

FIDDLER
Data Management Program
and Plotting Package
Users' Manual

Richard Booth
Sherman Fairchild Laboratory
Lehigh University
Bethlehem PA USA
IMEC, VZW

Kapeldreef 75
B-3001 Leuven BELGIUM

September 2, 1997

Contents

1	FIDDLER : Introduction	5
1.1	INTRODUCTION	6
2	FIDDLER : Plotting-package	9
2.1	Introduction	10
2.1.1	Brief Description of FIDDLER plotting-package Subroutines	10
2.1.2	Global Details and Conventions in FIDDLER plotting-package	12
2.2	FIDDLER plotting package subroutine specifications	14
2.2.1	XYPLOT	15
2.2.2	XPLOTE	18
2.2.3	CONTOUR	21
2.2.4	CONTOU2	23
2.2.5	CONTOU3	25
2.2.6	VECTORS	27
2.2.7	HISTOGM	29
2.2.8	PROBPLT	30
2.2.9	SURFACE	31
2.2.10	SURFAC2	33
2.2.11	SPACELN	35
2.2.12	PLTCHR	37
2.2.13	PLTVAL	39
2.2.14	PLOTN	40
2.3	Plotting package programming examples	44
2.3.1	XYPLOT Example: Polar-Parametric Curve	45
2.3.2	XYPLOT Example: Extra Label	47
2.3.3	XYPLOT Example: Line-types and LOG Axes	49
2.3.4	XYPLOT Example: Plotting Symbols and Characters	51
2.3.5	XPLOTE Example: Error Bars	53
2.3.6	XYPLOT Example: Error Bars without using XPLOTE	55
2.3.7	XYPLOT Example: border and options	57
2.3.8	CONTOUR Example: Family of Plane Curves	59
2.3.9	CONTOUR Example: Non-uniform Sampling Grid	61
2.3.10	HISTOGM Example: Random Samples with Transformation	63
2.3.11	HISTOGM Example: Wider Counting Intervals	65
2.3.12	PROBPLT Example: Exponential distribution	67
2.3.13	PROBPLT Example: Normal distribution	69
2.3.14	VECTORS Example: Sampled Vector Field	71
2.3.15	SURFACE Example: Opaque View at 30 Degrees	73
2.3.16	SURFACE Example: Translucent View at 10 Degrees	75
2.3.17	SURFACE Example: Plot Window Bigger than Sampling Window	77

2.3.18	SURFACE Example: Orthographic view	79
2.3.19	SURFACE Example: Oblique view - x-axis at front	81
2.3.20	SURFACE Example: Oblique view - y-axis at front	83
2.3.21	SURFACE Example: Border and options	85
2.3.22	SPACELN Example: Line Segments and Verticals	87
2.3.23	SPACELN Example: Plotting Symbols	89
2.3.24	PLTCHR Example: Rotation and Font Control	91
2.3.25	PLTCHR Example: Mathematical Expressions	93
3	FIDDLER : Interactive program	95
3.1	Introduction	96
3.1.1	Brief Description of FIDDLER Interactive program Capabilities	96
3.1.2	Global Details and Conventions in FIDDLER	97
3.2	FIDDLER Menu Organization	98
3.2.1	Summary of FIDDLER menus	98
3.2.2	[MAIN]: the main FIDDLER menu	101
3.2.3	[CHANGE]: change plotting parameters	103
3.2.4	[CHANGE.CHANNELS]: change x,y,z channels	104
3.2.5	[CHANGE.MODE]: change plotting mode	105
3.2.6	[CHANGE.WINDOW]: change plotting window	107
3.2.7	[CHANGE.ORIENTATION]: change labeling orientation	108
3.2.8	[CHANGE.SYMBOLS]: change plotting symbols, pen-colors, line-types	109
3.2.9	[CHANGE.PEN-CONTROL]: change pen-control parameters	110
3.2.10	[CHANGE.LABELING]: change labeling parameters	111
3.2.11	[CHANGE.EXTRA]: change routine-specific parameters	112
3.2.12	[EDIT]: edit data array	113
3.2.13	[EDIT.VIEW]: spread-sheet of data array	114
3.2.14	[EDIT.OPERATE]: operate on data array	115
3.2.15	[EDIT.OPERATE.UNARY]: unary operations	116
3.2.16	[EDIT.OPERATE.BINARY]: binary operations	117
3.2.17	[EDIT.OPERATE.KONSTANT]: channel/constant operations	118
3.2.18	[IMPORT]: load non-FIDDLER -format data files	119
3.2.19	[FUNCTION]: sample a one-parameter function	120
3.2.20	[MAKE]: make a hard-copy plot or printout	121
3.2.21	[ADD]: add labels to plot	122
3.2.22	[USER]: other applications	123
3.3	FIDDLER Examples and Applications	124
3.3.1	Example 1: Importing data from an ASCII data file.	125
3.3.2	Example 2: Importing data generated by FORTRAN program.	138
3.3.3	Example 3: Generating FIDDLER -format data file directly with a FORTRAN program.	142
3.3.4	Example 4: Entering Single points.	148
3.3.5	Example 5: Sampling Functions.	152
3.3.6	Example 6: Uploading Data generated by HP9836/HPseries300 version of FIDDLER	156
3.3.7	Example 7: The PostScript file generated by FIDDLER ; incorporating a graphic into a SCRIBE TM document	158
3.3.8	Example 8: The HPGL file generated by FIDDLER ; downloading and printing on an HPGL pen-plotter	161

<i>CONTENTS</i>	3
B FIDDLER : HELP files	179
C FIDDLER : Subroutine/function list	189
D FIDDLER : Installation	199
D.0.9 PREPARATION OF SUBDIRECTORIES FOR FIDDLER INSTALLATION . . .	200
D.0.10 INSTALLATION	203
E FIDDLER : Reference cards	205

Chapter 1

FIDDLER : Introduction

1.1 INTRODUCTION

The FIDDLER scientific data management program is a menu-driven system to input, edit, display, and plot data. FIDDLER performs many data management functions, such as:

Input

- Loads data from standard format (FIDDLER -format) files
- Imports data from 80-column space-or-comma-delimited ASCII files
- Imports data from other standard formats
- Samples functions
- Inputs data via manual entry

Edit

- Data may be edited on a point by point basis
- Data may be sorted, deleted, or filled in to columns or rows
- Linear and polynomial regression to give regression curve data
- Operations performed between columns of data to give another column of data

Change

- Change channels which are to be plotted
- Change plotting symbol, line-type, color, and labeling sizes
- Linear/Logarithmic axes
- Change plotting window
- Change plotting mode: x-y, contour, surface, space-line, histogram, etc.

Label

- Create and edit list of labels
- Interactively move, size, rotate, slant, and orient labels on plot

Output

- Save FIDDLER information to standard format (FIDDLER -format) file
- Preview plot on terminal
- Make hard copy of plot
- Create plot file of PostScript, HPGL, LN03 format plot commands.
- Create plot file of generic primitive graphics commands for post-processing by other printer drivers.

The original version of FIDDLER was developed on a Cyber 850 under the NOS/VE operating system. It is written in standard Fortran 77. It has been ported to many other mainframe and workstation environments: SUN/Sparc, Sony, Apollo workstation, DEC station, Convex, and VAX/VMS. Since this manual was originally written for the Cyber version of FIDDLER, small differences may apply to other environments.

FIDDLER is also a plotting package: a set of FORTRAN-callable subroutines which generate:

- X-Y plots: linear/logarithmic, error-bars
- Contour plots: labelled, color-band
- Surface plots: Orthographic, isometric, or oblique projections
- Histogram plots
- Probability plots
- Vector plots
- Space-curve plots
- Character string labels
- Primitive graphics functions: move, draw, fill, pen-color

FIDDLER is designed so that it may easily be ported to different environments:

- Only a minimal number of non-standard Fortran code is used (inline comments, for example).
- Machine-specific routines are confined to one file (speci.fsb).
- Graphics/graphical-text are generated using a small number of basic graphics operations. All graphics (including characters) by absolute moves and draws (vectors): only a very small subset of graphics functions is required from any terminal. Several terminal types and plotting devices are supported. A generic device type is also supported so that the graphical output may be post-processed and sent to any other printer or plotter.
- All terminal output passes through a single output stage routine. All terminal input passes through one of two routines: one for single-key entry (for possible "hot-key" implementations), and one for string entry.

FIDDLER plots and labels are generated by FORTRAN programs which need only:

- Define the type of terminal or plotting "device" in use. This requires one subroutine call.
- Generate (or Import data into) plotting arrays.
- Call one plotting subroutine.

The manual is divided into two parts: the first for the plotting-package routines and the second for the interactive FIDDLER program. In the first section, we discuss the global conventions used in the plotting package routines, describe the parameters required in each subroutine call, and present example FORTRAN programs which call the plotting routines. In the second section, we describe global conventions, present the menu hierarchy and menu specifications, and present several example FIDDLER sessions. The examples illustrate the use of the packages and various capabilities of the programs, but not all of them. Users are encouraged to explore further, beginning with these basic examples.

FIDDLER is offered to the public domain. The author requests only that the following be observed:

1. *No portion of* this manual or of the FIDDLER packages should be used for commercial gain without the written consent of the author.
2. The author would like to be notified of any new versions, improvements, modifications, implementations (on other computers) of FIDDLER . There is no formal registration, but I would appreciate knowing where, and on which machines FIDDLER has been successfully installed. Communication should reach me if it is addressed to: Richard Booth, Sherman Fairchild Center, Lehigh University, Bethlehem PA 18015 USA.
3. Please report any errors in the *original* code or manual to the author with adequate documentation.
4. Suggestions for improvement of existing routines or suggestions for new applications are appreciated.

Several random thoughts for the future:

- Line labels in FIDDLER : for pointing to curves.
- Left and Right axes: two scales on the same plot.
- Hot-key calculator.

Chapter 2

FIDDLER : Plotting-package

2.1 Introduction

2.1.1 Brief Description of FIDDLER plotting-package Subroutines

The FIDDLER plotting-package includes the following FORTRAN-callable subroutines:

1. **XYPLOT**: This routine plots y values versus x values. The y and x values are in two 1-d arrays. Linear and logarithmic modes are supported. Several curves may be plotted on the same set of axes by using successive calls to XYPLOT. A dotted-line grid may be plotted, if desired. Characters or lines may be used to plot the curve, one of 6 line types may be specified, one of 6 pens may be selected. Finally the plot axes may be flipped into any quadrant. (The labelling also flips.)
2. **XYPLOT2**: This routine performs all XYPLOT functions, with the addition of plotting an error-bar at each point.
3. **CONTOUR**: Contours of a 2-dimensional array of values are plotted. The 2-d array is specified at grid locations $(x, y)_i$. This grid is specified by two vectors: x-axis and y-axis values. (Each "element" of the grid is a rectangle, as for a finite-difference approach.) The grid-spacing need not be uniform and may be plotted with dotted-lines if desired. A plotting window is specified, allowing the user to zoom-in on various regions of the plot. The axes may be flipped as described in XYPLOT.
4. **CONTOU2**: This routine performs all CONTOUR functions, with the addition of labelling each contour with its value.
5. **CONTOU3**: Color bands are plotted along the contours of a two-dimensional function, as described for CONTOUR. The color bands are labelled at the right-hand side of the plot.
6. **SURFACE**: This routine plots an isometric 3-dimensional view of a 2-d array of values which is specified over a grid of $(x, y)_i$ values, as described for CONTOUR. The surface is mapped onto a uniform grid, within a plotting window which is specified by the calling program. The resolution of this uniform grid is also caller-specified. Other parameters which are sent by the calling program are: angle of tilt about the $x = y$ line (the view is one where the surface is first rotated 45 degrees so that the $x = y$ line is horizontal, then the axes are rotated about the $x = y$ line), hidden-line removal, which surface(s) to plot - top and/or bottom, and which octant to have axes fill (the flipping described in XYPLOT, extended to 3-d).
7. **SURFAC2**: This routine plots an orthographic or oblique 3-D view of a 2-D array of values. All of the features described for SURFACE are implemented, with the addition of a second angle of rotation, choice of no mapping onto a uniform grid, choice of oblique front axis, and plotting skirts at edges of the surface.
8. **VECTORS**: "Vectors" are plotted which have a line-segment and arrow-size which are scaled to sizes corresponding to the magnitude and pointing in the same direction of vectors of a "vector-field". The values of the vector-field are specified by two 2-d arrays which contain the x and y components of the vectors at the (not-necessarily uniform) grid of $(x, y)_i$ points. These components are mapped onto a uniform grid so that the plotted "vectors" are uniformly spaced.
9. **HISTOGRAM**: This routine displays a histogram of a vector of values. The calling program sends the vector of values to "count", and the counting window and interval. The counting window is enlarged a little bit so that axes labels will not be too long, and this enlarged window is split up into counting intervals. (The counting intervals are aligned with the calling specified lower

counting limit.) "Counts" are accumulated for each interval within the plotting window. That is, if x_i falls within a particular interval, the count for that interval is incremented. The final counts are displayed as rectangles on the plot.

10. **PROBPLT**: This routine displays a probability plot of a vector of values. The values are first sorted, then the cumulative distribution of values is plotted on a probability scale. Values of the vector may be plotted on the right hand of the plot.
11. **SPACELN**: A three-dimensional view of a space-line is plotted. The space-line is described by consecutive points $(x, y, z)_i$. These are sent by the calling-program in three 1-d arrays. The same scheme for plotting the 3-d view as described in SURFACE is employed and all of the parameters for changing the view are available for SPACELN. In addition, for better clarity, lines may be dropped from the space-line to the x-y plane at specified intervals of points.
12. **PLTCHR**: The character generator routine. This supports 12 different fonts which are a subset of the Hershey fonts. These are organized as four different font-families (Simplex, Complex, Duplex and Triplex), each containing a roman font, a script font, and a Greek/Gothic/or Cyrillic font. There are several control characters which are used for sub/super-scripts, backspacing, overprinting, font-changing, and in-line sizing. In addition, the characters may be rotated, slanted, and oriented in 9 positions with respect to the specified plot-location. All of the n-dimensional plotting routines make use of this routine, for labeling plots, so that greek, script, over-printed, superscripted, and sub-scripted characters may be used to label axes and the plot titles.
13. **PLTVAL**: The number value labeling routine. This supports two different ways of labeling characters. The "*" format prints 10^n using the superscript mode mentioned above. Other formats are specified with a fortran format specification. The n-dimensional plotting routines all make use of this routine to label the axes. The numbers may be rotated, slanted, or oriented as described for PLTCHR.
14. **PLOTN**: The graphics device-driver routine. The routine works in the following way. Only very primitive graphics functions are performed, such as clearing the graphics screen, moving or drawing to a pixel location, and changing pen colors. Characters are internally generated via calls to this subroutine. In addition, several different line-styles are internally supplied, which do not rely upon the line-styles which may be available on certain terminals. This means that it is easy to enter the proper parameters for other terminal types into this routine. Currently, the routines support the following graphics devices:
 - (a) Tektronix 40xx, 41xx, 42xx terminals
 - (b) Seiko 11xx terminals
 - (c) HPGL plotter (Hewlett-Packard Graphics Language)
 - (d) PostScript printer
 - (e) LN03 printer
 - (f) CALC: generic output device file for postprocessing and sending to any other type of plotter
15. A personal or desk-top computer, running a terminal emulator which traps and interprets graphics escape-code sequences for one of the graphics devices listed above, can display FIDDLER plots. Before any of the other routines are called, the user's calling program must call PLOTN to define the type of plotting device which is in use (or emulated). Internal to PLOTN, this amounts to setting a flag corresponding to a particular plot device. In addition, as mentioned in the last section, many default plotting parameters may be changed by calling PLOTN.

2.1.2 Global Details and Conventions in FIDDLER plotting-package

The FIDDLER plotting package employs only one subroutine to interface with the various graphics devices: PLOTN. PLOTN performs only very simple graphics commands, such as MOVE, DRAW and CHANGE PEN for each graphics device. PLOTN calls are also used to change internal plotting parameters used by FIDDLER. In this manner, the graphics images which are created on different graphics devices (terminals, pen plotters and printers) are very similar.¹ Line-types and characters are built into FIDDLER, rather than the use of character or line-types available on certain terminals. Additional graphics device drivers may be very quickly and easily incorporated into PLOTN, since only a very minimal number of graphics commands are necessary to generate a plot.

All of the MOVE and DRAW commands sent to PLOTN (ICMD=2 or 3) send graphics values in "plot units." These values are between 0 and 10 and correspond to the minimum and maximum horizontal and vertical extents of the graphics window. Once X,Y values (in plot units) reach PLOTN, they are:

1. "Clipped" to the current clipping window. This window is (0,0),(10,10) when the program is initialized. The clipping window is reduced to (2,2),(8,8) when data is plotted to 1- and 2-dimensional plots, so that plotted data lines or symbols do not appear outside of the plotting border. The clipping window may be modified by a PLOTN call, but may never be outside of the (0,0),(10,10) maximum (plot unit) window. This eliminates errors involving attempts to plot outside of the screen address space of the graphics device.
2. Scaled to values which will fit into the pixel address space of the current graphics device. In addition, the physical screen height to width ratio is used to scale the x,y values so that plotted circles will actually look like circles on the graphics device. Optionally, a PLOTN call may be used to make use of the entire graphics screen, rather than the portion necessary for a 1:1 aspect ratio.
3. Sent to the current graphics device in the form of an appropriate escape sequence or other command sequence.
4. Saved for future use: Some terminals DRAW by moving to the previous graphics location and filling in the straight line to the current graphics location.

There are only two subroutines which generate the axes and titles to which data is plotted:

1. FRAME2D : Used by XYplot, XYplotE, CONTOUR, CONTOU2, CONTOU3, VECTORS, HISTOgm and PROBPLT. These are the routines which create a two-dimensional view of the data which is plotted.
2. FRAME3D : Used by SURFACE, SURFAC2, and SPACELN. These are the routines which create a three-dimensional view of the data which is plotted.

The code for generating the various types of 2-D and 3-D plots is greatly simplified by the use of just these two routines, since the 2-D and 3-D routines each require the same sort of frame or axes. New applications might very easily make use of these two routines, greatly minimizing the programming effort. PLOTN has a series of commands which change the default plotting parameters used in FRAME2D and FRAME3D. These parameters include axes label sizes, numbering format, and border thickness, for example. If the user has a preference for a set of plotting parameters different from the default values, then the parameters can be changed just once at the beginning of each application program, since the FRAME2D and FRAME3D plotting parameters are separate variables.

FRAME2D and FRAME3D both provide the ability to "flip" the axes into any one of the four quadrants (2D plots) or one of the eight octants (3D plots). This is useful in the 3D plots, since a particular plotting view in one octant may obscure more of the surface than the view in another octant. When the axes of 2D plots are "flipped" into a particular quadrant of the x-y plane, (or when 3D plots are "flipped" into

¹The only differences are qualities such as size, color, pen thickness, and resolution.

a particular octant), the plotting window limits go from low values to high values in the same sense as the absolute values of the quadrant axes values. For example, a 2D view in the 2^{nd} quadrant would have the lower x-axis limit at the RIGHT, the upper x-axis limit at the LEFT, the lower y-axis limit at the BOTTOM, and the upper y-axis limit at the TOP. This flipping capability is accessed by the user via the IORIENT parameter in all of the n-dimensional plotting subroutines (except HISTOGRAM).

The routines XYPLOT and SPACELN plot data values as either discrete points (labeled by squares, triangles, or other symbols), or as line-segments connecting the data values. The line-segments are plotted using one of six line-type styles.

There are two formats for labeling the axes in FRAME2D and FRAME3D. The number labels can either be strictly an integer times a power of ten (default) or a format which can be accessed by a PLOTN call: the "*" format. This format labels powers of ten with 10^n instead of the E type format.

The character labeler PLTCHR has several control characters which will

- Change the font-family from simplex to duplex or complex or triplex.
- Change the fonts from Roman to Script or Greek/Gothic/or Cyrillic.
- Backspace.
- Overprint.
- Super-script or Sub-Script.
- Increase or decrease the size of following characters.

These control characters may be enabled or disabled in the call to PLTCHR. All of the n-dimensional plotting subroutines use PLTCHR to label the axes and titles, and enable these control characters.

2.2 FIDDLER plotting package subroutine specifications

This section describes each of the parameters in each subroutine call. The N-dimensional plotting routines are:

HISTOGM	plot histogram of array	x(t)
PROBPLT	plot probability plot of array	x(t)
XYPLOT	plot x vs y plot of points	[x(t),y(t)]
XYPLOTE	plot x vs y plot of points with error bars	[x(t),y(t) +/- e(t)]
CONTOUR	plot contours of surface	z(x,y)
CONTOU2	plot contours of surface (contours labelled with value)	z(x,y)
CONTOU3	plot contours of surface (color band plot)	z(x,y)
SURFACE	plot 3D view of surface (isometric projection)	z(x,y)
SURFAC2	plot 3D view of surface (orthographic or oblique projection)	z(x,y)
SPACELN	plot 3D view of points	[x(t),y(t),z(t)]
VECTORS	plot vectors of vector field	[zx(x,y),zy(x,y)]

The labeling and support routines are:

PLTCHR	plot strings
PLTVAL	plot numbers
PLOTN	perform primitive graphics commands

2.2.1 XYPLOT

This subroutine plots the points $(x, y)_i$ from the two arrays $x(*)$, and $y(*)$. The subroutine call is:

```
CALL XYPLOT(X,Y,N,
+          XLO,XHI,YLO,YHI,LINLOGX,LINLOGY,
+          LABELX,LABELY,LABELM,LABELS,
+          SYMBOL,LFIRST,LLAST,LGRID,
+          IORIENT,IPEN,ILINE)
```

where:

X(*),Y(*) [REAL(*)] are two 1-dimensional arrays containing the x- and y-coordinates of the data points to be plotted. The data arrays are dimensioned in the calling program, and need not be completely "full", since the parameter N indicates the number of valid data points.

N [INTEGER] is the number of valid data points in the arrays X(*) and Y(*).

XLO,XHI [REAL] are the lower and upper limits of the x-axis plotting window. These numbers are re-scaled to more convenient numbers for labeling the axes, if necessary.

YLO,YHI [REAL] are the lower and upper limits of the y-axis plotting window.

LINLOGX,LINLOGY [CHARACTER*3; 'LIN' or 'LOG'] These two parameters specify linear or logarithmic axes. There are no default values, and the strings must be uppercase.

LABELX,LABELY [CHARACTER*(*)] are the axes labels. These can be variable length strings or character variables. The control characters which are listed in the subroutine PLTCHR are enabled, so that these axes labels may contain greek characters, sub-/super-scripts, etc.

LABELM,LABELS [CHARACTER*(*)] are the main and sub-titles.

SYMBOL [CHARACTER*1] is the plotting symbol to be used. This may be a one-character string or character variable. There are three special plotting symbols:

[L] plot line-segments between successive points.

[S] plot squares at each point

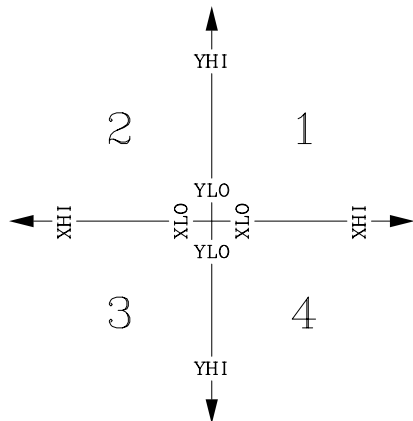
[T] plot triangles at each point

LFIRST [LOGICAL] If LFIRST is .TRUE., then the graphics device is initialized, cleared, and axes are plotted. This would be the case for the first subroutine call to XYPLOT in a series of overlaid curves or data sets plotted on the same axes, or any single line plot.

LLAST [LOGICAL] If LLAST is .TRUE., then the graphics device is closed. This would be the case for the last subroutine call to XYPLOT in a series of overlaid curves or data sets plotted on the same axes, or any single line plot.

LGRID [LOGICAL] If LGRID is .TRUE., then a grid is drawn on the plot with line-type 4, at major axes divisions.

IORIENT [INTEGER] is the quadrant to "flip" the axes into: axes values go from low to high values in the same sense as the absolute values of the respective quadrant. The IORIENT values, [1-4], are shown in the following figure along with the orientation of the respective window limits.



IPEN [INTEGER] is the number of the pen color used to draw the data points. There are six default pen color values in the pen table. Numbers which are greater than 6 or less than 1 are MOD'd to give values between these two limits. The pen colors corresponding to the pen color values [1-6] for each graphics device are:

IPEN:	Pen color value:	Kermit:	Seiko:	TK1:	CALCOMP:	PostScript:
1	1	WHITE	BLACK	WHITE	BLACK	BLACK
2	2	WHITE	WHITE	RED	RED	GRAY
3	3	WHITE	RED	GREEN	GREEN	GRAY
4	4	WHITE	GREEN	BLUE	BLUE	GRAY
5	5	WHITE	BLUE	CYAN	BLACK	GRAY
6	6	WHITE	CYAN	MAGENTA	RED	GRAY

ILINE [INTEGER] is the number of the line-type used to draw the data line and symbols. Data symbols are best drawn with with line-type 1 (solid), for clarity. The ILINE parameter is most useful when plotting a series of data sets with continuous lines (as opposed to data symbols) on the same set of axes. Each data set is assigned a different line-type number [1-6] in order to distinguish the curve of one set from another. There are six default line-type values in the line-type table, which may be interchanged using a PLOTN call.

ILINE: Line-type value: Description:

Line-type value	Description
1	solid
2	long segment long space
3	med. segment med. space
4	short segment short space
5	short segment med. space
6	short segment long space

The following figure shows the line-types and "colors" produced by FIDDLER when the graphics device is a PostScript file.

Index:	Line-type:	Pen-color:
1	—————	—————
2	-----	—————
3	-----	—————
4	—————
5	-----	—————
6	-----	—————

2.2.2 XYPLOTE

This subroutine plots the points $(x, y)_i$ from the two arrays $x^{(*)}$, and $y^{(*)}$ with error bars. The error values at each point are passed in the array $e^{(*)}$. The subroutine call is:

```
CALL XYPLOTE(X,Y,E,N,
+           XLO,XHI,YLO,YHI,LINLOGX,LINLOGY,
+           LABELX,LABELY,LABELM,LABELS,
+           SYMBOL,LFIRST,LLAST,LGRID,
+           IORIENT,IPEN,ILINE)
```

where:

X(*), **Y(*)** [REAL(*)] are two 1-dimensional arrays containing the x- and y-coordinates of the data points to be plotted. The data arrays are dimensioned in the calling program, and need not be completely "full", since the parameter N indicates the number of valid data points.

E(*) [REAL(*)] is a 1-dimensional arrays containing the error values at each point. The array is dimensioned in the calling program, and need not be completely "full", since the parameter N indicates the number of valid data points. The error bars are plotted at $[X(I), Y(I)-E(I); X(I), Y(I)+E(I)]$.

N [INTEGER] is the number of valid data points in the arrays **X(*)** and **Y(*)**.

XLO, **XHI** [REAL] are the lower and upper limits of the x-axis plotting window. These numbers are re-scaled to more convenient numbers for labeling the axes, if necessary.

YLO, **YHI** [REAL] are the lower and upper limits of the y-axis plotting window.

LINLOGX, **LINLOGY** [CHARACTER*3; 'LIN' or 'LOG'] These two parameters specify linear or logarithmic axes. There are no default values, and the strings must be uppercase.

LABELX, **LABELY** [CHARACTER*(*)] are the axes labels. These can be variable length strings or character variables. The control characters which are listed in the subroutine **PLTCHR** are enabled, so that these axes labels may contain greek characters, sub-/super-scripts, etc.

LABELM, **LABELS** [CHARACTER*(*)] are the main and sub-titles.

SYMBOL [CHARACTER*1] is the plotting symbol to be used. This may be a one-character string or character variable. There are three special plotting symbols:

L plot line-segments between successive points.

S plot squares at each point

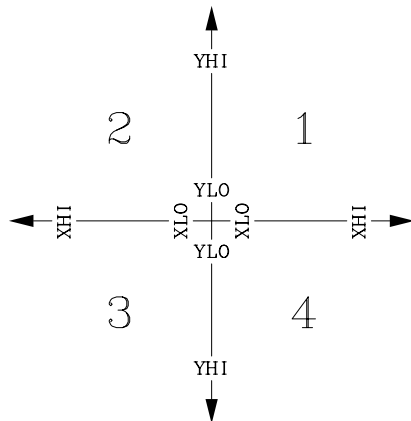
T plot triangles at each point

LFIRST [LOGICAL] If **LFIRST** is **.TRUE.**, then the graphics device is initialized, cleared, and axes are plotted. This would be the case for the first subroutine call to **XYPLOT** in a series of overlaid curves or data sets plotted on the same axes, or any single line plot.

LLAST [LOGICAL] If **LLAST** is **.TRUE.**, then the graphics device is closed. This would be the case for the last subroutine call to **XYPLOT** in a series of overlaid curves or data sets plotted on the same axes, or any single line plot.

LGRID [LOGICAL] If **LGRID** is **.TRUE.**, then a grid is drawn on the plot with line-type 4, at major axes divisions.

IORIENT [INTEGER] is the quadrant to "flip" the axes into: axes values go from low to high values in the same sense as the absolute values of the respective quadrant. The IORIENT values, [1-4], are shown in the following figure along with the orientation of the respective window limits.



IPEN [INTEGER] is the number of the pen color used to draw the data points. There are six default pen color values in the pen table. Numbers which are greater than 6 or less than 1 are MOD'd to give values between these two limits. The pen colors corresponding to the pen color values [1-6] for each graphics device are:

IPEN:	Pen color value:	Kermit:	Seiko:	TK1:	CALCOMP:	PostScript:
1	1	WHITE	BLACK	WHITE	BLACK	BLACK
2	2	WHITE	WHITE	RED	RED	GRAY
3	3	WHITE	RED	GREEN	GREEN	GRAY
4	4	WHITE	GREEN	BLUE	BLUE	GRAY
5	5	WHITE	BLUE	CYAN	BLACK	GRAY
6	6	WHITE	CYAN	MAGENTA	RED	GRAY

ILINE [INTEGER] is the number of the line-type used to draw the data line and symbols. Data symbols are best drawn with with line-type 1 (solid), for clarity. The ILINE parameter is most useful when plotting a series of data sets with continuous lines (as opposed to data symbols) on the same set of axes. Each data set is assigned a different line-type number [1-6] in order to distinguish the curve of one set from another. There are six default line-type values in the line-type table, which may be interchanged using a PLOTN call.

ILINE: Line-type value: Description:

Line-type value	Description
1	solid
2	long segment long space
3	med. segment med. space
4	short segment short space
5	short segment med. space
6	short segment long space

The following figure shows the line-types and "colors" produced by FIDDLER when the graphics device is a PostScript file.

Index:	Line-type:	Pen-color:
1	—————	—————
2	-----	—————
3	-----	—————
4	—————
5	-----	—————
6	-----	—————

2.2.3 CONTOUR

This subroutine plots contours of a three-dimensional surface specified in the 2-dimensional array $Z(*,*)$. It is called by:

```
CALL CONTOUR(Z,X,Y,C,NZ,NX,NY,NC,
+           XLO,XHI,YLO,YHI,FGRID,
+           LABELX,LABELY,LABELM,LABELS,
+           IORIENT)
```

where:

$Z(*,*)$ [REAL(NZ,*)] is a 2-dimensional array of values of the surface, specified at grid-locations $(X(I), Y(J)) : I = 1 \dots NX; J = 1 \dots NY$.

The dimensions of $Z(*,*)$ must be declared in the calling program and the first physical dimension of $Z(*,*)$, NZ, is passed to the CONTOUR subroutine.

$X(*), Y(*)$ [REAL(*)] are two 1-dimensional vectors which contain the values of the grid divisions at which $Z(*,*)$ is specified.

$C(*)$ [REAL(*)] is a 1-dimensional array which contains the contour values. Equivalence contours for each contour value $C(I):I=1..NC$ are drawn for the surface which is interpolated from the array of values $Z(*,*)$.

NZ [INTEGER] is the first physical dimension of the 2-dimensional array $Z(*,*)$. This is the value used in declaring $Z(*,*)$ in the calling routine.

NX, NY [INTEGER] are the number of x and y grid divisions.

NC [INTEGER] is the number of contour values in $C(*)$ to be drawn.

XLO, XHI [REAL] are the lower and upper limits of the x-axis plotting window. These numbers are re-scaled to more convenient numbers for labeling the axes, if necessary.

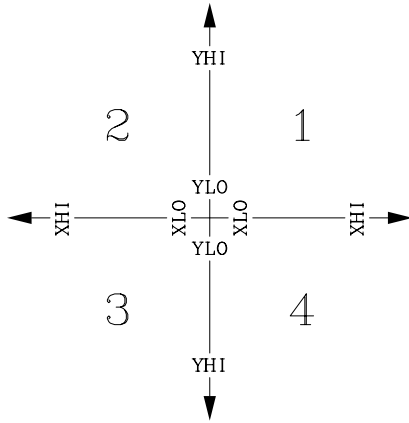
YLO, YHI [REAL] are the lower and upper limits of the y-axis plotting window.

$FGRID$ [LOGICAL] If $FGRID$ is .TRUE., then the grid of constant $X(*)$ and $Y(*)$ values is drawn using line-type 4.

$LABELX, LABELY$ [CHARACTER*(*)] are the x and y axes labels. These can be variable length strings or character variables. The control characters which are listed in the subroutine PLTCHR are enabled, so that the axes labels may contain greek characters, sub-/super-scripts, etc.

$LABELM, LABELS$ [CHARACTER*(*)] are the main and sub-titles.

IORIENT [INTEGER] is the quadrant to flip the axes into: axes values go from low to high values in the same sense as the absolute values of the respective quadrant. The IORIENT values, [1-4], are shown in the following figure along with the orientation of the respective window limits.



2.2.4 CONTOU2

This subroutine plots contours of a three-dimensional surface specified in the 2-dimensional array $Z(*,*)$. The contours are interrupted by labels indicating the contour value. It is called by:

```
CALL CONTOU2(Z,X,Y,C,NZ,NX,NY,NC,
+           XLO,XHI,YLO,YHI,FGRID,
+           LABELX,LABELY,LABELM,LABELS,
+           IORIENT)
```

where:

$Z(*,*)$ [REAL(NZ,*)] is a 2-dimensional array of values of the surface, specified at grid-locations $(X(I), Y(J)) : I = 1 \dots NX; J = 1 \dots NY$.

The dimensions of $Z(*,*)$ must be declared in the calling program and the first physical dimension of $Z(*,*)$, NZ, is passed to the CONTOUR subroutine.

$X(*), Y(*)$ [REAL(*)] are two 1-dimensional vectors which contain the values of the grid divisions at which $Z(*,*)$ is specified.

$C(*)$ [REAL(*)] is a 1-dimensional array which contains the contour values. Equivalence contours for each contour value $C(I):I=1..NC$ are drawn for the surface which is interpolated from the array of values $Z(*,*)$.

NZ [INTEGER] is the first physical dimension of the 2-dimensional array $Z(*,*)$. This is the value used in declaring $Z(*,*)$ in the calling routine.

NX, NY [INTEGER] are the number of x and y grid divisions.

NC [INTEGER] is the number of contour values in $C(*)$ to be drawn.

XLO, XHI [REAL] are the lower and upper limits of the x-axis plotting window. These numbers are re-scaled to more convenient numbers for labeling the axes, if necessary.

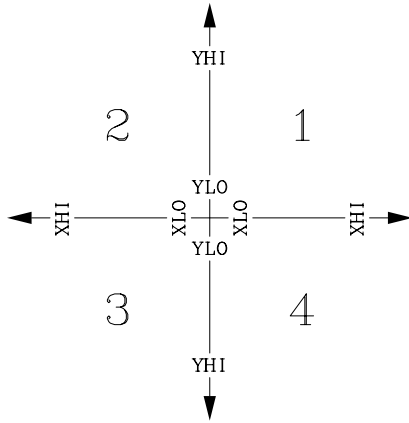
YLO, YHI [REAL] are the lower and upper limits of the y-axis plotting window.

$FGRID$ [LOGICAL] If $FGRID$ is .TRUE., then the grid of constant $X(*)$ and $Y(*)$ values is drawn using line-type 4.

$LABELX, LABELY$ [CHARACTER*(*)] are the x and y axes labels. These can be variable length strings or character variables. The control characters which are listed in the subroutine PLTCHR are enabled, so that the axes labels may contain greek characters, sub-/super-scripts, etc.

$LABELM, LABELS$ [CHARACTER*(*)] are the main and sub-titles.

IORIENT [INTEGER] is the quadrant to flip the axes into: axes values go from low to high values in the same sense as the absolute values of the respective quadrant. The IORIENT values, [1-4], are shown in the following figure along with the orientation of the respective window limits.



2.2.5 CONTOU3

This subroutine plots color band contours of a three-dimensional surface specified in the 2-dimensional array $Z(*,*)$. It is called by:

```
CALL CONTOU3(Z,X,Y,C,NZ,NX,NY,NC,
+           XLO,XHI,YLO,YHI,FSCALE,
+           LABELX,LABELY,LABELM,LABELS,
+           IORIENT)
```

where:

$Z(*,*)$ [REAL(NZ,*)] is a 2-dimensional array of values of the surface, specified at grid-locations $(X(I), Y(J)) : I = 1 \dots NX; J = 1 \dots NY$.

The dimensions of $Z(*,*)$ must be declared in the calling program and the first physical dimension of $Z(*,*)$, NZ, is passed to the CONTOUR subroutine.

$X(*), Y(*)$ [REAL(*)] are two 1-dimensional vectors which contain the values of the grid divisions at which $Z(*,*)$ is specified.

$C(*)$ [REAL(*)] is a 1-dimensional array which contains the contour values. Equivalence contours for each contour value $C(I):I=1..NC$ are drawn for the surface which is interpolated from the array of values $Z(*,*)$.

NZ [INTEGER] is the first physical dimension of the 2-dimensional array $Z(*,*)$. This is the value used in declaring $Z(*,*)$ in the calling routine.

NX,NY [INTEGER] are the number of x and y grid divisions.

NC [INTEGER] is the number of contour values in $C(*)$ to be drawn.

XLO,XHI [REAL] are the lower and upper limits of the x-axis plotting window. These numbers are re-scaled to more convenient numbers for labeling the axes, if necessary.

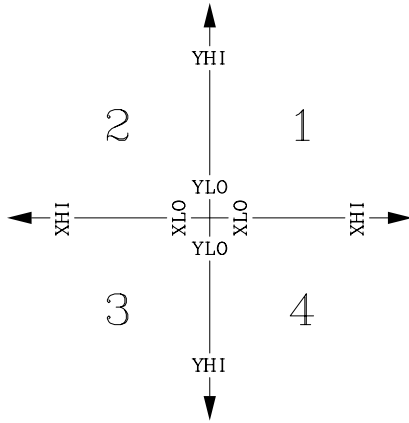
YLO,YHI [REAL] are the lower and upper limits of the y-axis plotting window.

FSCALE [LOGICAL] If FSCALE is .TRUE., then a scale of colored boxes and contour values is plotted at the right-hand side of the plot. The value indicates the lower value of the interval; the upper value of the interval is indicated by the value of the next higher contour value.

LABELX,LABELY [CHARACTER*(*)] are the x and y axes labels. These can be variable length strings or character variables. The control characters which are listed in the subroutine PLTCHR are enabled, so that the axes labels may contain greek characters, sub-/super-scripts, etc.

LABELM,LABELS [CHARACTER*(*)] are the main and sub-titles.

IORIENT [INTEGER] is the quadrant to flip the axes into: axes values go from low to high values in the same sense as the absolute values of the respective quadrant. The IORIENT values, [1-4], are shown in the following figure along with the orientation of the respective window limits.



2.2.6 VECTORS

This subroutine draws vectors with lengths and directions proportional to the values of a vector field specified in the two 2-dimensional arrays ARRAYX and ARRAYY. It is called by

```
CALL VECTORS(ZX,ZY,X,Y,NZ,NX,NY,NG,
+           XLO,XHI,YLO,YHI,
+           LABELX,LABELY,LABELM,LABELS,
+           IORIENT)
```

where:

ZX(*,*),ZY(*,*) [REAL(NZ,*)] are 2-dimensional arrays of values of the x- and y-components of the vector field, specified at grid-locations $[X(I),Y(J)]:I=1..NX;J=1..NY$. The dimensions of ZX(*,*) and ZY(*,*) must be declared in the calling program and the first physical dimension of both ZX(*,*) and ZY(*,*) is passed to the VECTORS subroutine.

X(*),Y(*) [REAL(*)] are two 1-dimensional vectors which contain the values of the grid divisions at which ZX(*,*) and ZY(*,*) are specified.

NZ [INTEGER] is the first physical dimension of the 2-dimensional array Z(*,*). This is the value used in declaring Z(*,*) in the calling routine.

NX,NY [INTEGER] are the number of x and y grid divisions.

NG [INTEGER] is the number of equally-spaced divisions in both x and y, over the plotting region specified by XLO, XHI, YLO, and YHI (after modifying the window limits to more convenient values). The values of the vector-arrays are mapped onto this uniform grid, and vector arrows with length and direction corresponding to the mapped values are drawn with the base of the arrow stationed at these grid locations.

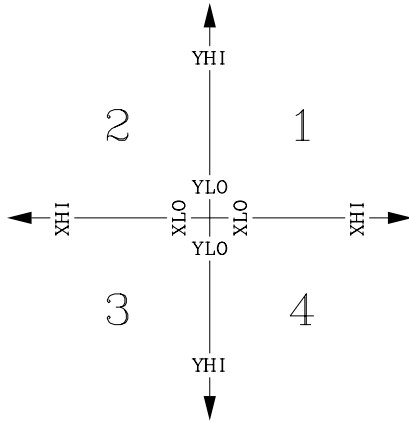
XLO,XHI [REAL] are the lower and upper limits of the x-axis plotting window. These numbers are re-scaled to more convenient numbers for labeling the axes, if necessary.

YLO,YHI [REAL] are the lower and upper limits of the y-axis plotting window.

LABELX,LABELY [CHARACTER*(*)] are the x- and y-axis labels. These can be variable length strings or character variables. The control characters which are listed in the subroutine PLTCHR are enabled, so the axes labels may contain greek characters, sub-/super-scripts, etc.

LABELM,LABELS [CHARACTER*(*)] are the main and sub-titles.

IORIENT [INTEGER] is the quadrant to flip the axes into: axes values go from low to high values in the same sense as the absolute values of the respective quadrant. The IORIENT values, [1-4], are shown in the following figure along with the orientation of the respective window limits.



2.2.7 HISTOGM

This subroutine plots a histogram of counts of a 1-dimensional array of values. The counting window and resolution is specified by the calling program. It is called by:

```
CALL HISTOGM(X,N,  
+           XLO,XHI,DX,  
+           LABELX,LABELM,LABELS)
```

where:

X(*) [REAL] is the vector of values which are to be counted.

N [INTEGER] is the number of valid entries in X(*) which are to be counted.

XLO,XHI [REAL] are the lower and upper limits of the counting window. This window is re-scaled, if necessary, so that convenient values may be labeled on the x-axis. The counting window is discretized into increments which line up with the edge of the original (unscaled) XLO value.

DX [REAL] is the increment which is used to discretize the counting window.

LABELX [CHARACTER*(*)] is the x-axis label. This can be a variable length string or character variable. The control characters which are listed in the subroutine PLTCHR are enabled, so that the axis label may contain greek characters, sub-/super-scripts, etc.

LABELM,LABELS [CHARACTER*(*)] are the main and sub-titles.

2.2.8 PROBPLT

This subroutine plots a probability plot of a 1-dimensional array of values. It is called by:

```
CALL PROBPLT(X,N,
+           SYMBOL,
+           LABELX,LABELM,LABELS,
+           FLHS,FRHS)
```

where:

X(*) [REAL] is the vector of values which are to be plotted.

N [INTEGER] is the number of valid entries in X(*) which are to be plotted.

SYMBOL [CHARACTER*1] is the plotting symbol to be used. This may be a one-character string or character variable. There are three special plotting symbols:

L plot line-segments between successive points.

S plot squares at each point

T plot triangles at each point

LABELX [CHARACTER*(*)] is the x-axis label. This can be a variable length string or character variable. The control characters which are listed in the subroutine PLTCHR are enabled, so that the axis label may contain greek characters, sub-/super-scripts, etc.

LABELM,LABELS [CHARACTER*(*)] are the main and sub-titles.

FLHS [LOGICAL] If .TRUE., the left hand side of the plot is labeled with the values of the probability divisions.

FRHS [LOGICAL] If .TRUE., the right hand side of the plot is labeled with the values of the array X(*)

2.2.9 SURFACE

This subroutine plots a 2-dimensional representation of a 3-dimensional surface. The view is an isometric projection. The subroutine call is:

```
CALL SURFACE(Z,X,Y,NZ,NX,NY,NG,
+           XLO,XHI,YLO,YHI,
+           LABELX,LABELY,LABELZ,LABELM,LABELS,
+           ANGLE,IMODE,LOPAQUE,IORIENT)
```

where:

Z(*,*) [REAL(NZ,*)] is a 2-dimensional array of values of the surface, specified at grid-locations $(X(I), Y(J)) : I = 1 \dots NX; J = 1 \dots NY$.

The dimensions of **Z(*,*)** must be declared in the calling program and the first physical dimension of **Z(*,*)**, **NZ**, is passed to the **CONTOUR** subroutine.

X(*),Y(*) [REAL(*)] are two 1-dimensional vectors which contain the values of the grid divisions at which **Z(*,*)** is specified.

NZ [INTEGER] is the first physical dimension of the 2-dimensional array **Z(*,*)**. This is the value used in declaring **Z(*,*)** in the calling routine.

NX,NY [INTEGER] are the number of x and y grid divisions.

NG [INTEGER] is the number of equally-spaced divisions in both x and y, over the plotting region specified by **XLO, XHI, YLO, and YHI** (after modifying the window limits to more convenient values). The values of the surface are mapped onto this uniform grid.

XLO,XHI [REAL] are the lower and upper limits of the x-axis plotting window. These numbers are re-scaled to more convenient numbers for labeling the axes, if necessary.

YLO,YHI [REAL] are the lower and upper limits of the y-axis plotting window.

LABELX,LABELY,LABELZ [CHARACTER*(*)] are the axes labels. These can be variable length strings or character variables. The control characters which are listed in the subroutine **PLTCHR** are enabled, so that the axes labels may contain greek characters, sub-/super-scripts, etc.

LABELM,LABELS [CHARACTER*(*)] are the main and sub-titles.

ANGLE [REAL] is the angle of rotation about the $x=y$ line (in degrees). The 3-dimensional view generated by **SURFACE** keeps the line $(x=y; z=0)$ aligned horizontally on the screen; the z-axis is tilted **ANGLE** degrees toward the front of the screen.

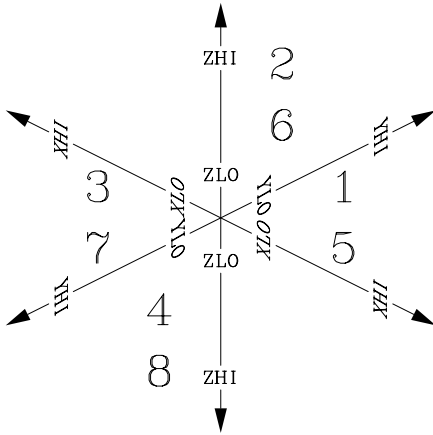
IMODE [INTEGER] is used to specify which surfaces are to be plotted:

IMODE: top surface: bottom surface:

```
-----
 1          X
 2                      X
 3          X          X
```

LOPAQUE [LOGICAL] If **LOPAQUE** is **.TRUE.**, then hidden lines are not plotted. Otherwise, the surface is 'translucent'.

IORIENT is the octant to "flip" the axes into: axes values go from low to high values in the same sense as the absolute values of the respective octant. The IORIENT values, [1-8], are shown in the following figure along with the orientation of the respective window limits.



2.2.10 SURFAC2

This subroutine plots a 2-dimensional representation of a 3-dimensional surface. Orthographic and oblique projections are supported. The subroutine call is:

```
CALL SURFAC2(Z,X,Y,NZ,NX,NY,NG,
+           XLO,XHI,YLO,YHI,
+           LABELX,LABELY,LABELZ,LABELM,LABELS,
+           PHI,THETA,IMODE,LOPAQUE,IORIENT)
```

where:

Z(*,*) [REAL(NZ,*)] is a 2-dimensional array of values of the surface, specified at grid-locations $(X(I), Y(J)) : I = 1 \dots NX; J = 1 \dots NY$.

The dimensions of **Z(*,*)** must be declared in the calling program and the first physical dimension of **Z(*,*)**, **NZ**, is passed to the **CONTOUR** subroutine.

X(*),Y(*) [REAL(*)] are two 1-dimensional vectors which contain the values of the grid divisions at which **Z(*,*)** is specified.

NZ [INTEGER] is the first physical dimension of the 2-dimensional array **Z(*,*)**. This is the value used in declaring **Z(*,*)** in the calling routine.

NX,NY [INTEGER] are the number of x and y grid divisions.

NG [INTEGER] is the number of equally-spaced divisions in both x and y, over the plotting region specified by **XLO, XHI, YLO, and YHI** (after modifying the window limits to more convenient values). The values of the surface are mapped onto this uniform grid.

XLO,XHI [REAL] are the lower and upper limits of the x-axis plotting window. These numbers are re-scaled to more convenient numbers for labeling the axes, if necessary.

YLO,YHI [REAL] are the lower and upper limits of the y-axis plotting window.

LABELX,LABELY,LABELZ [CHARACTER*(*)] are the axes labels. These can be variable length strings or character variables. The control characters which are listed in the subroutine **PLTCHR** are enabled, so that the axes labels may contain greek characters, sub-/super-scripts, etc.

LABELM,LABELS [CHARACTER*(*)] are the main and sub-titles.

PHI [REAL] is the angle of rotation about the vertical axis (in degrees). For orthographic projection, phi is the rotation of the projection plane. For oblique projection, phi is the rotation of the projection angle.

THETA [REAL] is the angle of rotation about the horizontal axis (in degrees). For orthographic projection, phi is the rotation of the projection plane. For oblique projection, phi is the rotation of the projection angle.

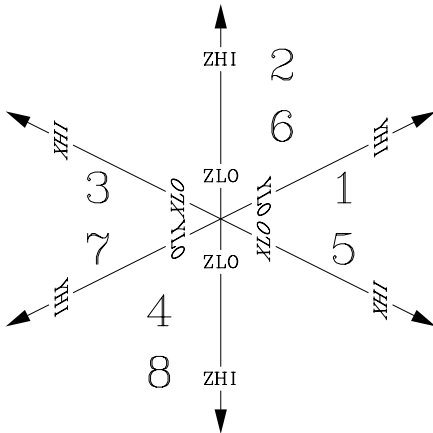
IMODE [INTEGER] is used to specify which surfaces are to be plotted, the type of projection, and whether or not skirts are to be plotted:

IMODE:	projection:	skirt:	top surface:	bottom surface:
1	orthographic		X	
2	orthographic			X
3	orthographic		X	X
4	orthographic	X	X	
5	orthographic	X		X
6	orthographic	X	X	X

11	oblique x-axis at front	X		
12	oblique x-axis at front			X
13	oblique x-axis at front	X		X
14	oblique x-axis at front	X	X	
15	oblique x-axis at front	X		X
16	oblique x-axis at front	X	X	X
21	oblique y-axis at front	X		
22	oblique y-axis at front			X
23	oblique y-axis at front	X		X
24	oblique y-axis at front	X	X	
25	oblique y-axis at front	X		X
26	oblique y-axis at front	X	X	X

LOPAQUE [LOGICAL] If LOPAQUE is .TRUE., then hidden lines are not plotted. Otherwise, the surface is 'translucent'.

IORIENT is the octant to "flip" the axes into: axes values go from low to high values in the same sense as the absolute values of the respective octant. The IORIENT values, [1-8], are shown in the following figure along with the orientation of the respective window limits.



2.2.11 SPACELN

This subroutine plots a 2-dimensional representation of a 3-dimensional space curve specified by three 1-dimensional arrays of consecutive values of the curve coordinates. It is called by:

```
CALL SPACELN(X,Y,Z,N,
+           XLO,XHI,YLO,YHI,ZLO,ZHI,
+           LABELX,LABELY,LABELZ,LABELM,LABELS,
+           SYMBOL,ISKIP,ANGLE,IORIENT)
```

where:

X(*),Y(*),Z(*) [REAL(*)] are 1-dimensional arrays of values of the x-, y-, and z-coordinates of consecutive points along the spaceline.

N [INTEGER] is the number of valid data points in X(*), Y(*), and Z(*).

XLO,XHI [REAL] are the lower and upper limits of the x-axis plotting window. These numbers are re-scaled to more convenient numbers for labeling the axes, if necessary. No clipping is performed, so the spaceline may exceed these plotting window limits.

YLO,YHI [REAL] are the lower and upper limits of the y-axis plotting window.

ZLO,ZHI [REAL] are the lower and upper limits of the z-axis plotting window.

LABELX,LABELY,LABELZ [CHARACTER*(*)] are the axes labels. These can be variable length strings or character variables. The control characters which are listed in the subroutine PLTCHR are enabled, so that the axes labels may contain greek characters, sub-/super-scripts, etc.

LABELM,LABELS [CHARACTER*(*)] are the main and sub-titles.

SYMBOL [CHARACTER*1] is the plotting symbol to be used. This may be a one-character string or character variable. There are three special plotting symbols:

L plot line-segments between successive points.

S plot squares at each point.

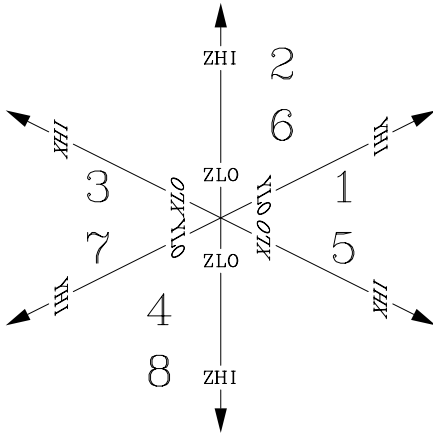
T plot triangles at each point.

Any other symbol is plotted as is.

ISKIP [INTEGER] Vertical lines are dropped between the spaceline and the x-y plane after every ISKIP points have been plotted. If ISKIP=0 then no vertical lines are dropped. The vertical lines are plotted with line-type 4.

ANGLE [REAL] is the angle of rotation about the x=y line (in degrees). The 3-dimensional view generated by SURFACE keeps the line (x=y;z=0) aligned horizontally on the screen; the z-axis is tilted ANGLE degrees toward the front of the screen.

IORIENT is the octant to "flip" the axes into: axes values go from low to high values in the same sense as the absolute values of the respective octant. The IORIENT values, [1-8], are shown in the following figure along with the orientation of the respective window limits.



2.2.12 PLTCHR

This subroutine labels character strings with several different attributes and fonts. The labeling is variable width per character. The subroutine is called by:

```
CALL PLTCHR(X,Y,H,ROTATE,SLANT,LORG,STRING,NCHAR)
```

where:

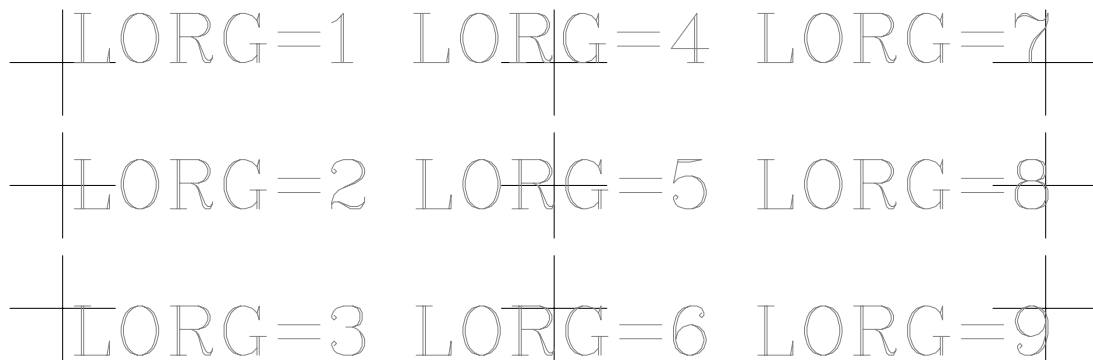
X,Y [REAL] are x and y coordinates of the location at which the string is to be labeled, in plot units : $0 \leq X, Y \leq 10$.

H [REAL] is the height of the string in plot units.

ROTATE [REAL] is the angle in degrees to which the string is rotated counter-clockwise about the point X,Y.

SLANT [REAL] is the angle in degrees to which the string is slanted forward from the vertical orientation.

LORG [INTEGER] specifies the orientation of the string with respect to the point X,Y. The strings in the following figure were labeled at the locations X,Y indicated by '+' and with the indicated LORG values.



(STRING) [CHARACTER*(*)] is the string to be labeled. It may be a variable length string or a character variable.

NCHAR [INTEGER] is the number of characters in the string. **NCHAR** is also used to enable or disable the **FIDDLER** "control" characters.² These control characters are inserted directly into **STRING** to control the characteristics of the *following* characters in the string. If **NCHAR** is negative, then the control characters are enabled, if **NCHAR** is positive then control characters are disabled. The **FIDDLER** control characters are:

- ^ Increment the sub/superscript index. The sub/superscript index is zero upon entry to **PLTCHR**. If the sub/superscript index is positive, then subsequent characters are shifted above the prevailing character direction line; if the index is negative, characters are shifted below the current direction line.
- Decrement the sub/superscript index.
- & This toggles the "font typeface" from n (normal) or s (script) to g (greek or gothic), or back again.
- \$ This toggles the "font typeface" from n (normal) to s (script), or back again.
- % This toggles the overprint mode. Subsequent characters in string will be labeled over each other until another % appears.
- # Decrease the size of subsequent characters by one inter-letter distance.
- @ Increase the size of subsequent characters by one inter-letter distance.
- " Backspace one inter-letter distance.
- ! Extended control sequence. If following character is:
 - 0 Change to font-family 0 (SIMPLEX)
 - 1 Change to font-family 1 (COMPLEX)
 - 2 Change to font-family 2 (DUPLEX)
 - 3 Change to font-family 3 (TRIPLEX)
 - = Toggle constant-letter-spacing (typewriter) mode.
 - Toggle connected-letter (no inter-letter spacing) mode.
 - | Stretch vertical scale by inter-letter distance.
 - ~ Shrink vertical scale by inter-letter distance.
 - . Turn off control character recognition for rest of line.

If ! is followed by any other character, that character is printed (including **FIDDLER** control characters).

The parameter **NCHAR** is also used to enable the single character mode. This mode is in effect when **STRING** has length 1 and **NCHAR** is greater than 1. The character which appears in the table of characters at position **NCHAR** is plotted.

²The control characters are all printable ASCII characters, not ASCII control characters.

2.2.13 PLTVAL

This subroutine plots number values using a specified format. The subroutine call is:

```
CALL PLTVAL(X,Y,H,ROTATE,SLANT,LORG,FORMAT,VALUE)
```

where:

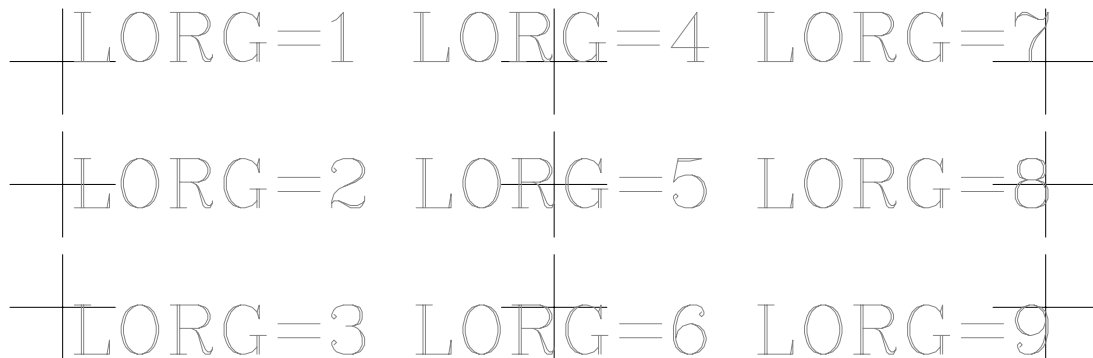
X,Y [REAL] are x and y coordinates of the location at which the string is to be labeled, in plot units : $0 \leq X, Y \leq 10$.

H [REAL] is the height of the string in plot units.

ROTATE [REAL] is the angle in degrees to which the string is rotated counter-clockwise about the point X,Y.

SLANT [REAL] is the angle in degrees to which the string is slanted forward from the vertical orientation.

LORG [INTEGER] specifies the orientation of the string with respect to the point X,Y. The strings in the following figure were labeled at the locations X,Y indicated by '+' and with the indicated LORG values.



FORMAT [CHARACTER*(*)] is a string containing the format which is to be used to label the value. This can be either a fortran-like format such as 'F7.0' or '1PE15.3', or a '*' format which plots numbers like 20000 as 2×10^4 instead of 2.E4 or 20,000. Numbers with powers of ten less than 4 and greater than -4 are plotted with no $I \times 10^N$ format but directly.

VALUE [REAL] is the value to be formatted and labeled.

2.2.14 PLOTN

This subroutine performs all of the primitive graphics functions for the FIDDLER plotting-package. This includes moving, drawing, clearing the screen, and changing pen and line-types. The subroutine call is:

```
CALL PLOTN(X,Y,ICMD,ISUB)
```

where X,Y are REAL values and ICMD,ISUB INTEGER values. The PLOTN commands are:

ICMD: ISUB: Operation performed:

```
-----
0      INITIATE      graphics device
1      CLEAR        screen (graphics or alpha)
2      DRAW         to X,Y
3      MOVE         to X,Y
4      FILL         area with color pattern
      1  BEGIN       area boundary
      2  CONTINUE    area boundary
      3  END         area boundary and fill with current are pattern
5      CLIP         window coordinates
      1  LOWER LEFT  clipping window coordinate (plot units) = X,Y
      2  UPPER RIGHT clipping window coordinate (plot units) = X,Y
      3  RESET       to maximum window (CLIP OFF)
6      SCREEN       specification
      1  RATIO=1:1   image is not warped
      2  RATIO=MAX   image fills screen
      3  LOWER LEFT  screen coordinate (inches) = X,Y
      4  UPPER RIGHT screen coordinate (inches) = X,Y
      5  LOWER LEFT  screen coordinate (pixel indices = X,Y
      6  UPPER RIGHT screen coordinate (pixel indices) = X,Y
7      PEN          color
      TEK4010 SEI1104 TEK4205 CALCOMP PostScript
      1  white  white  white  black  black
      2  white  red   red   red    gray
      3  white  green green green  gray
      4  white  blue  blue  blue   gray
      5  white          cyan          gray
      6  white          magenta        gray
      7  white          yellow         gray
      8  white          orange         gray
      9  white          green          gray
     10  white          green          gray
     11  white          blue           gray
     12  white          blue           gray
     13  white          red            gray
     14  white          gray           gray
     15  white          gray           gray
8      AREA PATTERN (TEK4205 only)
     -15..0  SOLID
      1..16  TEXTURED
     50..174 DITHERED
9      LINE-TYPE
      1  -----
      2  ---  ---
```

```

3      -- -- --
4      - - - - -
5      - - -
6      - - -
10     ALPHA/GRAPHICS
1      ALPHA    ON  (GRAPHICS OFF)
2      GRAPHICS ON (ALPHA    OFF)
11     SPECIFY PLOTTING SIZE IN "PLOTTING UNITS" >0 AND <10
1      LOWER LEFT graph window coordinate (plot units) = X,Y
2      UPPER RIGHT graph window coordinate (plot units) = X,Y
12     PEN      table index (used by FRAME2D, FRAME3D)
-1     TEXT PEN  pen color used for text, axes labels = X
0      GRID PEN  pen color used for grid and axes     = X
1..6   DATA PEN pen color used for data pen ISUB     = X
13     LINE-TYPE table index (used by FRAME2D, FRAME3D)
-1     TEXT LINE line-type used for text, axes labels = X
0      GRID LINE line-type used for grid and axes     = X
1..6   DATA LINE line-type used for data line ISUB   = X
14     <REMOVED FROM SERVICE>
15     TERMINAL definition (if X=0: load fonts, initialize)
        device name: alpha: graphics:
1      4010      VT102 Tektronix  4010 (40xx)
2      SEIK      VT102 Seiko      1104 (11xx)
3      4205      VT102 Tektronix  4205 (41xx,42xx)
4      CALC      VT102 CALC       file
5      POST      VT102 PostScript file
6      HPGL      VT102 HPGL       file
7      V240      VT240 Tektronix  4010 (40xx)
8      LN03      VT102 LN03       file
16     ENTER/EXIT GRAPHICS
1      ENTER     GRAPHICS
0      EXIT      GRAPHICS
17     PARAMETERS (default values in curly brackets)
1      NM =X {3}  number of mantissa digits in axes numbers
2      NE =X {3}  number of exponent digits in axes numbers
3      NP =X {3}  maximum power of 10 for axes numbers ('*'format)
4      NS =X {1}  ... NOT USED ...
5      DT2=X {.03} 2D relative tic          length
6      DL2=X {.03} 2D relative axes label   height
7      DH2=X {.02} 2D relative axes number  height
8      DM2=X {.20} 2D relative axes/label   distance
9      DC2=X {.05} 2D relative main title   height
10     DS2=X {.04} 2D relative sub title    height
11     DD2=X {.02} 2D relative date         height
12     DT3=X {.03} 3D relative tic          length
13     DL3=X {.03} 3D relative axes label   height
14     DH3=X {.02} 3D relative axes number  height
15     DM3=X {.40} 3D relative axes/label   distance
16     DC3=X {.05} 3D relative main title   height
17     DS3=X {.04} 3D relative sub title    height
18     DD3=X {.02} 3D relative date         height
19     FMT2 {1PG7.0}2D axes numbers: if X=0. '*' else '1PG7.0'

```

```

20   FMT3 {1PG7.0} 3D axes numbers: if X=0. '*' else '1PG7.0'
21   DUX=X {.005} XYPLOT  box symbol width
22   DVX=X {.005} XYPLOT  box symbol height
23   DSX=X {.010} XYPLOT  char symbol height
24   DUS=X {.005} SPACELN box symbol width
25   DVS=X {.005} SPACELN box symbol height
26   DSS=X {.010} SPACELN char symbol height
27   DRV=X {.800} VECTORS  relative line-segment length
28   DAV=X {.150} VECTORS  relative arrow          length
29   90DEG {T}      2D x-axis numbered at 90 degrees
30   0 DEG {F}      2D x-axis numbered at 0 degrees
31   BLANK {F}      2D main tics:   if X=0. blank else unblank
32   BLANK {F}      2D axes labels: if X=0. blank else unblank
33   BLANK {F}      2D axes numbers: if X=0. blank else unblank
34   BLANK {F}      2D main title:  if X=0. blank else unblank
35   BLANK {F}      2D sub  title:  if X=0. blank else unblank
36   BLANK {F}      2D date:        if X=0. blank else unblank
37   BLANK {F}      3D main tics:   if X=0. blank else unblank
38   BLANK {F}      3D axes labels: if X=0. blank else unblank
39   BLANK {F}      3D axes numbers: if X=0. blank else unblank
40   BLANK {F}      3D main title:  if X=0. blank else unblank
41   BLANK {F}      3D sub  title:  if X=0. blank else unblank
42   BLANK {F}      3D date:        if X=0. blank else unblank
43   BLANK {F}      2D frame:       if X=0. blank else unblank
44   BLANK {F}      3D frame:       if X=0. blank else unblank
45   BLANK {T}      2D minor tics:  if X=0. blank else unblank
46   NQ =X {8}      number of lines drawn in 2D border
47   XAXIS LABELING {0.}          if X=1. xlo is at right
48   YAXIS LABELING {0.}          if X=1. ylo is at top
49   PLOT OPENING CONTROL {0.}    if X=1. disable else enable
50   PLOT CLOSING CONTROL {0.}    if X=1. disable else enable
51   TIC ORIENTATION {0.}         if X=1. outward else inward
52   ZMIN=X,ZMAX=Y Surface clipping limits
18   ABS TEXT MOVE Absolute text move to ( COL=X, ROW=Y)
19   REL TEXT MOVE Relative text move by (#COL=X,#ROW=Y)
20   SAVE/RESTORE text cursor position
      1   SAVE          text cursor position
      0   RESTORE      text cursor position
21   FLUSH          Buffered output (CALL BUFPOUT)

```

The following figure shows the line-types and "colors" produced by FIDDLER when the graphics device is a PostScript file.

Index:	Line-type:	Pen-color:
1	—————	—————
2	- - - - -	—————
3	—————
4	—————
5	—————
6	- - - - -	—————

2.3 Plotting package programming examples

In this section, several example FORTRAN programs and resulting FIDDLER plots are shown. The examples illustrate many of the options available in the subroutine calls, *but not all of them*. Refer to the specifications of the subroutine parameters, (especially PLOTN), in order to fully explore the available plotting options.

In all of these examples, the plotting device is chosen as PostScript: device number 5 in the PLOTN(0.,0.,15,*) call within each example program. This choice results in a PostScript-format file which may be printed on a PostScript printer, or directly incorporated into a document (via SCRIBETM). All of the graphics in this manual were created in this way. Alternatively, the following two program lines may be substituted for the PLOTN(0.,0.,15,5) command in any example program:

```
GINIT()
UPSET(0)
```

This allows the plots to be previewed on a terminal, and then plotted out by choosing a hard-copy device, creating a plot file, and printing the plot file. The first command initializes the terminal to a default device (VT-100) so that the prompt for a terminal type may appear on the screen. The second command prompts for the terminal/output-device type, loads in the fonts, and sets a few default parameters.

The various graphics devices are: ³

Device #:	Device:	Alpha:	Graphics:
1	4010	VT-100	Tektronix 4010 (40xx)
2	SEIK	VT-100	Seiko 1104
3	4205	VT-100	Tektronix 4205 (41xx,42xx)
4	CALC	VT-100	CALC file
5	POST	VT-100	PostScript file
6	HPGL	VT-100	HPGL file
7	V240	VT-240	Tektronix 4010 (40xx)
8	LN03	VT-100	LN03 file

³The device number is used in the PLOTN(0.,0.,15,*) call to define the terminal type.

2.3.1 XYPLOT Example: Polar-Parametric Curve

In the following example, 1000 samples of a polar-parametric curve are collected and displayed using XYPLOT. It is necessary to choose the graphics device before calling XYPLOT, using the PLOTN call. The main and sub titles are character variables, containing FIDDLER control characters ("!", for example). The axes labels, however, are sent directly to XYPLOT as parameters in the subroutine call. The control character ! is used to change the font family (See PLTCHR specifications). The SYMBOL parameter in the XYPLOT call is "L", for line segments: successive points are joined by line segments in the plot.

```

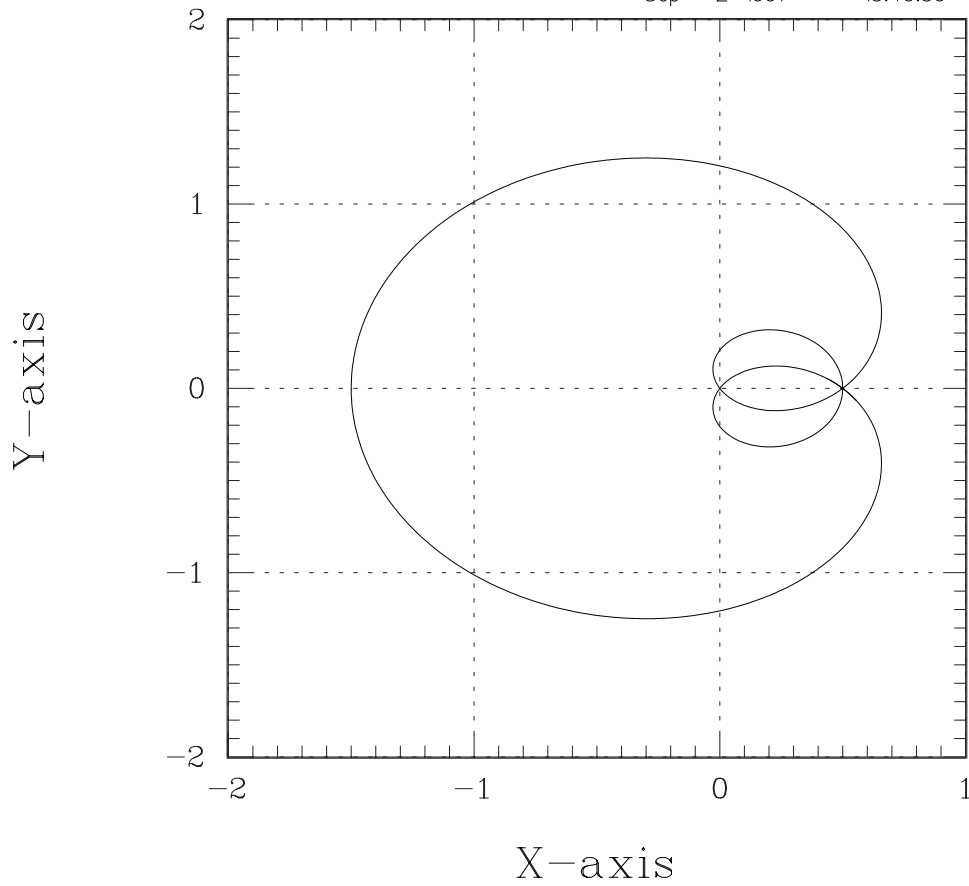
C#####
PROGRAM XXYP1
C *** X-Y PLOT: POLAR PARAMETRIC
C#####
  DIMENSION X(1000),Y(1000)
  CHARACTER MTIT*20,STIT*30,XTIT*10,YTIT*10
  DATA NPTS,TMIN,TMAX,XMIN,XMAX,YMIN,YMAX/1000,-3.2,3.2,-2,1,-2,2/
  DATA MTIT/'!3Nephroid of Freeth'/
  DATA STIT/'!r = sin(t)+1/2  &}& = 2t  '/
  DATA XTIT/'!X-axis  '/
  DATA YTIT/'!Y-axis  '/
C-----
C *** GATHER DATA
  DO 1 I=1,NPTS
    T=TMIN+(TMAX-TMIN)*REAL(I-1)/REAL(NPTS-1)
    R=SIN(T)+.5
    A=2.*T
    X(I)=R*COS(A)
    Y(I)=R*SIN(A)
  1 CONTINUE
C *** PLOT
  CALL PLOTN(0.,0.,15,5)      ! DEFINE GRAPHICS DEVICE
  CALL XYPLOT(X,Y,NPTS,XMIN,XMAX,YMIN,YMAX,'LIN','LIN',
+ XTIT,YTIT,MTIT,STIT,'L',.TRUE.,.TRUE.,.TRUE.,1,1,1)
  END

```

Nephroid of Freeth

$$r = \sin(t) + 1/2 \quad \theta = 2t$$

Sep- 2-1997 15:49:39



2.3.2 XYPLOT Example: Extra Label

The following FORTRAN program creates a Lissajous pattern with line-type 4, and places an extra label on the plot. 1000 samples of the curve are first collected in X and Y. The first PLOTN call under the "PLOT" comment defines the graphics device.⁴ The next PLOTN call sets the thickness of the border of the plot to 3 lines thick (first parameter). The XYPLOT call has the parameter LFIRST .true. and LLAST .false. (the parameters following 'L'). With LFIRST .true., a new plot is created (the graphics screen is cleared or the plot file is opened). With LLAST .false., additional plots or labels may be overlaid on the first plot (the plot file is not closed or the graphics screen is not cleared). The PLOTN and PLTCHR calls following the XYPLOT call place an additional label on the plot. The PLOTN call sets the line-type index back to 1 (solid), since it was changed in the XYPLOT program to 4 (the last parameter in the XYPLOT call). The PLTCHR call places the label at location .5,.5 on the plotting window. These coordinates are in "plot" units, which are between 0. and 10., for the minimum and maximum screen distance. The PLTCHR call contains FIDDLER control characters, (! and &), to change font families and to change to greek font, respectively). These are enabled by the negative NCHAR (last parameter). If NCHAR were positive, the control characters would have been labeled instead of interpreted. The plot is finally closed with the last PLOTN call.

```

C#####
PROGRAM XXYP2
C *** X-Y PLOT: LISSAJOUS PATTERN
C#####
PARAMETER(PI=3.1415926535,NPTS=1000)
DIMENSION X(NPTS),Y(NPTS)
DATA TMIN,TMAX,XMIN,XMAX,YMIN,YMAX/ -PI,PI,-2.,2.,-2.,2./
C
-----
C *** GATHER DATA
DO 1 I=1,NPTS
T=TMIN+(TMAX-TMIN)*REAL(I-1)/REAL(NPTS-1)
X(I)=SIN(T*5.)
Y(I)=SIN(T*7.+PI/4.)
1 CONTINUE
C *** PLOT
CALL PLOTN (0.,0.,15, 5) ! DEFINE GRAPHICS DEVICE
CALL PLOTN (3.,0.,17,46) ! 2-D BORDER: 3 LINES THICK
CALL XYPLOT(X,Y,NPTS,XMIN,XMAX,YMIN,YMAX,'LIN','LIN',
+ '!1X-axis','!1Y-axis',
+ '!3Lissajous Pattern','!1x = sin(5t); y = sin(7t + &p&/4)',
+ 'L',.TRUE.,.FALSE.,.FALSE.,3,1,4)
C *** EXTRA LABEL
CALL PLOTN (0.,0.,9,1) ! GET LINE-TYPE 1 BACK
CALL PLTCHR(.5,.5,.2,0.,0.,1,'!3&This is an extra label',-25)
C *** CLOSE PLOT
CALL PLOTN (0.,0.,16,0)
END

```

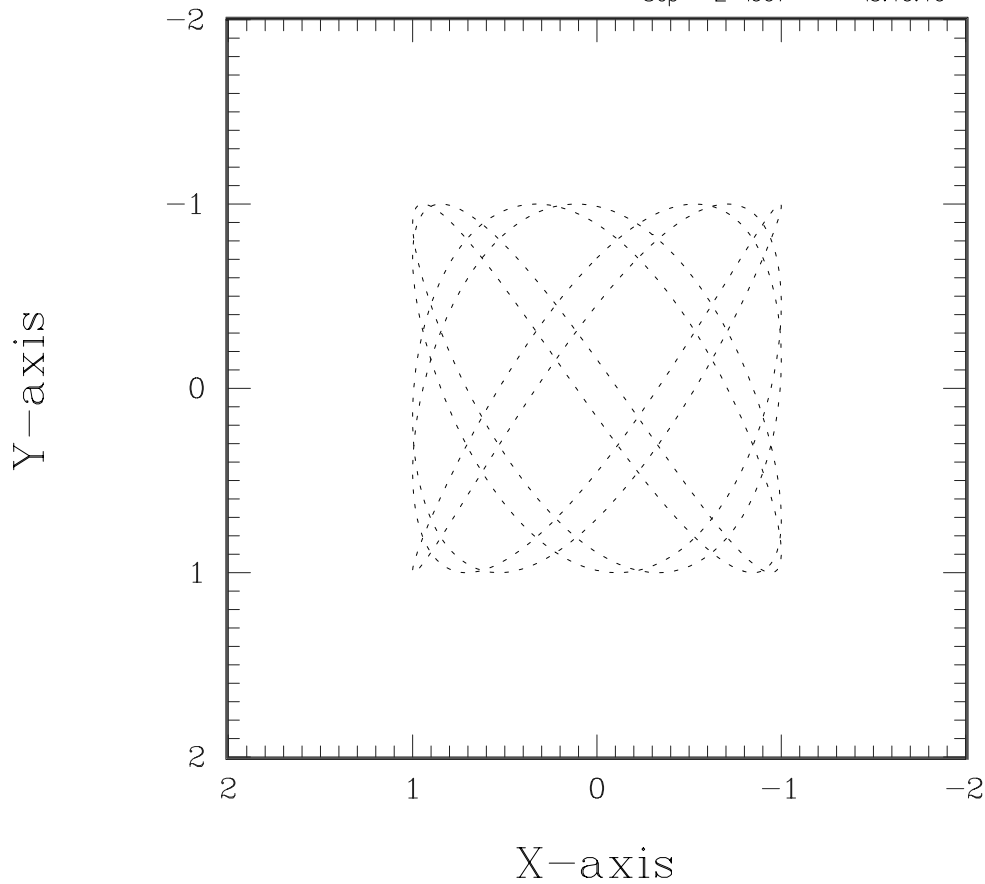
⁴Always necessary at the beginning of one or a series of FIDDLER plots!

Lissajous Pattern

$$x = \sin(5t); \quad y = \sin(7t + \pi/4)$$

Sep- 2-1997

15:49:40



This is an extra label

2.3.3 XYPLOT Example: Line-types and LOG Axes

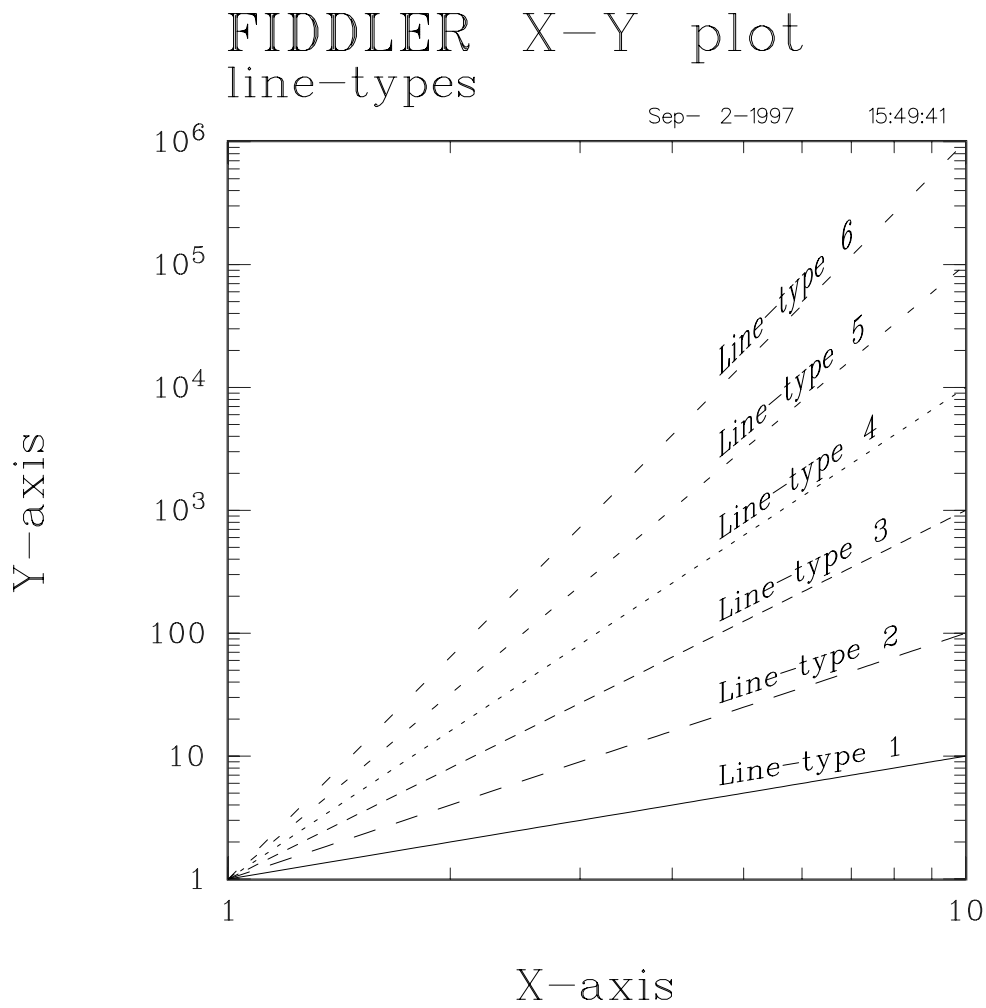
The following example program displays the six FIDDLER line-types, and labels them using PLTCHR. The first PLOTN call defines the graphics device. ⁵ For each line style, a line is drawn on a LOG-LOG plot. The series of XYPLOT calls (from within the loop) is made with the parameter LFIRST .true. only when the first call is made (LFIRST=(J.EQ.1)), and LLAST always .false. (so that the labels may also be overlaid on the plot). Following each XYPLOT call, a PLTCHR call labels the respective line. The label location and angle are determined by using the fact that the XYPLOT frame is drawn from ("plot" units) (2,2) to 8,8). Finally, the plot is closed using the last PLOTN call, outside of the loop.

```

C#####
PROGRAM XXYP3
C *** X-Y PLOT: LINE-TYPES
C#####
PARAMETER(NPTS=100,RAD2DEG=180./3.14159)
DIMENSION X(NPTS),Y(NPTS)
CHARACTER*1 CJ
DATA XMIN,XMAX,YMIN,YMAX/1.,9.99,1.,1.E6/
C-----
C *** DEFINE GRAPHICS DEVICE
CALL PLOTN(0.,0.,15,5)
C *** FOR EACH LINE-TYPE
DO 2 J=1,6
C *** GATHER DATA
RJ=REAL(J)
CJ=CHAR(ICCHAR('0')+J)
DO 1 I=1,NPTS
X(I)=XMIN+(XMAX-XMIN)*REAL(I-1)/REAL(NPTS-1)
Y(I)=X(I)**RJ
1 CONTINUE
C *** PLOT
CALL XYPLOT(X,Y,NPTS,XMIN,XMAX,YMIN,YMAX,'LOG','LOG',
+ '!1X-axis','!1Y-axis','!3FIDDLER !1X-Y plot','!1line-types',
+ 'L',(J.EQ.1),.FALSE.,.FALSE.,1,1,J)
CALL PLOTN(0.,0.,9,1) ! BACK TO LINE-TYPE 1
C *** LABEL
ANGLE=RAD2DEG*ATAN(RJ/6.)
CALL PLTCHR(6.,2.1+RJ*2./3.,.15,ANGLE,ANGLE,1,
+ '!1Line-type '//CJ,-13)
2 CONTINUE
C *** CLOSE PLOT
CALL PLOTN(0.,0.,16,0)
END

```

⁵Always necessary at the beginning of one or a series of FIDDLER plots!



2.3.4 XYPLOT Example: Plotting Symbols and Characters

XYPLOT has three reserved values for the parameter SYMBOL. These are: L, S, and T, for line-segments, squares, and triangles, respectively. If SYMBOL has any other value, the plotting symbol is the respective character labeled in Simplex normal font. The series of PLOTN calls in the example define the graphics device ⁶, blank out the date labeling, and disable minor axes tics. These house-cleaning options are usually performed at the beginning of any application program, to tailor the plot to the preferences of the user. Ten different plotting symbols are used to plot data curves on the plot. Note that the size of triangles (SYMBOL='T') is modified from the default value by the use of PLOTN calls (IF(J.EQ.4) ...). The size of character symbols (SYMBOL = any character except L, S, or T) is changed by the PLOTN call (IF(J.EQ.10) ...). The XYPLOT call has the LFIRST parameter .true. only on the first plot, so that several data curves may be overlaid. The LLAST parameter is .true. only on the last call.

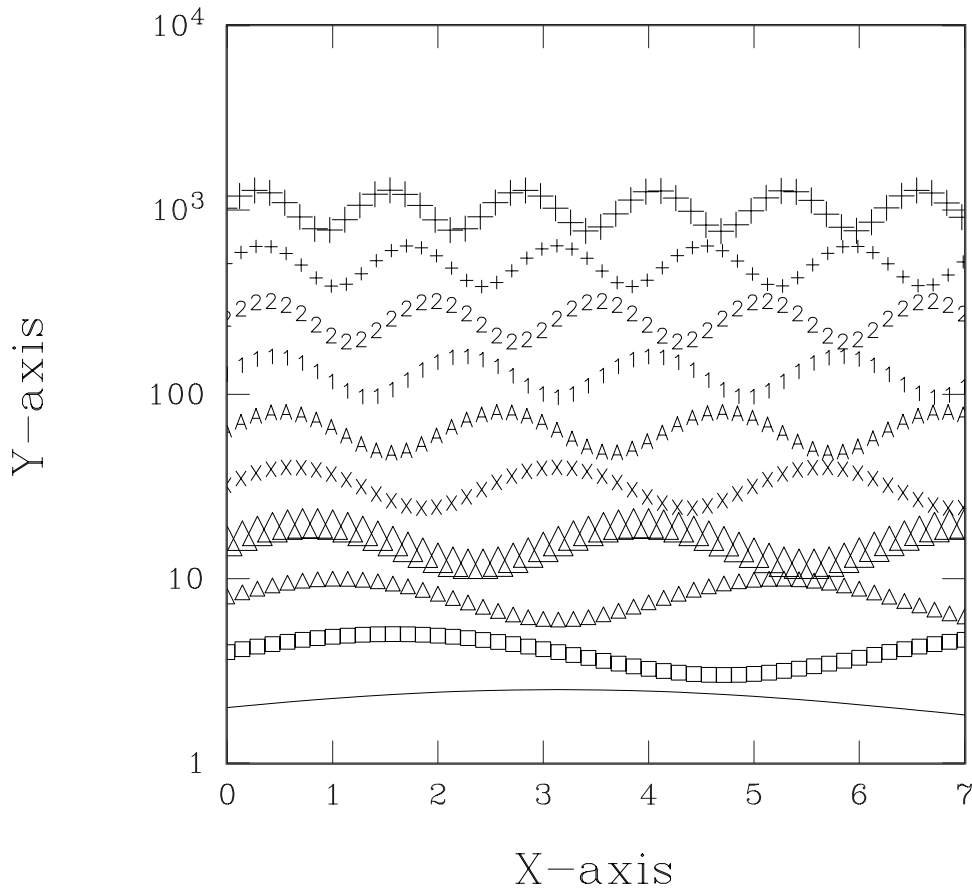
```

C#####
PROGRAM XXYP4
C *** X-Y PLOT: CHARACTERS
C#####
      DIMENSION X(100),Y(100)
      CHARACTER SYMBOL*10
      DATA NPTS,XMIN,XMAX,YMIN,YMAX/50,0.,7.,1.,9.E3/
      DATA SYMBOL/'LSTTXA12++'/
C-----
      CALL PLOTN(0.,0.,15,5)           ! DEFINE GRAPHICS DEVICE
      CALL PLOTN(0.,0.,17,36)          ! BLANK DATE
      CALL PLOTN(0.,0.,17,45)          ! NO MINOR TICS
C *** FOR TEN DIFFERENT CHARACTERS
      DO 2 J=1,10
         RJ=REAL(J)
C *** GATHER DATA
      DO 1 I=1,NPTS
         X(I)=XMIN+(XMAX-XMIN)*REAL(I-1)/REAL(NPTS-1)
         Y(I)=2.**RJ*(1.+SIN(RJ/2.*X(I))/4.)
      1 CONTINUE
C *** MODIFY CHARACTER SIZES
      IF(J.EQ. 4)CALL PLOTN(.02,0.,17,21)
      IF(J.EQ. 4)CALL PLOTN(.02,0.,17,22)
      IF(J.EQ.10)CALL PLOTN(.04,0.,17,23)
C *** PLOT
      CALL XYPLOT(X,Y,NPTS,XMIN,XMAX,YMIN,YMAX,'LIN','LOG',
+ ' !1X-axis', '!1Y-axis', '!3FIDDLER !1X-Y plot',
+ ' !1characters and symbols',
+ SYMBOL(J:J),(J.EQ.1),(J.EQ.10),.FALSE.,1,1,1)
      2 CONTINUE
      END

```

⁶Always necessary at the beginning of one or a series of FIDDLER plots!

FIDDLER X-Y plot characters and symbols

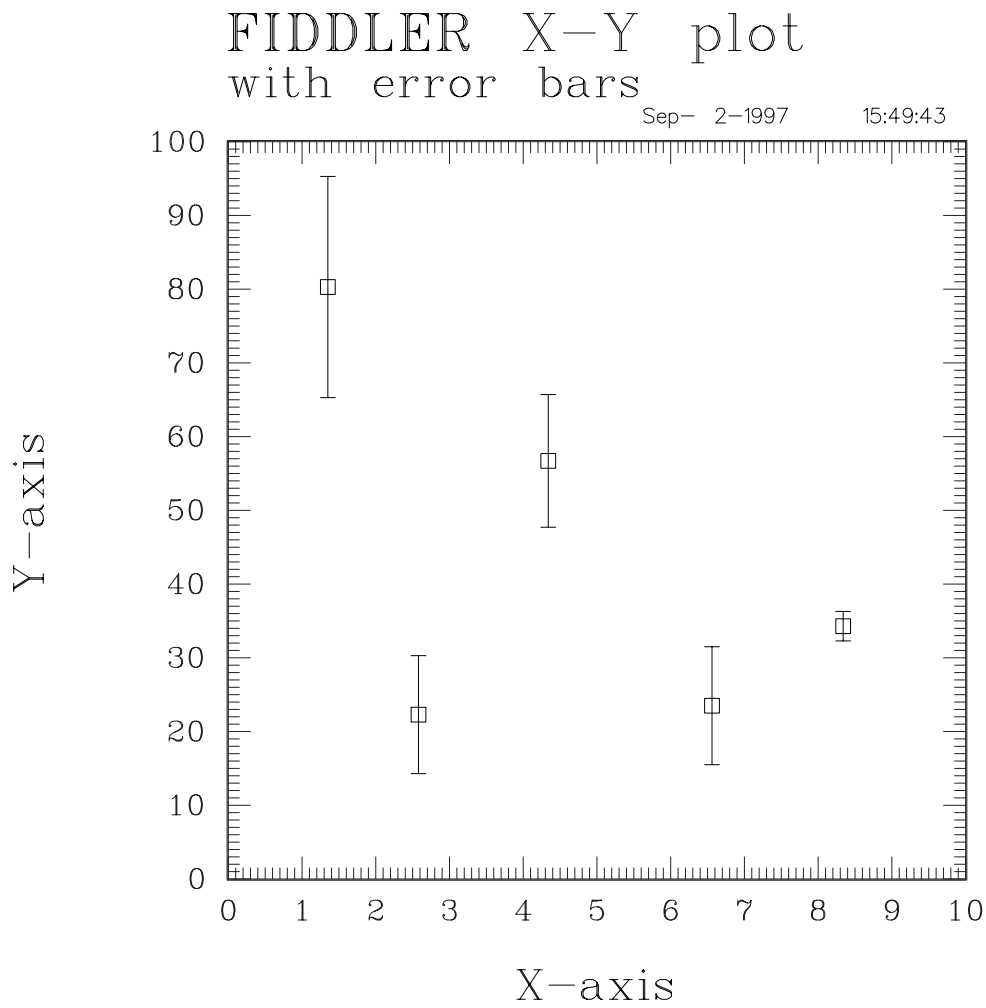


2.3.5 XYPLOTE Example: Error Bars

```

C#####
PROGRAM XXYP5
C *** X-Y PLOT: ERROR-BARS USING XYPLOTE
C#####
PARAMETER(NPTS=5)
DIMENSION X(NPTS),Y(NPTS),E(NPTS)
DATA XMIN,XMAX,YMIN,YMAX/.1,10.,1.,100./
DATA (X(I),I=1,NPTS)/1.35,2.58,4.34,6.56,8.34/
DATA (Y(I),I=1,NPTS)/80.3,22.3,56.7,23.5,34.3/
DATA (E(I),I=1,NPTS)/15.0,8.00,9.00,8.00,2.00/
C-----
CALL PLOTN(0.,0.,15,5)          ! DEFINE GRAPHICS DEVICE
CALL XYPLOTE(X,Y,E,NPTS,XMIN,XMAX,YMIN,YMAX,'LIN','LIN',
+ '!1X-axis','!1Y-axis','!3FIDDLER !1X-Y plot',
+ '!1with error bars','S',.TRUE.,.TRUE.,.FALSE.,1,1,1)
END

```



2.3.6 XYPLOT Example: Error Bars without using XYPLOTE

The following example illustrates a technique of accessing the positions of the data points on the XYPLOT graph. In order to accomplish this, the file 'cgrap.inc' (found in the CODE directory) must be included in the program. The common block in cgrap.inc includes the necessary transformation parameters. In order to convert data units to plot units, the FIDDLER routine T2D is used, as shown in the program.⁷ The first 2 parameters of T2D are the coordinates of a point in data units. The second 2 parameters are the coordinates in plot units which are used by PLOTN to plot the error bars. Note that two XYPLOT calls are made, one with SYMBOL='L' for line-segments, and one with SYMBOL='S' for squares. The error bars are overlaid on the resulting symbol/line x-y plot.

```

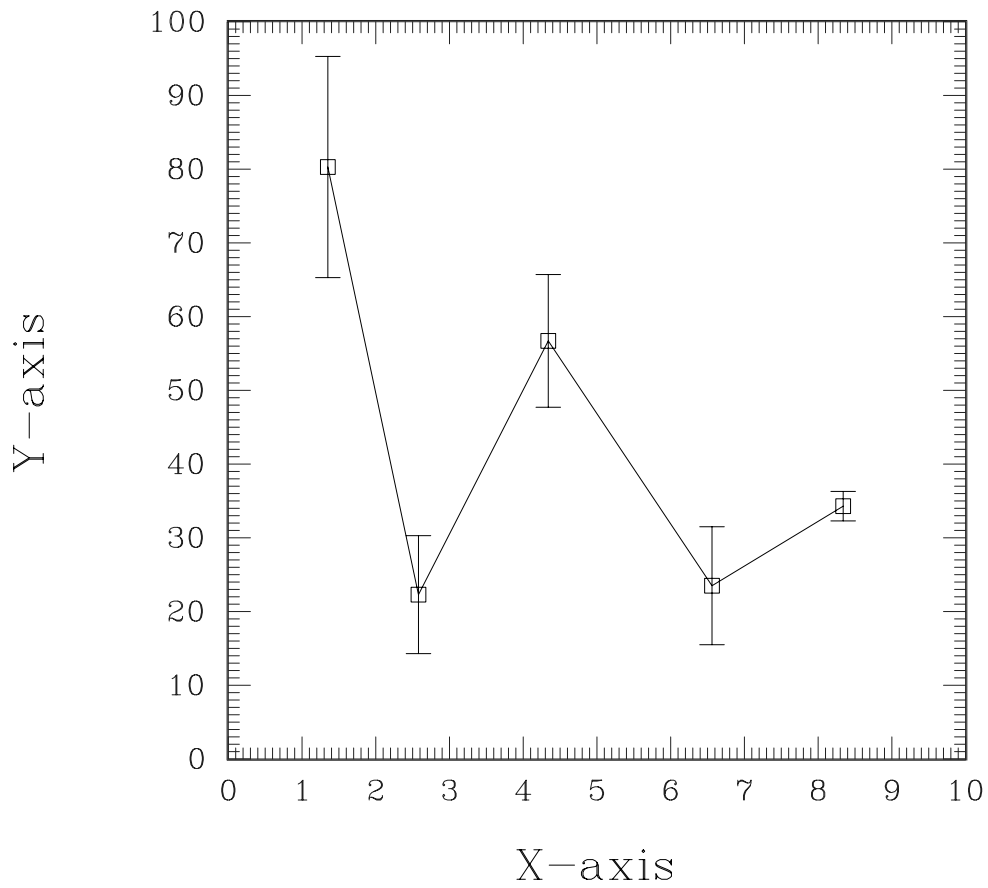
C#####
PROGRAM XXYP6
C *** X-Y PLOT: ERROR-BARS WITHOUT USING XYPLOTE
C#####
      include 'cgrap.inc'
      PARAMETER(NPTS=5)
      DIMENSION X(NPTS),Y(NPTS),E(NPTS)
      DATA XMIN,XMAX,YMIN,YMAX/.1,10.,1.,100./
      DATA (X(I),I=1,NPTS)/1.35,2.58,4.34,6.56,8.34/
      DATA (Y(I),I=1,NPTS)/80.3,22.3,56.7,23.5,34.3/
      DATA (E(I),I=1,NPTS)/15.0,8.00,9.00,8.00,2.00/

C-----
C *** DEFINE GRAPHICS DEVICE
      CALL PLOTN(0.,0.,15,5)
      CALL PLOTN(0.,0.,17,36)          ! BLANK DATE
C *** PLOT LINE SEGMENTS AND SYMBOLS
      CALL XYPLOT(X,Y,NPTS,XMIN,XMAX,YMIN,YMAX,'LIN','LIN',
+ '!1X-axis','!1Y-axis','!3FIDDLER !1X-Y plot',
+ '!1with error bars','L',.TRUE.,.FALSE.,.FALSE.,1,1,1)
      CALL XYPLOT(X,Y,NPTS,XMIN,XMAX,YMIN,YMAX,'LIN','LIN',
+ ' ',' ',' ',' ','S',.FALSE.,.FALSE.,.FALSE.,1,1,1)
C *** PLOT ERROR BARS
      DO 1 I=1,NPTS
          CALL T2D(X(I),Y(I)-E(I),UL,VL)
          CALL T2D(X(I),Y(I)+E(I),UH,VH)
          CALL PLOTN(UL,VL,3,0)          ! VERTICAL
          CALL PLOTN(UH,VH,2,0)
          CALL PLOTN(UH-.1,VH,3,0)      ! TOP HORIZ
          CALL PLOTN(UH+.1,VH,2,0)
          CALL PLOTN(UL-.1,VL,3,0)      ! BOT HORIZ
          CALL PLOTN(UL+.1,VL,2,0)
      1 CONTINUE
C *** CLOSE PLOT
      CALL PLOTN(0.,0.,16,0)
      END

```

⁷There is also a T3D for 3-D plots.

FIDDLER X-Y plot with error bars

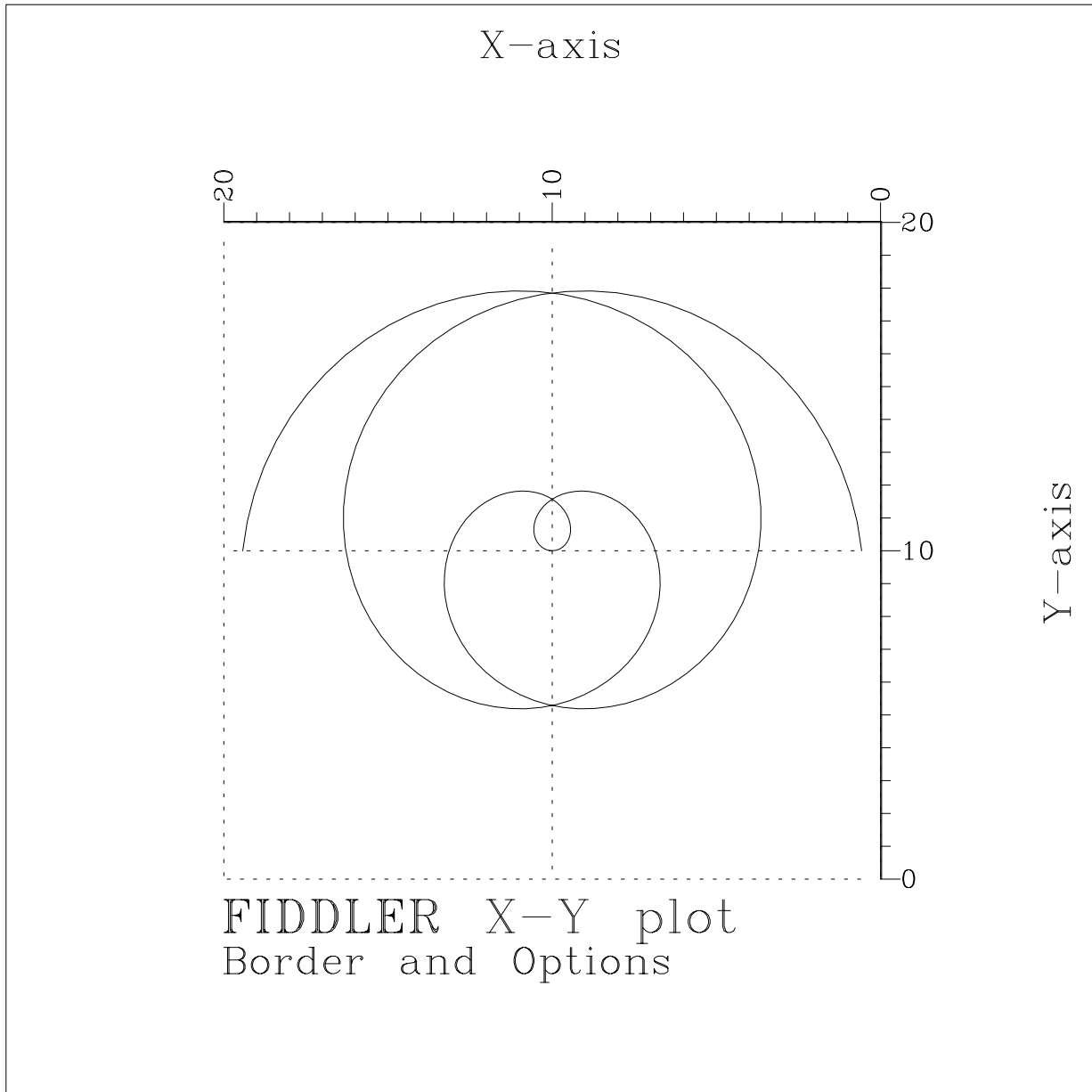


2.3.7 XYPLOT Example: border and options

```

C#####
PROGRAM XXYP7
C *** X-Y PLOT: BORDER AND OPTIONS
C#####
PARAMETER(NPTS=200,PI3=3.1415926535*3)
DIMENSION X(NPTS),Y(NPTS)
DATA TMIN,TMAX,XMIN,XMAX,YMIN,YMAX/PI3,PI3,0.,20.,0.,20./
C
-----
C *** SELECT PLOT DEVICE
CALL GINIT()           ! INITIALIZE PLOT PACKAGE
CALL UPSET(0)          ! PROMPT FOR GRAPHICS DEVICE
CALL PLOTN( 0.,0.,17,36) ! BLANK 2-D DATE
CALL PLOTN(-2.,0.,17,46) ! BORDER: 2 LINES (|_ FORMAT)
CALL PLOTN( 0.,0.,17,29) ! 90 DEGREES X AXIS LABEL
CALL PLOTN( 1.,0.,17,51) ! TIC ORIENTATION OUTWARDS
CALL PLOTN( 1.,0.,17,47) ! XAXIS LABELING AT RIGHT
CALL PLOTN( 1.,0.,17,48) ! YAXIS LABELING AT TOP
IORIENT=2              ! XMIN AT RIGHT, YMIN AT BOTTOM
C *** GATHER DATA
DO 1 I=1,NPTS          ! VALUES
  T=TMIN+(TMAX-TMIN)*REAL(I-1)/REAL(NPTS-1)
  X(I)=T*COS(T)+10.
  Y(I)=T*SIN(T)+10.
1 CONTINUE
C *** PLOT
CALL XYPLOT(X,Y,NPTS,XMIN,XMAX,YMIN,YMAX,'LIN','LIN',
+ '!1X-axis','!1Y-axis','!3FIDDLER !1X-Y plot',
+ '!1Border and Options','L',.TRUE.,.FALSE.,.TRUE.,2,1,1)
C *** PLOT BORDER
CALL PLOTN(0.01,0.01,3,0)
CALL PLOTN(0.01,9.99,2,0)
CALL PLOTN(9.99,9.99,2,0)
CALL PLOTN(9.99,0.01,2,0)
CALL PLOTN(0.01,0.01,2,0)
C *** CLOSE PLOT
CALL PLOTN(0.,0.,16,0)
END

```



2.3.8 CONTOUR Example: Family of Plane Curves

The contours of a sampled surface $Z(X,Y)$ are drawn by CONTOUR in the following example. In this example, the contours are special plane-curves called Cassinian Ovals.⁸ The first portion of the example performs the sampling of the function Z , at various X and Y values. In addition, a set of contour values is calculated and stored in C . The calculated set of contour values is uniformly-spaced in this example. The contour plot is generated by a single CONTOUR call, after first defining the graphics device.⁹

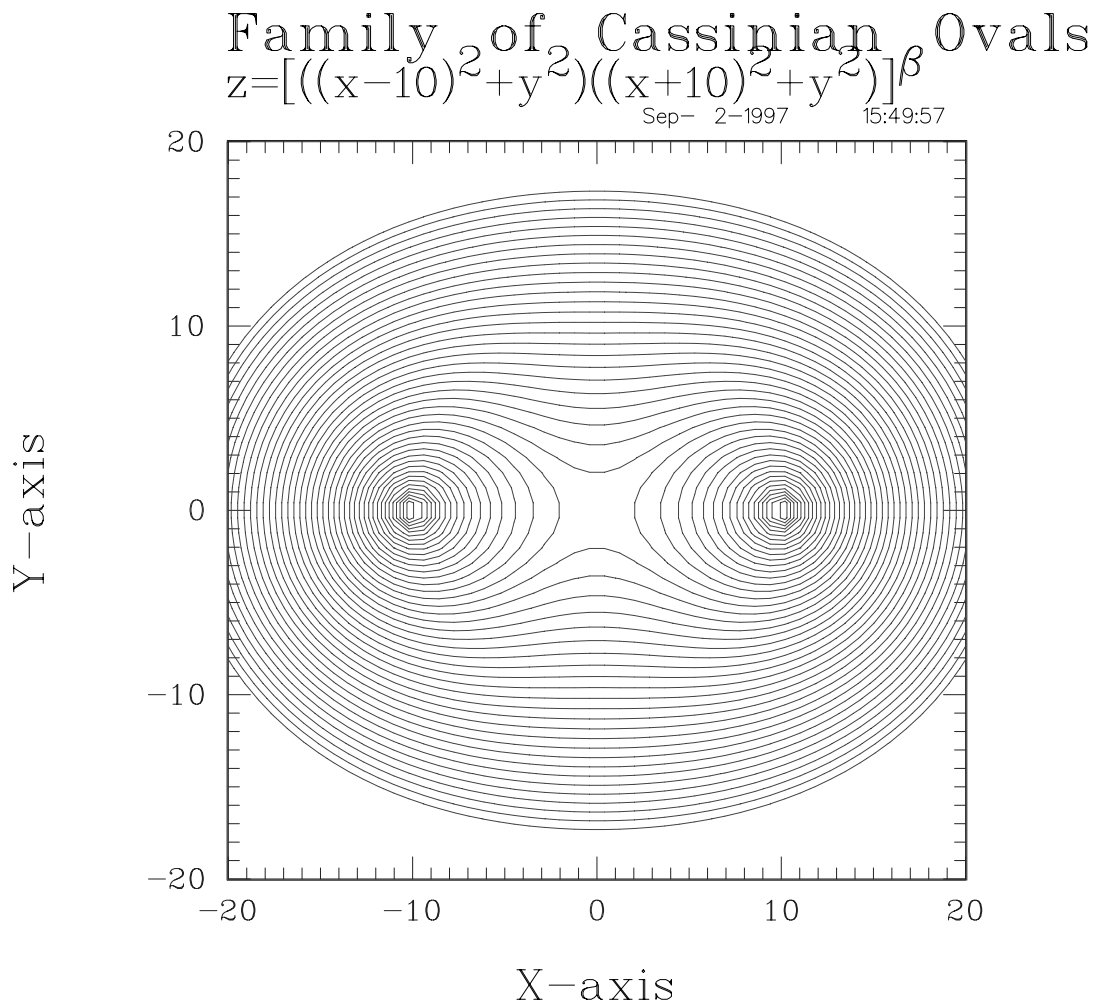
```

C#####
PROGRAM XCON1
C *** GRAPHER EXAMPLE : CONTOUR PLOT
C#####
PARAMETER(NP=100)
DIMENSION X(NP),Y(NP),Z(NP,NP),C(NP)
CHARACTER*80 MTIT,STIT,XTIT,YTIT
DATA NX,NY,NC/50,50,50/
DATA XMIN,XMAX,YMIN,YMAX,CMIN,CMAX/-20.,20.,-20.,20.,0.,20./
DATA MTIT/'!3Family of Cassinian Ovals  '/
DATA STIT/'!1z=[((x-10)^2+y^2)((x+10)^2+y^2)]^&b&  '/
DATA XTIT/'!1X-axis'/
DATA YTIT/'!1Y-axis'/
SURF(RX,RY)=(((RX-10.)**2+RY**2)*((RX+10.)**2+RY**2))**.25
C-----
C *** GENERATE DATA
DO 1 I=1,NX
1 X(I)=XMIN+(XMAX-XMIN)*REAL(I-1)/REAL(NX-1)
DO 2 I=1,NY
2 Y(I)=YMIN+(YMAX-YMIN)*REAL(I-1)/REAL(NY-1)
DO 3 I=1,NC
3 C(I)=CMIN+(CMAX-CMIN)*REAL(I-1)/REAL(NC-1)
DO 4 I=1,NX
DO 4 J=1,NY
4 Z(I,J)=SURF(X(I),Y(J))
C *** PLOT
CALL PLOTN(0.,0.,15,5)          ! DEFINE GRAPHICS DEVICE
CALL CONTOUR(Z,X,Y,C,NP,NX,NY,NC,XMIN,XMAX,YMIN,YMAX,
+ .FALSE.,XTIT,YTIT,MTIT,STIT,1)
END

```

⁸In this case, the contour values are the value of the parameter in the equation for the curve $C=Z(X,Y)$.

⁹Always necessary at the beginning of one or a series of FIDDLER plots!



2.3.9 CONTOUR Example: Non-uniform Sampling Grid

Contours of the complex sine argument are generated by the following example. The grid over which $Z(X,Y)$ is non-uniform, and more samples are stored for the higher values of X . The grid is superimposed upon the graph, using line-type 4, (the LGRID parameter is .true. in the CONTOUR call.) In addition, the plot is flipped into the 4th quadrant: Axes numbers go from low values to high values in the same sense as the absolute values of the 4th quadrant.

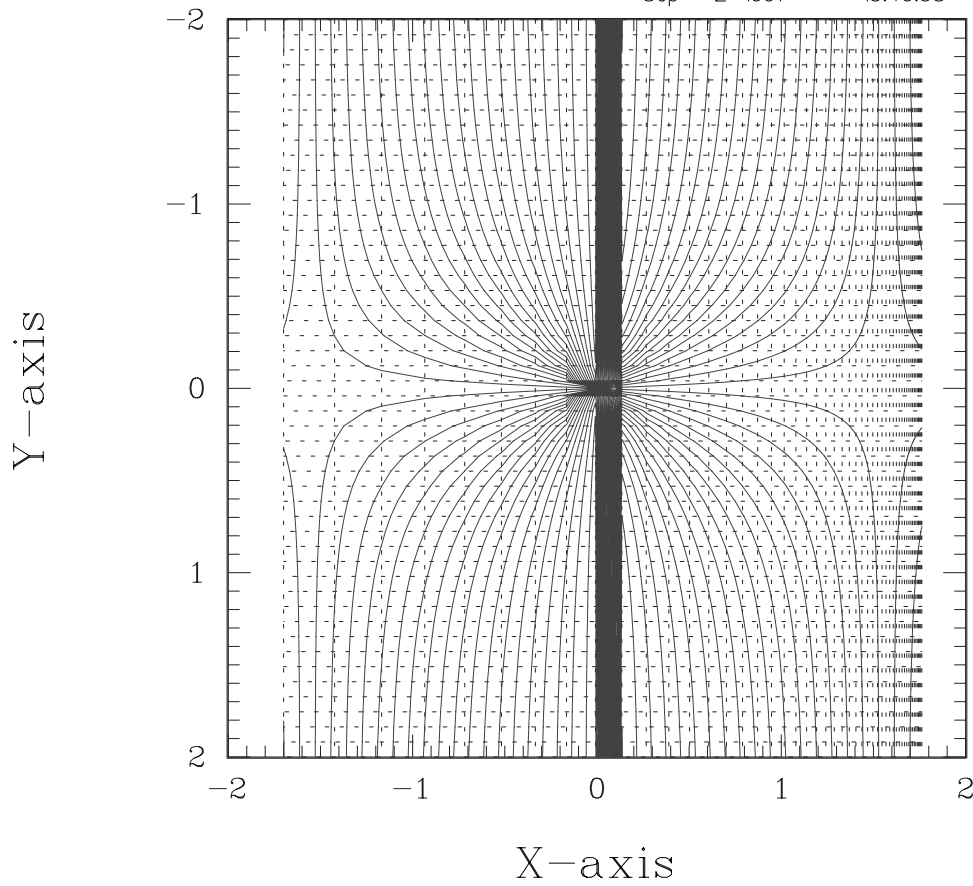
```

C#####
PROGRAM XCON2
C *** CONTOUR PLOT: NON-UNIFORM GRID
C#####
PARAMETER(NP=100)
DIMENSION X(NP),Y(NP),Z(NP,NP),C(NP)
CHARACTER*40 MTIT,STIT,XTIT,YTIT
DATA NX,NY,NC/50,50,50/
DATA XMIN,XMAX,YMIN,YMAX,CMIN,CMAX/-2.,2.,-2.,2.,-2.,2./
DATA MTIT/'!3Argument of Complex Sine  '/
DATA STIT/'!2z = tan^-1_[tanh(y)/tan(x)]'/
DATA XTIT/'!1X-axis                      '/
DATA YTIT/'!1Y-axis                      '/
SURF(RX,RY)=ATAN((EXP(2.*RY)-1.)/(EXP(2.*RY)+1.)/TAN(RX))
C-----
C *** GENERATE DATA
X0=XMIN
DO 1 I=1,NX
XO=XO+.3*EXP(-(XMAX-XMIN)*REAL(I-1)/REAL(NX-1))
1 X(I)=XO
DO 2 I=1,NY
2 Y(I)=YMIN+(YMAX-YMIN)*REAL(I-1)/REAL(NY-1)
DO 3 I=1,NC
3 C(I)=CMIN+(CMAX-CMIN)*REAL(I-1)/REAL(NC-1)
DO 4 I=1,NX
DO 4 J=1,NY
4 Z(I,J)=SURF(X(I),Y(J))
C *** PLOT
CALL PLOTN(0.,0.,15,5)          ! DEFINE GRAPHICS DEVICE
CALL CONTOUR(Z,X,Y,C,NP,NX,NY,NC,XMIN,XMAX,YMIN,YMAX,
+ .TRUE.,XTIT,YTIT,MTIT,STIT,4)
END

```

Argument of Complex Sine
 $z = \tan^{-1}[\tanh(y)/\tan(x)]$

Sep- 2-1997 15:49:58



2.3.10 HISTOGM Example: Random Samples with Transformation

The following example creates a histogram of the counts of the values of a random variable which appear within a set of .05 wide intervals of the x-axis. The first portion of the example gathers random values which are stored in X. The transformation of the uniformly-distributed random samples, to the exponentially distributed random samples, X, is performed in the first loop. The counting intervals are aligned with the XMIN parameter in the HISTOGM call (XMIN=0. in this case). So the counting intervals are: (0.,0.05), (0.05,0.10), and so on. If the XMIN parameter were shifted slightly from 0., say at 0.004, then the counting intervals would be aligned with XMIN, as (0.004,0.054), (0.054,0.104), and so on, while the plotting window would be unchanged. The section of the example following PLOT HISTOGRAM involves a terminal definition call to PLOTN.¹⁰, a series of modifications to the default plotting parameters via PLOTN, and one call to HISTOGM.

```

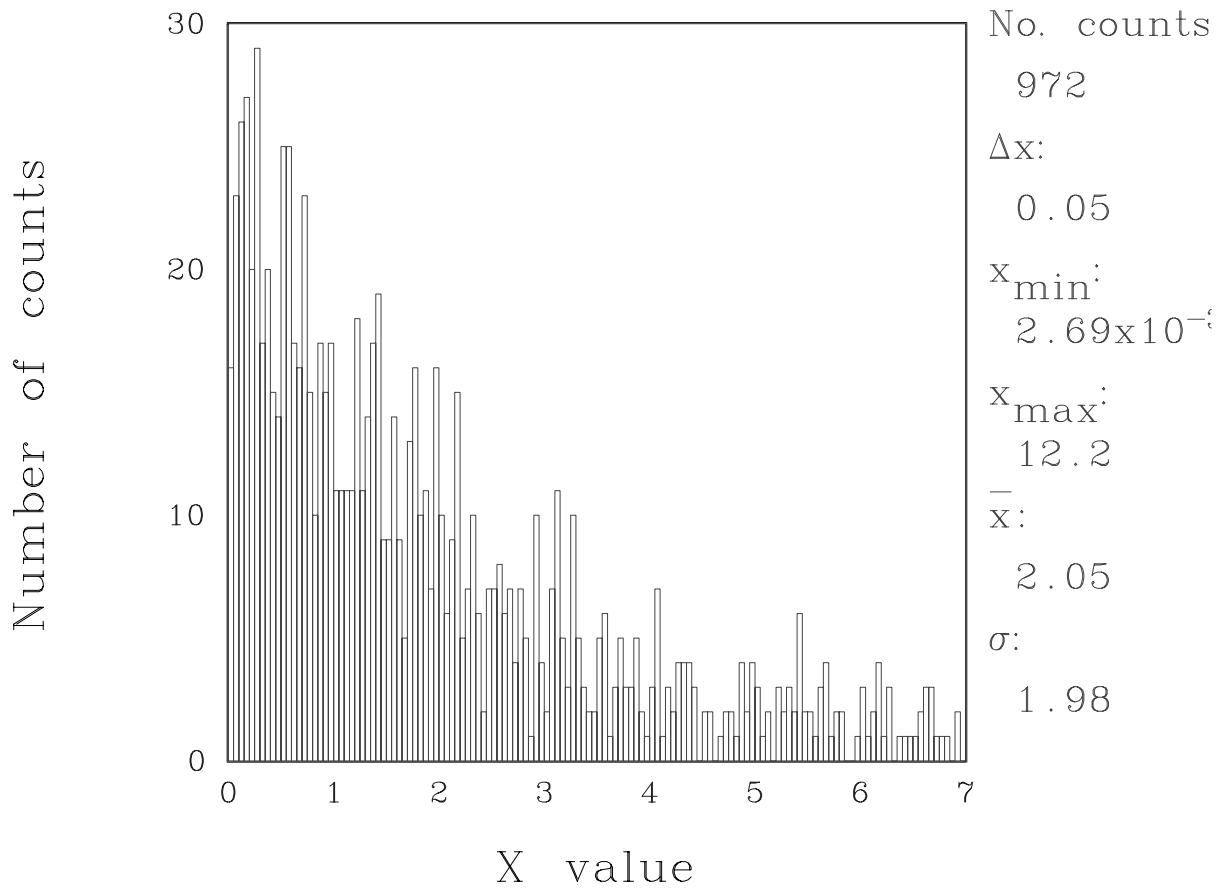
C#####
PROGRAM XHIS1
C *** HISTOGRAM
C#####
PARAMETER(NPTS=1000)
DIMENSION X(NPTS)
INTEGER*4 ISEED
DATA ISEED/93201/
DATA XMIN,XMAX,DX/0.,7.,.05/
C-----
C *** GENERATE DATA
DO 1 I=1,NPTS
2 R=SYSRND(ISEED)          ! RANDOM-NUMBER GENERATOR
  IF(R.EQ.0.)GOTO 2
  X(I)=2.*LOG(1./R)        ! EXPONENTIAL DISTRIBUTION
1 CONTINUE
C *** PLOT HISTOGRAM
CALL PLOTN(0.,0.,15, 5)    ! DEFINE GRAPHICS DEVICE
CALL PLOTN(0.,0.,17,31)   ! BLANK AXES TICS
CALL PLOTN(0.,0.,17,36)   ! BLANK DATE
CALL HISTOGM(X,NPTS,XMIN,XMAX,DX,'!1X value',
+ '!3FIDDLER !1Histogram','!1Exponential distribution',1)
END

```

¹⁰Always necessary at the beginning of one or a series of FIDDLER plots!

FIDDLER Histogram

Exponential distribution



2.3.11 HISTOGM Example: Wider Counting Intervals

The following example is the same as the previous HISTOGM example, except that the counting intervals are wider, .25 instead of .05. The number of counts in each interval is typically much higher than in the previous example, since there is higher probability that a random value appears in the larger intervals. The histogram is not normalized to the total number of counts. If this is necessary, a routine to accomplish this might be modeled after the HISTOGM subroutine, with slight modifications.

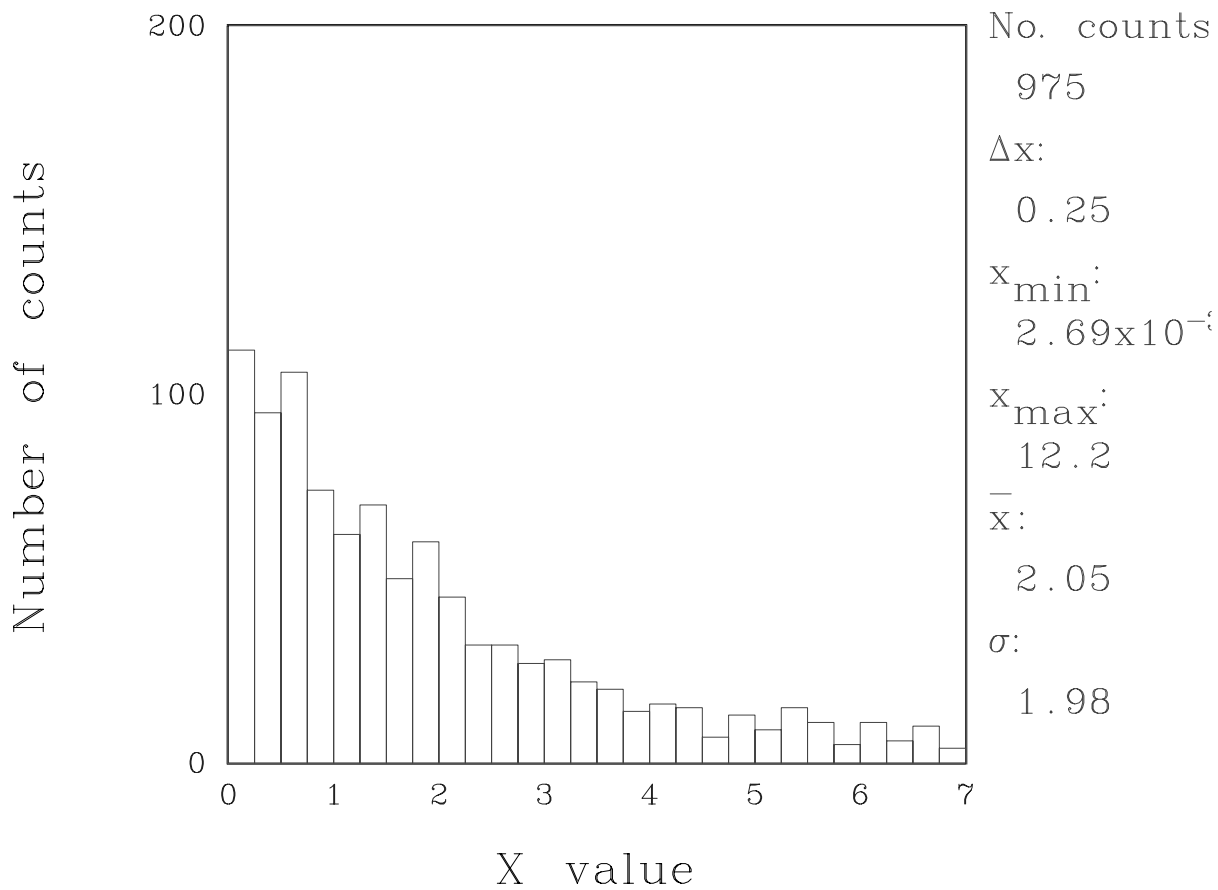
```

C#####
  PROGRAM XHIS1
C *** HISTOGRAM
C#####
  PARAMETER(NPTS=1000)
  DIMENSION X(NPTS)
  INTEGER*4 ISEED
  DATA ISEED/93201/
  DATA XMIN,XMAX,DX/0.,7.,.25/
C-----
C *** GENERATE DATA
  DO 1 I=1,NPTS
    2  R=SYSRND(ISEED)          ! RANDOM-NUMBER GENERATOR
      IF(R.EQ.0.)GOTO 2
      X(I)=2.*LOG(1./R)        ! EXPONENTIAL DISTRIBUTION
    1 CONTINUE
C *** PLOT HISTOGRAM
  CALL PLOTN(0.,0.,15, 5)      ! DEFINE GRAPHICS DEVICE
  CALL PLOTN(0.,0.,17,31)     ! BLANK AXES TICS
  CALL PLOTN(0.,0.,17,36)     ! BLANK DATE
  CALL HISTOGM(X,NPTS,XMIN,XMAX,DX,'!1X value',
+ '!3FIDDLER !1Histogram','!1Exponential distribution',1)
  END

```

FIDDLER Histogram

Exponential distribution



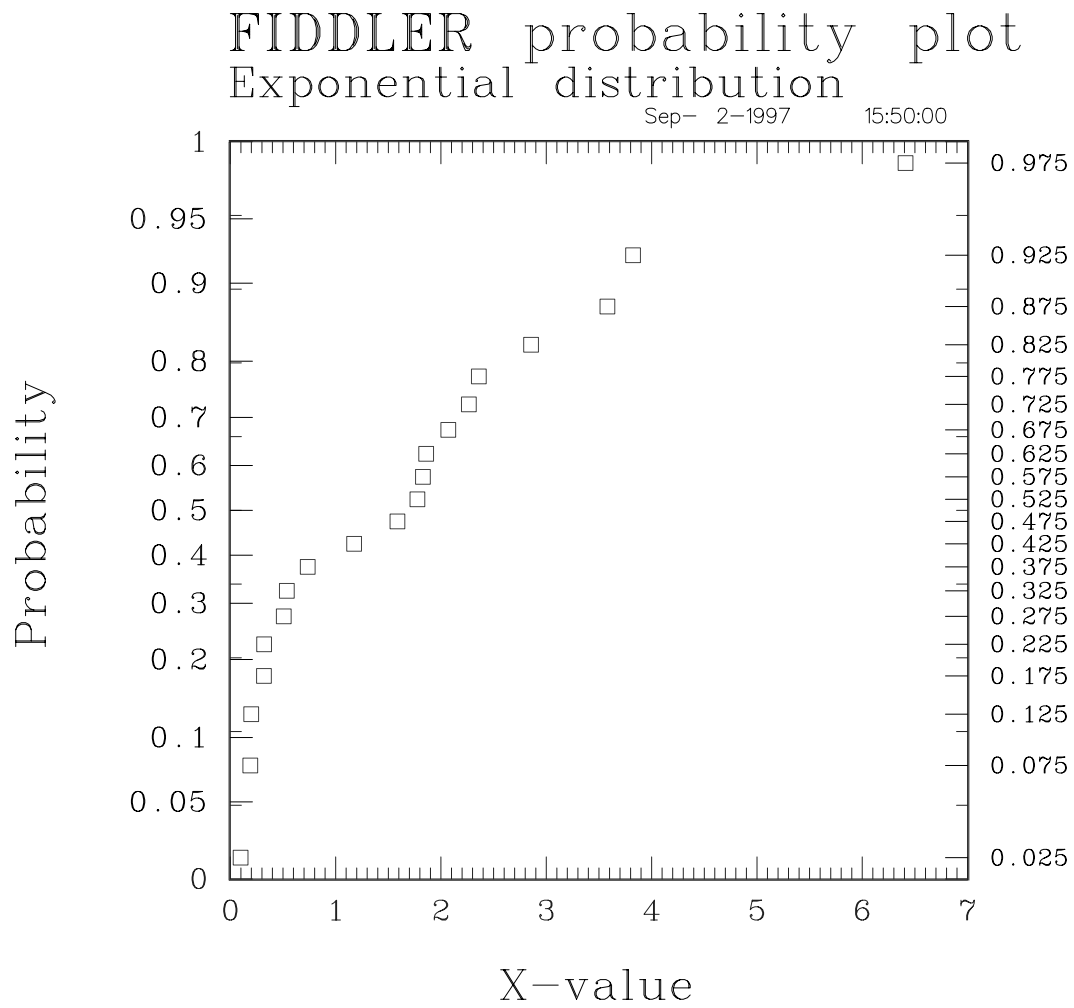
2.3.12 PROBPLT Example: Exponential distribution

```

C#####
PROGRAM XPROB
C *** PROBABILITY PLOT
C#####
PARAMETER(NPTS=20)
DIMENSION X(NPTS)
INTEGER*4 ISEED
DATA ISEED/90321/

C-----
C *** GENERATE DATA
DO 1 I=1,NPTS
  2  R=SYSRND(ISEED)          ! RANDOM-NUMBER GENERATOR
    IF(R.EQ.0.)GOTO 2
    X(I)=2.*LOG(1./R)        ! EXPONENTIAL DISTRIBUTION
  1 CONTINUE
C *** PLOT
CALL PLOTN(0.,0.,15,5) ! DEFINE GRAPHICS DEVICE
CALL PROBPLT(X,NPTS,'S','!X-value',
+ '!3FIDDLER !probability plot','!1Exponential distribution',
+ .TRUE.,.TRUE.)
END

```



2.3.13 PROBPLT Example: Normal distribution

```

C#####
  PROGRAM XPROB
C *** PROBABILITY PLOT
C#####
  PARAMETER(NPTS=1000)
  DIMENSION X(NPTS)
  INTEGER*4 ISEED
  DATA ISEED/90321/

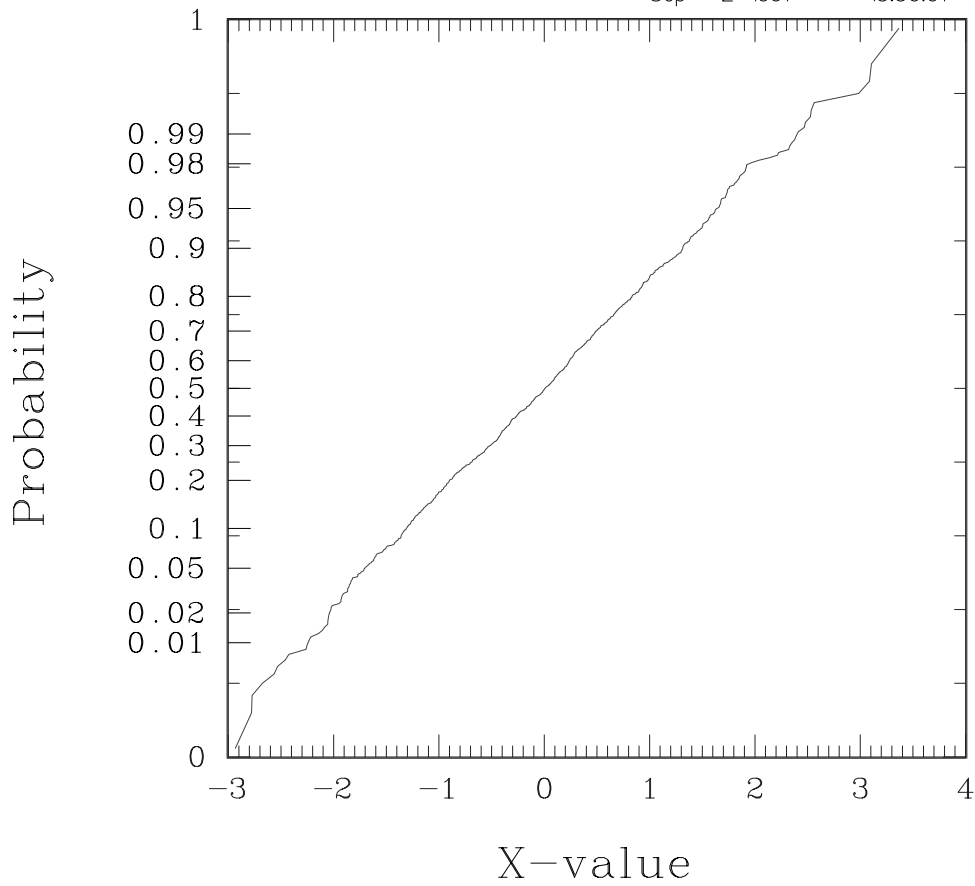
C-----
C *** GENERATE DATA
  DO 1 I=1,NPTS
    2  U=SYSRND(ISEED)*2.-1.
      V=SYSRND(ISEED)*2.-1.
      R=U*U+V*V
      IF(R.GE.1..OR.R.LE.0.)GOTO 2
      X(I)=U*SQRT(-2.*LOG(R)/R)
  1 CONTINUE
C *** PLOT
  CALL PLOTN(0.,0.,15,5) ! DEFINE GRAPHICS DEVICE
  CALL PROBPLT(X,NPTS,'L','!1X-value',
+ '!3FIDDLER !1probability plot','!1Normal distribution',
+ .TRUE.,.FALSE.)
  END

```

FIDDLER probability plot

Normal distribution

Sep- 2-1997 15:50:01



2.3.14 VECTORS Example: Sampled Vector Field

Arrows representing the length and direction of a sampled vector field are plotted in the following example. The vector field components are calculated in the portion of the program below GENERATE DATA. The vector field components are specified at a fine grid of uniformly spaced nodes.¹¹ Once the grid locations and vector components are calculated, the vector plot is generated, under PLOT. First, a terminal definition call is made via PLOTN.¹² The VECTORS call has the parameter NC¹³ equal to 20. The vector arrows are drawn with the base of each arrow at each of these course grid locations. The length and direction of each arrow represents the relative magnitude and direction of the vector field at the course grid locations, interpolated from the sampled grid.

```

C#####
      PROGRAM XVEC1
C *** VECTOR PLOT
C#####
      PARAMETER(NX=50,NY=50)
      DIMENSION X(NX),Y(NY),ZX(NX,NY),ZY(NX,NY)
      CHARACTER*30 MTIT,STIT,XTIT,YTIT
      DATA NC,XMIN,XMAX,YMIN,YMAX/20,-2.,2.,-2.,2./
      DATA MTIT/' !3FIDDLER !1vector plot      '/
      DATA STIT/' !1zx=xsin(xy); zy=cos(xy)    '/
      DATA XTIT/' !1X-axis                      '/
      DATA YTIT/' !1Y-axis                      '/
C-----
C *** GENERATE DATA
      DO 1 I=1,NX
1    X(I)=XMIN+(XMAX-XMIN)*REAL(I-1)/REAL(NX-1)
      DO 2 I=1,NY
2    Y(I)=YMIN+(YMAX-YMIN)*REAL(I-1)/REAL(NY-1)
      DO 3 I=1,NX
      DO 3 J=1,NY
          ZX(I,J)=SIN(X(I)*Y(J))*X(I)
          ZY(I,J)=COS(X(I)*Y(J))
3    CONTINUE
C *** PLOT
      CALL PLOTN(0.,0.,15,5)          ! DEFINE GRAPHICS DEVICE
      CALL VECTORS(ZX,ZY,X,Y,NX,NX,NY,NC,XMIN,XMAX,YMIN,YMAX,
+ XTIT,YTIT,MTIT,STIT,1)
      END

```

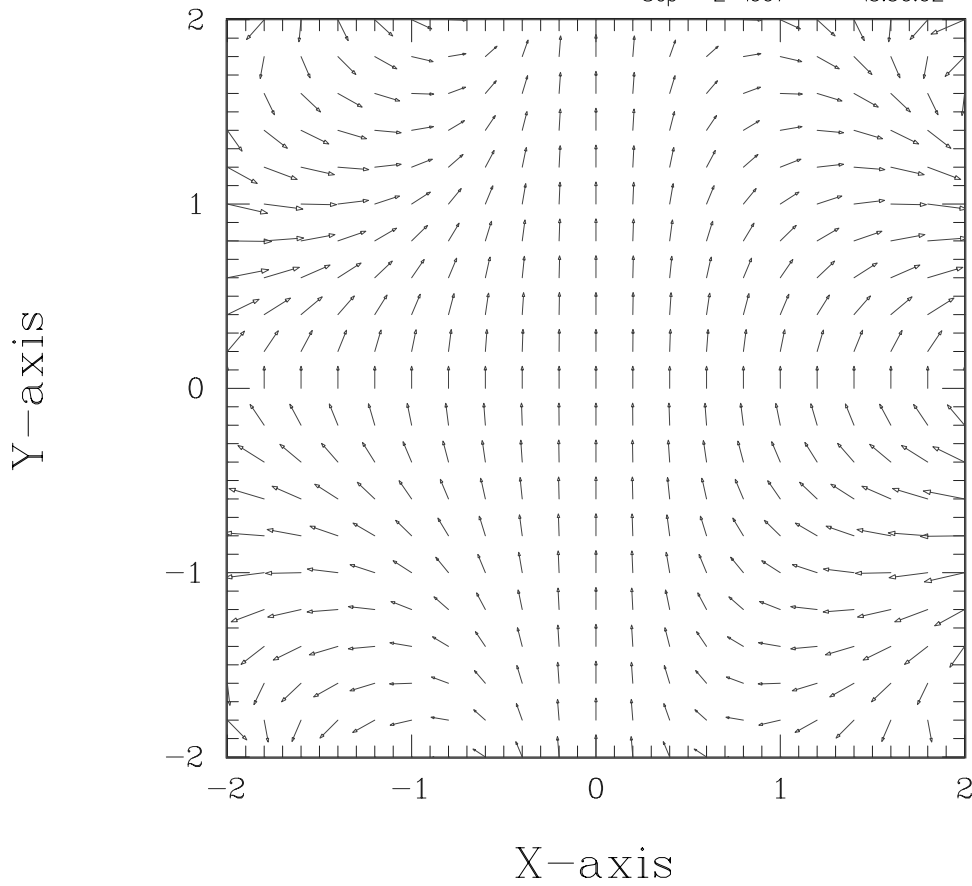
¹¹ The sampling grid need not be uniformly spaced.

¹² Always necessary at the beginning of one or a series of FIDDLER plots!

¹³ NG in the VECTORS specification section

FIDDLER vector plot
 $zx = x \sin(xy)$; $zy = \cos(xy)$

Sep- 2-1997 15:50:02



2.3.15 SURFACE Example: Opaque View at 30 Degrees

In this example, a 3-dimensional surface $Z(X,Y)$ is sampled over a region of the X-Y plane, at uniformly-spaced grid locations ¹⁴ The X-axis and Y-axis values of the (rectangular) sampling grid are stored in X and Y. Once the sampled values of the grid and surface are collected, the SURFACE plot is generated. SURFACE interpolates sampled surface values onto a uniformly-spaced grid which fills the re-scaled X-Y plotting region. The number of divisions is specified by the NC parameter in this example (50). ¹⁵ The portion of the example following PLOT involves one terminal definition call to PLOTN ¹⁶, and one call to SURFACE. The MODE parameter (3^{rd} from last) in this example is 3, so that both top and bottom faces of the surface are plotted (using different colors). The LOPAQ parameter (2^{nd} from last) is .true., so that the hidden mesh lines are not plotted. The view is an isometric projection.

```

C#####
PROGRAM XSUR1
C *** SURFACE PLOT
C#####
PARAMETER(NP=100)
DIMENSION X(NP),Y(NP),Z(NP,NP)
CHARACTER*30 MTIT,STIT,XTIT,YTIT,ZTIT
LOGICAL LOPAQ
DATA NX,NY,NC,XMIN,XMAX,YMIN,YMAX/50,50,50,-2.,2.,-2.,2./
DATA ANGLE,IORIENT,MODE,LOPAQ/30.,1,3,.TRUE./
DATA MTIT/'!3Argument of Complex Sine '/
DATA STIT/'!2z = Tan^-1_[Tanh(y)/Tan(x)] '/
DATA XTIT/'!1X-axis '/
DATA YTIT/'!1Y-axis '/
DATA ZTIT/'!1Z-axis '/
SURF(RX,RY)=ATAN((EXP(2.*RY)-1.)/(EXP(2.*RY)+1.)/TAN(RX))
C-----
C *** GENERATE DATA
DO 1 I=1,NX
1 X(I)=XMIN+(XMAX-XMIN)*REAL(I-1)/REAL(NX-1)
DO 2 I=1,NY
2 Y(I)=YMIN+(YMAX-YMIN)*REAL(I-1)/REAL(NY-1)
DO 3 I=1,NX
DO 3 J=1,NY
3 Z(I,J)=SURF(X(I),Y(J))
C *** PLOT
CALL PLOTN(0.,0.,15,5) ! DEFINE GRAPHICS DEVICE
CALL SURFACE(Z,X,Y,NP,NX,NY,NC,XMIN,XMAX,YMIN,YMAX,
+ XTIT,YTIT,ZTIT,MTIT,STIT,ANGLE,MODE,LOPAQ,IORIENT)
END

```

¹⁴The sampling grid need not be uniformly spaced.

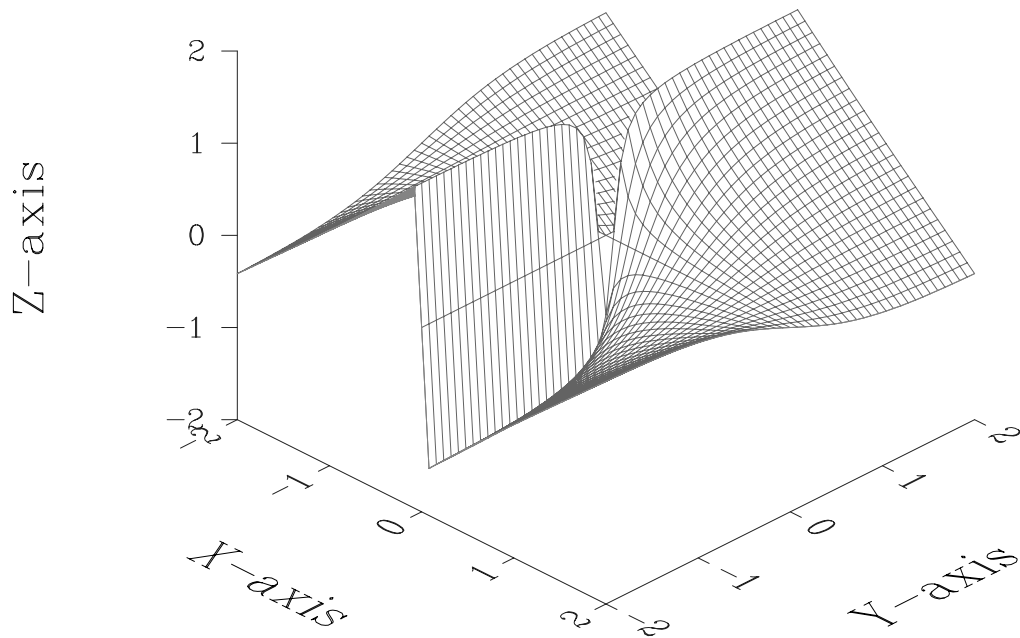
¹⁵NG in the SURFACE specification section.

¹⁶Always necessary at the beginning of one or a series of FIDDLER plots!

Argument of Complex Sine

$$z = \text{Tan}^{-1}[\text{Tanh}(y)/\text{Tan}(x)]$$

Sep- 2-1997 15:50:03



2.3.16 SURFACE Example: Translucent View at 10 Degrees

This example program is identical to the last SURFACE example, with the exception of the parameters ANGLE, MODE, LOPAQ, and IORIENT. These are the last four parameters in the call to SURFACE. The ANGLE parameter controls the "tilt" of the 3-dimensional view. The view used by SURFACE (And SPACELN) is an isometric projection: the X=Y line is horizontal, and the Z-Axis is tilted toward the front by ANGLE degrees. The angle in this example is 10. degrees: a quite shallow angle, so that the view is almost from the side, and the axes-labels are considerably slanted. The MODE parameter is 1, so that only the top face of the surface is plotted. The LOPAQ parameter is .false. in the example, so that the view is "translucent": no line segments are hidden. Finally, the view is flipped into the 7th octant of X-Y-Z space (Z values go from high values at the bottom of the Z-Axis to low values at the top of the Z-Axis, and so on.)

```

C#####
PROGRAM XSUR1
C *** SURFACE PLOT
C#####
PARAMETER(NP=100)
DIMENSION X(NP),Y(NP),Z(NP,NP)
CHARACTER*30 MTIT,STIT,XTIT,YTIT,ZTIT
LOGICAL LOPAQ
DATA NX,NY,NC,XMIN,XMAX,YMIN,YMAX/50,50,50,-2.,2.,-2.,2./
DATA ANGLE,IORIENT,MODE,LOPAQ/10.,7,1,.FALSE./
DATA MTIT/'!3Argument of Complex Sine '/
DATA STIT/'!2z = Tan^-1_[Tanh(y)/Tan(x)] '/
DATA XTIT/'!1X-axis '/
DATA YTIT/'!1Y-axis '/
DATA ZTIT/'!1Z-axis '/
SURF(RX,RY)=ATAN((EXP(2.*RY)-1.)/(EXP(2.*RY)+1.)/TAN(RX))
C-----
C *** GENERATE DATA
DO 1 I=1,NX
1 X(I)=XMIN+(XMAX-XMIN)*REAL(I-1)/REAL(NX-1)
DO 2 I=1,NY
2 Y(I)=YMIN+(YMAX-YMIN)*REAL(I-1)/REAL(NY-1)
DO 3 I=1,NX
DO 3 J=1,NY
3 Z(I,J)=SURF(X(I),Y(J))
C *** PLOT
CALL PLOTN(0.,0.,15,5) ! DEFINE GRAPHICS DEVICE
CALL SURFACE(Z,X,Y,NP,NX,NY,NC,XMIN,XMAX,YMIN,YMAX,
+ XTIT,YTIT,ZTIT,MTIT,STIT,ANGLE,MODE,LOPAQ,IORIENT)
END

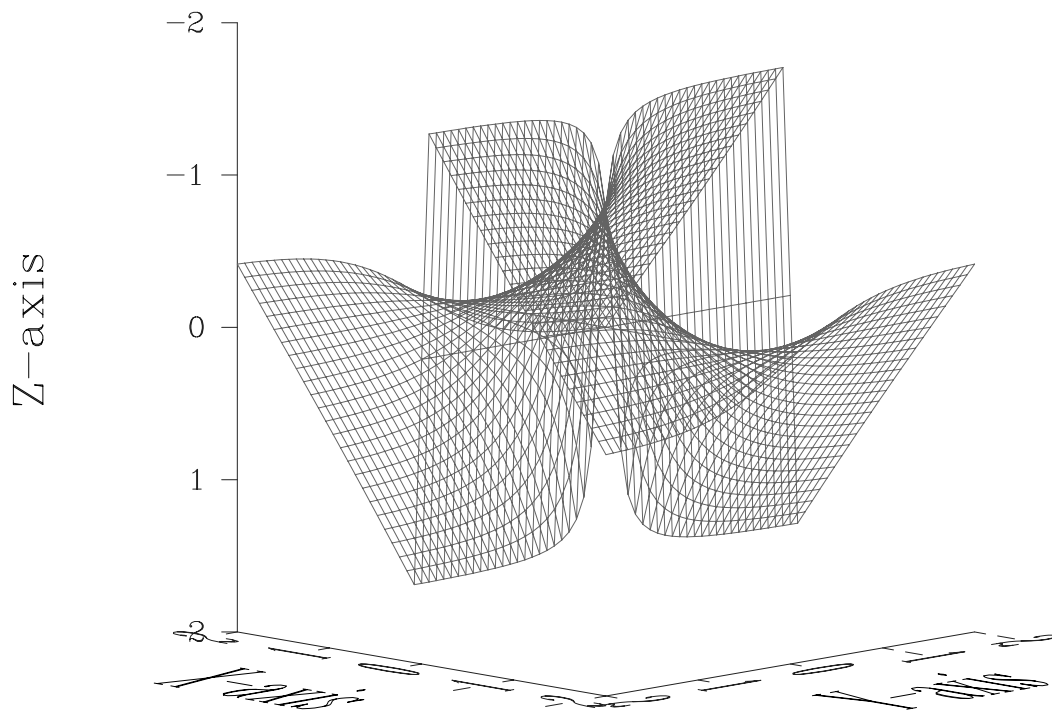
```

Argument of Complex Sine

$$z = \text{Tan}^{-1}[\text{Tanh}(y)/\text{Tan}(x)]$$

Sep- 2-1997

15:50:04



2.3.17 SURFACE Example: Plot Window Bigger than Sampling Window

The ANGLE parameter in the call to SURFACE in this example is 45 degrees, which gives a nearly top-view of the surface. In this example, the sampling window is within the plotting window, so the surface is plotted for a subregion of the plotting region.

```

C#####
      PROGRAM XSUR3
C *** SURFACE PLOT
C#####
      PARAMETER(NP=100)
      DIMENSION    X(NP),Y(NP),Z(NP,NP)
      CHARACTER*30 MTIT,STIT,XTIT,YTIT,ZTIT
      LOGICAL      LOPAQ
      DATA NX,NY,NC,XMIN,XMAX,YMIN,YMAX/50,50,30,-2.,2.,-2.,2./
      DATA ANGLE,IORIENT,MODE,LOPAQ/45.,1,3,.TRUE./
      DATA MTIT/'!3Arrow function'      '/'
      DATA STIT/'!2z = x/y -y/x'        '/'
      DATA XTIT/'!1X-axis'              '/'
      DATA YTIT/'!1Y-axis'              '/'
      DATA ZTIT/'!1Z-axis'              '/'
C-----
C *** GENERATE DATA
      DO 1 I=1,NX
1    X(I)=XMIN+(XMAX-XMIN)*REAL(I-1)/REAL(NX-1)
      DO 2 I=1,NY
2    Y(I)=YMIN+(YMAX-YMIN)*REAL(I-1)/REAL(NY-1)
      DO 3 I=1,NX
      DO 3 J=1,NY
      Z(I,J)=0.
3    IF(X(I)*Y(J).NE.0.)Z(I,J)=X(I)/Y(J)-Y(J)/X(I)
C *** PLOT
      CALL PLOTN(0.,0.,15,5)           ! DEFINE GRAPHICS DEVICE
      CALL SURFACE(Z,X,Y,NP,NX,NY,NC,XMIN,XMAX,-4.,4.,
+ XTIT,YTIT,ZTIT,MTIT,STIT,ANGLE,MODE,LOPAQ,IORIENT)
      END

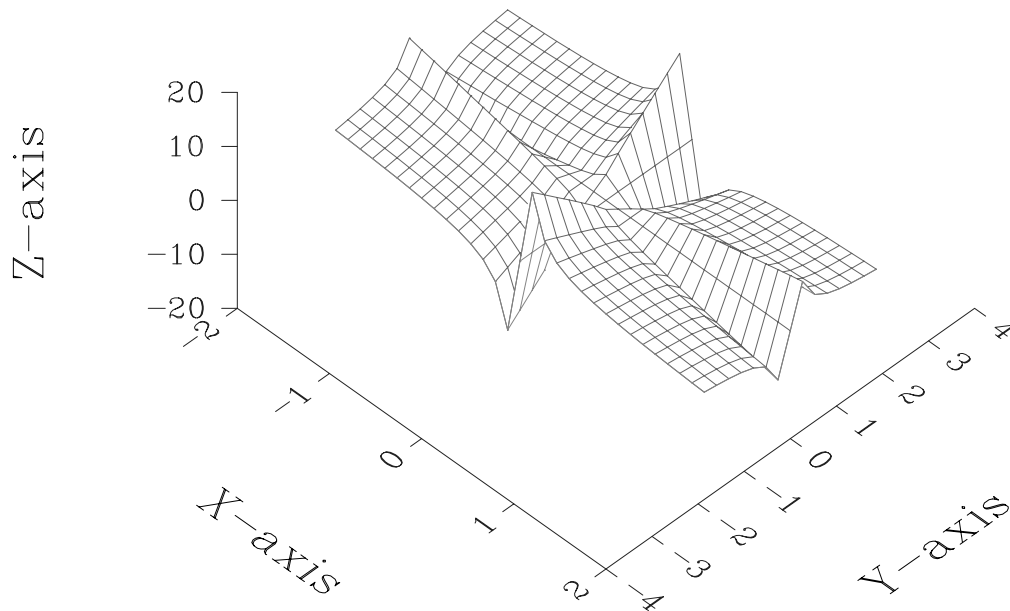
```

Arrow function

$$z = x/y - y/x$$

Sep- 2-1997

15:50:05



2.3.18 SURFACE Example: Orthographic view

```

C#####
PROGRAM XSUR4A
C *** SURFACE PLOT: ORTHOGRAPHIC
C#####
PARAMETER(NP=100)
DIMENSION X(NP),Y(NP),Z(NP,NP)
CHARACTER*30 MTIT,STIT,XTIT,YTIT,ZTIT
LOGICAL LOPAQ
DATA NX,NY,NC,XMIN,XMAX,YMIN,YMAX/50,50,0,-2.,2.,-2.,2./
DATA PHI,THETA,IORIENT,MODE,LOPAQ/20.,30.,1,6,.TRUE./
DATA MTIT/'!3Arrow function' '/'
DATA STIT/'!2z = x/y -y/x' '/'
DATA XTIT/'!1X-axis' '/'
DATA YTIT/'!1Y-axis' '/'
DATA ZTIT/'!1Z-axis' '/'
C-----
C *** GENERATE DATA
DO 1 I=1,NX
1 X(I)=XMIN+(XMAX-XMIN)*REAL(I-1)/REAL(NX-1)
DO 2 I=1,NY
2 Y(I)=YMIN+(YMAX-YMIN)*REAL(I-1)/REAL(NY-1)
DO 3 I=1,NX
DO 3 J=1,NY
Z(I,J)=0.
3 IF(X(I)*Y(J).NE.0.)Z(I,J)=X(I)/Y(J)-Y(J)/X(I)
C *** PLOT
CALL PLOTN(0.,0.,15,5) ! DEFINE GRAPHICS DEVICE
CALL SURFAC2(Z,X,Y,NP,NX,NY,NC,XMIN,XMAX,-4.,4.,
+ XTIT,YTIT,ZTIT,MTIT,STIT,PHI,THETA,MODE,LOPAQ,IORIENT)
END

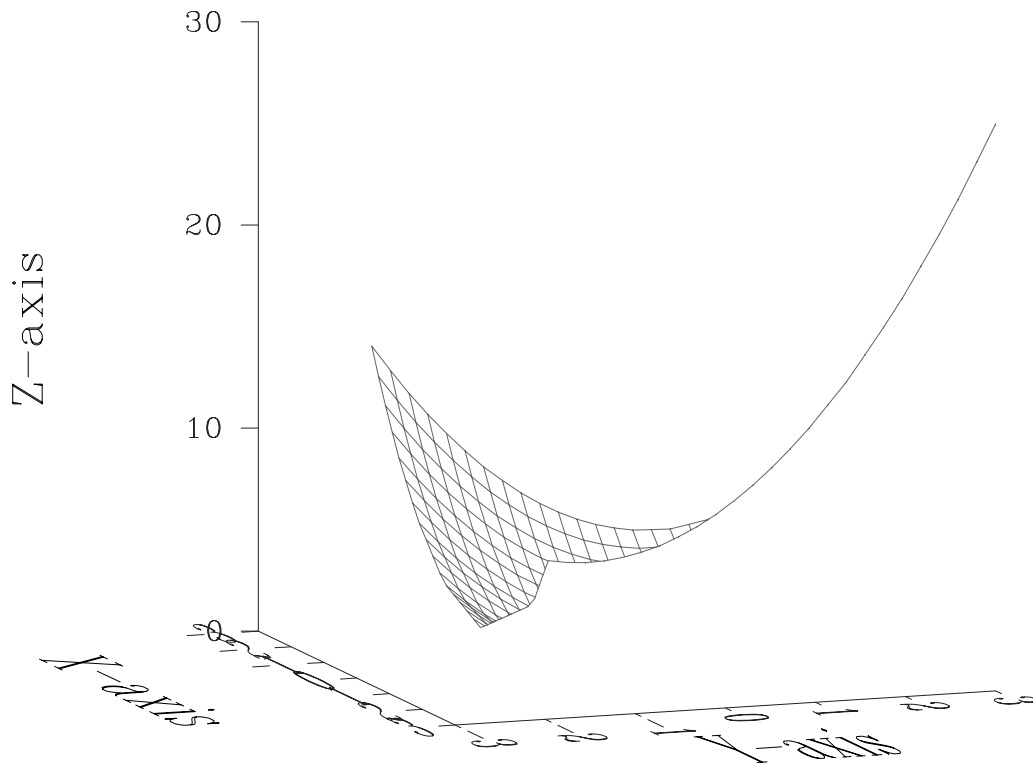
```

Arrow function

$$z = x/y - y/x$$

Sep- 2-1997

15:50:27



2.3.19 SURFACE Example: Oblique view - x-axis at front

```

C#####
PROGRAM XSUR4B
C *** SURFACE PLOT: OBLIQUE
C#####
PARAMETER(NP=100)
DIMENSION X(NP),Y(NP),Z(NP,NP)
CHARACTER*30 MTIT,STIT,XTIT,YTIT,ZTIT
LOGICAL LOPAQ
DATA NX,NY,NC,XMIN,XMAX,YMIN,YMAX/50,50,0,-2.,2.,-2.,2./
DATA PHI,THETA,IORIENT,MODE,LOPAQ/20.,30.,1,16,.TRUE./
DATA MTIT/'!3Arrow function' '/'
DATA STIT/'!2z = x/y -y/x' '/'
DATA XTIT/'!1X-axis' '/'
DATA YTIT/'!1Y-axis' '/'
DATA ZTIT/'!1Z-axis' '/'
C-----
C *** GENERATE DATA
DO 1 I=1,NX
1 X(I)=XMIN+(XMAX-XMIN)*REAL(I-1)/REAL(NX-1)
DO 2 I=1,NY
2 Y(I)=YMIN+(YMAX-YMIN)*REAL(I-1)/REAL(NY-1)
DO 3 I=1,NX
DO 3 J=1,NY
Z(I,J)=0.
3 IF(X(I)*Y(J).NE.0.)Z(I,J)=X(I)/Y(J)-Y(J)/X(I)
C *** PLOT
CALL PLOTN(0.,0.,15,5) ! DEFINE GRAPHICS DEVICE
CALL SURFAC2(Z,X,Y,NP,NX,NY,NC,XMIN,XMAX,-4.,4.,
+ XTIT,YTIT,ZTIT,MTIT,STIT,PHI,THETA,MODE,LOPAQ,IORIENT)
END

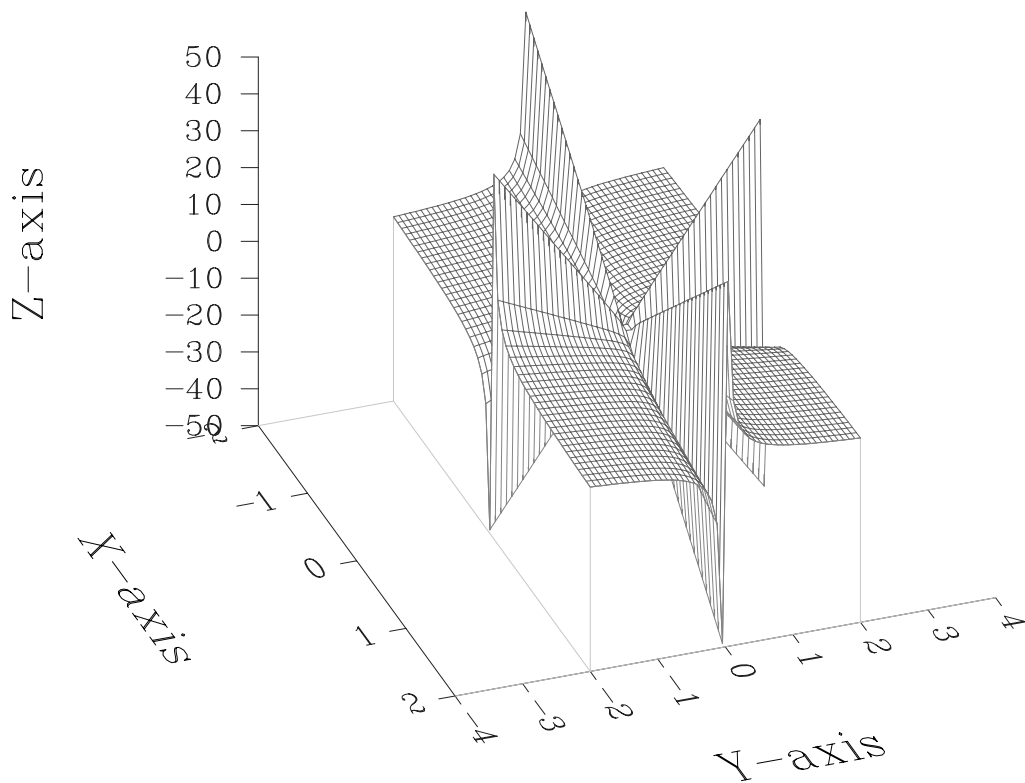
```

Arrow function

$$z = x/y - y/x$$

Sep- 2-1997

15:50:28



2.3.20 SURFACE Example: Oblique view - y-axis at front

```

C#####
PROGRAM XSUR4C
C *** SURFACE PLOT: OBLIQUE
C#####
PARAMETER(NP=100)
DIMENSION X(NP),Y(NP),Z(NP,NP)
CHARACTER*30 MTIT,STIT,XTIT,YTIT,ZTIT
LOGICAL LOPAQ
DATA NX,NY,NC,XMIN,XMAX,YMIN,YMAX/50,50,0,-2.,2.,-2.,2./
DATA PHI,THETA,IORIENT,MODE,LOPAQ/20.,30.,1,26,.TRUE./
DATA MTIT/'!3Arrow function' '/'
DATA STIT/'!2z = x/y -y/x' '/'
DATA XTIT/'!1X-axis' '/'
DATA YTIT/'!1Y-axis' '/'
DATA ZTIT/'!1Z-axis' '/'
C-----
C *** GENERATE DATA
DO 1 I=1,NX
1 X(I)=XMIN+(XMAX-XMIN)*REAL(I-1)/REAL(NX-1)
DO 2 I=1,NY
2 Y(I)=YMIN+(YMAX-YMIN)*REAL(I-1)/REAL(NY-1)
DO 3 I=1,NX
DO 3 J=1,NY
Z(I,J)=0.
3 IF(X(I)*Y(J).NE.0.)Z(I,J)=X(I)/Y(J)-Y(J)/X(I)
C *** PLOT
CALL PLOTN(0.,0.,15,5) ! DEFINE GRAPHICS DEVICE
CALL SURFAC2(Z,X,Y,NP,NX,NY,NC,XMIN,XMAX,-4.,4.,
+ XTIT,YTIT,ZTIT,MTIT,STIT,PHI,THETA,MODE,LOPAQ,IORIENT)
END

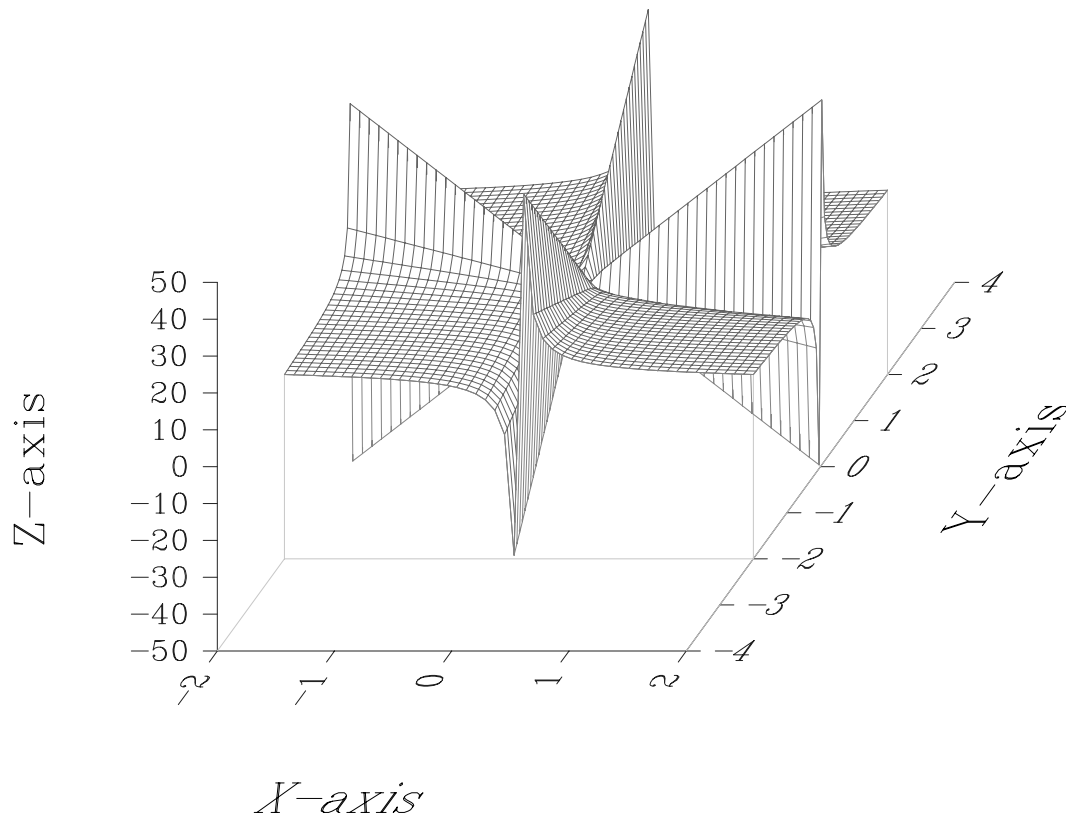
```

Arrow function

$$z = x/y - y/x$$

Sep- 2-1997

15:50:29



2.3.21 SURFACE Example: Border and options

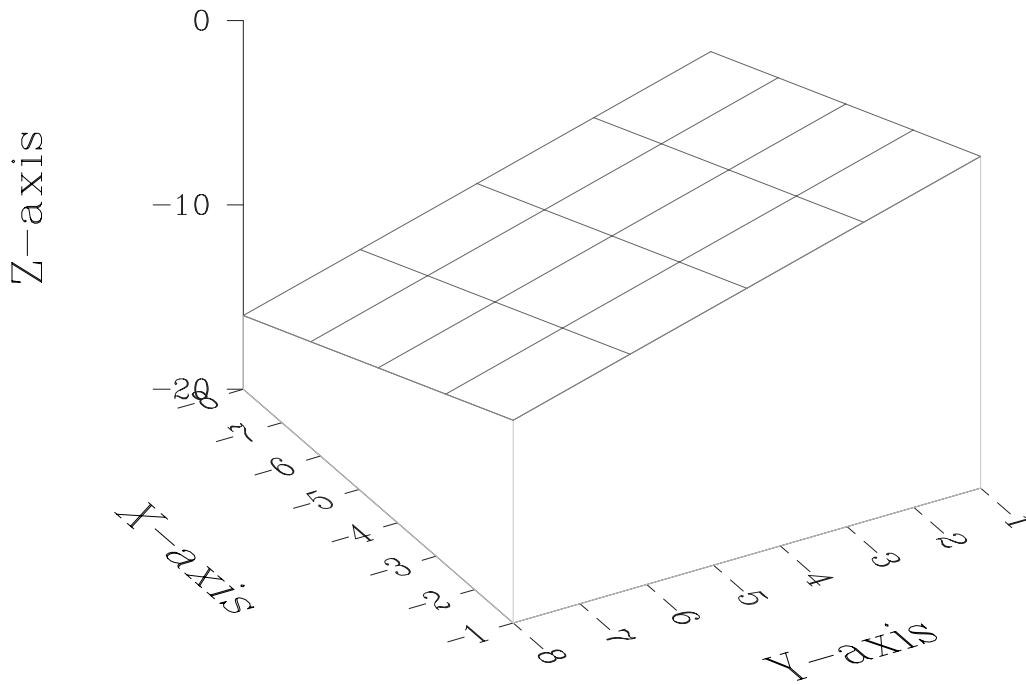
```

C#####
PROGRAM XSUR5
C *** SURFACE PLOT: ADDITIONAL BORDER
C#####
PARAMETER (NP=100)
DIMENSION X(NP),Y(NP),Z(NP,NP)
CHARACTER*80 MTIT,STIT,XTIT,YTIT,ZTIT
DATA NX,NY,NC/5,5,5/
DATA XMIN,XMAX,YMIN,YMAX/-8.,-1.,-8.,-1./
DATA MTIT/'!3FIDDLER !1Surface plot'/
DATA STIT/'!1with border'/
DATA XTIT/'!1X-axis'/
DATA YTIT/'!1Y-axis'/
DATA ZTIT/'!1Z-axis'/
C-----
C *** GENERATE DATA
DO 1 I=1,NX
1 X(I)=XMIN+(XMAX-XMIN)*REAL(I-1)/REAL(NX-1)
DO 2 I=1,NY
2 Y(I)=YMIN+(YMAX-YMIN)*REAL(I-1)/REAL(NY-1)
DO 4 I=1,NX
DO 4 J=1,NY
4 Z(I,J)=X(I)+Y(J)
C *** PLOT
CALL PLOTN(0.,0.,15,5) ! DEFINE GRAPHICS DEVICE
CALL PLOTN(1.,0.,17,50) ! DISABLE PLOT CLOSING
CALL SURFAC2(Z,X,Y,NP,NX,NY,NC,XMIN,XMAX,YMIN,YMAX,
+ XTIT,YTIT,ZTIT,MTIT,STIT,30.,30.,6.,.TRUE.,1)
C *** BORDER
CALL PLOTN(0.01,0.01,3,0)
CALL PLOTN(0.01,9.99,2,0)
CALL PLOTN(9.99,9.99,2,0)
CALL PLOTN(9.99,0.01,2,0)
CALL PLOTN(0.01,0.01,2,0)
C *** CLOSE PLOT
CALL PLOTN(0.,0.,16,0)
END

```

FIDDLER Surface plot with border

Sep- 2-1997 15:50:33



2.3.22 SPACELN Example: Line Segments and Verticals

A space-helix is plotted by the following example FORTRAN program. 1000 values for the helix function are gathered in the portion below GENERATE DATA. The plot is generated by one terminal definition call to PLOTN¹⁷, and one call to SPACELN. In the example, vertical lines are dropped from the curve to the X-Y plane after each interval of ten samples. These vertical lines are drawn with line-type 4 by default, so that the curve is not obscured. This line-type may be changed, however, using a call to PLOTN.

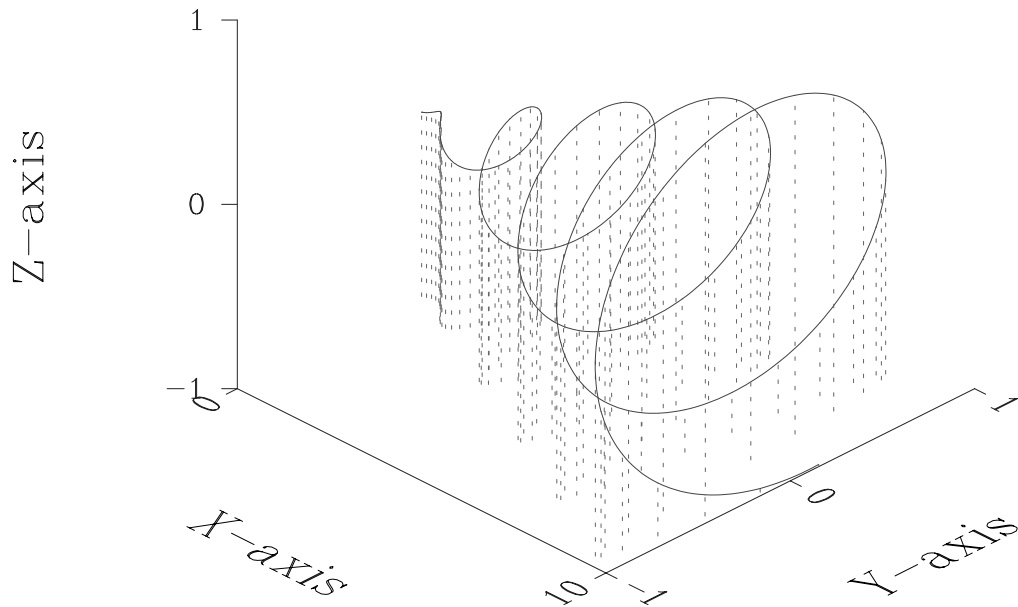
```

C#####
PROGRAM XSPC1
C *** SPACE-LINE PLOT
C#####
PARAMETER(NPTS=1000)
DIMENSION X(NPTS),Y(NPTS),Z(NPTS)
CHARACTER*80 MTIT,STIT,XTIT,YTIT,ZTIT
DATA TMIN,TMAX/0.,10./
DATA XMIN,XMAX,YMIN,YMAX,ZMIN,ZMAX/0.,10.,-1.,1.,-1.,1./
DATA MTIT/'!3FIDDLER !1Space-line plot'/
DATA STIT/'!1x(t)=t; y(t)=a cos(t) z(t)=b sin(t)'/
DATA XTIT/'!1X-axis'/
DATA YTIT/'!1Y-axis'/
DATA ZTIT/'!1Z-axis'/
C-----
C *** GENERATE DATA
DO 1 I=1,NPTS
  T=TMIN+(TMAX-TMIN)*REAL(I-1)/REAL(NPTS-1)
  X(I)=T
  Y(I)=T/10.*COS(3.*T)
  Z(I)=T/10.*SIN(3.*T)
1 CONTINUE
C *** PLOT
CALL PLOTN(0.,0.,15,5) ! DEFINE GRAPHICS DEVICE
CALL SPACELN(X,Y,Z,NPTS,XMIN,XMAX,YMIN,YMAX,ZMIN,ZMAX,
+ XTIT,YTIT,ZTIT,MTIT,STIT,'L',10,30.,1)
END

```

¹⁷Always necessary at the beginning of one or a series of FIDDLER plots!

FIDDLER Space-line plot
 $x(t)=t; y(t)=a \cos(t) z(t)=b \sin(t)$
Sep- 2-1997 15:50:34



2.3.23 SPACELN Example: Plotting Symbols

SPACELN is used in the following example to plot a space-Lissajous pattern. 200 samples are taken of the curve, and the samples are represented with squares on the plot, instead of line-segments, as in the previous SPACELN example. Vertical lines are dropped from each data sample, by setting the parameter ISKIP (3^{rd} to last) to 1. These vertical lines may be suppressed by setting ISKIP to 0.

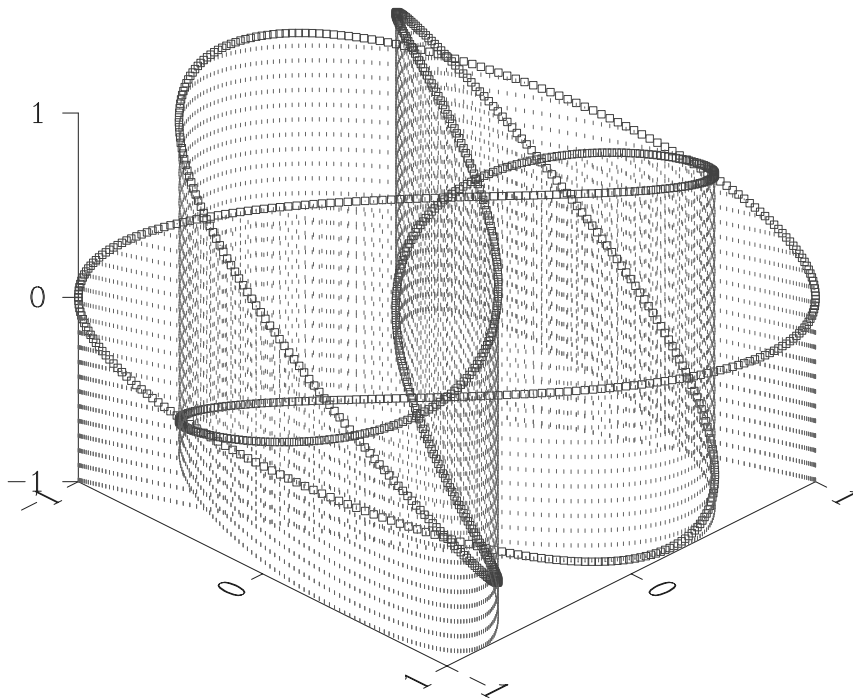
```

C#####
PROGRAM XSPC2
C *** GRAPHER EXAMPLE : SPACE-LINE PLOT
C#####
PARAMETER(NPTS=1000,PI=3.1415926535)
DIMENSION X(NPTS),Y(NPTS),Z(NPTS)
CHARACTER*80 MTIT,STIT,XTIT,YTIT,ZTIT
DATA TMIN,TMAX/-PI,PI/
DATA XMIN,XMAX,YMIN,YMAX,ZMIN,ZMAX/-1.,1.,-1.,1.,-1.,1./
DATA MTIT/'!3Space-Lissajous pattern'/
DATA STIT/'!1x=cos(5t) y=cos(3t) z=sin(3t)'/
C-----
C *** GENERATE DATA
DO 1 I=1,NPTS
T=TMIN+(TMAX-TMIN)*REAL(I-1)/REAL(NPTS-1)
X(I)=COS(5.*T)
Y(I)=COS(3.*T)
Z(I)=SIN(3.*T)
1 CONTINUE
C *** PLOT
CALL PLOTN(0.,0.,15,5) ! DEFINE GRAPHICS DEVICE
CALL SPACELN(X,Y,Z,NPTS,XMIN,XMAX,YMIN,YMAX,ZMIN,ZMAX,
+ XTIT,YTIT,ZTIT,MTIT,STIT,'S',1,30.,1)
END

```

Space-Lissajous pattern
 $x=\cos(5t)$ $y=\cos(3t)$ $z=\sin(3t)$

Sep- 2-1997 15:50:35



2.3.24 PLTCHR Example: Rotation and Font Control

This example illustrates the use of PLTCHR to label strings containing FIDDLER control characters to change the font. The string S is labeled at the mid-screen location $X, Y = (5., 5.)$ in plot units, with height .2, and no slant. The font is selected by the use of the control characters !, & , and \$. There are 4 font families: simplex, complex, duplex, and triplex, each containing 3 fonts: n (normal), g (greek or gothic), and s (script or italic). The 12 font representations of the capital alphabet string s are rotated at 30 degrees intervals.

```

C#####
PROGRAM XPLC1
C *** PLTCHR: PLOT ROTATED ALPHABET IN DIFFERENT FONTS
C#####
CHARACTER S*28,CFONT(0:3)*1,CFACE(0:2)*1
DATA S/' ABCDEFGHIJKLMNOPQRSTUVWXYZ'/
DATA CFONT/'0','1','2','3'/
DATA CFACE/'!','&','$'/
C-----
CALL PLOTN(0.,0.,15,5) ! DEFINE GRAPHICS DEVICE
CALL PLOTN(0.,0.,0,0) ! INITIALIZE GRAPHICS DEVICE
ROTATE=0.
DO 1 I=0,3           ! FOR EACH FONT
DO 1 J=0,2           ! FOR EACH FACE
  CALL PLTCHR(5.,5.,.2,ROTATE,0.,2,' ' //CFONT(I)//CFACE(J)//S,-31)
  ROTATE=ROTATE+30.
1 CONTINUE
CALL PLOTN(0.,0.,16,0) ! EXIT GRAPHICS
END

```


2.3.25 PLTCHR Example: Mathematical Expressions

The program example below simply labels strings S(*) defined in the data statement. These strings contain many FIDDLER control characters, which change fonts, shift characters up and down, and overprint. The first PLOTN call defines the graphics device, and the second initializes the device for plotting. All of the N-Dimensional routines perform this step before beginning to plot.

```

C#####
  PROGRAM XPLC2
C *** PLTCHR: CONTROL CHARACTERS
C#####
  PARAMETER(NS=2)
  CHARACTER*80 S(NS)
  DATA (S(I),I=1,NS)/
+ 'e^x/y_ + & ;&#_3^6__@% [x+&a&y]',
+ '&p&r^2_/_ [36&[&%2^!__% + 4]^ = %t^!_'/
C-----
  CALL PLOTN(0.,0.,15,5) ! DEFINE GRAPHICS DEVICE
  CALL PLOTN(0.,0.,0,0) ! INITIALIZE GRAPHICS DEVICE
  CALL PLOTN(0.,0.,7,3) ! SELECT PEN COLOR {DEFAULT IS 1}
  DO 1 I=1,NS           ! FOR EACH STRING
1  CALL PLTCHR(.5,REAL(I)*2.,.4,0.,0.,2,S(I),-LEN(S(I)))
  CALL PLOTN(0.,0.,16,0) ! EXIT GRAPHICS
  END

```

$$\pi r^2 / [36\sqrt{2} + 4] = \bar{t}$$

$$e^{x/y} + \sum_3^6 [x + \alpha y]$$

Chapter 3

FIDDLER : Interactive program

3.1 Introduction

3.1.1 Brief Description of FIDDLER Interactive program Capabilities

3.1.2 Global Details and Conventions in FIDDLER

FIDDLER uses the subroutines XYPlot to generate a plot. The plotting package has many parameters which can be changed to modify the appearance of the graphics output. FIDDLER allows the user to enter these changes via commands entered from menus. The menus are organized in a hierarchy which reflects the data entry and plot generation process: Input and output functions are performed from the main menu. Editing, Changing, and Labeling are performed using commands from sub-menus of the main menu. For "hot-key" implementations (Cyber/NOS/VE and VAX/VMS) the commands are all initiated by simply typing a key on the terminal, with no necessary carriage return following the command. Other implementations require a carriage return following the typed character. Menu commands are presented in mnemonic format: C means "Change" when it is issued from the main menu, for example. Commands may often be typed ahead, so that moving into a "deep" sub-menu may be quickly accomplished.¹ For example, the following series of one-keystroke commands initiates a binary operations command (multiplication between two channels):

USER INPUT:	COMMENTS:
E DIT	Go to EDIT sub-menu
O PERATE	Go to OPERATIONS sub-menu
B INARY	Go to BINARY OPERATIONS sub-menu
*	Multiply two channels

where the command letter surrounded by a box is the "hot-key" which is pressed to initiate the particular command.

The files which are generated by FIDDLER are ASCII files, with record lengths not exceeding 80 characters, so that a standard text-editor may be used to edit the files. These include:

- FIDDLER -format files: Files which store FIDDLER -information
- PostScript-format plot files: Encapsulated PostScript plot files which may be incorporated into documents.
- HPGL-format plot files: Files which contain Hewlett-Packard Graphics Language plotting commands.

¹Unfortunately, each menu must be completely painted before the next deeper sub-menu is displayed.

3.2 FIDDLER Menu Organization

The hierarchy of FIDDLER menus is shown in the next section, and each menu is listed and discussed in the following sections.

3.2.1 Summary of FIDDLER menus

The FIDDLER menu structure is shown in the following diagram. Menus are indicated by square brackets surrounding the label, while commands are not bracketed. The menu or command is accessed by the single capitalized letter shown in the diagram. The entries of the menu (or summaries of entries in some cases, indicated by parentheses surrounding the summary) are shown one level deeper than the menu level. One or more of the entries may be (sub-)menus. For example, the [main] menu has the command Load, and menu [Change]. The menus which are displayed by FIDDLER, shown in the following sections, are surrounded by a colored border which incorporates the menu level. For example, the menu [Operate] is surrounded by a border of 2's, indicating menu level 2. A complete command path is indicated by [menu1.menu2. . .]command. For example, the command path to add two columns of data together is: [main.Edit.Operate.Binary]+

As the diagram shows, most menus and commands are accessed by a mnemonic character which begins or is part of the menu or command keyword. Thus changing the plotting window lower x-limit to 1.5 takes the form of issuing the "sentence":

Change Window Left (to) 1.5

These "sentences" may be written to a script file (one keyword or answer to a line) which may be executed by issuing [main]Batch followed by the name of the script file. FIDDLER has one comand-line parameter which is the name of a script file. If FIDDLER is called with this parameter, the script file is first executed, then the program returns to interactive mode.

```

=====
MENU LEVEL:
  0      1      2      3      4
=====
[main]__Help
  |__Plot
  |__Save
  |__Load
  |__Data
  |__Terminal
  |__eXecute
  |__Batch
  |
  |_[Change]_____View
  |      |__Name
  |      |__Titles
  |      |_[Channels]_____(x,y,z channels)
  |      |_[Window]_____(plotting limits)
  |      |_[Symbols]_____(line-types,symbols,pens)
  |      |_[Orientation]_(axes-orientation)
  |      |_[Pen-control]_(secondary sweep control)
  |      |_[Labels]_____(label sizes,format)
  |      |_[Mode]_____(plotting mode)
  |      |_[Extra]_____(specific plotting parameters)
  |
  |_[Edit]_____Insert
  |      |__Delete
  |      |__3-convert
  |      |__Sort
  |      |__Linear-regression
  |      |__Polynomial-regression
  |      |_[View]_____(spreadsheet commands)
  |      |_[Operate]_____Endpoints
  |      |      |__eXchange
  |      |      |__Index
  |      |      |__Constant
  |      |      |_[Unary]_____(unary operations)
  |      |      |_[Binary]_____(binary operations)
  |      |      |_[Konstant]___(channel/constant operations)
  |
  |_[Import]_____Ascii
  |      |__Suprem2
  |      |__supreM3
  |      |__Iv
  |      |__1-d
  |      |__3-d
  |      |__simPar
  |      |__pRism
  |
  |_[Function]___Help
  |      |__View
  |      |__T-limits

```

```
|      |__Enter
|      |__Insert
|      |__Delete
|      |__Use
|      |__Reset
|      |__Load
|      |__Save
|      |__Go
|      |__Plot
|[Make]-----Hpgl
|      |__Post
|      |__Ln03
|      |__Calc
|      |__Ascii
|[Add]-----Help
|      |__View
|      |__Enter
|      |__Insert
|      |__Delete
|      |__Use
|      |__Kill
|      |__Place
|[User]-----[A-demo]
|      |__B-calculator
|      |__C-calendar
|      |__[D-draw]
|      |__[E-poster]
```

3.2.2 [MAIN]: the main FIDDLER menu

The main FIDDLER menu provides entries for input/output or creation of data, plotting (screen previewing and hard-copy plots), label editing, and other options. Sub-menus of MAIN are: [CHANGE] (change plotting parameter values), [EDIT] (view/alter/enter data), [IMPORT] (load non-FIDDLER-format files), [FUNCTION] (sample a one-parameter function), [MAKE] (make a hard-copy of the FIDDLER plot, or printout data), [ADD] (add and place labels on the current plot), and [USER] (several application programs).

```
#####
#0#  [MAIN] FIDDLER main menu:      #0#
#0#  [.]: ....                      #0#
#0#  [H]: Help                      #0#
#0#  [P]: Plot                      #0#
#0#  [S]: Save fiddler data file    #0#
#0#  [L]: Load fiddler data file    #0#
#0#  [I]> Import other data file    #0#
#0#  [D]: Data point entry         #0#
#0#  [F]> Function sample           #0#
#0#  [C]> Change parameters         #0#
#0#  [E]> Edit data                 #0#
#0#  [M]> Make hard copy            #0#
#0#  [A]> Add labels                #0#
#0#  [T]: Terminal definition       #0#
#0#  [X]: eXecute system command    #0#
#0#  [B]: Batch file               #0#
#0#  [U]> User programs             #0#
#0#  [Q]: Quit [MAIN]              #0#
#####
```

H elp	Displays the help file HMAIN.HLP one page at a time.
P lot	Plot the current plot-view.
S ave	Save the current FIDDLER -information: (data arrays, channel names, titles, and labels) in FIDDLER -format data file. This is a formatted ASCII file, with record length < 80 characters.
L oad	Load a FIDDLER -format data file. The data from the specified file may either replace or be appended to the current FIDDLER -information, (data/titles/names/labels), which has already been loaded or created.
I mport	Entry to the [IMPORT] sub-menu. Various file formats may be imported, including ASCII files with columns of data separated by commas or white-space.
D ata	Enter data points, value by value. Titles and channel names are specified before entering the data values.
F unction	Sample a user specified function. Function lines of a FORTRAN program are entered interactively, limits for sampling are chosen, and the data is generated.
C hange	Entry to the [CHANGE] sub-menu: change plotting parameters.
E dit	Entry to the [EDIT] sub-menu: view/edit data values, or operate on data channels.
M ake	Entry to the [MAKE] sub-menu: make hard-copy of plot-view.
A dd	Entry to the [ADD] sub-menu: add labels and place on current plot-view.
T erminal	Specify graphics device.
X EQ	Execute a system-level command.
B atch	Perform the commands in a script file. For "hot-key" commands, only the first character of the line is significant. After executing the commands, the user can then use FIDDLER interactively (unless the commands 'Quit' and 'Yes' are executed from the main menu.)
U ser	Entry to the [USER] sub-menu: application programs.
Q uit	Quit FIDDLER .

3.2.3 [CHANGE]: change plotting parameters

The [CHANGE] sub-menu provides entries for changing the parameters of the current plot-setup: channel numbers, LIN/LOG mode, plotting window, symbols, line-types, pen-colors. The channel names and titles may be changed. The sub-menus of CHANGE are: [CHANGE.CHANNELS] (change the selected channels), [CHANGE.MODE] (change the plotting mode to surface, x-y plot, ...), [CHANGE.WINDOW] (change the plotting window), [CHANGE.ORIENTATION] (change the plot labeling orientation), [CHANGE.SYMBOLS] (change the plotting symbols, line-types, and pen-colors), [CHANGE.PEN-CONTROL] (turn on pen-control option: channel with name "PEN_CONTROL" indicates a MOVE or DRAW in line-segment curve plots), [CHANGE.LABELING] (change sizes of axes numbers, titles, spacings), [CHANGE.EXTRA] (change parameters specific to certain plotting routines).

```
#####
#1# [CHANGE] #1#
#1# [C]> Channels #1#
#1# [M]> plotting Mode #1#
#1# [W]> plotting Window #1#
#1# [O]> axes Orientation #1#
#1# [S]> Symbol,line,pen #1#
#1# [P]> Pen-control #1#
#1# [L]> Labeling options #1#
#1# [E]> Extra parameters #1#
#1# [N]: channel Name #1#
#1# [T]: Title/subtitle #1#
#1# [V]: View parameter values #1#
#1# [Q]: Quit [CHANGE] #1#
#####
```

Channels

Entry to the [CHANGE.CHANNELS] sub-menu: change or turn on the plot-setup curve parameters. There are six available plot-setup curves. After data is loaded in, the first curve only is turned on, with X-channel = channel 1, and Y-channel = channel 2. These can be changed to appropriate values.

Mode

Change plotting mode to X-Y plot, Surface, etc.

Window

Entry to the [CHANGE.WINDOW] sub-menu: alter current plot-setup window.

Orientation

Entry to the [CHANGE.ORIENTATION] sub-menu: change axes labelling directions.

Symbols

Entry to the [CHANGE.SYMBOLS] sub-menu: change the plot-setup symbols, line-types, and pen-colors.

Pen control

Entry to the [CHANGE.PEN-CONTROL] sub-menu: turn pen-control option on or off.

Labeling

Entry to the [CHANGE.LABELING] sub-menu: change axes number and title sizes, and other options.

Extra pars

Change various other settings.

Name

Change one channel name at a time.

Title

Change main and sub titles.

View

Display parameter values of current plot-setup and statistics of current FIDDLER -information.

Quit

Quit [CHANGE]: Return to [MAIN] menu.

3.2.4 [CHANGE.CHANNELS]: change x,y,z channels

The [CHANGE.CHANNELS] menu provides commands for associating (for X-Y plots) one x-axis channel and one y-axis channel with one of up to ten curves. The first data curve is always plotted, but the curves [2]-[10] are disabled, unless one of the commands [2]-[10] are used to associate x- and y-axis channels with the particular curve.

```
#####
#2#  [CHANGE.CHANNELS]                #2#
#2#  [N]:  list channel Names         #2#
#2#  [V]:  View current channels      #2#
#2#  [R]:  Reset                       #2#
#2#  [1]:  curve 1                     #2#
#2#  [2]:  curve 2                     #2#
#2#  [3]:  curve 3                     #2#
#2#  [4]:  curve 4                     #2#
#2#  [5]:  curve 5                     #2#
#2#  [6]:  curve 6                     #2#
#2#  [7]:  curve 7                     #2#
#2#  [8]:  curve 8                     #2#
#2#  [9]:  curve 9                     #2#
#2#  [0]:  curve 10                   #2#
#2#  [Q]:  Quit [CHANNELS]            #2#
#####
```

N ames	Display the channel names associated with all of the FIDDLER channels.
V iew	Display the current settings for the x-axis and y-axis channels of all of the data-curves.
R eset	Reset the channel settings to disable curves [2]-[6]. Each may be re-enabled by one of the commands [2]-[6].
1	Specify the x-axis and the y-axis channels for data-curve [1].
2	Specify the x-axis and the y-axis channels for data-curve [2].
3	Specify the x-axis and the y-axis channels for data-curve [3].
4	Specify the x-axis and the y-axis channels for data-curve [4].
5	Specify the x-axis and the y-axis channels for data-curve [5].
6	Specify the x-axis and the y-axis channels for data-curve [6].
7	Specify the x-axis and the y-axis channels for data-curve [7].
8	Specify the x-axis and the y-axis channels for data-curve [8].
9	Specify the x-axis and the y-axis channels for data-curve [9].
0	Specify the x-axis and the y-axis channels for data-curve [10].
Q uit	Quit [CHANGE.CHANNNELS]: Return to [CHANGE] menu.

3.2.5 [CHANGE.MODE]: change plotting mode

The [CHANGE.MODE] menu provides commands for changing the plotting mode. The various plotting routines make use of several different data formats:

- 1COL** one data column (1COL.X):
x-values.
- 2COL** two data columns with the same number of rows (2COL.X,2COL.Y):
x- and y-values.
- 2PEN** three data columns with the same number of rows (2PEN.X,2PEN.Y,2PEN.P):
x-, y- and PEN_CONTROL - values.
- 3COL** three data columns with the same number of rows (3COL.X,3COL.Y,3COL.Z):
x-, y- and z-values.
- 3PEN** four data columns with the same number of rows (3PEN.X,3PEN.Y,3PEN.Z,3PEN.P):
x-, y-, z- and PEN_CONTROL -values.
- 3DIM** one column with NX rows, one column with NY rows, and NX columns with NY rows (3DIM.X,3DIM.Y,3DIM.Z):
x-axis, y-axis and z(x,y) values.

```
#####
#2# [CHANGE.MODE] #2#
#2# [X]: X-Y plot #2#
#2# [E]: X-Y plot with Error bars #2#
#2# [3]: space-line plot #2#
#2# [S]: Surface plot #2#
#2# [I]: surface plot-Isometric #2#
#2# [C]: Contour plot #2#
#2# [L]: Labelled contour plot #2#
#2# [O]: Color-band contour plot #2#
#2# [P]: Probability plot #2#
#2# [H]: Histogram plot #2#
#2# [V]: Vector plot #2#
#2# [B]: Blank plot #2#
#2# [Q]: Quit [MODE] #2#
#####
```

X Yplot	X-Y plot. Uses 2COL-format (x,y) or 2PEN-format (x,y,pen-control). Calls XYPLOT routine.
E rror	X-Y plot with error bars. Uses 3COL-format (x,y,error) or 3PEN-format (x,y,error,pen-control). The points (x,y) will be plotted with an error bar extending from $y - error$ to $y + error$. Calls XYPLOTE routine.
3 space-line	Space-Line plot. Uses 3COL-format (x,y,z) or 3PEN-format (x,y,z,pen-control). Calls SPACELN routine.
S urface	Surface plot. Uses 3DIM-format (x,y,z). The values may be interpolated onto a uniform grid, as needed. The plot may be tilted and rotated, and skirts may be added. In addition, oblique plots may be selected by the command [CHANGE.EXTRA]Mode. Calls SURFAC2 routine.
I sometric	Isometric surface plot. Uses 3DIM-format (x,y,z). The values are interpolated onto a uniform grid. The plot may be tilted. Calls SURFACE routine.
C ontour	Contour plot. Uses 3DIM-format (x,y,z). Calls CONTOUR routine.
L abelled	Labelled Contour plot. Uses 3DIM-format (x,y,z). Contour lines are interrupted and labelled with the contour value. Calls CONTOU2 routine.
O color band	Color-Band Contour plot. Uses 3DIM-format (x,y,z). Contour bands are plotted in various colors. The color scale is labelled at the right hand side of the plot. Calls CONTOU3 routine.
P robability	Probability plot. Uses 1COL-format (samples). Calls PROBPLT routine.
H istogram	Histogram plot. Uses 1COL-format (samples). Calls HISTOGM routine.
V ector	Vector plot. This option is not currently supported.
B lank	Blank plot. Plots labels only.
Q uit	Quit [CHANGE.MODE]: Return to [CHANGE] menu.

3.2.6 [CHANGE.WINDOW]: change plotting window

The [CHANGE.WINDOW] menu provides commands for changing the selected plotting window. Values for each of the four boundaries may be entered separately or all together. Autoscaling chooses the maximum and minimum of both the horizontal and vertical channel as the plotting window. Central view chooses a view centered about 3 standard deviations above and below the means of both horizontal and vertical channels.

```
#####
#2# [CHANGE.WINDOW]                #2#
#2# [L]: Left (xlo)                 #2#
#2# [R]: Right (xhi)                #2#
#2# [B]: Bottom (ylo)               #2#
#2# [T]: Top (yhi)                  #2#
#2# [U]: Under (zlo)                #2#
#2# [O]: Over (zhi)                 #2#
#2# [W]: Window (LRBTUO)            #2#
#2# [A]: Autoscaled view            #2#
#2# [C]: Central view               #2#
#2# [V]: View current window        #2#
#2# [Q]: Quit [WINDOW]              #2#
#####
```

L	left	Enter the left edge of the plotting window.
R	right	Enter the right edge of the plotting window.
B	bottom	Enter the bottom edge of the plotting window.
T	top	Enter the top edge of the plotting window.
U	under	Enter the lower Z-limit for 3-D data.
O	over	Enter the upper Z-limit for 3-D data.
W	window	Enter all limit values.
A	autoscale	Minimum and maximum values of the horizontal and vertical channels are chosen as the plotting window edges.
C	central	Plotting window edges are 3 standard deviations below and above the mean value of the horizontal and vertical channels.
V	view	Display the current settings of the plotting window.
Q	quit	Quit [CHANGE.WINDOW]: Return to [CHANGE] menu.

3.2.7 [CHANGE.ORIENTATION]: change labeling orientation

The [CHANGE.ORIENTATION] menu provides commands for changing the orientation of the x- and y-axes and labels.

```
#####
#2# [CHANGE.ORIENTATION]          #2#
#2# [L]: Y-axis at Left           #2#
#2# [B]: X-axis at Bottom         #2#
#2# [R]: Y-axis at Right          #2#
#2# [T]: X-axis at Top            #2#
#2# [^]: Y-values ascending       #2#
#2# [V]: Y-values descending      #2#
#2# [>]: X-values ascending        #2#
#2# [<]: X-values descending        #2#
#2# [Q]: Quit [ORIENTATION]      #2#
#####
```

L eft	Place Y-axis numbers and title on left side of plot.
R ight	Place Y-axis numbers and title on right side of plot.
B ottom	Place X-axis numbers and title on bottom of plot.
T op	Place X-axis numbers and title on top of plot.
^	Y-axis numbers go from smaller to larger values, from bottom to top of axis.
V	Y-axis numbers go from smaller to larger values, from top to bottom of axis.
>	X-axis numbers go from smaller to larger values, from left to right of axis.
<	X-axis numbers go from smaller to larger values, from right to left of axis.
Q uit	Quit [CHANGE.ORIENTATION]: Return to [CHANGE] menu.

3.2.8 [CHANGE.SYMBOLS]: change plotting symbols, pen-colors, line-types

The [CHANGE.SYMBOLS] menu provides commands for changing the plotting symbols, pen-colors or line-types for each curve. Use the [I]ndex command to choose one of the curves, then one of the other commands to change the attributes of the curve.

```
#####
#2# [CHANGE.SYMBOLS] #2#
#2# [I]: curve Index 1..9,0=10 #2#
#2# [L]: Line #2#
#2# [S]: Square #2#
#2# [T]: Triangle #2#
#2# [O]: Other symbol #2#
#2# [%]: line plus other symbol #2#
#2# [P]: Pen color index #2#
#2# [V]: View current settings #2#
#2# [H]: Help: show lines,pens #2#
#2# [Q]: Quit [SYMBOLS] #2#
#####
```

I ndex	Specify the number of the data-line (y-vector) which is referred to by the other commands in the SYMBOL menu.
L ine	Specify that the data-line is to be labeled with a series of line-segments connecting the points of the data-line. A line-type is also specified.
S quare	Specify that the data-line is to be labeled with a square at each point. The size of the squares may be changed by the commands [CHANGE.LABELING]Height and [CHANGE.LABELING]Width.
T riangle	Specify that the data-line is to be labeled with a triangle at each point. The size of the triangles may be changed by the commands [CHANGE.LABELING]Height and [CHANGE.LABELING]Width.
O ther	Specify that the data-line is to be labeled with any other character besides a square or a triangle. The size of the symbols may be changed by the command [CHANGE.LABELING]Character.
% line/symbol	Specify that the data-line is to be labeled with both symbols at each point and line segments between consecutive points. The symbol and line-type are both specified.
P en	Specify the pen-color for labeling the data-line.
V iew	Display the current settings of each data-line for pen-color, symbol, and line-type.
H elp	Display a graphic of the various pen-color and line-type options.
Q uit	Quit [CHANGE.SYMBOLS]: Return to [CHANGE] menu.

3.2.9 [CHANGE.PEN-CONTROL]: change pen-control parameters

The [CHANGE.PEN-CONTROL] menu provides commands for turning the pen-control option on or off, and changing the lower and upper sweep limits. The pen-control option allows multiple-sweep plots to be displayed on the same plot, using a continuous line for each sweep, and a discontinuous step between each sweep. The FIDDLER -information must include a channel labeled (exactly) "PEN_CONTROL ". This channel consists of ones and zeros. Zeros mark the positions of the first points in the various sweeps. Using the pen-control option allows more than ten data curves to be displayed on the same plot, since only one x-channel and one y-channel are selected (one data curve). If a FIDDLER -format file is loaded in which contains one channel with the title "PEN_CONTROL ", then the pen-control option is automatically turned on.

```
#####
#2# [CHANGE.PEN-CONTROL]          #2#
#2# [Y]: turn on pen-control      #2#
#2# [N]: turn off pen-control     #2#
#2# [I]: pen-control channel      #2#
#2# [L]: Lowest secondary step    #2#
#2# [H]: Highest secondary step   #2#
#2# [Q]: Quit [PEN-CONTROL]      #2#
#####
```

Y	es	Turn pen-control ON.
N	o	Turn pen-control OFF.
I	ndex	Select the column index of the pen-control channel.
L	ow	Specify lowest secondary step.
H	igh	Specify highest secondary step.
Q	uit	Quit [CHANGE.PEN-CONTROL]: Return to [CHANGE] menu.

3.2.10 [CHANGE.LABELING]: change labeling parameters

The [CHANGE.LABELING] menu provides commands for changing the size of the axes labels and numbers, title sizes, and several other plotting parameters.

```
#####
#2# [CHANGE.LABELING] #2#
#2# [N]: axes Number size #2#
#2# [A]: Axes title size #2#
#2# [M]: Main title size #2#
#2# [S]: Sub title size #2#
#2# [D]: Date title size #2#
#2# [T]: axes Tic length #2#
#2# [L]: Length between axes,tit #2#
#2# [H]: Height of box symbol #2#
#2# [W]: Width of box symbol #2#
#2# [C]: size of Char symbol #2#
#2# [X]: X-axis number orient. #2#
#2# [B]: number of Border lines #2#
#2# [F]: axes number Format #2#
#2# [V]: axes number font #2#
#2# [P]: any Plotn command #2#
#2# [R]: Restore intial options #2#
#2# [Q]: Quit [LABELING] #2#
#####
```

N	Number size	Specify relative size for labeling axes numbers.
A	axes	Specify relative size for labeling axes labels.
M	Main	Specify relative size for main title.
S	Sub	Specify relative size for sub title.
D	Date	Specify relative size for the date label.
T	Tic	Specify relative tic length.
L	Length	Specify relative distance between axes and axes labels.
H	Height	Specify relative height of box symbols (squares or triangles).
W	Width	Specify relative width of box symbols (squares or triangles).
C	Char	Specify relative size of character symbols (anything except squares or triangles).
X	X-axis	Specify orientation of numbers on x-axis: 0 degrees or 90 degrees.
B	Border	Specify the number of lines drawn in the border surrounding the plot. One line is the default. Specifying several lines produces a thick border.
F	Format	Set the formats used in labeling axes.
V	font	Set the font used in labeling axes.
P	Plotn	Call PLOTN (See FIDDLER plotting-package section).
R	Restore	Restore the options to default values: as set when FIDDLER is initiated.
Q	Quit	Quit [CHANGE.LABELING]: Return to [CHANGE] menu.

3.2.11 [CHANGE.EXTRA]: change routine-specific parameters

The [CHANGE.EXTRA] menu provides commands for changing parameters which are specific to particular plotting modes (plotting-package routines).

```
#####
#2#  [CHANGE.EXTRA]                #2#
#2#  [A]:  3-D Elevation Angle     #2#
#2#  [R]:  3-D Rotation  Angle     #2#
#2#  [O]:  3-D Octant              #2#
#2#  [P]:  Surface Opacity         #2#
#2#  [M]:  Surface Mode            #2#
#2#  [S]:  Surface Mesh size       #2#
#2#  [C]:  # contours              #2#
#2#  [I]:  # spaceline pts. to skip #2#
#2#  [D]:  Histogram delta-x       #2#
#2#  [G]:  Grid on/off            #2#
#2#  [L]:  Color-band scale on/off #2#
#2#  [Q]:  Quit [EXTRA]           #2#
#####
```

A ngle	Change 3-D elevation angle (ANGLE parameter in SPACELN, SURFACE, and SURFAC2.)
R otation	Change 3-D rotation angle (PHI parameter in SURFAC2.)
O ctant	Change 3-D octant (IORIENT parameter in SPACELN, SURFACE and SURFAC2.)
P opacity	Select hidden-line or transparent surface mode (SURFACE, SURFAC2.)
M ode	Select surface mode: plot top or bottom, isometric or oblique, skirts or no skirts (SURFACE, SURFAC2.)
S urface	Number of divisions for uniform mesh when interpolating surface (SURFACE, SURFAC2.)
C ontours	Number of contours (CONTOUR, CONTOU2, CONTOU3.)
I skip	Number of points to skip between verticals on space-line plot (ISKIP parameter in SPACELN.)
D x	Size of counting interval in histogram plots (DX parameter in HISTOGM.)
G rid	Turn grid on or off (XYPLOT, XYPLOTE, CONTOUR, CONTOU2, CONTOU3.)
L color scale	Keep or remove color scale from right side of color-band contour plots.
Q uit	Quit [CHANGE.EXTRA]: Return to [CHANGE] menu.

3.2.12 [EDIT]: edit data array

The [EDIT] menu has entries for viewing and altering the FIDDLER -information (data arrays, channel-names, and titles). The sub-menus of [EDIT] are: [EDIT.VIEW] (spread-sheet of FIDDLER -information), and [EDIT.OPERATE] (operate on data channels).

```
#####
#1# [EDIT] #1#
#1# [V]> View data arrays #1#
#1# [O]> Operate on data arrays #1#
#1# [S]: Sort by channel values #1#
#1# [I]: Insert row or channel #1#
#1# [D]: Delete row or channel #1#
#1# [L]: Linear-regression #1#
#1# [P]: Polynomial-regression #1#
#1# [3]: convert 3COL to 3DIM #1#
#1# [Q]: Quit [EDIT] #1#
#####
```

View

Entry to the [EDIT.VIEW] (background) sub-menu: view data channels. On entry, a spread-sheet of the FIDDLER data is displayed, and the menu is not displayed. If any key is pressed which is not a recognized [EDIT.VIEW] command, the menu is displayed.

Operate

Entry to the [EDIT.OPERATE] sub-menu: operations on and between data channels. For example, channel 1 may be multiplied by channel 3, with the result put into channel 8.

Sort

Choose One or more channels to use as the basis of ordering the entire data set. For example, values in channel 8 may originally appear in the data set in random order, with repetitions of certain values. The data, once reordered with channel 8 with highest ordering priority, and channel 9 with next highest ordering priority, will result in channel 8 values increasing (or decreasing) monotonically. On repeated channel 8 values, channel 9 values will be ordered monotonically increasing (or decreasing).

Insert

Insert row(s) or column(s), filling the new row with a specified constant.

Delete

Delete a range of rows or columns.

Linear

Perform a linear regression of the data in two specified channels. The regression line is placed in a new channel (at the end of the data set) with channel name REGRESSION_LINE, and regression statistics are printed out.

Polynomial

Perform a polynomial regression of the data in two specified channels to a polynomial of specified degree. The regression curve is placed in a new channel (at the end of the data set) with channel name REGRESSION_LINE, and regression statistics are printed out.

3 convert

Convert (Xi,Yi,Zi) 3-column data format to (Xi),(Yj),(Zij) 3-D format.

Quit

Quit [EDIT] : Return to [MAIN] menu.

3.2.13 [EDIT.VIEW]: spread-sheet of data array

The [EDIT.VIEW] menu provides commands for moving around the spread-sheet view of the FIDDLER-information. Four columns and 15 rows of the data arrays are displayed on the screen. In addition, channel names, main title and sub-title can be altered. Values can also be edited in the data array (fiddling with the data!)

```
#####
#2#  [EDIT.VIEW]                #2#
#2#  [U]:  Up      15 rows      #2#
#2#  [D]:  Down   15 rows      #2#
#2#  [L]:  Left    4 columns    #2#
#2#  [R]:  Right   4 columns    #2#
#2#  [T]:  Top     row          #2#
#2#  [B]:  Bottom  row          #2#
#2#  [H]:  Home    column       #2#
#2#  [E]:  End     column       #2#
#2#  [P]:  Point   goto row     #2#
#2#  [C]:  Channel goto column  #2#
#2#  [V]:  Values  change cvalues #2#
#2#  [N]:  Name    change cname  #2#
#2#  [M]:  Main    change main tit #2#
#2#  [S]:  Sub     change sub tit #2#
#2#  [Q]:  Quit    [VIEW]       #2#
#####
```

U	p	Move up 15 rows in the spreadsheet (if possible).
D	own	Move down 15 rows in the spreadsheet (if possible).
L	eft	Move left 4 columns in the spreadsheet (if possible).
R	ight	Move right 4 columns in the spreadsheet (if possible).
T	op	Go to the top row of the spreadsheet.
B	ottom	Go to the bottom row of the spreadsheet.
H	ome	Go to the home column of the spreadsheet.
E	nd	Go to the end column of the spreadsheet.
P	oint	Go to a specified point (row) in the spreadsheet.
C	hannel	Go to a specified channel in the spreadsheet.
V	alues	Specify the beginning place (row and column) to begin modifying the existing data. Values are entered into one channel only.
N	ame	Change a specified channel name.
M	ain	Change the main title.
S	ub	Change the sub title.
Q	uit	Quit [EDIT.VIEW]: Return to [EDIT] menu.

3.2.14 [EDIT.OPERATE]: operate on data array

The [EDIT.OPERATE] menu provides commands for selecting the type of operation which is to be performed on one or two data channels to produce another data channel, (or modify an existing one). The sub-menus of [EDIT.OPERATE] are: [EDIT.OPERATE.UNARY] (unary operations), [EDIT.OPERATE.BINARY] (binary operations), and [EDIT.OPERATE.KONSTANT] (operations between a channel and a constant).

```
#####
#2# [EDIT.OPERATE] #2#
#2# [E]: low, high Endpoints #2#
#2# [C]: Constant Z=const #2#
#2# [U]> Unary Z=f(X) #2#
#2# [B]> Binary Z=f(X,Y) #2#
#2# [K]> channel,K Z=f(X,K) #2#
#2# [X]: eXchange Z<=>X #2#
#2# [P]: Point index Z=index #2#
#2# [Q]: Quit [OPERATE] #2#
#####
```

E ndpoints	Specify low and high points for the operation.
C onstant	Set one channel equal to a constant.
U nary	Unary operations, $Z=f(X)$.
B inary	Binary operations, $Z=f(X,Y)$.
K onstant	Operations involving one channel and a constant, $Z=f(X,k)$.
X change	Exchange one channel with another.
P oint	Place the row (point) index into a channel.
Q uit	Quit [EDIT.OPERATE]: Return to [EDIT] menu.

3.2.15 [EDIT.OPERATE.UNARY]: unary operations

The [EDIT.OPERATE.UNARY] menu provides access to operations which are performed on one channel to produce another channel. The X channel and the Z channel may be identical. Once the operation has been performed, the channel title is changed to indicate the type of operation which has been performed.

```
#####
#3# [EDIT.OPERATE.UNARY]          #3#
#3# [A]: Z=ABS(X)                 #3#
#3# [S]: Z=SIN(X)                 #3#
#3# [C]: Z=COS(X)                 #3#
#3# [T]: Z=TAN(X)                 #3#
#3# [E]: Z=EXP(X)                 #3#
#3# [L]: Z= LN(X)                 #3#
#3# [G]: Z=LOG10(X)               #3#
#3# [I]: Z=INT(X)                 #3#
#3# [1]: Z=1/X                    #3#
#3# [2]: Z=X^2                    #3#
#3# [3]: Z=X^3                    #3#
#3# [R]: Z=SQRT(X)               #3#
#3# [-]: Z=-X                     #3#
#3# [+]: Z=+X                     #3#
#3# [$]: Z=SGN(X)                 #3#
#3# [~]: Z=NOT(X)                 #3#
#3# [#]: Z=SUMi(X)                #3#
#3# [P]: Z=PRODi(X)              #3#
#3# [=]: Zi=(Xi.EQ.Xi-1)         #3#
#3# [Q]: Quit [UNARY]            #3#
#####
```

A	BS	Absolute value.
S	IN	Sine.
C	OS	Cosine.
T	AN	Tangent.
E	XP	Exponential Function.
L	N	Natural Logarithm.
G	LGT	Common Logarithm.
I	NT	Integer Value.
1	1/X	Inverse.
2	X^ 2	Square.
3	X^ 3	Cube.
R	OOT	Square-root.
-		Negative.
+		Positive.
\$	SGN	SGN function: -1 if $X < 0$; 0 if $X = 0$; $+1$ if $X > 0$.
~	NOT	NOT function: $+1$ if $X = 0$; 0 if $X <> 0$.
#	SUM	SUM: $Z(i) = X(i) + X(i - 1) + \dots + X(1)$.
P	ROD	PRODUCT: $Z(i) = X(i) * X(i - 1) * \dots * X(1)$.
=		1 if $X(i) = X(i - 1)$; 0 otherwise. Useful for creating a PEN_CONTROL channel.
Q	uit	Quit [EDIT.OPERATE.UNARY]: Return to [EDIT.OPERATE] menu.

3.2.16 [EDIT.OPERATE.BINARY]: binary operations

The [EDIT.OPERATE.BINARY] menu provides access to operations which are performed on two channels to produce another channel. Any of the X, Y or Z channels may be identical. Once the operation has been performed, the channel title is changed to indicate the type of operation which has been performed.

```
#####
#3# [EDIT.OPERATE.BINARY]          #3#
#3# [+]: Z=Y+X                     #3#
#3# [-]: Z=Y-X                     #3#
#3# [*]: Z=Y*X                     #3#
#3# [/]: Z=Y/X                     #3#
#3# [^]: Z=Y^X                     #3#
#3# [1]: Z=dY/dX                   #3#
#3# [2]: Z=d2Y/dX2                 #3#
#3# [I]: Z=integral(Y)dX           #3#
#3# [A]: Z=Y.AND.X                 #3#
#3# [O]: Z=Y.OR.X                  #3#
#3# [X]: Z=Y.XOR.X                 #3#
#3# [Q]: Quit [BINARY]             #3#
#####
```

+	Addition.
-	Subtraction.
*	Multiplication.
/	Division.
^	Exponentiation.
1	First Derivative of Y with respect to X.
2	Second Derivative of Y with respect to X.
I	Integral Integration of Y with respect to X.
A	AND AND Function: 1 if X <> 0 and Y <> 0; 0 otherwise.
O	OR OR Function: 1 if X <> 0 or Y <> 0; 0 otherwise.
X	XOR XOR Function: 1 if (X <> 0 and Y = 0) or (X = 0 and Y <> 0); 0 otherwise.
Q	Quit [EDIT.OPERATE.BINARY]: Return to [EDIT.OPERATE] menu.

3.2.17 [EDIT.OPERATE.KONSTANT]: channel/constant operations

The [EDIT.OPERATE.KONSTANT] menu provides access to operations which are performed on one channel, and using one constant, to produce another channel. The X channel and the Z channel may be identical. Once the operation has been performed, the channel title is changed to indicate the type of operation which has been performed.

```
#####
#3# [EDIT.OPERATE.KONSTANT]      #3#
#3# [M]: Z=MOD(X,const)         #3#
#3# [D]: Z=DIV(X,const)         #3#
#3# [=]: Z=(X=const)           #3#
#3# [<]: Z=(X<const)           #3#
#3# [>]: Z=(X>const)           #3#
#3# [L]: Z=(X<=const)          #3#
#3# [G]: Z=(X>=const)          #3#
#3# [+]: Z=X+const              #3#
#3# [-]: Z=X-const              #3#
#3# [*]: Z=X*const              #3#
#3# [/]: Z=X/const              #3#
#3# [^]: Z=X^const              #3#
#3# [Q]: Quit [KONSTANT]       #3#
#####
```

M	OD	The remainder of the integer division of X by const.
D	IV	The result of the integer division of X by const.
=		1 if $X = constant$; 0 otherwise.
<		1 if $X < constant$; 0 otherwise.
>		1 if $X > constant$; 0 otherwise.
L	E	1 if $X \leq constant$; 0 otherwise.
G	E	1 if $X \geq constant$; 0 otherwise.
+		Addition.
-		Subtraction.
*		Multiplication.
/		Division.
^		Exponentiation.
Q	uit	Quit [EDIT.OPERATE.KONSTANT]: Return to [EDIT.OPERATE] menu.

3.2.18 [IMPORT]: load non-FIDDLER -format data files

The [IMPORT] menu provides commands for entering various formats of data. Example files for importing can be found in the example directory (EXAM).

```
#####
#1# [IMPORT] #1#
#1# [H]: Help #1#
#1# [A]: generic ASCII data file #1#
#1# [S]: SUPREM-2 ASCII/binary #1#
#1# [M]: TMA-SUPREM-3 #1#
#1# [I]: MINIMOS4 IV data file #1#
#1# [1]: GE-PLOT 1 data file #1#
#1# [3]: GE-PLOT 3 data file #1#
#1# [P]: SIMPAR input file #1#
#1# [R]: PRISM summary file #1#
#1# [Q]: Quit [IMPORT] #1#
#####
```

H	elp	Display HIMPT.HLP help file, one page at a time. This describes the format of the ASCII file.
A	SCII	Import ASCII free-format data.
S	UPREM-2	Import SUPREM-2 SAVE file (ASCII or binary).
M	TMA SUPREM-3	Import SUPREM-3 SAVE file.
I	V	Import MINIMOS I-V ASCII save file
1	GEPLOT1	Import GE-PLOT 1 format data file.
3	GEPLOT3	Import GE-PLOT 3 format data file.
P	SIMPAR	Import SIMPAR format data file.
R	PRISM	Import PRISM summary file.
Q	uit	Quit [IMPORT]: Return to [MAIN] menu.

3.2.19 [FUNCTION]: sample a one-parameter function

The [FUNCTION] menu provides commands for entering and editing FORTRAN-style program lines for a function which is to be sampled at discrete intervals of the independent variable T. The program lines assign a value to the dependent variable F. The range of sampling, the number of data samples, and the type of sampling, linear or logarithmic, are specified. Once the program lines, and sampling range values are set, the command [G]o initiates compilation of a short FORTRAN program built around the program lines and sampling range values. The FORTRAN program is run, and produces function samples, which are loaded in as the new (or appended, if specified) FIDDLER -information.

```
#####
#1# [FUNCTION]                #1#
#1# [H]:  Help                #1#
#1# [V]:  View                #1#
#1# [T]:  enter Tmin,Tmax,npts #1#
#1# [E]:  Enter function lines #1#
#1# [I]:  Insert function line #1#
#1# [D]:  Delete function line #1#
#1# [U]:  Use screen text-editor #1#
#1# [R]:  Reset               #1#
#1# [L]:  Load function       #1#
#1# [S]:  Save function        #1#
#1# [G]:  Go                  #1#
#1# [P]:  Plot                #1#
#1# [Q]:  Quit [FUNCTION]     #1#
#####
```

H elp	Display HFUNC.HLP help file, one page at a time.
V iew	List Program lines for function and limits of parameter T.
T	Specify Tmin, Tmax, and NPTS.
E nter	Enter several function lines.
I nsert	Insert one function line before a specified function line number.
D elete	A specific function line.
U se	Use the system-supplied text-editor to edit function lines. The text-editor is defined by the environmental parameter SCEDIT (see the FIDDLER command script fiddler.unx or fiddler.vms).
R eset	Delete all function lines.
L oad	Load a FIDDLER -function-save file.
S ave	Save a FIDDLER -function-save file.
G o	Perform function sampling. Tmin, Tmax, NPTS and all function lines must be correct before performing the sampling.
P lot	Plot the function after it has been generated.
Q uit	Quit [FUNCTION]: Return to [MAIN] menu.

3.2.20 [MAKE]: make a hard-copy plot or printout

The [MAKE] menu provides commands for producing a hard-copy plot of the current FIDDLER -plot or a printout of the data array. The plot files may be saved for later use – downloading or incorporating into documents. The CALC-format is a simple generic format which may be post-processed and sent to other types of plotters.²

```
#####
#1#  [MAKE]                               #1#
#1#  [H]:  HPGL  >file/plotter           #1#
#1#  [P]:  POST  >file/printer           #1#
#1#  [L]:  LN03  >file/printer           #1#
#1#  [C]:  CALC  >file/plotter           #1#
#1#  [A]:  ASCII >file/printer           #1#
#1#  [Q]:  Quit  [MAKE]                   #1#
#####
```

- H** PGL Produce an HPGL-format plot file and send to an HPGL plotter. The system plot command for sending the plot file to the plotter is set by the environmental parameter HPGLPLOT (see the file fiddler.unx or fiddler.vms.) The record length of the file is less than 80 characters, so that a standard text editor may be used to make changes to the file.
- P** OST Produce a PostScript-format file and send to a PostScript printer. The system plot command for sending the plot file to the printer is set by the environmental parameter POSTPLOT (see the file fiddler.unx or fiddler.vms.) The record length of the file is less than 80 characters, so that a standard text editor may be used to make changes to the file.
- L** N03 Produce an LN03-format file and send to a LN03 printer. The system plot command for sending the plot file to the printer is set by the environmental parameter LN03PLOT (see the file fiddler.unx or fiddler.vms.)
- C** ALC Produce a CALC-format plot file, then send it to a color plotter. The system plot command for sending the plot file to the plotter is set by the environmental parameter CALCPLOT (see the file fiddler.unx or fiddler.vms.) Each line of a CALC-format is a one-character command followed by one or two numerical parameters. This generic plot file can easily be converted to other plot formats.
- A** SCII Produce an ASCII file containing all channels of data written out in column form. The system print command for sending the file to the printer is set by the environmental parameter ASCIPRNT (see the file fiddler.unx or fiddler.vms.) The record length of the file is less than 80 characters, so that a standard text editor may be used to make changes to the file.
- Q** uit Quit [MAKE]: Return to [MAIN] menu.

²At IMEC, the CALC-format files are sent to a VERSATEC color plotter.

3.2.21 [ADD]: add labels to plot

The [ADD] menu provides commands for entering labels which are to be placed on the current FIDDLER -plot. A miniature text editor is provided to [E]nter, [I]nset, [D]elete, and [K]ill labels. Labels may also be edited using the screen editor provided by the system, via the command [U]se. The text and attributes of each label are stored along with the FIDDLER -information in a FIDDLER -format file (with identifying first character of "@"). The [P]lot command produces the current FIDDLER -plot (including previously entered labels), and recently entered labels at default positions and default size. Once the plot has been produced, the labels may be moved, sized, and oriented by the use of the one-letter commands which are listed at the bottom of the plot (see the help file HLABE.HLP).

```
#####
#1#  [ADD]                                #1#
#1#  [H]:  Help                            #1#
#1#  [V]:  View  labels                    #1#
#1#  [E]:  Enter  labels                   #1#
#1#  [I]:  Insert label                   #1#
#1#  [D]:  Delete label                   #1#
#1#  [U]:  Use screen text editor         #1#
#1#  [K]:  Kill  labels                   #1#
#1#  [P]:  Plot and Place labels         #1#
#1#  [Q]:  Quit  [ADD]                   #1#
#####
```

- | | | |
|---|-------|--|
| H | elp | Display HLABE.HLP help file, one page at a time. This describes the various LABEL EDITOR PLOT commands which are used to move, size, and orient the labels on the current FIDDLER -plot. |
| V | iew | List the current labels. |
| E | nter | Enter several labels at one time. To stop entering, use "QUIT" or "quit". |
| I | nsert | Insert one label before a specified label number. |
| D | elete | Delete a specific label. |
| U | se | Use the system-supplied text-editor to edit labels. The text-editor is defined by the environmental parameter SCEDIT (see the FIDDLER command script fiddler.unx or fiddler.vms). |
| K | ill | Get rid of all labels. |
| P | lot | Enter the PLOT mode of the Label editor. LABEL EDITOR PLOT commands are enabled for moving, sizing, and orienting labels. The LABEL EDITOR PLOT commands are listed in the help file HLABE.HLP , shown on the next page. |
| Q | uit | Quit [ADD]: Return to [MAIN] menu. |

3.2.22 [USER]: other applications

The [USER] menu provides access to applications which are not associated with the production of a FIDDLER plot.

```
#####
#1# [USER] #1#
#1# [A]: demo #1#
#1# [B]: calculator (rev.polish) #1#
#1# [C]: calendar #1#
#1# [D]: flowcharts,slides #1#
#1# [E]: poster composer #1#
#1# [Q]: Quit [USER] #1#
#####
```

- A USER program A: PROGLOT - Demo program for plotting-package. Displays menu of different types of plots, character displays.
- B USER program B: CALCULATOR - A scientific calculator.
- C USER program C: CALENDAR - Produce a calendar plot.
- D USER program D: FLOWCHARTS,SLIDES - Produce drawings using a simple flowchart language.
- E USER program E: POSTER COMPOSER - Arrange CALC-format files into a poster.
- Quit Quit [USER]: Return to [MAIN] menu.

3.3 FIDDLER Examples and Applications

In this section, several examples of using FIDDLER are presented. In these examples, the key associated with a certain FIDDLER function is indicated by a single character surrounded by a box, followed with the rest of the key's mnemonic. To access a command, the key is pressed with no carriage return³. In addition, all other user responses which need a carriage return are indicated by surrounding the response with a box, along with the carriage return, indicated by `C/R`. Both computer and user responses are shown in **TYPEWRITER** typeface.

The following areas are typically encountered in a FIDDLER session, and are covered in the examples:

Input Loading, Importing, or Generating Data.

Edit Generating new data from old; Sorting, Deleting, Regressing, and Editing data points.

Change Changing the parameters used to generate the X-Y plot.

Label Creating and placing plot labels.

Output Plot previewing, hard-copy plots, and plot file generation.

We present the first example in script format: step-by-step computer and user actions, along with comments. The other examples are presented in a much more brief format. The reader can gain a feel for what is happening with FIDDLER simply by reading the examples, but might profit from doing the examples on-line.

The figure which is generated by each FIDDLER example is shown at the end of the example discussion.

³This is a "hot" key. If the hot-key feature is not installed, then follow the key with a carriage return.

3.3.1 Example 1: Importing data from an ASCII data file.

One method of accessing data values via FIDDLER is to "import" a file. FIDDLER accepts data from an ASCII data file which has numbers listed on each line, separated by commas or spaces. The numbers may be listed in integer, fixed decimal point, floating decimal point, and/or exponential formats (each line may contain a mixture of these different formats.) Each line of the file must contain the same number of values: one value for the coordinate of each data vector which is to be filled. In this example, there are three columns of data, so each line of the file to be imported has three values, separated by spaces. In this version of FIDDLER, the ASCII file is assumed to have a record length of 80 columns, since this makes it easier to use a standard text-editor to modify the values in the file. If there are too many columns of data to be written into 80-column format, then one option is to split the data into several 80-column data files and then to APPEND them, upon importing into FIDDLER. In this example, the data was generated by the use of a PC/digitizing-tablet setup, and uploaded to the mainframe computer. Another way to create the ASCII data file is simply to use a full-screen editor to type in the values. This is a listing of the ASCII data file:

```
2.355 2.648 .822
2.751 3.210 2.015
3.011 3.836 2.212
3.167 4.163 1.926
3.574 4.767 1.219
4.013 5.177 .435
4.603 5.039 -.719
5.156 4.389 -2.036
5.536 3.390 -2.578
5.722 2.915 -2.352
5.962 2.413 -.805
6.252 2.630 1.028
6.652 3.196 2.020
6.885 3.749 2.284
7.105 4.233 2.114
7.311 4.652 2.096
```

FIDDLER EXAMPLE 1: STARTING OUT

In this portion of the example, FIDDLER is run, the terminal type is selected, and the MAIN menu appears. Again, note that the user input is surrounded by a box, and that no C/R is needed after issuing the "hot" key. The computer response is also shown.

Run FIDDLER with the command "fiddler" from the system prompt:

PROMPT>fiddler

The screen clears and following menu appears:

```
-----
|  FIDDLER : Richard Booth 8-1-88  |
-----
Jun- 1-1992   12:52:22
%%%%%%%%%%
%%% INPUT TERMINAL TYPE:          %%%
%%% #: model:      text:  graphics: %%%
%%% 1: TEK-4010   (VT-100/TEK-4010) %%%
%%% 2: SEIKO     (VT-100/SEI-1104)  %%%
%%% 3: TEK-4205   (VT-100/TEK-4205) %%%
%%% 4: CALCOMP   (VT-100/CALC FILE) %%%
%%% 5: POSTSCRIPT(VT-100/POST FILE) %%%
%%% 6: HPGL      (VT-100/HPGL FILE) %%%
%%% 7: DEC-VT240 (VT-100/TEK-4010) %%%
%%% 8: LN03      (VT-100/LN03 FILE) %%%
%%%%%%%%%%
>>>>>
```

Choose the terminal/terminal-emulator type which you are using. If you are running KERMIT on a PC, choose terminal type 1. If you are at a workstation, you can run FIDDLER under XTERM, and choose terminal type 7. If you do not require plot-previewing capability, but only need to generate hard-copy plots, choose any terminal type.

1

[C]olor or [B]lack default >>>>>

Choose [C]OLOR for color screen menus and colored hard-copy plots:

COLOR

The FIDDLER MAIN menu appears.

FIDDLER EXAMPLE 1: INPUTING DATA

In this section, the importation is performed. We have already created a file named EXAMPLE.DAT which contains the data to be imported, and it is in the default directory.

Choose the IMPORT command from the MAIN menu:

I MPORT

Choose ASCII data format - free-format, space-delimited data in columns

A SCII

INPUT FILE TO BE IMPORTED?

Enter the name of the ASCII data file to be imported. In this example, the file is in the current directory. If it were in a different directory, then you must specify the full path to the file.

EXAMPLE.DAT

Recognize control lines? [Y]es or [No] default

Titles may be specified in the generic ASCII file by preceding them with a ! (for a channel name) \$ for the main title and % for the subtitle. In this example, we do not use this feature.

C/R

After pressing [C/R], the importation begins. The end of importation is signaled by the re-appearance of the MAIN FIDDLER menu.

FIDDLER EXAMPLE 1: EDITING THE DATA

We want to look at the data which was imported, so that we can see if there was any problem with the process. One problem which might appear is: if there were any blank lines in the ASCII file, there would be additional points generated with indeterminate values. These points need to be deleted. The data points in this example are time, position, and velocity values. We want to create a new channel for acceleration by differentiating the velocity with respect to time. Finally, we want to change the channel names, since the import assigned default names.

Choose the EDIT command from the MAIN menu:

E **D**IT

The EDIT menu appears. Choose the VIEW command from the EDIT menu:

V **I**EW

A "Spread-sheet" of the data appears:

```
POINT:  !1X_1      :  !1X_2      :  !1X_3      :
          1          2          3
  1      2.35500E+00  2.64800E+00  8.22000E-01
  2      2.75100E+00  3.21000E+00  2.01500E+00
  3      3.01100E+00  3.83600E+00  2.21200E+00
  4      3.16700E+00  4.16300E+00  1.92600E+00
  5      3.57400E+00  4.76700E+00  1.21900E+00
  6      4.01300E+00  5.17700E+00  4.35000E-01
  7      4.60300E+00  5.03900E+00 -7.19000E-01
  8      5.15600E+00  4.38900E+00 -2.03600E+00
  9      5.53600E+00  3.39000E+00 -2.57800E+00
 10     5.72200E+00  2.91500E+00 -2.35200E+00
 11     5.96200E+00  2.41300E+00 -8.05000E-01
 12     6.25200E+00  2.63000E+00  1.02800E+00
 13     6.65200E+00  3.19600E+00  2.02000E+00
 14     6.88500E+00  3.74900E+00  2.28400E+00
 15     7.10500E+00  4.23300E+00  2.11400E+00
 16     7.31100E+00  4.65200E+00  2.09600E+00
```

Command? >>>>

The VIEW FIDDLER menu is not shown, unless some key not on the menu is pressed, such as [^C/_R]. In this example, the entire data set is visible on one screen. If there were more than 4 channels or more than 16 data points, we could move up, down, left, or right with VIEW commands. We will change the channel names here. Choose the NAME command from the VIEW menu:

N **A**ME

ENTER CHANNEL NUMBER TO GIVE NEW NAME TO

1

CURRENT CHANNEL NAME: X1

ENTER THE NEW NAME

!1 Time [s]

The "!" is a FIDDLER control sequence to choose font family 1 (see FIDDLER plotting-package section, PLTCHR description.)

The "Spreadsheet" is re-painted. Choose the NAME command from the VIEW menu:

N **A**ME

ENTER CHANNEL NUMBER TO GIVE NEW NAME TO

2

CURRENT CHANNEL NAME: X2

ENTER THE NEW NAME

The "Spreadsheet" is re-painted. Choose the NAME command from the VIEW menu:

ENTER CHANNEL NUMBER TO GIVE NEW NAME TO

CURRENT CHANNEL NAME: X3

ENTER THE NEW NAME

The "Spreadsheet" is re-painted. QUIT from VIEW:

UIT

The EDIT menu appears. We will now differentiate channel 3 with respect to channel 1. Choose the OPERATE command from the EDIT menu:

PERATE

The OPERATE menu appears. Choose the BINARY command from the OPERATE menu.

INARY

The BINARY menu appears. Choose the [1]ST DERIVATIVE command from the BINARY menu:

ST DERIVATIVE

Z-CHANNEL, Y-CHANNEL, X-CHANNEL 0 ABORTS

Note that these parameters are entered separately with a [C/R] following each entry; commas or spaces separating values will not work.

The EDIT menu re-appears. Choose VIEW command from the EDIT menu:

IEW

The "Spread-sheet" is displayed with the new column of data. We will re-name this column to "acceleration." Note that OPERATE commands create a default channel name which describes the type of operation which was performed, and which channels were involved.

POINT:	!1 Time [s]	!1 Position [c]	!1 Velocity [c]	CHAN#Cd^1CHAN#:
	1	2	3	4
1	2.35500E+00	2.64800E+00	8.22000E-01	4.37384E+00
2	2.75100E+00	3.21000E+00	2.01500E+00	1.65142E+00
3	3.01100E+00	3.83600E+00	2.21200E+00	-8.61699E-01
4	3.16700E+00	4.16300E+00	1.92600E+00	-1.80667E+00
5	3.57400E+00	4.76700E+00	1.21900E+00	-1.76057E+00
6	4.01300E+00	5.17700E+00	4.35000E-01	-1.85843E+00
7	4.60300E+00	5.03900E+00	-7.19000E-01	-2.17563E+00
8	5.15600E+00	4.38900E+00	-2.03600E+00	-1.81537E+00
9	5.53600E+00	3.39000E+00	-2.57800E+00	3.47042E-01
10	5.72200E+00	2.91500E+00	-2.35200E+00	3.49892E+00
11	5.96200E+00	2.41300E+00	-8.05000E-01	6.38916E+00
12	6.25200E+00	2.63000E+00	1.02800E+00	4.70649E+00
13	6.65200E+00	3.19600E+00	2.02000E+00	1.62884E+00
14	6.88500E+00	3.74900E+00	2.28400E+00	1.52814E-01
15	7.10500E+00	4.23300E+00	2.11400E+00	-4.18791E-01
16	7.31100E+00	4.65200E+00	2.09600E+00	2.44034E-01

Command? >>>>

Choose the NAME command from the VIEW menu:

AME

ENTER CHANNEL NUMBER TO GIVE NEW NAME TO

```
4
CURRENT CHANNEL NAME: CHAN#Cd^1CHAN#A
ENTER THE NEW NAME
!1 Acceleration [cm/s^2_]
The "Spread-sheet" is re-painted. QUIT from VIEW and EDIT:
QUIT
QUIT
The MAIN menu appears.
```

FIDDLER EXAMPLE 1: CHANGING PLOTTING PARAMETERS

We want to alter the default plotting parameters, so that three data curves are displayed on the same plot, different symbols are used, and the plotting window is large enough for the curves.

Choose the CHANGE command from the MAIN menu:

CHANGE

The CHANGE menu appears. Choose the CHANNELS command from the CHANGE menu:

CHANNELS

The CHANNELS menu appears. Choose the X1-Y1 command from the CHANNELS menu. This is the first curve which will be plotted: Y1-values vs X1-values. Note that the default X1,Y1 settings were channels 1 and 2, and setting them here is redundant.

1ST CURVE

Old X1-channel: 1 !1 Time [s]

Enter New X1-channel >>>

1

X1-channel : 1 !1 Time [s]

Old Y1-channel: 2 !1 Position [cm]

Enter New Y1-channel >>>

2

Y1-channel : 2 !1 Position [cm]

Hit any key to continue ...

Type [C/R] to continue. The CHANNELS menu reappears. Select the X2-Y2 command from the CHANNELS menu:

2ND CURVE

Old X2-channel: 0

Enter New X2-channel >>>

1

X2-channel : 1 !1 Time [s]

Old Y2-channel: 0

Enter New Y2-channel >>>

3

Y2-channel : 3 !1 Velocity [cm/s]

Hit any key to continue ...

Type [C/R] to continue. The CHANNELS menu reappears. Select the X3-Y3 command from the CHANNELS menu:

3RD CURVE

Old X3-channel: 0

Enter New X3-channel >>>

1

X3-channel : 1 !1 Time [s]

Old Y3-channel: 0

Enter New Y3-channel >>>

4

Y3-channel : 4 !1 Acceleration [cm/s^ 2_]

Hit any key to continue ...

Type [C/R] to continue. The CHANNELS menu reappears. QUIT from CHANNELS:

QUIT

The CHANGE menu appears. Choose the TITLES command from the CHANNELS menu:

TITLES

CURRENT MAIN TITLE: !1Imported data file

This is the default title.

ENTER NEW MAIN TITLE

!

which nullifies the main title. Do the same for the sub-title:

CURRENT SUB TITLE: !!!EXAMPLE.DAT

ENTER NEW SUB TITLE

!

The CHANGE menu appears.

Choose the SYMBOLS command from the CHANGE menu.

S YMBOLS

The SYMBOLS menu appears. Choose the INDEX command from the SYMBOLS menu. This is the "curve-index". We have defined three curves: Curve 1 is (X-values in Channel 1, Y-values in Channel 2); Curve 2 is (X-values in Channel 1, Y-values in Channel 3); and Curve 3 is (X-values in Channel 1, Y-values in Channel 4).

I NDEX

WHICH Y-VECTOR (1 .. 9,0=10)?

1 ST CURVE

The SYMBOLS menu reappears. Choose the LINE PLUS OTHER SYMBOL command from the SYMBOLS menu:

% LINE PLUS OTHER SYMBOL

ENTER 1-CHARACTER SYMBOL >>>

S QUARES

Capital S is for Squares.

ENTER LINE-TYPE (1 .. 6) >>>

1 ST LINE-TYPE

Line-type 1 is for solid line-segments.

The SYMBOLS menu reappears. Choose the INDEX command from the SYMBOLS menu:

I NDEX

WHICH Y-VECTOR (1 .. 9,0=10)?

2 ND CURVE

The SYMBOLS menu reappears. Choose the LINE PLUS OTHER SYMBOL command from the SYMBOLS menu:

% LINE PLUS OTHER SYMBOL

ENTER 1-CHARACTER SYMBOL >>>

T RIANGLES

Capital T is for Triangles.

ENTER LINE-TYPE (1 .. 6) >>>

1 ST LINE-TYPE

Line-type 1 is for solid line-segments.

The SYMBOLS menu reappears. Choose the INDEX command from the SYMBOLS menu:

I NDEX

WHICH Y-VECTOR (1 .. 9,0=10)?

3 RD CURVE

The SYMBOLS menu reappears. Choose the LINE PLUS OTHER SYMBOL command from the SYMBOLS menu:

% LINE PLUS OTHER SYMBOL

ENTER 1-CHARACTER SYMBOL >>>

O

Capital "o"

ENTER LINE-TYPE (1 .. 9,0=10) >>>

1 ST LINE-TYPE

Line-type 1 is for solid line-segments.

The SYMBOLS menu reappears. QUIT from SYMBOLS:

QUIT

The CHANGE menu appears. Select the WINDOW command from the CHANGE menu:

WINDOW

The WINDOW menu appears. Select the WINDOW command from the WINDOW menu to enter all four endpoints:

WINDOW

Xlo= 2.355000E+00 Enter new Xlo:

0

Xhi= 7.311000E+00 Enter new Xhi:

8

Ylo= -2.413000E+00 Enter new Ylo:

-3

Yhi= 5.177000E+00 Enter new Yhi:

6.9

The WINDOW menu reappears. Note that 6.9 was selected above, so that the distance between ylo and yhi would be slightly less than 10. If the difference were greater or equal to 10, then the y-axis scale would be labeled from -10 to 10. In this case, however, the axis will be labeled from -3 to 7.

QUIT from WINDOW:

QUIT

The CHANGE menu appears. Select the LABELING OPTIONS command from the CHANGE menu:

LABELING OPTIONS

The LABELING OPTIONS menu appears. Choose the CHARACTER symbol size command:

CHARACTER

ENTER CHAR SYMBOL SIZE init.=.02

.03

The LABELING OPTIONS menu reappears. QUIT from LABELING OPTIONS and CHANGE:

QUIT

QUIT

The MAIN menu reappears.

FIDDLER EXAMPLE 1: LABELING PLOT

We will label each curve on the plot itself in order to identify the channels: position, velocity, and acceleration.

Select the LABEL EDITOR command from the MAIN menu.

ADD

The LABEL EDITOR menu appears. Select the ENTER LABELS command from the LABEL EDITOR menu:

ENTER

ENTER Label # 1 or QUIT

!position [cm]

ENTER Label # 2 or QUIT

!velocity [cm/s]

ENTER Label # 3 or QUIT

!acceleration [cm/s^2]

ENTER Label # 4 or QUIT

QUIT

The LABEL EDITOR menu reappears.

Select the PLOT AND PLACE LABELS command from the LABEL EDITOR menu. The plot is previewed, and labels are placed at default locations on the plot. To move the labels around: use L, R, U and D for left, right, up and down, respectively. Type one of 1 through 9 to give relatively larger or smaller movement (and sizing, slanting, and rotation). The x location, y location, and angle may be typed in directly by typing the commands X, Y and A, followed by the value. The screen is not erased until the C command is issued.

PLOT

Once the labels have been placed, QUIT from the plot mode of the LABEL EDITOR by issuing Q. A question mark appears: type [^C/_R] to go back to text mode. The LABEL EDITOR menu reappears.

QUIT from the LABEL EDITOR menu:

QUIT

The MAIN menu reappears.

FIDDLER EXAMPLE 1: PLOT OUTPUT

We would like to preview the plot before making a hard copy plot. Also, we will generate a PostScript format file so that the figure may be incorporated into a document. PostScript files may also be printed directly using the `PRINT` command on the `CYBER: PRINT <FILENAME.EPS > QUEUE=POST`

Select the `PLOT` command from the `MAIN` menu to preview the plot on the screen:

`PLOT`

Once the plot is displayed, a question mark appears: type [`C/R`] to go back to text mode. The `MAIN` menu appears.

Select the `MAKE` command from the `MAIN` menu to make a hard-copy plot, or create a PostScript file of plot commands.

`MAKE`

The `MAKE` menu appears. Select the PostScript command from the `MAKE` menu:

`POSTSCRIPT`

After the file is written, the following prompt appears:

Making POSTSCRIPT File [POST.DAT] ...

RENAME POST.DAT ? [Y]es [N]o=default

`YES`

FILENAME ?

Encapsulated PostScript format is indicated by the file extension `EPS`.

The `MAIN` menu appears.

FIDDLER EXAMPLE 1: FINISHING UP

Finally, to stop FIDDLER ...

QUIT from FIDDLER MAIN menu:

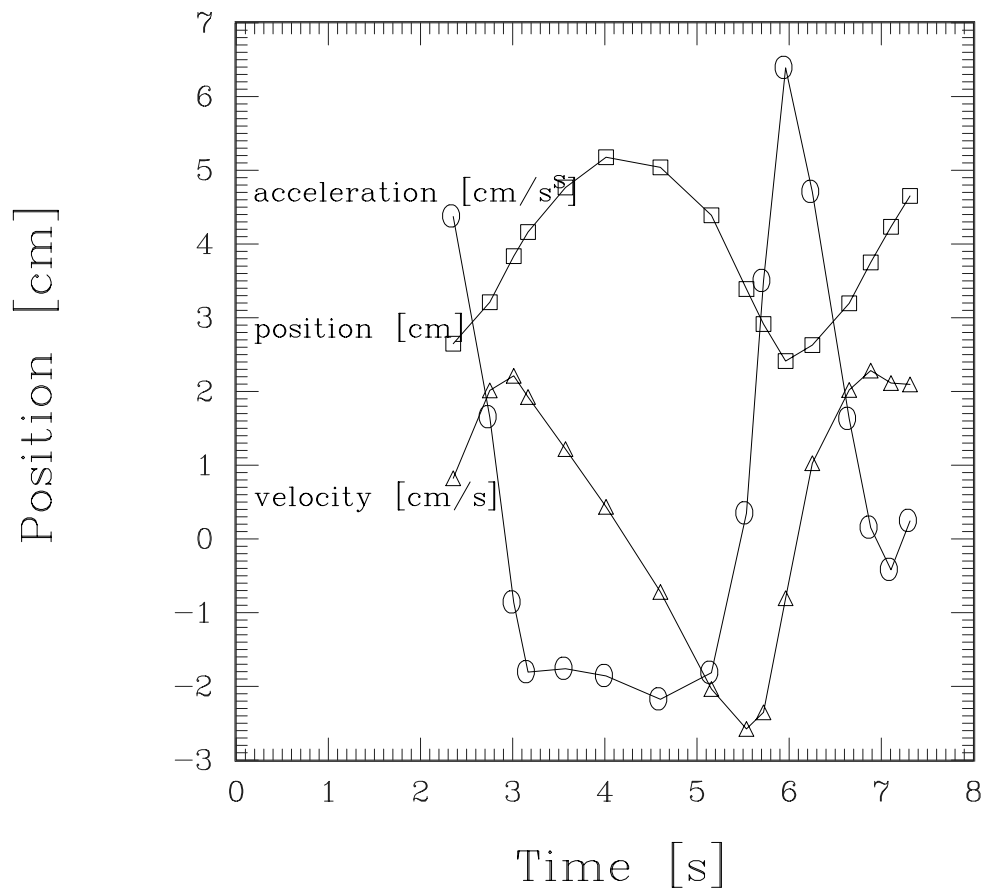
QUIT

The following message appears:

QUIT?? Are You Sure?? [Y]es, [N]o=default

YES

The PostScript plot which was generated by Example 1 is shown here:



3.3.2 Example 2: Importing data generated by FORTRAN program.

The following FORTRAN program generates data values which are written to a file EXAMPLE.DAT . We use FIDDLER to import this file and to plot the results.

```

C#####
PROGRAM EXAMPLE
C + (INPUT,OUTPUT,TAPE5=INPUT,TAPE6=OUTPUT) ! FOR CYBER NOS/VE
C PROGRAM TO GENERATE MULTIPLE COLUMN DATA FOR FIDDLER IMPORT
C#####
DATA LUOUT,TMIN,TMAX,NPTS/8,0.,10.,100/
C-----
OPEN(LUOUT,FILE='EXAMPLE_DAT',STATUS='UNKNOWN')
DO 1 I=1,NPTS
  T=TMIN+(TMAX-TMIN)*REAL(I-1)/REAL(NPTS-1)
  F0=BESSEL(0,T)
  F1=BESSEL(1,T)
  F2=BESSEL(2,T)
  WRITE(LUOUT,*)T,F0,F1,F2
1 CONTINUE
CLOSE(LUOUT)
END
C#####
FUNCTION BESSEL(N,T)
C BESSEL'S FUNCTION, ORDER N (NON-ASYMPTOTIC)
C#####
PARAMETER(SMALL=1.E-9)
C-----
S=T*.5
U=-S*S
R=1.
DO 1 I=2,N
1 R=R/REAL(I)
V=R
DO 2 I=1,100
  R=R*U/(REAL(I)*REAL(I+N))
  IF(ABS(R).LT.SMALL)GOTO 3
  V=V+R
2 CONTINUE
3 BESSEL=V
IF(N.NE.0)BESSEL=V*S**REAL(N)
RETURN
END

```

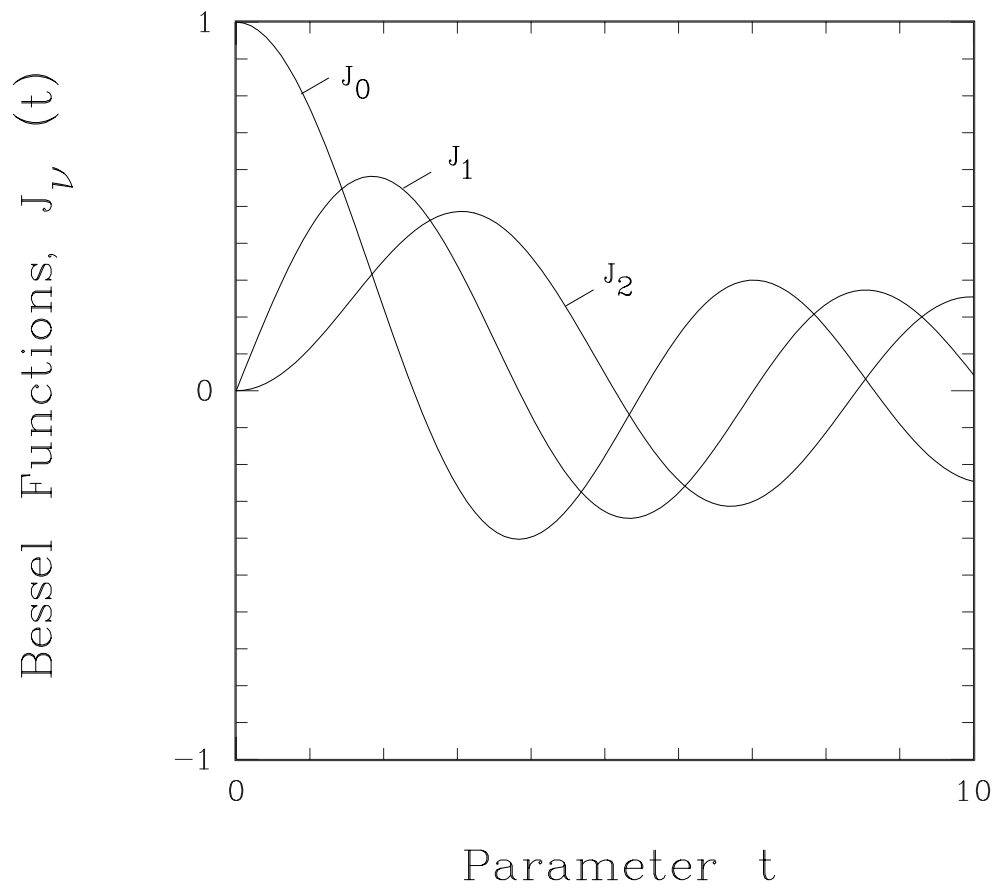
The example is presented in a simple script format, with user input followed by comments on the performed actions (computer response is not shown.)

USER INPUT:	EXAMPLE 2 SCRIPT	COMMENTS:
PROMPT> FORTRAN EXAMPLE.FOR	C/R	Compile FORTRAN program
PROMPT> RUN EXAMPLE	C/R	Run FORTRAN program
PROMPT> fiddler	C/R	Run FIDDLER
1		Terminal type 1 (or other)
COLOR		Color menus and hard-copy plots
IMPORT		Start importation
ASCII		ASCII file
EXAMPLE.DAT		File name to import
C/R		Don't need to recognize control lines
EDIT		Change channel names after importation:
VIEW		View the data
NAME		Change name
1		Rename first channel
!1 Parameter t		New channel name
NAME		Change name
2		Rename second channel
!1 Bessel Functions, J _n (t)		New Channel name
QUIT		QUIT from VIEW
QUIT		QUIT from EDIT
CHANGE		Change plotting parameters
CHANNELS		Change channels
1ST CURVE		Specify x and y channels for 1st curve
1		x-channel
2		y-channel
C/R		Hit C/R to continue
2ND CURVE		Specify x and y channels for 2nd curve
1		x-channel
3		y-channel
C/R		Hit C/R to continue
3RD CURVE		Specify x and y channels for 3rd curve
1		x-channel
4		y-channel
C/R		Hit C/R to continue
QUIT		QUIT from CHANNELS

EXAMPLE 2 SCRIPT – CONTINUED

USER INPUT:	COMMENTS:
T ITLES	Enter new main-title and sub-title
!	New Title
!	New Sub-title
Q UIT	QUIT from CHANGE
A DD	ADD labels
E NTER	Enter labels
!1J_0	!1 is FIDDLER control sequence
!1J_1	for font family 1; _ and ^ are
!1J_2	for sub and super-scripts
!_	A literal underline for pointers
!_	Quote character followed by a
!_	FIDDLER control character is printed
quit	End of entering labels
P LOT	Go to LABEL EDITOR graphics mode
	Place and size labels with graphics-mode commands
Q UIT	QUIT from graphics mode of LABEL EDITOR
C/R	Return to text mode
Q UIT	QUIT from LABEL EDITOR
M AKE	MAKE a PostScript-format plot file
P OSTSCRIPT	PostScript command from MAKE
Y ES	Rename POST.DAT
EXAMPLE.EPS	to new file name
Q UIT	QUIT from FIDDLER
Y ES	YES to QUIT???

The PostScript plot which was generated by Example 2 is shown here. Note that the default line-types for the curves are the same as the curve indices (eg., curve 3 is shown in line-type 3, short dashes).



3.3.3 Example 3: Generating FIDDLER -format data file directly with a FORTRAN program.

FIDDLER uses a particular format to store data in an ASCII file. ⁴ FIDDLER -format data files are read in by using the '[L]OAD' command from the MAIN menu. Channel names, titles, data endpoints, and label information are stored in the FIDDLER -format data file. ⁵

A FIDDLER -format file contains records with 80 or fewer characters, so that a standard text editor may be used to modify (or directly create) the file. The FIDDLER -format has:

1. A control character in column one of each 80-(or less)-column record:

`$` is for the main title

`%` is for the sub-title

`!` is for channel names

`<` is for the lower point index for the following data values

`>` is for the upper point index for the following data values

`space` is for numbers

`@` is for the first label line (followed by label information record)

`"` or any other character is for comments

2. The rest of the line is the value of the particular FIDDLER variable.

`$ % !` Titles and channel names are character values

`< >` Lower and upper point indices are integers, written in I6 format

`space` Numbers are real, written in E20.10 format

3. Specifying the channel name with a record beginning with ! also increments the channel index. If endpoints are specified, FIDDLER assumes that the following (upper - lower) records are numbers, and ignores the first column. If the lower index is different from 1, then the point index begins with an offset. This is useful for lining up data vectors to a single column of data values.

Application FORTRAN programs can be designed to write data files in FIDDLER -format. This has the advantage that titles and labels may be written directly to the FIDDLER -format file. The FORTRAN program on the following page generates 9 channels of data, which are stored in a 2-dimensional array. FIDDLER -information is sent to the subroutine FIDFILE, which stores the information. In this application, no label information is stored. Label data is typically generated and stored by using the LABEL EDITOR from the MAIN FIDDLER menu, and re-storing the FIDDLER -information using the [S]TORE command from the MAIN FIDDLER menu. ⁶

The example script is listed following the FORTRAN program listing.

⁴Using the '[S]TORE' command from the MAIN menu.

⁵We use the standard file extension '.FID ' to indicate a FIDDLER -format data file. We refer to the information which is stored in such a file as FIDDLER -information.

⁶If interested, study the source listing of the FIDDLER subroutine FDISK in the appendices to discover how label information is stored.

```

C#####
  PROGRAM EXAMPLE
C  +(INPUT,OUTPUT,TAPE5=INPUT,TAPE6=OUTPUT) ! FOR CYBER NOS/VE
C  PROGRAM TO GENERATE MULTIPLE COLUMN DATA FOR FIDDLER LOAD
C  FIDDLER INFORMATION:
C  D(I,J)   = DATA ARRAY; I=CHANNEL INDEX, J=POINT INDEX
C  CTIT     = MAIN TITLE
C  CSUB     = SUB TITLE
C  CNAME(I) = CHANNEL NAME; I=CHANNEL INDEX
C  MAXCHAN  = 1ST PHYSICAL DIMENSION OF D(*,*)
C  NPTS     = NUMBER OF POINTS IN D(*,*)
C  NCHAN    = NUMBER OF CHANNELS IN D(*,*)
C#####
  PARAMETER(MAXCHAN=50,MAXPTS=500)
  DIMENSION D(MAXCHAN,MAXPTS)
  CHARACTER*80 CTIT,CSUB,CNAME(MAXCHAN)
  DATA TMIN,TMAX/-3.2,3.2/
C-----
  NPTS=200          ! USING 200 OUT OF POSSIBLE 500 POINTS
  NCHAN=9           ! USING 9 OUT OF POSSIBLE 50 CHANNELS
  CTIT= ' !1FIDDLER Example '
  CSUB= ' !1FORTRAN Program to .FID file'
  CNAME(1)= ' !1Parameter T '
  CNAME(2)= ' !1X Leminiscate of Bernoulli '
  CNAME(3)= ' !1Y Leminiscate of Bernoulli '
  CNAME(4)= ' !1X Leminiscate of Gerono '
  CNAME(5)= ' !1Y Leminiscate of Gerono '
  CNAME(6)= ' !1X Hypotrochoid '
  CNAME(7)= ' !1Y Hypotrochoid '
  CNAME(8)= ' !1X Epitrochoid '
  CNAME(9)= ' !1Y Epitrochoid '
  DO 1 I=1,NPTS
    T=TMIN+(TMAX-TMIN)*REAL(I-1)/REAL(NPTS-1)
    C1=COS(T)
    S1=SIN(T)
    C7=COS(7.*T)
    S7=SIN(7.*T)
    D(1,I)=T
    D(2,I)= C1/(S1*S1+1.) -1.
    D(3,I)=S1*C1/(S1*S1+1.) -1.
    D(4,I)=C1 -1.
    D(5,I)=S1*C1 +1.
    D(6,I)=.7*C1+.2*C7 +1.
    D(7,I)=.7*S1-.2*S7 -1.
    D(8,I)=.7*C1-.2*C7 +1.
    D(9,I)=.7*S1-.2*S7 +1.
1  CONTINUE
  CALL FIDFILE(D,MAXCHAN,NCHAN,NPTS,CNAME,CTIT,CSUB)
  END

```

```

C#####
  SUBROUTINE FIDFILE(DATARAY,MAXCHAN,NCHAN,NPTS,CNAME,CTIT,CSUB)
C THIS SUBROUTINE GENERATES FIDDLER-FORMAT DATA FILE
C REAL      DATARAY(NP,*) = DATA POINTS (CHAN.INDEX,POINT INDEX)
C INTEGER   NP           = FIRST PHYSICAL DIMENSION OF DATARAY
C INTEGER   NCHAN       = NUMBER OF CHANNELS
C INTEGER   NPTS        = NUMBER OF POINTS
C CHARACTER*(*) CNAME(*) = CHANNEL NAMES (CHAN.INDEX)
C CHARACTER*(*) CTIT    = MAIN TITLE
C CHARACTER*(*) CSUB    = SUB TITLE
C#####
  PARAMETER(LUOUT=8)
  DIMENSION DATARAY(MAXCHAN,*)
  CHARACTER*(*) CTIT,CSUB,CNAME(*)
  CHARACTER*80 OUTFILE
1000 FORMAT(A)
1001 FORMAT(A,A)
1002 FORMAT(A,I6)
1003 FORMAT(A,E20.10)
C-----
  PRINT *, ' ENTER FIDDLER FORMAT OUTPUT FILE NAME >>>'
  READ (*,1000) OUTFILE
  OPEN(LUOUT,FILE=OUTFILE,STATUS='UNKNOWN')
  WRITE(LUOUT,1001)'$',CTIT
  WRITE(LUOUT,1001)'%',CSUB
  DO 1 I=1,NCHAN
    WRITE(LUOUT,1001)'!',CNAME(I)
    WRITE(LUOUT,1002)'<',1
    WRITE(LUOUT,1002) '>',NPTS
    WRITE(LUOUT,1003)(' ',DATARAY(I,J),J=1,NPTS)
1 CONTINUE
  CLOSE(LUOUT)
  RETURN
  END

```

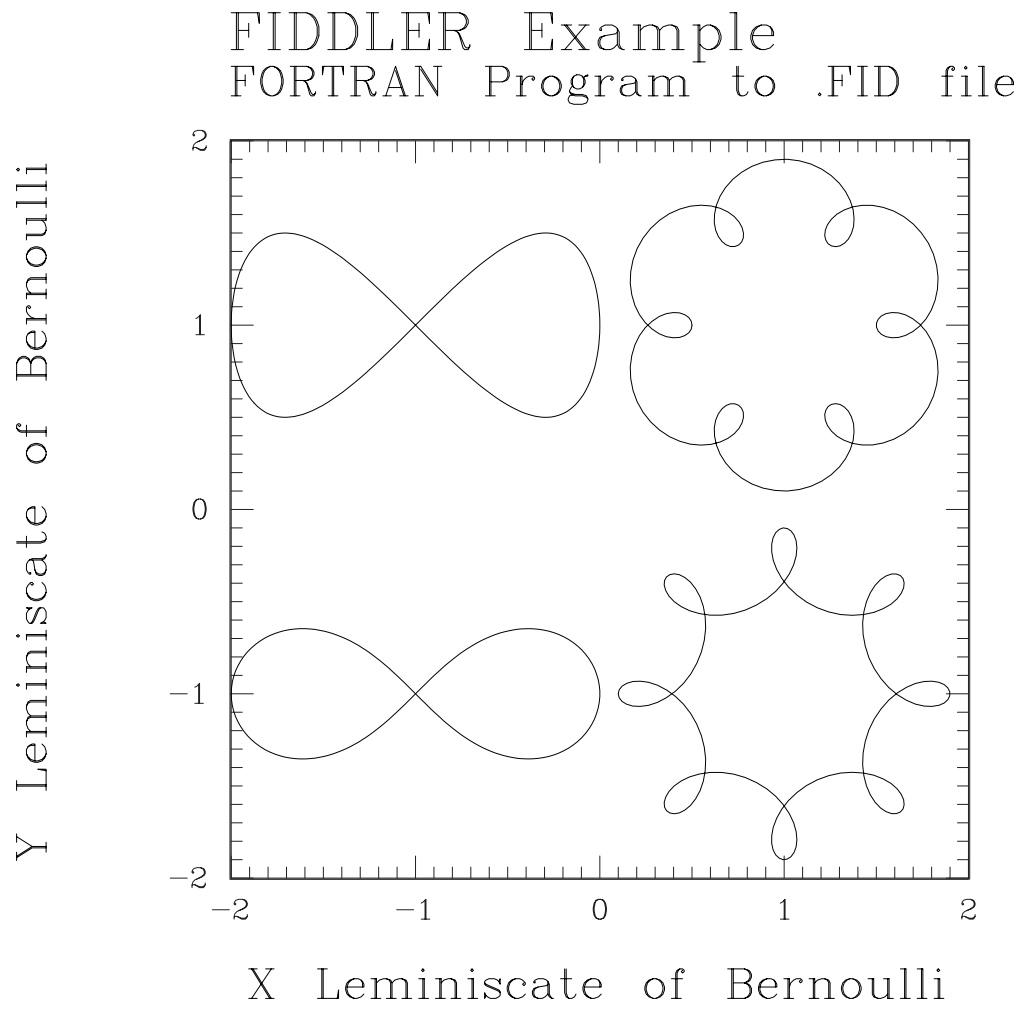
USER INPUT:	EXAMPLE 3 SCRIPT	COMMENTS:
PROMPT> FORTRAN EXAMPLE.FOR	C/R	Compile FORTRAN program
PROMPT> RUN EXAMPLE	C/R	Run FORTRAN program
ENTER FIDDLER FORMAT OUTPUT FILE NAME >>>		FORTRAN program prompt
EXAMPLE.FID		File name
PROMPT> FIDDLER		Run FIDDLER
1		select 4010 terminal type (or other)
COLOR		Color menus and hard-copy plots
LOAD		Load FIDDLER -format data file
EXAMPLE.FID		File name
CHANGE		Change channels
CHANNELS		Channels sub-menu
1ST CURVE		Select x and y channels of 1st curve
2		x-channel
3		y-channel
C/R		return to CHANNELS menu
2ND CURVE		Select x and y channels of 2nd curve
4		x-channel
5		y-channel
C/R		return to CHANNELS menu
3RD CURVE		Select x and y channels of 3rd curve
6		x-channel
7		y-channel
C/R		return to CHANNELS menu
4TH CURVE		Select x and y channels of 4th curve
8		x-channel
9		y-channel
C/R		return to CHANNELS menu
QUIT		QUIT from CHANNELS
WINDOW		Change window
WINDOW		All 4 endpoints
-2		Lower x
2		Upper x
-2		Lower y
2		Upper y
QUIT		QUIT from WINDOW
QUIT		QUIT from CHANGE
PLOT		Preview plot
C/R		Return to text mode after plot is done

EXAMPLE 3 SCRIPT - CONTINUED

USER INPUT: COMMENTS:

<input type="checkbox"/> M	AKE	Make hard-copy
<input type="checkbox"/> P	OSTSCRIPT	PostScript plot
<input type="checkbox"/> Y	ES	Yes to print file
<input type="checkbox"/> Y	ES	Yes to rename file
figfid3.ps1		File name
<input type="checkbox"/> Q	UIT	QUIT from FIDDLER
<input type="checkbox"/> Y	ES	YES to QUIT???

The PostScript plot which was generated by Example 3 is shown here:



3.3.4 Example 4: Entering Single points.

The MAIN FIDDLER menu provides a command for entering FIDDLER -information manually ⁷, as opposed to generation by FORTRAN programs, or uploading from local computers. In this example, this [D]ATA command is used to input data values for plotting. In addition, the PEN_CONTROL option (or data structure) is illustrated. In many applications, the data points are best illustrated by connecting consecutive points by line-segments. In example 1, for instance, the curve representing 'position' is drawn with line-segments connecting the points (in addition to squares at each point). ⁸ The data set in example 1 has more than one curve, however, which is illustrated with line-segments connecting consecutive points of each curve, but with no connecting line-segment joining the curve endpoints. In example 1, we used a separate channel for each curve's y-value, while the x-values of each curve were in a common channel. An alternative technique, shown in this example, is to use two channels only for x- and y-values for every curve (or 'sweep'), and to use a third channel (the PEN_CONTROL channel) to indicate the start of each sweep. The standard implemented in FIDDLER is:

1. The PEN_CONTROL channel has channel name 'PEN_CONTROL ': all capitals and an underbar. No other name is supported.
2. The PEN_CONTROL channel contains 0 for the start of a sweep, and 1 for a continuation of a sweep. Plotting with PEN_CONTROL '0' is like a (HP) BASIC 'MOVE' command and a PEN_CONTROL '1' is like a (HP) BASIC 'DRAW' command.
3. The PEN_CONTROL flag is turned 'ON' when the file is loaded, if any channel is found with the name 'PEN_CONTROL '; otherwise, this flag is 'OFF'. If, during the course of a FIDDLER session, a channel is re-named to 'PEN_CONTROL ', and the PEN_CONTROL flag is 'OFF', it may be turned on by using the 'PEN_CONTROL ' command from the CHANGE menu.

The example script is shown on the following pages.

⁷ Titles, channel names, and data values

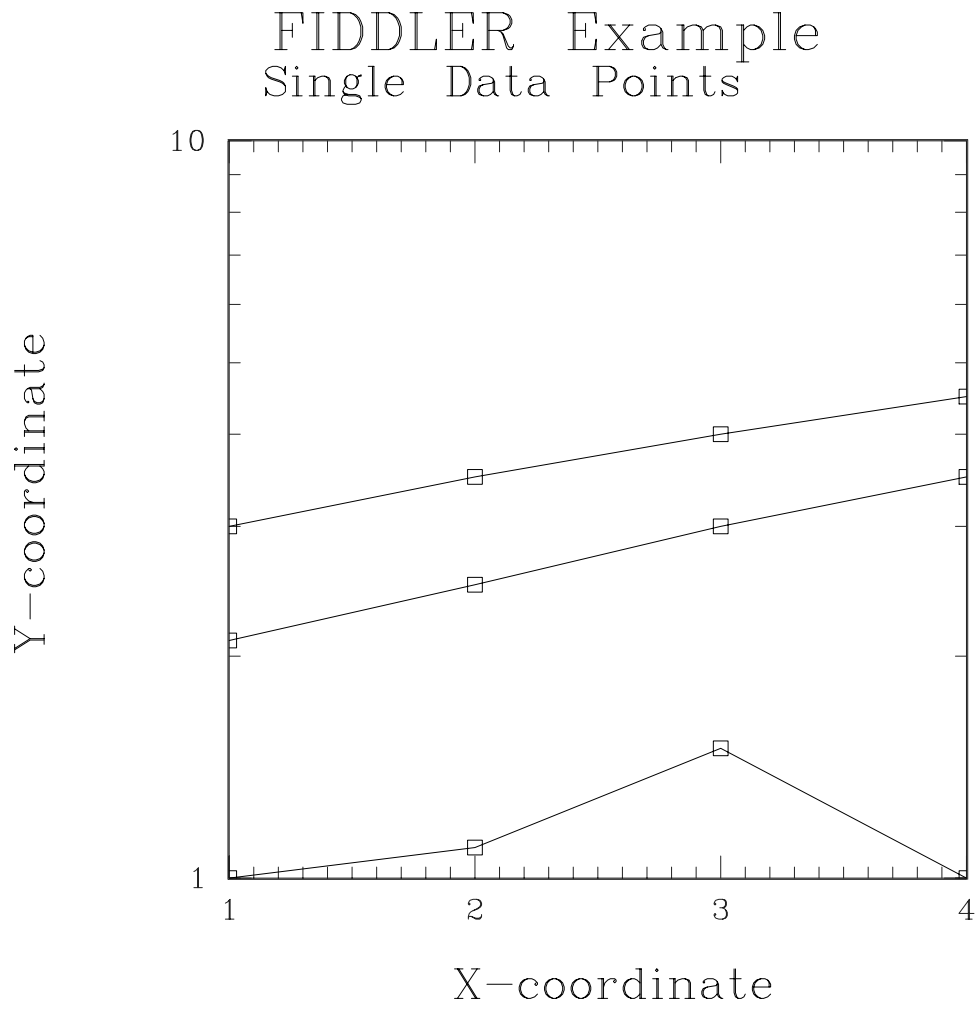
⁸ Randomly positioned data is probably best illustrated with a symbol at each point, with no intervening line-segments.

USER INPUT:	EXAMPLE 4 SCRIPT COMMENTS:
PROMPT> fiddler ^{C/R}	Run FIDDLER
1	Terminal type 1 (or other)
C OLOR	Color menus and plots
D ATA	Enter single Data points
!1 FIDDLER Example	Title
!1 Single Data Points	Sub-title
x	1st Channel name
y	2nd Channel name
PEN_CONTROL	3rd Channel name
quit	End of channels
1	x
1	y
0	PEN_CONTROL
2	x
1.1	y
1	PEN_CONTROL
3	x
1.5	y
1	PEN_CONTROL
4	x
1	y
1	PEN_CONTROL
1	x
2.1	y
0	PEN_CONTROL
2	x
2.5	y
1	PEN_CONTROL
3	x
3	y
1	PEN_CONTROL
4	x
3.5	y
1	PEN_CONTROL
1	x
3	y
0	PEN_CONTROL
2	x
3.5	y
1	PEN_CONTROL
3	x
4	y
1	PEN_CONTROL
4	x
4.5	y
1	PEN_CONTROL
QUIT	Signal end of data entry

EXAMPLE 4 SCRIPT – CONTINUED

USER INPUT:	COMMENTS:
C HANGE	Change plotting parameters
L ABELING OPTIONS	Re-size axes labels,etc.
N UMBERS	Number size
.027	A little larger than default
Q UIT	QUIT from LABELING OPTIONS
S YMBOL	Change plotting symbol
% LINE PLUS OTHER SYMBOL	Line-segments/symbols
S QUARES	Square
1 ST LINE-TYPE	Solid lines
Q UIT	QUIT from SYMBOLS
M ODE	LIN/LOG
X YPLOT	X-Y plot
1 LINEAR	x-axis
2 LOGARITHMIC	y-axis
N AME	Change channel name
N O	Don't list channel names
1	1st channel
!1 X-coordinate	New name
N AME	Change channel name
N O	Don't list channel names
2	2nd channel
!1 Y-coordinate	New name
Q UIT	QUIT from CHANGE
P LOT	Preview plot
C/R	Return to text mode after plot is done
M AKE	Make hard-copy of plot
P OSTSCRIPT	PostScript format plot file
Y ES	Yes to print file
Y ES	Yes to rename file
figfid4.ps1	New file name
Q UIT	QUIT from FIDDLER
Y ES	YES to QUIT???

The PostScript plot which was generated by Example 4 is shown here:



3.3.5 Example 5: Sampling Functions.

In this example, we use the function sampling command available at the MAIN FIDDLER menu. We sample

$$f(t) = 0.5 \frac{\sin(10 t)}{t} - 0.3$$

over this range of values of t :

$$-3 < t < 3$$

The function sampling command provides a way of generating FIDDLER -information by entering only the 'function' part of a function-sampling FORTRAN program (and also the range and type of sampling). In the example, 5 function lines are entered. The function sampler generates a FORTRAN program, compiles and runs it, and then loads the FIDDLER -information. The FORTRAN program is `FUNC.FOR`. The user may find it useful to generate a function sampling FORTRAN program, using the function sampling command, and then to modify the program later, after FIDDLER has been stopped. The example script is shown on the following pages.

USER INPUT:	EXAMPLE 5 SCRIPT COMMENTS:
PROMPT> FIDDLER ^{C/R}	Run FIDDLER
1	KERMIT terminal type
COLOR	Color menus and plots
FUNCTION	Function sampler
ENTER	Enter function lines
A=.5	1st function line
B=.3	2nd function line
C=10.	3rd function line
F=A*C-B	4th function line
IF(T.NE.0.)F=A*SIN(C*T)/T-B	5th function line
QUIT	End of function line entry
TMIN,TMAX	Enter parameter limits
-3	tmin
3	tmax
300	Number of samples
1 LINEAR	Linear sampling
GD	Start sampling
QUIT	Quit from FUNCTION SAMPLE
CHANGE	Change plotting parameters
NAMES	Change channel name
ND	No need to list names
1	1st channel
!2 Parameter t	New name
NAMES	Change channel name
ND	No need to list names
2	2nd channel
!2 a sin(ct) - b	New name
QUIT	QUIT from CHANGE
PLOT	Preview plot
MAKE	Make hard-copy
POSTSCRIPT	Make PostScript plot file
YES	YES to print file
YES	YES to rename file
figfid5.ps1	New file name
QUIT	QUIT from FIDDLER
YES	YES to QUIT???

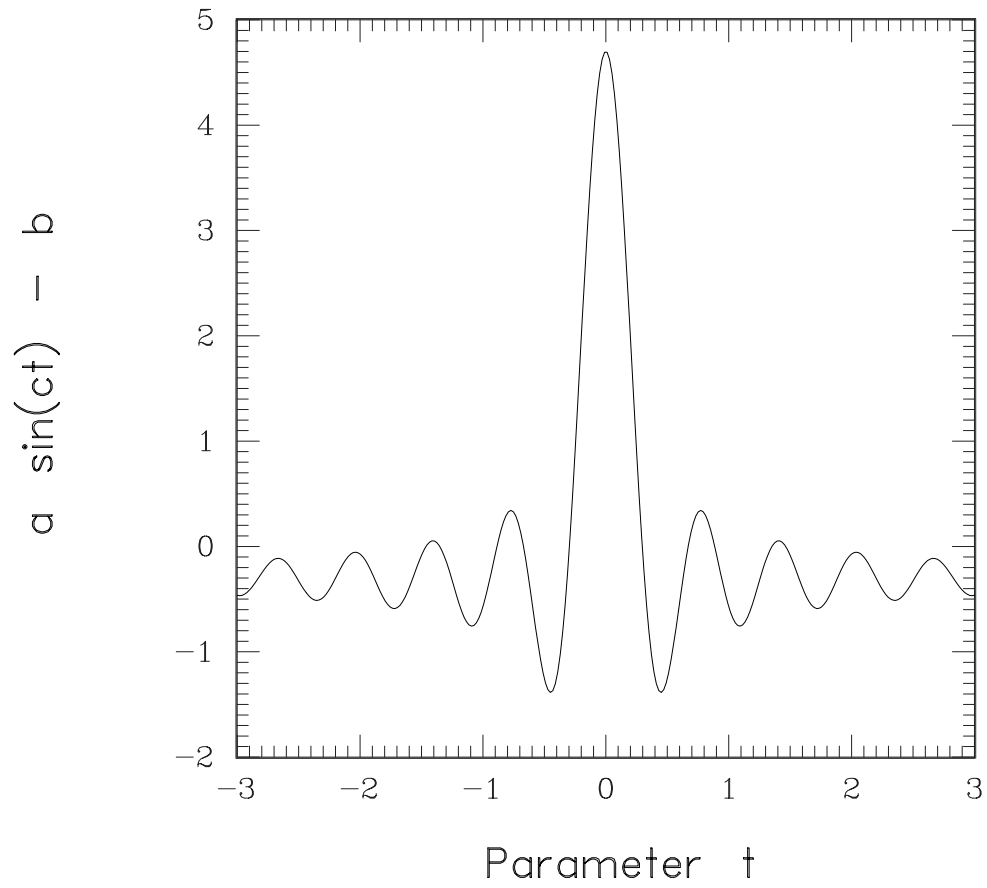
The program which is generated by the function sampling FIDDLER command is shown below:

```

C#####
PROGRAM SAMFUNC
C LOCAL PROGRAM GENERATED BY FIDDLER          #
C CALCULATES T,F(T)                            #
C#####
REAL DATA(2,1000),F,T,TMIN,TMAX
INTEGER NPTS
CHARACTER*50 MTIT,STIT,CNAME(2)
C-----
CNAME(1)='T'
CNAME(2)='F(T)'
MTIT=' '
STIT=' '
NCHAN=2
NPTS= 300
TMIN= -.3000000000E+01
TMAX= .3000000000E+01
T=0.
F=0.
DO 1 I=1,NPTS
  IF(NPTS.LE.1)D=1.
  IF(NPTS.GT.1)D=FLOAT(NPTS-1)
  D=FLOAT(I-1)/D
  T=TMIN+(TMAX-TMIN)* D
  a=.5
  b=.3
  c=10.
  f=a*c-b
  if(t.ne.0.)f=a*sin(c*t)/t-b
  DATA(1,I)=T
  DATA(2,I)=F
1 CONTINUE
OPEN(8,FILE='$LOCAL.SAMFUNC_FID',STATUS='UNKNOWN')
WRITE(8,'(A1,A)') '$',MTIT
WRITE(8,'(A1,A)') '%',STIT
DO 2 I=1,NCHAN
  WRITE(8,'(A1,A)') '! ',CNAME(I)
DO 3 J=1,NPTS
  WRITE(8,'(A1,E20.10)') '#',DATA(I,J)
3 CONTINUE
2 CONTINUE
CLOSE(8)
RETURN
END

```

The PostScript plot which was generated by Example 5 is shown here:



3.3.6 Example 6: Uploading Data generated by HP9836/HPseries300 version of FIDDLER .

There are several versions of FIDDLER which run on HP9836 and HP-series 300 desktop computers. These local versions of FIDDLER can generate ASCII data files in the same format used by the CYBER version of FIDDLER . Thus, the desktop FIDDLER applications, which are typically involved in gathering of measurement data, can communicate easily with the mainframe FIDDLER program. The procedure is this:

1. Use HP version of FIDDLER to gather measurement data. In this example, we have digitized a signature, using the digitizing tablet.
2. Store the HP-FIDDLER -information as an ASCII file, using the [S]TORE command at the MAIN menu.
3. Use a data-conversion program to translate the HP format of ASCII data (LIF format) to IBM pc-compatible format.
4. Log in to the CYBER using NETDIAL/KERMIT terminal emulation program running on an IBM pc-compatible computer.
5. Run KERMIT on the CYBER by issuing

```
KERMIT
```

```
SERVER
```

6. Escape to the local KERMIT command page by typing ALT-X (or CTRL-] C).
7. Issue this command:

```
SEND < filename.FID >
```

where < filename.FID > is the name of the converted data file.

8. Once the file has been transferred, issue:

```
FINISH
```

```
CONNECT
```

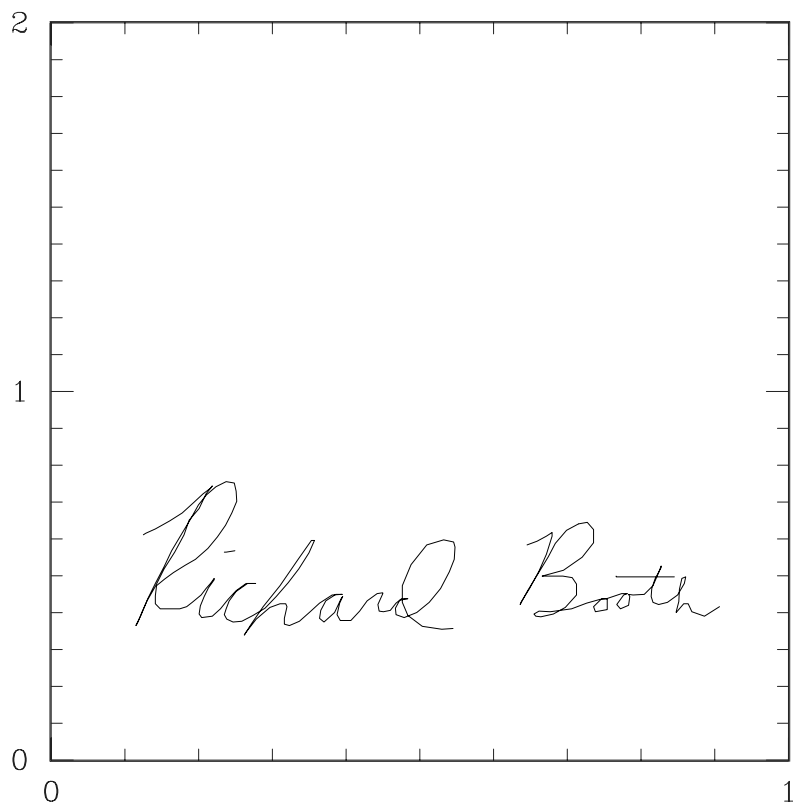
9. Issue:

```
QUIT
```

to exit from (mainframe) KERMIT.

10. Run FIDDLER and [L]OAD < filename.FID > .

The PostScript plot which was generated by Example 6 is shown here:



3.3.7 Example 7: The PostScript file generated by FIDDLER ; incorporating a graphic into a SCRIBETM document

FIDDLER generates⁹ an encapsulated PostScript file which may be incorporated directly into a SCRIBETM document *with no changes* . In addition, the file may be printed directly on the PostScript printer, as shown in the previous examples. The generated file contains several PostScript macros (in the front portion of the PostScript "program") to re-size the graphic when it is printed on the printer.

The front-end and trailing-end of the PostScript file which was generated by a previous example is shown on the following page. The first nine lines of the listing are standard PostScript encapsulated format. The following two lines of underlines preceded by a percent sign are comments, and are ignored by the PostScript printer. The next section defines several macros which are used in the rest of the "program." The last portion of the header defines some useful macros which may be invoked by the user before printing. To use them, one must simply edit the PostScript file and enter the macro into the program. Or remove the comment character (%) from one of the lines following the R command (see comments in the EPS file itself). For example, the graphic which is produced by printing the file directly turns out to be 8x8 inches in size. To half size the plot, use the /INCHWIDTH command: **4 INCHWIDTH**

A procedure to incorporate the graphic into a SCRIBE file is:

1. Produce the PostScript format file using FIDDLER , as illustrated in the previous examples.
2. Port the file over to the VAX from the CYBER: use FTP from either the VAX or the CYBER; or (more time-consumingly) download the file using KERMIT and upload it to the VAX using KERMIT.
3. Within the front end of your SCRIBE document, define a macro for incorporating graphics:

```
@TEXTFORM {GRAL="@BEGIN (CENTER)
@GRAPHIC (WIDTH=6IN,SCALETOFIT,BOUNDINGBOX=FILE,POSTSCRIPT='@PARM<TEXT>')
@END (CENTER)"}

```

4. In the portion of the SCRIBE document where the graphic is to appear, include the command:


```
@GRALFIGURE.EPS
```

⁹Via the [M]AKE command from the MAIN menu.

The front and tail ends of the Encapsulated PostScript file generated by a previous example are listed here.

```

%!
%%Creator: FIDDLER (convex/unix VERSION 2.0)
%%For: Learned Extractor
%%Pages: 1
%%DocumentFonts: Courier
%%Title: Tropic of Calculus
%%CreationDate: Jun-18-1992 13:47:41
%%BoundingBox: 0 0 576 576
%%EndComments
%-----
/C {currentpoint stroke moveto} def
/F {currentpoint fill moveto} def
/D {lineto} def
/M {moveto} def
/d {lineto C} def
/m {moveto C} def
/e {lineto F} def
/P {1 sub 7.7 div setgray} def
/R {.072 .072 scale 2 setlinewidth
 1 setlinecap 1 setlinejoin} def
%=====
% USEFUL DEFINITIONS:
% ISSUE THESE AFTER "R" COMMAND IN THIS PROLOG
%
% 1. MAKE PLOT FIT IN InchDim. x InchDim.
% InchDimension INCHWIDTH -
/INCHWIDTH {8 div dup scale} bind def
%
% 2. ELIMINATE SHOWPAGE FOR ENCAPSULATED PS
/NOSHOWPAGE {/showpage {} def} def
%
% 3. landscape/portrait, etc:
/PORT {1.38 1.38 scale 150 150 translate} def
/POR1 {0.97 0.97 scale 200 2000 translate} def
/PORH {0.97 1.20 scale 200 900 translate} def
/PORW {0.97 1.38 scale 200 200 translate} def
/LAND { 90 rotate 0 -8000 translate
 1.38 1.38 scale 150 0 translate} def
/LAN1 { 90 rotate 0 -8000 translate
 0.97 0.97 scale 1950 0 translate} def
/LANH { 90 rotate 0 -8000 translate
 1.20 0.97 scale 850 0 translate} def
/LANW { 90 rotate 0 -8000 translate
 1.38 0.97 scale 150 0 translate} def
%=====
R
LANH
% 4 INCHWIDTH % 1/2 SIZE (8 INCHES DEFAULT)

```

```
% NOSHOWPAGE      % ENCAPSULATED POST-SCRIPT
%%EndProlog
%-----
%%Page: -ONE- 1
newpath
      6 P  800 4800 M  840 5080 M  840 4800 D  853 5080 M  853 4800 D
      800 5080 M  893 5080 D  800 4800 M 1000 4800 D 1000 4880 D  987 4800 D
      1160 5080 M 1120 5067 D 1093 5040 D 1080 5013 D 1067 4960 D 1067 4920 D
      1080 4867 D 1093 4840 D 1120 4813 D 1160 4800 D 1187 4800 D 1227 4813 D

      ....

      4267 3200 D 7200 4800 M 7124 5143 M 7124 4457 D 6781 4800 M 7467 4800 D
      7200 4000 M 7124 4343 M 7124 3657 D 6781 4000 M 7467 4000 D 7200 3200 m
      7124 3543 M 7124 2857 D 6781 3200 M 7467 3200 D
%-----
stroke          % WRITE DATA
showpage        % SEND TO PRINTER
```

3.3.8 Example 8: The HPGL file generated by FIDDLER ; downloading and printing on an HPGL pen-plotter

FIDDLER generates ¹⁰ a Hewlett-Packard Graphics Language (HPGL) plot file which can be downloaded and issued to an HP pen plotter. The front-end and trailing-end of an HPGL file are shown on the following page. Note that the records of the file are not longer than 80 characters, so that a standard text-editor may be used to edit the file. This is useful for changing the pen indices.

The procedure to generate the HPGL graphic is

1. Log into the CYBER using NETDIAL/KERMIT communications program running on a personal computer. Run FIDDLER , load or generate FIDDLER -information, and generate a FIDDLER -plot.
2. Produce the HPGL format file: choose the HP[G]L command from the MAKE menu. The file will be named HPGL.DAT .
3. Quit FIDDLER .
4. Download the file using KERMIT. Run KERMIT on the CYBER by issuing:

```
KERMIT
SERVER
```

5. Escape to the local KERMIT command page by typing ALT-X (or CTRL-] C).
6. Issue this command:

```
GET HPGL.DAT
```

7. Once the file has been transferred, issue:

```
FINISH
CONNECT
```

8. Issue:

```
QUIT
```

to exit from (mainframe) KERMIT.

9. Logout from the mainframe.
10. The file HPGL.DAT is now in the default directory on the PC.
11. If the HP pen-plotter is connected to the serial port of the PC, issue:


```
COPY HPGL.DAT COM1:
```

 from the DOS prompt.
12. If the HP pen-plotter is connected to an HP desk-top computer, use a data-conversion program to translate the IBM pc-compatible format of ASCII data to HP format (LIF format). A short program must be written on the HP computer to read the ASCII data file and to send the data to the plotter. A program written in HP BASIC 2.0 is listed following the HPGL file listing.

¹⁰Via the [M]AKE command from the MAIN menu.

Shown below are the front and tail ends of the HPGL file generated by the MAKE HPGL command with data from a previous example.

```

1
PU;IN;VS10;
PU 250, 278;
SP 2;
PU 1690, 1719;PD 6010, 1719;PD 6010, 6039;PD 1690, 6039;PD 1690, 1719;
PU 1641, 1594;PD 1628, 1589;PD 1620, 1577;PD 1616, 1556;PD 1616, 1544;
PD 1620, 1524;PD 1628, 1511;PD 1641, 1507;PD 1649, 1507;PD 1661, 1511;
PD 1669, 1524;PD 1674, 1544;PD 1674, 1556;PD 1669, 1577;PD 1661, 1589;

...

PD 5316, 3387;PD 5300, 3132;PD 5296, 2842;PD 5279, 2603;PD 5279, 2586;
PD 5328, 2791;PD 5357, 2791;PD 5381, 2603;PD 5463, 2501;PD 5561, 2723;
PD 4888, 3438;PD 4888, 3421;PD 4904, 3421;PD 4981, 3421;PD 5051, 3421;
PD 5112, 3421;PD 5157, 3421;PD 5222, 3421;PD 5267, 3421;PU 250, 279;
SP 1;
PU;SP;VS;

```

The following program reads an HPGL-format ASCII data file and writes the data to an HP pen-plotter via an HPIB interface. The program runs on an HP desktop computer (HP9836 or HP series 300 computer) under BASIC2.0 or higher.

```

1000 !#####
1010 !# HPGL ASCII FILE -> HPIB PLOTTER
1020 !# R.BOOOTH 3-18-89
1040 !#####
1050 DIM Buffer$(1024),Filename$(30)
1060 !-----
1070 INPUT "HPGL ASCII FILE NAME?",Filename$
1080 ASSIGN @File TO Filename$&"":HP9895,702,0";FORMAT ON
1090 ASSIGN @Plot TO 705
1100 ON ERROR GOTO Endofit
1110 Nextline: !
1120 ENTER @File;Buffer$
1130 OUTPUT @Plot USING "#,K";Buffer$
1140 GOTO Nextline
1150 Endofit: !
1160 OUTPUT @Plot USING "#,K";"PU;"
1170 ASSIGN @File TO *
1180 ASSIGN @Plot TO *
1190 BEEP
1200 PRINT "FINISHED"
1210 END

```

Appendix A

FIDDLER : Character set

The character set which is incorporated in the GRAPHER plotting package is a subset of the (public-domain) Hershey font set, organized into four font families. These font families are: Simplex, Complex, Duplex and Triplex. ¹ Each font family consists of three component fonts: n (for normal), g (for greek or gothic), and s (for script or italic). Each of the character set sub-fonts are accessed by the use of GRAPHER control characters ² inserted into the string which is to be printed. The control sequences for accessing each font are:

GRAPHER control sequence:	FONT FAMILY:	FONT:
!0	Simplex	n (normal)
!0&	Simplex	g (greek)
!0\$	Simplex	s (script)
!1	Complex	n (normal)
!1&	Complex	g (greek)
!1\$	Complex	s (script)
!2	Duplex	n (normal)
!2&	Duplex	g (cyrillic)
!2\$	Duplex	s (italic)
!3	Triplex	n (normal)
!3&	Triplex	g (gothic)
!3\$	Triplex	s (italic)

The table of characters which follows was generated by the FORTRAN program listed on the following page. Each single character is generated by concatenating the "prefix" string to the "suffix" string, in the call to the PLTCHR subroutine.

¹ Simplex, etc. indicates the "weighting" or thickness of the component strokes of each character.

² Don't worry, these "control" characters are also printable ASCII characters.

```

C#####
PROGRAM XPLC3
C *** PLOT TABLES OF CHARACTERS USING PLTCHR
C#####
CHARACTER*1 C,CFONT(4),CFACE(3)
CHARACTER*3 CASC,CPRE,CSUF
CHARACTER*8 CTYP
DATA CFONT,CFACE,CTYP/'0','1','2','3',' ','&','$','!2!=!-.'/
DATA RL,RU,RM,RN,RR,RT,RV/.4,1.1,2.,2.05,9.2,.2,8.2,9.2/
DATA HC,HH/.4,.2/

C-----
PRINT*,'GRAPHICS DEVICE?'
READ(*,*)ITERM
PRINT*,'PAGE? [1-12]'
READ(*,*)IPAGE
C *** INITIATE GRAPHICS DEVICE
CALL PLOTN(0.,0.,15,ITERM) ! DEFINE GRAPHICS DEVICE
CALL PLOTN(0.,0.,0,0) ! INITIATE GRAPHICS DEVICE
C *** LINE-SEGMENTS
CALL PLOTN(RL,RT,3,0) ! BOX AROUND ENTIRE SET
CALL PLOTN(RL,RT,2,0)
CALL PLOTN(RR,RT,2,0)
CALL PLOTN(RR,RB,2,0)
CALL PLOTN(RL,RB,2,0)
CALL PLOTN(RM,RB,3,0) ! BETWEEN ASCII INDICES AND CHARACTERS
CALL PLOTN(RM,RT,2,0)
CALL PLOTN(RN,RB,3,0)
CALL PLOTN(RN,RT,2,0)
CALL PLOTN(RU,RB,3,0) ! BETWEEN PREFIX AND ASCII INDICES
CALL PLOTN(RU,RT,2,0)
C *** LABEL TOP OF PLOT
CALL PLTCHR(RL+.3,RT,HH,45.,0.,1,CTYP//'Suffix',-19)
CALL PLTCHR(RU+.3,RT,HH,45.,0.,1,CTYP//'ASCII (dec)',-19)
CALL PLTCHR(5.,RV,HH,0.,0.,4,CTYP//'Prefix:',-19)
C *** PLOT TABLE OF CHARACTERS
DO 1 K=1,8 ! 8 CHARACTERS PER PAGE
I=K+(IPAGE-1)*8+31 ! ASCII INDEX
C=CHAR(I)
C *** PLOT HORIZONTAL LINE-SEGMENTS
Y=RT-REAL(K)
CALL PLOTN(RL,Y,3,0)
CALL PLOTN(RR,Y,2,0)
C *** PLOT LEFT HEADINGS
Y=Y+.3
WRITE(CASC,'(I3)')I
CSUF=' '//C
IF (C.EQ.'!' .OR. C.EQ.'@' .OR. C.EQ.'#' .OR. C.EQ.'$' .OR. C.EQ.'%'
+ .OR. C.EQ.'~' .OR. C.EQ.'&' .OR. C.EQ.'_' .OR. C.EQ.'"' )
+ CSUF=' !'//C
IF(I.EQ.32)CSUF=' spc'
IF(I.EQ.127)CSUF=' del'
CALL PLTCHR(RM-.1,Y,HH,0.,0.,7,CTYP//CASC,-11)

```

```
      CALL PLTCHR(RU-.1,Y,HH,0.,0.,7,CTYP//CSUF,-11)
C *** PLOT CHARACTERS/TOP HEADINGS
      DO 25 IFONT=1,4 ! FOR EACH FONT
      DO 25 IFACE=1,3 ! FOR EACH FACE
      CPRE=' '//CFONT(IFONT)//CFACE(IFACE)
      X=RM+.6*REAL((IFONT-1)*3+IFACE)
      IF(K.EQ.1)THEN ! PLOT TOP HEADINGS
        CALL PLOTN(X,RB,3,0)
        CALL PLOTN(X,RT,2,0)
        CALL PLTCHR(X-.3,RT,HH,45.,0.,1,CTYP//CPRE,-11)
      ENDIF
      IF(I.NE.32.AND.I.NE.127)
+     CALL PLTCHR(X,Y,HC,0.,0.,7,CPRE//CSUF,-6)
25     CONTINUE
1     CONTINUE
C *** END OF PLOT
      CALL PLOTN(0.,0.,16,0) ! EXIT GRAPHICS
      END
```

Suffix ASCII		Prefix:												
		!0	!0&	!0\$!1	!1&	!1\$!2	!2&	!2\$!3	!3&	!3\$	
spc	32													
!!	33	!	!	!	!	!	!	!	!	!	!	!	!	
!"	34	"	☐	†	"	⊙	"	"	...	u	"	"	"	
!#	35	#	←	‡	#	♀	#	#	∅	v	#	#	#	
!\$	36	\$	↓	✳	\$	♀	\$	\$	©	w	\$	\$	\$	
!%	37	%	⊥	🌲	%	⊕	%	%	®	x	%	%	%	
!&	38	&	∠	♣	&	♂	&	&	£	y	&	&	&	
'	39	'	∴	👤	'	4	'	'	$\frac{3}{8}$	z	'	'	'	

		Prefix:												
Suffix	ASCII	!0	!0&	!0\$!1	!1&	!1\$!2	!2&	!2\$!3	!3&	!3\$	
(40	(⊂	♠	(h	(($\frac{1}{4}$	0	(((
)	41)	⊃	♥)	⊕))	$\frac{7}{8}$	1)))	
*	42	*	∪	♦	*	Ψ	*	*	$\frac{5}{8}$	2	*	*	*	
+	43	+	∩	♣	+	ℙ	+	+	$\frac{3}{8}$	3	+	+	+	
,	44	,	∈	♣	,	☾	,	,	$\frac{2}{8}$	4	,	,	,	
-	45	-	→	♣	-	♠	-	-	$\frac{1}{8}$	5	-	-	-	
.	46	.	↑	○	.	♠	.	.	$\frac{1}{3}$	6	.	.	.	
/	47	/	♫	□	/	♠	/	/	$\frac{1}{2}$	7	/	/	/	

Suffix		Prefix:												
ASCII		!0	!0&	!0\$!1	!1&	!1\$!2	!2&	!2\$!3	!3&	!3\$	
0	48	0	△	0	↗	0	0	.	0	0	0	0	0	
1	49	1	◇	1	↙	1	1)	1	1	1	1	1	
2	50	2	☆	2	↖	2	2)	2	2	2	2	2	
3	51	3	+	3	↘	3	3	o	3	3	3	3	3	
4	52	4	●	4	⋈	4	4	o	4	4	4	4	4	
5	53	5	■	5	♋	5	5	●	5	5	5	5	5	
6	54	6	▲	6	♋	6	6	#	6	6	6	6	6	
7	55	7	◀	7	♋	7	7	⋈	7	7	7	7	7	

		Prefix:												
Suffix		!0	!0&	!0\$!1	!1&	!1\$!2	!2&	!2\$!3	!3&	!3\$	
ASCII	(dec)													
8	56	8	8	▽	8	Ω	8	8	♭	8	8	8	8	
9	57	9	9	▷	9	♯	9	9	—	9	9	9	9	
:	58	:	∏	★	:	Ω	:	:	=	8	:	:	:	
;	59	;	Σ	+	;	♯	;	;	♯	9	;	;	;	
<	60	<	<	©	<	♯	<	<	♯	i	<	<	<	
=	61	=		☆	=	♯	=	=	♯	j	=	=	=	
>	62	>	>	🔔	>	♯	>	>	♯	m	>	>	>	
?	63	?	☆	§	?	♯	?	?	♯	n	?	?	?	

		Prefix:											
Suffix		!0	!0&	!0\$!1	!1&	!1\$!2	!2&	!2\$!3	!3&	!3\$
ASCII	(dec)												
!@	64	@	∞	◦	@	∞	◦	@	Т	◦	@	@	◦
A	65	A	A	À	A	A	À	A	A	A	A	À	A
B	66	B	B	Ɓ	B	B	Ɓ	B	Б	B	B	Ɓ	B
C	67	C	X	Ɔ	C	X	Ɔ	C	Ч	C	C	Ɔ	C
D	68	D	Δ	Ɖ	D	Δ	Ɖ	D	Д	D	D	Ɖ	D
E	69	E	E	Ǝ	E	E	Ǝ	E	E	E	E	Ǝ	E
F	70	F	ϕ	Ƒ	F	ϕ	Ƒ	F	Ф	F	F	Ƒ	F
G	71	G	Г	Ɠ	G	Г	Ɠ	G	Г	G	G	Ɠ	G

		Prefix:												
		Suffix ASCII (dec)												
		!0	!0&	!0\$!1	!1&	!1\$!2	!2&	!2\$!3	!3&	!3\$	
H	72	H	H	<i>H</i>	H	H	<i>H</i>	H	HO	H	H	<i>H</i>	H	
I	73	I	I	<i>I</i>	I	I	<i>I</i>	I	II	I	I	<i>I</i>	I	
J	74	J	<i>J</i>	<i>J</i>	J	<i>J</i>	<i>J</i>	J	JK	J	J	<i>J</i>	J	
K	75	K	K	<i>K</i>	K	K	<i>K</i>	K	K	K	K	<i>K</i>	K	
L	76	L	Λ	<i>L</i>	L	Λ	<i>L</i>	L	Л	L	L	<i>L</i>	L	
M	77	M	M	<i>M</i>	M	M	<i>M</i>	M	MM	M	M	<i>M</i>	M	
N	78	N	N	<i>N</i>	N	N	<i>N</i>	N	HN	N	N	<i>N</i>	N	
O	79	O	O	<i>O</i>	O	O	<i>O</i>	O	O	O	O	<i>O</i>	O	

Suffix		Prefix:											
ASCII	(dec)	!0	!0&	!0\$!1	!1&	!1\$!2	!2&	!2\$!3	!3&	!3\$
P	80	P	Π	ρ	P	Π	ρ	P	Π	P	P	Ⓟ	P
Q	81	Q	⊖	Q	⊕	Q	⊖	Q	⊖	Q	Q	Ⓟ	Q
R	82	R	P	R	R	P	R	R	P	R	R	℞	R
S	83	S	Σ	S	Σ	ſ	S	C	S	S	ſ	S	S
T	84	T	T	T	T	T	T	T	T	T	T	Ⓟ	T
U	85	U	Υ	U	Υ	U	U	Y	U	U	Ⓟ	U	U
V	86	V	∇	V	∇	V	V	B	V	V	Ⓟ	V	V
W	87	W	Ω	W	Ω	W	W	W	W	W	Ⓟ	W	W

		Prefix:												
Suffix ASCII (dec)		!0	!0&	!0\$!1	!1&	!1\$!2	!2&	!2\$!3	!3&	!3\$	
X	88	X	Ξ	ℳ	X	Ξ	ℳ	X	X	X	X	ℳ	X	
Y	89	Y	Ψ	ϳ	Y	Ψ	ϳ	Y	Я	Y	Y	ϳ	Y	
Z	90	Z	Z	ℷ	Z	Z	ℷ	Z	3	Z	Z	Z	Z	
[91	[√	‘	[√	‘	[Ы	‘	[[‘	
\	92	\	×	÷	\	×	÷	\	Ь	÷	\	\	÷	
]	93]	√	’]	√	’]	Э	’]]	’	
!^	94	^	˘	±	^	˘	±	^	˘	±	^	^	±	
!_	95	_	˘	≠	_	˘	≠	_	˘	≠	_	_	≠	

		Prefix:											
Suffix		(dec)											
ASCII		!0	!0&	!0\$!1	!1&	!1\$!2	!2&	!2\$!3	!3&	!3\$
'	96	'	≈	≠	'	≈	≠	'	Ђ	≠	'	'	≠
a	97	a	α	a	a	α	a	a	a	a	a	а	α
b	98	b	β	b	b	β	b	b	б	b	b	б	b
c	99	c	χ	c	c	χ	c	c	ч	c	c	с	c
d	100	d	δ	d	d	δ	d	d	д	d	d	д	d
e	101	e	ε	e	e	ε	e	e	e	e	e	е	e
f	102	f	φ	f	f	φ	f	f	ф	f	f	ф	f
g	103	g	γ	g	g	γ	g	g	Г	g	g	г	g

Suffix		Prefix:											
ASCII		!0	!0&	!0\$!1	!1&	!1\$!2	!2&	!2\$!3	!3&	!3\$
(dec)													
p	112	p	π	p	p	π	p	Π	p	p	p	p	p
q	113	q	ϑ	q	q	ϑ	q	Ц	q	q	q	q	q
r	114	r	ρ	r	r	ρ	r	P	r	r	r	r	r
s	115	S	σ	s	S	σ	s	S	C	S	S	§	S
t	116	t	τ	t	t	τ	t	†	T	t	t	†	t
u	117	u	υ	u	u	υ	u	u	y	u	u	u	u
v	118	V	◻	v	V	ℵ	v	V	B	v	V	v	v
w	119	W	ω	w	W	ω	w	W	III	w	W	w	w

Appendix B

FIDDLER : HELP files

MAIN help file: HMAIN.HLP

```
=====
FIDDLER - VERSION 2.0 - 23-AUG-1991
```

```
(C) RICHARD BOOTH
```

```

SHERMAN FAIRCHILD LAB          IMEC,VZW
  LEHIGH UNIVERSITY            ==>>  KAPELDREEF 75
BETHLEHEM, PA 18015 USA        B3001 LEUVEN, BELGIUM
  (215)-758-3950                (016)-281-238
E-MAIL: RVBO@LEHIGH.BITNET     BOOTH@IMEC.BE
=====
```

```
%
```

```
=====
FIDDLER and GRAPHER are offered to the public domain. The following
conditions must be observed:
```

1. No portion of the software or manual should be used for commercial gain without the written consent of the author.
2. Modifications to the software can be freely made. It is suggested that modified versions of FIDDLER be renamed to reflect the fact that changes have been made. Please notify the author of any changes, improvements, implementations on other computers or terminals, and applications. Please also report any bugs found in the *original* software.
3. The author is not responsible for any loss of data, or other problems which are encountered by the use of FIDDLER/GRAPHER.

```
%
```

```
FIDDLER IS A PROGRAM WHICH CAN BE USED TO:
```

- * EDIT EXISTING DATA
- * CREATE NEW DATA:
 - SINGLE POINT ENTRY
 - FUNCTION SAMPLING
 - OPERATIONS USING EXISTING DATA
- * PLOT DATA:
 - TERMINAL (TEKTRONIX TERMINALS OR TERMINAL EMULATOR, OR SEIKO)
 - POSTSCRIPT PRINTER
 - HPGL PLOTTER
- * VIEW DATA:
 - SPREADSHEET-LIKE VIEW WITH EDITOR
 - PLOT WINDOWING, LIN/LOG MODE CHANGING

```
IT IS BASED ON THE GRAPHER PLOTTING PACKAGE WHICH SUPPORTS
XY, CONTOUR, SURFACE, SPACELINE, HISTOGRAM, PROBABILITY, AND VECTOR PLOTTING
```

```
%
```

```
=====
FIDDLER MENU STRUCTURE:
```

```
Menu keywords are surrounded by [ ].
```

Menu entry summaries are surrounded by ().
 Menu/Command is accessed by capitalized character.
 Menu level is incorporated into displayed menu border.

=====

MENU LEVEL:

0 1 2 3 4
 =====

```
[main]__Help
|__Plot
|__Save
|__Load
|__Data
|__Terminal
|__eXecute
|__Batch
|
|_[Change]_____View
|          |__Name
|          |__Titles
|          |_[Channels]_____(x,y,z channels)
|          |_[Window]_____(plotting limits)
|          |_[Symbols]_____(line-types,symbols,pens)
|          |_[Orientation]_(axes-orientation)
|          |_[Pen-control]_(secondary sweep control)
|          |_[Labels]_____(label sizes,format)
|          |_[Mode]_____(plotting mode)
|          |_[Extra]_____(specific plotting parameters)
|
|_[Edit]_____Insert
|          |__Delete
|          |__3-convert
|          |__Sort
|          |__Linear-regression
|          |__Polynomial-regression
|          |_[View]_____(spreadsheet commands)
|          |_[Operate]_____Endpoints
|                          |__eXchange
|                          |__Index
|                          |__Constant
|                          |_[Unary]_____(unary operations)
|                          |_[Binary]_____(binary operations)
|                          |_[Konstant]___(channel/constant operations)
|
|_[Import]_____Ascii
|          |__Suprem2
|          |__supreM3
|          |__Iv
|          |__1-d
|          |__3-d
|          |__simPar
|          |__pRism
|
```

```

|_[Function]__Help
|      |__View
|      |__T-limits
|      |__Enter
|      |__Insert
|      |__Delete
|      |__Use
|      |__Reset
|      |__Load
|      |__Save
|      |__Go
|      |__Plot
|
|_[Make]_____Hpgl
|      |__Post
|      |__Ln03
|      |__Calc
|      |__Ascii
|
|_[Add]_____Help
|      |__View
|      |__Enter
|      |__Insert
|      |__Delete
|      |__Use
|      |__Kill
|      |__Place
|
|_[User]_____ [A-demo]
|      |__B-calculator
|      |__C-calendar
|      |_[D-draw]
|      |_[E-poster]

```

%

```

=====
FIDDLER-format file information:
First
character
in line:      Rest of line:
=====
| $           main title
| %           sub-title
| !           channel name
| <           lower point index of following points
| >           upper point index of following points
| space       number
| @           label
| next line: LENG,LORG,X,Y,HEIGHT,ANGLE,SLANT
|   where: LENG  number of characters in the label
|           LORG  label orientation [1-9]
|           X,Y   origin of label in plot units [1-10]

```

```

|         HEIGHT height of label in plot units [1-10]
|         ANGLE  angle  of label in degrees
|         SLANT  slant  of label in degrees
| *         (blank - plotting parameters follow)
| next line:IMODE, IGRID, IPENC,IASCA, ILCLR,IOPAQ,KSTEPL,KSTEPH,
|           IPCHAN,IORIENT,ISKIP,IORIEN3,NSURF,NCONT,ISMODE
| next line:ANGLE,DXHIST
| next 6 lines (for each of the six curves):
|           IX,IY,IZ,ILINE,IPEN,ILSYMB,SYMB
| next line:XLO,XHI,XMIN,XMAX
| next line:XAVG,XSTD
| next line:YLO,YHI,YMIN,YMAX
| next line:YAVG,YSTD
| next line:ZLO,ZHI,ZMIN,ZMAX
| next line:ZAVG,ZSTD
| next line:XTYP,YTYP
|     where:IMODE  plotting mode:
|               0=blank
|               1=xyplot
|               2=histogram
|               3=probability
|               4=contour
|               5=vectors
|               6=surface
|               7=spaceline
|               8=contours with labels
|     IGRID  grid flag: (0=off,1=on)
|     IPENC  pen_control flag: (0=off,1=on)
|     IASCA  auto_scale flag: (0=off,1=on)
|     ILCLR  label-editor clear-label flag: (0=off,1=on)
|     IOPAQ  surface opacity flag: (0=off,1=on)
|     KSTEPL lower pen_control step
|     KSTEPH upper pen_control step
|     IPCHAN pen_control channel
|     IORIENT quadrant for 2d plot flipping
|     ISKIP  spaceline points to skip between verticals
|     IORIEN3 octant for 3d plot flipping
|     NCONT  number of contours
|     NSURF  number of surface grid points
|     ISMODE surface plot mode (1:top 2:bottom 3:both surfaces)
|     ANGLE  rotation angle of 3d plots
|     DXHIST histogram interval
|     IX     x-channel of curve
|     IY     y-channel of curve
|     IZ     z-channel of curve
|     ILINE  line type of curve
|     IPEN   pen color of curve
|     ILSYMB line-plus-symbol flag: (0=no,1=yes)
|     SYMB   xyplot plotting symbol
|     XLO,XHI,YLO,YHI,ZLO,ZHI      : plotting window
|     XMIN,XMAX,YMIN,YMAX,ZMIN,ZMAX : limits of data curve 1
|     XAVG,XSTD,YAVG,YSTD,ZAVG,ZSTD : average/stddev of data curve 1

```

=====
%

IMPORT help file: HIMPT.HLP

IMPORT ASCII DATA FILE
EACH LINE OF THE DATA FILE HAS ONE OR MORE
NUMBERS (NO SPECIFIC FORMAT) SEPARATED BY
COMMAS OR SPACES (OR MULTIPLES THEREOF)
ALL LINES OF THE DATA FILE MUST HAVE THE SAME
NUMBER OF VALUES

EXAMPLE:

```
1.23  2.35 -2.2 2E-4 4.5555555          3
-1   2      2    4E-7   5              9
  2   1.23  3    3E-9   2.222         10
... etc ...
```

IT IS ASSUMED THAT THERE ARE ONLY 80 COLUMNS OF ASCII DATA (PUNCH CARD-LIKE)
IF THERE ARE MORE THAN THIS, THEN THE PROGRAM ITSELF MUST BE MODIFIED
- OR - REWRITE YOUR ASCII DATA SO THAT THERE ARE ONLY 80 COLUMNS
%

FUNCTION help file: HFUNC.HLP

Function Entry:

1. The independent parameter is the variable T.
 2. The dependent parameter is the variable F.
 3. The function you will enter is a series of FORTRAN lines. These will be embedded in a loop with the parameter T running between the limits TMIN and TMAX which you also must specify. You also choose lin. or log. sampling. The source file is \$LOCAL.SAMFUN
 4. EXAMPLE:
(intent) :
f(t)=4.3*t + 3*t^2 + 2.8*exp(-2*(t-1))
(implementation) :
A=4.3*T
B=3.*T*T
C=2.8*EXP(-2.*(T-1.))
F=A+B+C
(OR ...)
F=4.3*T+3*T**2.+2.8*EXP(-2*(T-1))
 5. So far there is no error-trapping, so a bad function will cause the program to bomb
- %

LABEL-EDITOR help file: HLABE.HLP

LABEL EDITOR:

USE THE MENU COMMANDS TO ENTER,INSERT,OR DELETE LABELS

THERE IS A MAXIMUM OF 20 LABELS

USE PLOT TO MOVE,ORIENT,SIZE,OR SLANT LABELS

%

PLOTTER COMMANDS FOR EDITING LABELS

> SELECT NEXT LABEL

< SELECT PREV LABEL

L MOVE LEFT

R MOVE RIGHT

U MOVE UP

D MOVE DOWN

1 .. 9 SIZE OF MOVE/ANGLE INCREMENTS

C CLEAR SCREEN AND REPLOT

X SET X (0. .. 10.)

Y SET Y (0. .. 10.)

A SET ANGLE (DEGREES)

S SET SLANT (DEGREES)

L SET LONG (1 .. 9)

H SET HEIGHT

INCREMENT HEIGHT

\ INCREMENT ANGLE

% INCREMENT SLANT

Q QUIT

Appendix C

FIDDLER : Subroutine/function list

R=REAL FUNCTION
 I=INTEGER FUNCTION
 L=LOGICAL FUNCTION
 C=CHARACTER*1 FUNCTION
 Q=CHARACTER*25 FUNCTION
 S=SUBROUTINE
 |

FILE:	V ROUTINE:	COMMENTS:
=====		
d4010.fsb		TEKTRONIX 40XX TERMINAL DRIVER
	S T010NEW	OPEN PLOT
	S T010PLT (IX,IY,ISUB)	MOVE, DRAW
	S T010PEN (IPEN)	CHANGE PEN COLOR INDEX
	S T010PAT (IPAT)	CHANGE PATTERN INDEX
	S T010END	CLOSE PLOT
	S T010CLS	CLEAR SCREEN
	S T010GRA	GOTO GRAPHICS MODE
	S T010ALP	GOTO ALPHANUMERIC MODE
d4205.fsb		TEKTRONIX 41XX 42XX TERMINAL DRIVER
	S T205NEW	OPEN PLOT
	S T205PLT (IX,IY,ISUB)	MOVE, DRAW
	S T205PEN (IPEN)	CHANGE PEN COLOR INDEX
	S T205PAT (IPAT)	CHANGE PATTERN INDEX
	S T205END	CLOSE PLOT
	S T205CLS	CLEAR SCREEN
	S T205GRA	GOTO GRAPHICS MODE
	S T205ALP	GOTO ALPHANUMERIC MODE
dcalc.fsb		CALC (generic) DRIVER
	S CALCNEW	OPEN PLOT
	S CALCPLT (IX,IY,ISUB)	MOVE, DRAW
	S CALCPEN (IPEN)	CHANGE PEN COLOR INDEX
	S CALCPAT (IPAT)	CHANGE PATTERN INDEX
	S CALCEND	CLOSE PLOT
dhppl.fsb		HPGL PLOTTER DRIVER
	S HPGLNEW	OPEN PLOT
	S HPGLPLT (IX,IY,ISUB)	MOVE, DRAW
	S HPGLPEN (IPEN)	CHANGE PEN COLOR INDEX
	S HPGLPAT (IPAT)	CHANGE PATTERN INDEX
	S HPGLEND	CLOSE PLOT
dln03.fsb		LN03 PRINTER DRIVER

	S LN03NEW	OPEN PLOT
	S LN03PLT (IX,IY,ISUB)	MOVE, DRAW
	S LN03PEN (IPEN)	CHANGE PEN COLOR INDEX
	S LN03PAT (IPAT)	CHANGE PATTERN INDEX
	S LN03END	CLOSE PLOT
	S LN03CLS	CLEAR SCREEN
	S LN03GRA	GOTO GRAPHICS MODE
	S LN03ALP	GOTO ALPHANUMERIC MODE
	S LN03WRI (STRING)	BUFFERED WRITE
dpost.fsb		POSTSCRIPT DRIVER
	S POSTNEW	OPEN PLOT
	S POSTEND	CLOSE PLOT
	S POSTPLT (IX,IY,ISUB)	MOVE, DRAW
	S POSTPEN (IPEN)	CHANGE PEN INDEX
	S POSTPAT (IPAT)	CHANGE PATTERN INDEX
	S POSTWRT (STRING)	BUFFERED WRITE
dseik.fsb		SEIKO 110X TERMINAL DRIVER
	S SEIKNEW	OPEN PLOT
	S SEIKPLT (IX,IY,ISUB)	MOVE, DRAW
	S SEIKPEN (IPEN)	CHANGE PEN COLOR INDEX
	S SEIKPAT (IPAT)	CHANGE PATTERN INDEX
	S SEIKEND	CLOSE PLOT
	S SEIKCLS	CLEAR SCREEN
	S SEIKGRA	GOTO GRAPHICS MODE
	S SEIKALP	GOTO ALPHANUMERIC MODE
dv240.fsb		DEC VT-240 TERMINAL DRIVER
	S V240NEW	OPEN PLOT
	S V240PLT (IX,IY,ISUB)	MOVE, DRAW
	S V240PEN (IPEN)	CHANGE PEN COLOR INDEX
	S V240PAT (IPAT)	CHANGE PATTERN INDEX
	S V240END	CLOSE PLOT
	S V240CLS	CLEAR SCREEN
	S V240GRA	GOTO GRAPHICS MODE
	S V240ALP	GOTO ALPHANUMERIC MODE
f3c3d.fsb	S F3C3D (IX,IY,IZ)	CONVERT 3-CHANNEL TO 3-D DATA FORMAT
fchan.fsb	S FCHAN	[CHANGE.CHANNELS]
fdefa.fsb	S FDEFA	SET DEFAULT FIDDLER-INFORMATION VALUES

fdisk.fsb	S	FDISK (CACTION,CFILE)	SAVE/LOAD FIDDLER-FORMAT FILES
fedit.fsb	S	FEDIT	[EDIT] DATA
ffunc.fsb	S	FFUNC	[FUNCTION] SAMPLE
fimp1.fsb	S	FIMP1	IMPORT GEPLIT-1 DATA FILES
fimp3.fsb	S	FIMP3	IMPORT GEPLIT-3 DATA FILES
fimpa.fsb	S	FIMPA	IMPORT ASCII DATA FILES
fimpi.fsb	S	FIMPI	IMPORT MINIMOS I-V FILES
fimpm.fsb	S	FIMPM	IMPORT TMA-SUPREM-3 SAVE FILES
	I	IRDTMA (FILENAM,NPTS,YAXIS, RDON,RACC,MAT)	IMPORT TMA-SUPREM-3 SAVE FILES
fimpp.fsb	S	FIMPP	IMPORT SIMPAR DATA FILES
fimpr.fsb	S	FIMPR	IMPORT PRISM SUMMARY FILES
fimps.fsb	S	FIMPS	IMPORT SUPREM-2 SAVE FILES
fimpt.fsb	S	FIMPT	[IMPORT] DATA
finil.fsb	S	FINIL	INITIALIZE LABELS
finit.fsb	S	FINIT	INITIALIZE PARAMETERS
flabe.fsb	S	FLABE	[CHANGE.LABELING-OPTIONS]
fledi.fsb	S	FLEDI	[ADD] LABELS
	S	SETLAB (J)	SET LABEL PARAMETERS
flreg.fsb	S	FLREG	LINEAR REGRESSION
fmain.fsb	S	FIDDLER	FIDDLER INTERACTIVE PROGRAM MAIN
fmake.fsb	S	FMAKE	[MAKE] HARD COPY
fmenu.fsb			FIDDLER MENUS
	C	MENU1	
	C	MENU2	
	C	MENU3	
	C	MENU4	
	C	MENU5	
	C	MENU6	
	C	MENU7	
	C	MENU8	
	C	MENU9	
	C	MENUA	
	C	MENUB	
	C	MENUC	
	C	MENUD	
	C	MENUE	
	C	MENUF	
	C	MENUG	
	C	MENUH	
	C	MENUI	
	C	MENUJ	
	C	MENUK	
	C	MENUL	
fmode.fsb	S	FMODE	[CHANGE.MODE]
foper.fsb	S	FOPER	[EDIT.OPERATE]
fornt.fsb	S	FORNT	[CHANGE.ORIENTATION]
fpars.fsb	S	FPARS	[CHANGE]
fpenc.fsb	S	FPENC	[CHANGE.PEN-CONTROL]

```

fplot.fsb S FPLOT          [PLOT] DATA
          S PLTFIG        PLOT A FIGURE
            (X,Y,H,A,S,ITYPE)
          S ROTATE        ROTATE COORDINATES
            (XO,YO,XC,YC,
             XA,YA,ANGLE)
fppar.fsb S FPPAR          [CHANGE.EXTRA] PLOTTING PARAMETERS
fpreg.fsb S FPREG          POLYNOMIAL REGRESSION
fsing.fsb S FSING          [DATA] ENTER SINGLE POINTS
fsort.fsb S FSORT          SORT DATA ARRAYS
fstat.fsb S FSTATA        GET PLOTTING LIMITS AVERAGE, STDD
          S FSTATC        GET STATISTICS OF PLOTTING CHANNELS
          S FSTATW        GET PLOTTING WINDOW
fsymb.fsb S FSYMB          [CHANGE.SYMBOLS]
fuser.fsb S FUSER          [USER] APPLICATION PROGRAMS
futil.fsb
          S BEEP          CHAR(07)
          S CLS          CLEAR SCREEN
          S FIDFILE        GENERATE FIDDLER-FORMAT DATA FILE
            (DATARAY,NP,NCHAN,
             NPTS,CNAME,MTIT,STIT)
          S HELP          PRINT HELP FILES
            (FILENAM)
          S HITAKEY        PROMPT, WAIT FOR A CARRIAGE RETURN
          S HLINE          PLOT LINE-TYPES, PEN COLORS
          S ULREG          LINEAR REGRESSION
            (X,Y,I1,I2,UNKN,
             XTYP,YTYP,B,Q,R,
             QX,QY,DX,DY,DXY)
          S UMATI          MATRIX INVERSION
            (A,AI,N,NP,DET)
          S UMATM          MATRIX MULTIPLICATION
            (A,B,C,NA,NB,NC,
             NPA,NPB,NPC)
          S UNEWP          PRINT STRING AT PROGRAM START
            (STRING)
          S UPREG          POLYNOMIAL REGRESSION
            (X,Y,W,I1,I2,M)
          S UPSET          PROMPT, SET UP TERMINAL
            (IT)
          S STRUPC          STRING TO UPPER-CASE
            (C)
          C DEFONCR        RETURN DEFAULT IF CR PRESSED
            (STRING,DEFANS)
          C UMENU          PRINT MENU, ACCEPT RESPONSE
            (LAB,TIT,RES,N,L,
             LEV,LST,ICB,ICT,ICM)
          R UNUDE          NON-UNIFORM GRID DIFFERENTIATION
            (X,Y,NL,NH,IAT,IDEGREE)
          R XTOI          X TO POWER I
            (X,I)
          C CUPCASE        CHARACTER TO UPPER-CASE

```

```

(C)
L LNUMER          IF C = "0" TO "9" OR "-"
(C)
I NEXTCHR        FIND NEXT NON-DELIMITER CHARACTER
  (BUFFER,I,J)
I NEXTDEL        FIND NEXT DELIMITER CHARACTER
  (BUFFER,I,J)
I NEXTSCH        FIND NEXT SPECIFIC CHARACTER
  (BUFFER,I,J,C)
I LASTCHR        FIND LAST CHARACTER
  (BUFFER)
L GETKEYS        GET STRING FROM KEYWORD=STRING
  (KEYWORD,BUFFER,
   I,J,IBEG,IEND)
L GETKEYR        GET REAL FROM KEYWORD=REAL
  (KEYWORD,BUFFER,
   I,J,REALNO)
L GETKEYI        GET INTEGER FROM KEYWORD=INTEGER
  (KEYWORD,BUFFER,
   I,J,INTNO)
L GETKEYL        GET LOGICAL FROM KEYWORD='Y','N'
  (KEYWORD,BUFFER,
   I,J,LOGVAL)
L FINDKEY        FIND INDICES OF KEYWORD
  (KEYWORD,BUFFER,
   I,J,IBEG,IEND)
fview.fsb S FVIEW          [EDIT.VIEW]
fwind.fsb S FWIND         [CHANGE.WINDOW]
gcon2.fsb S CONTOU2       CONTOUR PLOT WITH LABELS
  (ARRAY,XAXIS,YAXIS,
   CONT,NA,NX,NY,NC,
   X1,X2,Y1,Y2,FGRID,
   XTIT,YTIT,CTIT,STIT,
   IORIG)
S ANGLEN        GET ANGLE AND LENGTH OF VECTOR DX,DY
  (ANGLE,RLENG,DX,DY)
gcon3.fsb S CONTOU3       CONTOUR PLOT WITH COLOR BANDS
  (ARRAY,XAXIS,YAXIS,
   CONT,NA,NX,NY,NC,
   X1,X2,Y1,Y2,FSCALE,
   XTIT,YTIT,CTIT,STIT,
   IORIG)
gcont.fsb S CONTOUR       CONTOUR PLOT
  (ARRAY,XAXIS,YAXIS,
   CONT,NA,NX,NY,NC,
   X1,X2,Y1,Y2,FGRID,
   XTIT,YTIT,CTIT,STIT,
   IORIG)
gfrm2.fsb S FRAME2D       FRAME FOR 1-D, 2-D PLOTS
  (X1,X2,Y1,Y2,XTYP,YTYP,
   XTIT,YTIT,CTIT,STIT,
   FGRID,IORIG)

```

```

gfrm3.fsb S FRAME3D          FRAME FOR 3-D PLOTS
          (X1,X2,Y1,Y2,Z1,Z2,
           XTIT,YTIT,ZTIT,
           CTIT,STIT,PHI,THETA,
           IORIG,I PROJ)
ghist.fsb S HISTOGM        HISTOGRAM
          (X,NX,X1,X2,DX,XTIT,
           CTIT,STIT,IPEN)
ginit.fsb S GINIT          INITIALIZE GRAPHICS PARAMETERS
gpltc.fsb S PLTCHR         PLOT CHARACTER STRING
          (XP,YP,HP,ROTATE,
           SLANT,LORG,STRING,NC)
gpltn.fsb S PLOTN          PLOTTING PRIMITIVES
          (X,Y,ICMD,ISUB)
          L CLIPPER        CLIPPING ROUTINE
          (X1,Y1,X2,Y2,
           XMIN,XMAX,YMIN,YMAX)
gpltv.fsb S PLTVAL        PLOT VALUES
          (XP,YP,HP,ROTATE,
           SLANT,LORG,FORMA,
           VALUE)
gprob.fsb S PROBPLT       PROBABILITY PLOT
          (X,NX,SYMB,XTIT,
           CTIT,STIT,FLHS,FRHS)
          S SORT           SORT VALUES
          (X,Y,NPTS)
          R FUNCINV        INVERT FUNCTION
          (X)
          R FUNCT          FUNCTION TO INVERT
          (X)
          R ERFC           ERROR FUNCTION
          (X)
gspac.fsb S SPACELN       SPACELINE PLOT
          (XVALUES,YVALUES,
           ZVALUES,NVAL,
           X1,X2,Y1,Y2,Z1,Z2,
           XTIT,YTIT,ZTIT,
           CTIT,STIT,SYMB,
           ISKIP,ANGLE,IORIG)
gsur2.fsb S SURFAC2       SURFACE PLOT
          (ARRAY,XAXIS,YAXIS,
           NA,NX,NY,NG,
           X1,X2,Y1,Y2,
           XTIT,YTIT,ZTIT,
           CTIT,STIT,PHI,THETA,
           IMODE,FOPAQ,IORIG)
gsurf.fsb S SURFACE       SURFACE PLOT (ISOMETRIC)
          (ARRAY,XAXIS,YAXIS,
           NA,NX,NY,NG,
           X1,X2,Y1,Y2,
           XTIT,YTIT,ZTIT,
           CTIT,STIT,ANGLE,

```

```

        IMODE,FOPAQ,IORIG)
gutil.fsb      GRAPHICS ROUTINE UTILITIES
S MINAX1      MINIMUM, MAXIMUM OF X
  (X,NX,XMIN,XMAX)
S MINAX2      MINIMUM, MAXIMUM OF A
  (A,NA,NX,NY,AMIN,AMAX)
S SWITCH      INTERCHANGE A AND B
  (A,B)
S FINDEX      LOCATE INDEX OF X : XI<VALUE<XI+1
  (X,N,VALUE,I,A,B)
S MAPPER      INTERPOLATE VALUES OF A ONTO B
  (A,B,X,Y,NA,NB,NX,NY,
  NG,X1,X2,Y1,Y2)
S SCALER      FIND VALUES FOR AXES
  (XLO,XHI,XTYP,
  XMIN,XMAX,XDEL,NDIV)
S WAITFCR     WAIT FOR CR AT END OF PLOT
S T2D         2-D TRANSFORMATION
  (X,Y,U,V)
S T3D         3-D TRANSFORMATION
  (X,Y,Z,U,V)
I IFTRUE      0 IF L IS FALSE 1 IF L IS TRUE
  (L)
I ILASTA      LAST CHARACTER OF BUFFER
  (BUFFER)
gvect.fsb S VECTORS      VECTOR PLOT
  (ARRAYX,ARRAYY,
  XAXIS,YAXIS,
  NA,NX,NY,NG,
  X1,X2,Y1,Y2,
  XTIT,YTIT,CTIT,STIT,
  IORIG)
gxype.fsb S XYPLOT      X-Y PLOT WITH ERROR BARS
  (XVALUES,YVALUES,
  EVALUES,NVAL,
  X1,X2,Y1,Y2,XTYP,YTYP,
  XTIT,YTIT,CTIT,STIT,
  SYMB,LAXES,LLAST,
  LGRID,IORIG,IPEN,ILIN)
gxypl.fsb S XYPLOT      X-Y PLOT
  (XVALUES,YVALUES,NVAL,
  X1,X2,Y1,Y2,XTYP,YTYP,
  XTIT,YTIT,CTIT,STIT,
  SYMB,LAXES,LLAST,
  LGRID,IORIG,IPEN,ILIN)
speci.fsb      SYSTEM-SPECIFIC ROUTINES
S SYSCMD      PERFORM SYSTEM COMMAND
  (C)
S LOGUSE      LOG PROGRAM USAGE
  (CBE)
S GETEVAR     GET ENVIRONMENTAL PARAMETER
  (CNAME,CVALU)

```

	S NUMARGS	GET COMMAND-LINE PARAMETER
	(IARGCNT)	
	S SINIT	INITIALIZE SYSTEM-DEPENDENT PARAMETERS
	S BTERMC	OPEN BUFFERED TERMINAL
	S BTERMD	CLOSE BUFFERED TERMINAL
	S BUFPACK	PACK BUFFER FOR BUFFERED TERMINAL OUTPUT
	(CVAL)	
	S RE	READ STRING FROM KEYBOARD
	(STRING)	
	S WR	WRITE STRING TO TERMINAL
	(STRING)	
	S WRITEPT	PRINT BUFFER TO BUFFERED TERMINAL
	(STRING,NCHR)	
	R SYSRND	GET RANDOM NUMBER FROM SYSTEM-SUPPLIED RNG
	(ISEED)	
	R SYSSEC	GET SYSTEM TIME
	(TIMEO)	
	R SYSCPU	GET CPU TIME
	Q CTIMDAT	GET DATE/TIME
	I IRD1E	GET 1-CHARACTER INTEGER VIA HOTKEY
	C URD1C	GET 1-CHARACTER CHARACTER VIA HOTKEY
	I IRD1D	GET 1-CHARACTER INTEGER DIGIT VIA HOTKEY
	I INTREAD	READ INTEGER FROM KEYBOARD
	R REAREAD	READ REAL FROM KEYBOARD
zcalc.fsb	S CALCULA	CALCULATOR APPLICATION
zcale.fsb	S ZCALE	CALENDAR APPLICATION
zchrt.fsb	S ZCHRT	POSTER-COMPOSER APPLICATION
	S PLOTBOX	PLOT BOUNDING BOX
	(XC,YC,XD,YD)	
	C MENCHRT	MENU FOR POSTER-COMPOSER
zflow.fsb	S ZFLOW	FLOW-CHART APPLICATION
	S PLOTFLO	PLOT STRING WITH BOX
	S PLOTSQR	PLOT SQUARE
	(X,Y,W,H,FPAT)	
	S PLOT CIR	PLOT CIRCLE
	(X,Y,H,FPAT)	
	S PLOTVEC	PLOT VECTOR
	(X1,Y1,X2,Y2,IDIR,H)	
	S ROTAT	ROTATE COORDINATE ABOUT ANGLE
	(XO,YO,XC,YC,	
	XA,YA,ANGLE)	
	R RANGLE	DETERMINE ANGLE OF VECTOR DX,DY
	(X1,Y1,X2,Y2)	
	C MENFLOW	MENU FOR FLOW
zpplt.fsb	S PROGPLT	DEMONSTRATION APPLICATION
	C MENUPP	MENU FOR DEMONSTRATION

Appendix D

FIDDLER : Installation

D.0.9 PREPARATION OF SUBDIRECTORIES FOR FIDDLER INSTALLATION

MSDOS FORMAT DISK/PKZIPPED DISTRIBUTION

1. Each directory on the MSDOS format disk contains a PKZIPPED file which must be uncompressed locally before uploading it to the system where Fiddler is to be installed. Use PKUNZIP followed by the file name to extract all of the files which are contained in the archive file. Since there will be no room on the diskette to receive the extracted files, they should be extracted to another disk. Keep each set of files separate so that they can be loaded into separate directories on the workstation or mainframe.
2. Create the following directory structure on the workstation or mainframe:

(UNIX environment):	(VMS environment):	
fiddler/BACK	[.fiddler.back]	backup
fiddler/CODE	[.fiddler.code]	source code
fiddler/DOCS	[.fiddler.docs]	manual
fiddler/EXAM	[.fiddler.exam]	applications
fiddler/FONT	[.fiddler.font]	ascii font file
fiddler/HELP	[.fiddler.help]	help files
fiddler/MAKE	[.fiddler.make]	script files
fiddler/MISC	[.fiddler.misc]	filter, misc
fiddler/XAPO/SPECIFIC	[.fiddler.xapo.specific]	apollo specific
fiddler/XCVX/SPECIFIC	[.fiddler.xcvx.specific]	convex specific
fiddler/XDEC/SPECIFIC	[.fiddler.xdec.specific]	dec specific
fiddler/XSNY/SPECIFIC	[.fiddler.xsny.specific]	sony specific
fiddler/XSUN/SPECIFIC	[.fiddler.xsun.specific]	sun specific
fiddler/XVAX/SPECIFIC	[.fiddler.xvax.specific]	vax specific

3. Upload all of the files from the respective PC archive to the workstation or mainframe subdirectory.


```
Subject: FID_TZU.af [fiddler file: 6 / 15]
Subject: FID_TZU.ag [fiddler file: 7 / 15]
Subject: FID_TZU.ah [fiddler file: 8 / 15]
Subject: FID_TZU.ai [fiddler file: 9 / 15]
Subject: FID_TZU.aj [fiddler file: 10 / 15]
Subject: FID_TZU.ak [fiddler file: 11 / 15]
Subject: FID_TZU.al [fiddler file: 12 / 15]
Subject: FID_TZU.am [fiddler file: 13 / 15]
Subject: FID_TZU.an [fiddler file: 14 / 15]
Subject: FID_TZU.ao [fiddler file: 15 / 15]
```

4. The result of uuencoding is the file FIDDLER.tar.Z This is a compressed tape-archive. First uncompress it:

```
uncompress FIDDLER.tar.Z
```

5. The result is FIDDLER.tar Put this file in the directory where you would like the fiddler main directory to reside. Change to this directory and issue

```
tar -xvf FIDDLER.tar
```

6. The following file structure should be set up. If tar gives protection errors, you may have to create these directories yourself first:

```
fiddler/BACK
fiddler/CODE
fiddler/DOCS
fiddler/EXAM
fiddler/FONT
fiddler/HELP
fiddler/MAKE
fiddler/MISC
fiddler/XAPO/SPECIFIC
fiddler/XCVX/SPECIFIC
fiddler/XDEC/SPECIFIC
fiddler/XSNY/SPECIFIC
fiddler/XSUN/SPECIFIC
fiddler/XVAX/SPECIFIC
```

D.0.10 INSTALLATION

UNIX

1. Change to the directory fiddler/MAKE. The script gethosttype is used to automatically determine the host type (Apollo Convex DEC-risc SONY or SUN). Verify that this program works, or set the host type keyword directly in the first script line. It must be one of (APO CVX DEC SONY or SUN).
2. Everything should be ok to compile, but just to be sure:
 - (a) Check to see that the proper directory path to the fortran compiler is correctly specified in make.unx
 - (b) The line defining LOB (logging objects) should be commented out, since these programs do not come with the distribution (they log program use).
 - (c) Check speci.fsb 's routine LOGGER to see that the two calls to `START_LOG` and `STOP_LOG` are commented out. Again, these are for logging program use.

3. To compile the entire library:

(make.unx should be executable)

```
make.unx -A -C -F
```

The parameter -A indicates that the library AFIDD.a, which contains all of the object files, is to be created. The parameter -C indicates that the interactive FIDDLER program is to be compiled. the parameter -F indicates that the binary font file is to be created from the ASCII font file.

The file AFIDD which should be executable if successful. If there are errors, the file make.log should show what happened.

If one *.fsb file is to be re-compiled and the library updated, use

```
make.unx afile.fsb -C
```

4. The file fiddler/MAKE/fiddler.unx is a script file to run fiddler. It first detects the host type, as does make.unx. A much simpler script file can be constructed if this feature is not needed. You should alter the directory definition to correspond to the proper fiddler directory where AFIDD, AFIDD.a, and AFONT.FON are.

Change the environmental settings for the printers that you use also - they are system commands that you use to print a post-script file or LN03 printer file, etc.

For convenience, you might want to define an alias for fiddler in your .cshrc file. For example,

```
alias fiddler ~/fiddler/MAKE/fiddler.unx
```

5. The file fiddler/MAKE/forge.unx is a script file for compiling applications such as those in the fiddler/EXAM directory. To compile an application,

```
forge.unx <application name[.for]>
```

Make sure that the directory name for the fiddler directory is correctly specified in forge.unx.

VMS

1. Change to the directory [.fiddler.xvax]
2. copy [-.CODE]*.* * (copy all CODE fortran files)
 copy [-.MAKE]make.vms * (copy make script)
 copy [-.FONT]*.* * (copy font file)
 copy [-.HELP]*.* * (copy help files)
 copy [SPECIFIC]*.* * (copy machine specific files)
3. Everything should be ok to compile, but just to be sure:
 - (a) The line defining LOB (logging objects) should be commented out, since these programs do not come with the distribution (they log program use).
 - (b) Check speci.fsb 's routine LOGGER to see that the two calls to START_LOG and STOP_LOG are commented out. Again, these are for logging program use.
4. To compile the entire library:


```
@make.vms *.fsb -C -F
```

This makes the library AFIDD.OLB containing all of the object files, compiles the interactive program AFIDD.EXE, and makes the binary font file AFONT.FOA.
5. If everything is done, you can delete MAKE.VMS, AFONT.FOA, *.FSB, *.INC, *.FOR. You should have these files: AFONT.FON AFIDD AFIDD.a *.HLP.
6. The file [.fiddler.MAKE]fiddler.vms is a script file to run fiddler. You should alter the directory definition to correspond to the proper fiddler directory where AFIDD.EXE AFONT.FON, *.HLP are. Change the environmental settings for the printers that you use also - they are system commands that you use to print a post-script file or LN03 printer file, etc.

 For convenience, you might want to define an alias for fiddler in your login.com file.
7. The file [.fiddler.MAKE]forge.vms is a script file for compiling applications such as those in the [.fiddler.EXAM] directory. To compile an application,


```
@forge.vms <application name[.for]>
```

Make sure that the directory name for the fiddler directory is correctly specified in forge.vms.

Appendix E

FIDDLER : Reference cards

FIDDLER: plotting-package reference Card

subroutine call examples

```
CALL HISTOGM(X,N,XLO,XHI,DX,XTIT,MTIT,STIT)

CALL PROBPLT(X,N,SYMBOL,
+           XTIT,MTIT,STIT,FLHS,FRHS)

CALL XYPLT(X,Y,N,XLO,XHI,YLO,YHI,
+           XTYP,YTYP,XTIT,YTIT,MTIT,STIT,
+           SYMBOL,LFIRST,LLAST,LGRID,
+           IORIENT,IPEN,ILINE)

CALL XYPLTE(X,Y,E,N,XLO,XHI,YLO,YHI,
+           XTYP,YTYP,XTIT,YTIT,MTIT,STIT,
+           SYMBOL,LFIRST,LLAST,LGRID,
+           IORIENT,IPEN,ILINE)

CALL CONTOUR(Z,X,Y,C,NZ,NX,NY,NC,
+           XLO,XHI,YLO,YHI,FGRID,
+           XTIT,YTIT,MTIT,STIT,IORIENT)

CALL CONTOU2(Z,X,Y,C,NZ,NX,NY,NC,
+           XLO,XHI,YLO,YHI,FGRID,
+           XTIT,YTIT,MTIT,STIT,IORIENT)

CALL CONTOU3(Z,X,Y,C,NZ,NX,NY,NC,
+           XLO,XHI,YLO,YHI,FSCALE,
+           XTIT,YTIT,MTIT,STIT,IORIENT)

CALL SURFACE(Z,X,Y,NZ,NX,NY,NG,
+           XLO,XHI,YLO,YHI,
+           XTIT,YTIT,ZTIT,MTIT,STIT,ANGLE,
+           IMODE,LOPAQUE,IORIENT)

CALL SURFAC2(Z,X,Y,NZ,NX,NY,NG,
+           XLO,XHI,YLO,YHI,
+           XTIT,YTIT,ZTIT,MTIT,STIT,
+           PHI,THETA,
+           IMODE,LOPAQUE,IORIENT)

CALL SPACELN(X,Y,Z,N,
+           XLO,XHI,YLO,YHI,ZLO,ZHI,
+           XTIT,YTIT,ZTIT,MTIT,STIT,SYMBOL,
+           ISKIP,ANGLE,IORIENT)

CALL VECTORS(ZX,ZY,X,Y,NZ,NX,NY,NG,
+           XLO,XHI,YLO,YHI,
+           XTIT,YTIT,MTIT,STIT,IORIENT)
```

```
CALL PLTCHR (X,Y,H,ROT,SLT,LORG,STRING,NCHAR)
```

```
CALL PLTVAL (X,Y,H,ROT,SLT,LORG,FORMAT,VALUE)
```

```
CALL PLOTN (X,Y,ICMD,ISUB)
```

Plotting Devices

The device number is used in the PLOTN(0.,0.,15,*) call to define the terminal type.

Index:	Device:	Alpha:	Graphics:
1	4010	VT-100	Tektronix 40xx
2	SEIK	VT-100	Seiko 11xx
3	4205	VT-100	Tektronix 42xx
4	CALC	VT-100	CALC file
5	POST	VT-100	PostScript file
6	HPGL	VT-100	HPGL file
7	V240	VT-240	Tektronix 40xx
8	LN03	VT-100	LN03 file

PLTCHR control characters

Used within STRING parameter of PLTCHR call. NCHR must be negative to enable control characters.

- ^ Superscript.
- _ Subscript.
- & Toggle to/from g typeface (greek or gothic).
- \$ Toggle to/from s typeface (script).
- % Toggle overprint mode.
- # Decrease character size.
- @ Increase character size.
- " Backspace.
- ! Extended control sequence. following character:
 - 0 font-family 0 (SIMPLEX)
 - 1 font-family 1 (COMPLEX)
 - 2 font-family 2 (DUPLEX)
 - 3 font-family 3 (TRIPLEX)
 - ≡ Toggle constant-letter-spacing (typewriter) mode.
 - Toggle connected-letter (no inter-letter spacing) mode.
 - | Stretch vertical scale.
 - ~ Shrink vertical scale.
 - Turn off control character recognition.

PLOTN parameters

Used in PLOTN(X,Y,ICMD,ISUB) calls.

ICMD: ISUB: Operation performed:

```

-----
0      INITIATE      graphics device
1      CLEAR        screen (graphics or alpha)
2      DRAW         to X,Y
3      MOVE         to X,Y
4      FILL         area with color pattern
      1  BEGIN       area boundary
      2  CONTINUE    area boundary
      3  END         area boundary and fill with current are pattern
5      CLIP         window coordinates
      1  LOWER LEFT  clipping window coordinate (plot units) = X,Y
      2  UPPER RIGHT clipping window coordinate (plot units) = X,Y
      3  RESET       to maximum window (CLIP OFF)
6      SCREEN       specification
      1  RATIO=1:1   image is not warped
      2  RATIO=MAX   image fills screen
      3  LOWER LEFT  screen coordinate (inches) = X,Y
      4  UPPER RIGHT screen coordinate (inches) = X,Y
      5  LOWER LEFT  screen coordinate (pixel indices) = X,Y
      6  UPPER RIGHT screen coordinate (pixel indices) = X,Y
7      PEN          color
      TEK4010 SEI1104 TEK4205 CALCOMP PostScript
      1  white white white black black
      2  white red red red gray
      3  white green green green gray
      4  white blue blue blue gray
      5  white cyan cyan gray
      6  white magenta magenta gray
      7  white yellow yellow gray
      8  white orange orange gray
      9  white green green gray
     10  white green green gray
     11  white blue blue gray
     12  white blue blue gray
     13  white red red gray
     14  white gray gray gray
     15  white gray gray gray
8      AREA PATTERN (TEK4205 only)
     -15..0  SOLID
      1..16  TEXTURED
     50..174 DITHERED
9      LINE-TYPE
      1  -----
      2  --- ---
      3  - - - -
      4  - - - -
      5  - - -
      6  - - -
10     ALPHA/GRAPHICS
      1  ALPHA ON (GRAPHICS OFF)
      2  GRAPHICS ON (ALPHA OFF)
11     SPECIFY PLOTTING SIZE IN "PLOTTING UNITS" >0 AND <10
      1  LOWER LEFT graph window coordinate (plot units) = X,Y

```

```

12      2      UPPER RIGHT  graph window coordinate (plot units) = X,Y
      PEN      table index (used by FRAME2D, FRAME3D)
      -1      TEXT PEN     pen color used for text, axes STIT = X
      0      GRID PEN     pen color used for grid and axes    = X
1..6    DATA PEN     pen color used for data pen ISUB    = X
13      LINE-TYPE  table index (used by FRAME2D, FRAME3D)
      -1      TEXT LINE   line-type used for text, axes STIT = X
      0      GRID LINE   line-type used for grid and axes    = X
1..6    DATA LINE   line-type used for data line ISUB    = X
14      <REMOVED FROM SERVICE>
15      TERMINAL  definition (if X=0: load fonts, initialize)
      device name: alpha: graphics:
      1      4010      VT102 Tektronix 4010 (40xx)
      2      SEIK      VT102 Seiko    1104 (11xx)
      3      4205      VT102 Tektronix 4205 (41xx,42xx)
      4      CALC      VT102 CALC    file
      5      POST      VT102 PostScript file
      6      HPGL      VT102 HPGL    file
      7      V240      VT240 Tektronix 4010 (40xx)
      8      LN03      VT102 LN03    file
16      ENTER/EXIT GRAPHICS
      1      ENTER    GRAPHICS
      0      EXIT     GRAPHICS
17      PARAMETERS (default values in curly brackets)
      1      MM =X {3} number of mantissa digits in axes numbers
      2      NE =X {3} number of exponent digits in axes numbers
      3      NP =X {3} maximum power of 10 for axes numbers ('*'format)
      4      NS =X {1} ... NOT USED ...
      5      DT2=X {.03} 2D relative tic          length
      6      DL2=X {.03} 2D relative axes label height
      7      DH2=X {.02} 2D relative axes number height
      8      DM2=X {.20} 2D relative axes/label distance
      9      DC2=X {.05} 2D relative main title height
     10      DS2=X {.04} 2D relative sub title height
     11      DD2=X {.02} 2D relative date          height
     12      DT3=X {.03} 3D relative tic          length
     13      DL3=X {.03} 3D relative axes label height
     14      DH3=X {.02} 3D relative axes number height
     15      DM3=X {.40} 3D relative axes/label distance
     16      DC3=X {.05} 3D relative main title height
     17      DS3=X {.04} 3D relative sub title height
     18      DD3=X {.02} 3D relative date          height
     19      FMT2 {1PG7.0}2D axes numbers: if X=0. '*' else '1PG7.0'
     20      FMT3 {1PG7.0}3D axes numbers: if X=0. '*' else '1PG7.0'
     21      DUX=X {.005} XYPLOT box symbol width
     22      DVX=X {.005} XYPLOT box symbol height
     23      DSX=X {.010} XYPLOT char symbol height
     24      DUS=X {.005} SPACELN box symbol width
     25      DVS=X {.005} SPACELN box symbol height
     26      DSS=X {.010} SPACELN char symbol height
     27      DRV=X {.800} VECTORS relative line-segment length
     28      DAV=X {.150} VECTORS relative arrow          length
     29      90DEG {T}    2D x-axis numbered at 90 degrees
     30      0 DEG {F}    2D x-axis numbered at 0 degrees
     31      BLANK {F}    2D main tics: if X=0. blank else unblank
     32      BLANK {F}    2D axes STIT: if X=0. blank else unblank

```

```

33     BLANK {F}      2D axes numbers: if X=0. blank else unblank
34     BLANK {F}      2D main title:   if X=0. blank else unblank
35     BLANK {F}      2D sub title:    if X=0. blank else unblank
36     BLANK {F}      2D date:         if X=0. blank else unblank
37     BLANK {F}      3D main tics:    if X=0. blank else unblank
38     BLANK {F}      3D axes STIT:    if X=0. blank else unblank
39     BLANK {F}      3D axes numbers: if X=0. blank else unblank
40     BLANK {F}      3D main title:   if X=0. blank else unblank
41     BLANK {F}      3D sub title:    if X=0. blank else unblank
42     BLANK {F}      3D date:         if X=0. blank else unblank
43     BLANK {F}      2D frame:        if X=0. blank else unblank
44     BLANK {F}      3D frame:        if X=0. blank else unblank
45     BLANK {T}      2D minor tics:   if X=0. blank else unblank
46     NQ =X {8}      number of lines drawn in 2D border
47     XAXIS LABELING {0.}             if X=1. xlo is at right
48     YAXIS LABELING {0.}             if X=1. ylo is at top
49     PLOT OPENING CONTROL {0.}       if X=1. disable else enable
50     PLOT CLOSING CONTROL {0.}       if X=1. disable else enable
51     TIC ORIENTATION {0.}            if X=1. outward else inward
52     ZMIN=X,ZMAX=Y Surface clipping limits
18     ABS TEXT MOVE Absolute text move to ( COL=X, ROW=Y)
19     REL TEXT MOVE Relative text move by (#COL=X,#ROW=Y)
20     SAVE/RESTORE text cursor position
      1     SAVE text cursor position
      0     RESTORE text cursor position
21     FLUSH Buffered output (CALL BUFPOUT)

```

FIDDLER interactive program reference card

To run FIDDLERPROMPT> **fiddler** ^C/R**FIDDLER-format data file**

1. Control character in column one:

\$ Main title

% Sub-title

! Channel name

< Lower point index

> Upper point index

SPACE Number

@ Label (next record is label information)

" Comment

2. Rest of line is value:

\$ % ! Titles and channel names are character values

< > Lower and upper point indices are integers, written in I6 format

SPACE Numbers are real, written in E20.10 format **1 .. 9** Relative size of Move/Size/Angle

3. Specifying the channel name with a record beginning with ! also increments the channel index. If endpoints are specified, FIDDLER assumes that the following (upper - lower) records are numbers, and ignores the first column (so # and space work equally well). If the lower index is different from 1, then the point index begins with an offset. This is useful for lining up data vectors to a single column of data values.

IMPORT command

Each 80-or less column line of ASCII data file contains the same number of values, separated by commas or spaces.

Example:

```
1.23  2.35 -2.2 2E-4 4.5555555      3
-1    2      2    4E-7  5          9
 2    1.23  3    3E-9  2.222      10
```

FUNCTION sampling command

1. The independent parameter is the variable T.
2. The dependent parameter is the variable F.
3. The function you will enter is a series of FORTRAN lines. These will be embedded in a loop with the parameter T running between the limits TMIN and TMAX which you also must specify. You also choose lin. or log. sampling. The source file is \$ LOCAL.SAMFUN

4. Example:

```
intent :
  f(t)=4.3*t + 3*t^2 + 2.8*exp(-2*(t-1))
implementation :
  A=4.3*T
  B=3.*T*T
  C=2.8*EXP(-2.*(T-1.))
  F=A+B+C
or :
  F=4.3*T+3*T**2.+2.8*EXP(-2*(T-1))
```

5. There is no error-trapping.

LABEL EDITOR graphics mode commands

> Next Label

< Previous label

L Left

R Right

U Up

D Down

C Clear and repaint

X Set X (0. .. 10.)

Y Set Y (0. .. 10.)

A Set angle (degrees)

S Set slant (degrees)

L Set LORG (1 .. 9)

H Set height

Increment height

**** Increment angle

% Increment slant

Q Quit graphics mode

PEN_CONTROL

The pen-control option allows multiple-sweep plots to be displayed on the same plot, using a continuous line for each sweep, and a discontinuous step between each sweep. The "PEN_CONTROL" channel consists of ones and zeros. Zeros mark the positions of the first points in each sweep.

FIDDLER menu structure

```
=====
MENU LEVEL:
```

```
0      1      2      3      4
=====
```

```
[main]__Help
|__Plot
|__Save
|__Load
|__Data
|__Terminal
|__eXecute
|__Batch
|
|_ [Change]_____View
|   |__Name
|   |__Titles
|   |_ [Channels]____(x,y,z channels)
|   |_ [Window]_____ (plotting limits)
|   |_ [Symbols]_____ (line-types,symbols,pens)
|   |_ [Orientation]_ (axes-orientation)
|   |_ [Pen-control]_ (secondary sweep control)
|   |_ [Labels]_____ (label sizes,format)
|   |_ [Mode]_____ (plotting mode)
|   |_ [Extra]_____ (specific plotting parameters)
|
|_ [Edit]_____ Insert
|   |__Delete
|   |__3-convert
|   |__Sort
|   |__Linear-regression
|   |__Polynomial-regression
|   |_ [View]_____ (spreadsheet commands)
|   |_ [Operate]_____ Endpoints
|   |   |__eXchange
|   |   |__Index
|   |   |__Constant
|   |   |_ [Unary]_____ (unary operations)
|   |   |_ [Binary]_____ (binary operations)
|   |   |_ [Konstant]___ (channel/constant operations)
|
|_ [Import]_____ Ascii
|   |__Suprem2
|   |__supreM3
|   |__Iv
|   |__1-d
|   |__3-d
|   |__simPar
|   |__pRism
|
|_ [Function]___ Help
|   |__View
|   |__T-limits
|   |__Enter
|   |__Insert
|   |__Delete
```

```
|          |__Use
|          |__Reset
|          |__Load
|          |__Save
|          |__Go
|          |__Plot
|
|_ [Make]_-----Hpg1
|          |__Post
|          |__Ln03
|          |__Calc
|          |__Ascii
|
|_ [Add]_-----Help
|          |__View
|          |__Enter
|          |__Insert
|          |__Delete
|          |__Use
|          |__Kill
|          |__Place
|
|_ [User]_-----[A-demo]
|          |__B-calculator
|          |__C-calendar
|          |__ [D-draw]
|          |__ [E-poster]
```