Chapter 1

Introduction

1.1 Overview

The *Quick and Dandy Plotter (QDP)* program reads ASCII files containing various plotting commands and data. *QDP* then calls the *PLT* subroutine which then executes the commands and plots the data. At this point the "`PLT>`" prompt appears and the user can then proceed to enter additional PLT commands which can:

- Display information on about the interactive commands *via* `HElp`,
- Override various PLT defaults,
- Override the PLT commands found in the QDP file,
- Add/remove labels,
- Plot data with various combinations of lines, markers, and error bars,
- Change the appearance/style of the of the plot, for example converting all text into the Roman Font,
- Plot the data as a function of a different x variable,
- Change the number of panels in which the data is plotted,
- Define models and calculate the 'best fit' parameter values,
- Generate a hardcopy.

Thus the interactive PLT commands allow you to both tailor the plot to your needs/taste and to do some simple analyses of the data. PLT commands can be placed in the QDP file, in an indirect command and/or in a command array created by the calling program. For example, if you have a set of commands that you commonly use, you can place those commands in a file, and then have PLT execute the commands that it finds in that file. Since exactly the same command syntax is used, it is not necessary to learn a special programming language to write software that uses PLT. Programmers can try out PLT commands interactively to find a set that works best with the type of data being plotted, and then make these commands the default values.

The PLT software is highly portable. It uses the PGPLOT Graphics Subroutine Library written by T. J. Pearson at the California Institute of Technology. PGPLOT has been ported to many systems ranging from MS-DOS machines to UNICOS Crays. PLT is actively supported on VAX VMS, SUN UNIX, and NeXT systems. The code is in standard Fortran and so can easily be ported to other systems and runs on MS-DOS, PRIME, and IBM RS/6000 systems.

This manual provides an overview of how to use PLT. Not all commands will be described in the overview. However, Appendix B includes the contents of the on-line help which does contain every command.

The rest of this introduction defines a few terms and discusses the syntax conventions used. If you wish to get started quickly, you should skip to the next chapter. Once you have mastered the basics, you should come back and read the following sections.

1.2 Definitions

PLT operates on quantities called *vectors* which can consist of one, two, or three columns in this rectangular array. If the data contains no errors, then each column is a vector. If the data contains symmetric errors, then it takes two columns to denote a single vector. Likewise, if you have two-sided errors (*e.g.*, +5,−2), it will take 3 columns to denote a vector. If one number in a vector has an error, then all numbers in that vector must have the same type of error. The vectors are independent of each other, and so some vectors can have errors and others not.

The PLT default is to make each vector an independent plot group. The PLT `SKip` command can be used if you have just two vectors and you wish to create several plot groups within those vectors.

Viewport denotes the physical area of the plotting surface that you are using. PLT (*via* PGPLOT) uses device-independent coordinates to denote the viewport, with (0.0,0.0) denoting the bottom left corner of the display surface, and (1.0,1.0) the top right corner.

PLT can be used to fit a model to the data. A *model* consists of one or more components which are added together. Each component must have one or more parameters; when fitting the data, the parameters are varied to minimize χ^2. There is no way to multiply the built-in components together.

A *COD file* is an ASCII text file that contains a function written in the COD programming language. PLT allows you to define a model in which one of the components is a function contained a COD file. When FIT evaluates that component, the COD function is called and should return with the function evaluated for the current parameter set.

1.3 Syntax

PLT does not distinguish between upper and lower case. When PLT matches the characters you type with possible commands, it only matches characters in the *shortest unique abbreviation* which, in this documentation, is denoted by upper case. Thus both `color` and `colour` will match the `COlor` command. Of course, some caution is required as `cosmopolitan` will also match `COlor`. As new commands are added, a previously acceptable abbreviation could refer to one of the new commands. To avoid such potential conflicts, you are encouraged to use three letter abbreviations.

In this documentation, a name in all lower case name is a mnemonic and should not be entered. For example, in `Rescale X xmin,xmax` both "`xmin`" and "`xmax`" should be replaced with numbers.

In PLT, arguments can be separated by a comma and/or any number of spaces and tab characters. Thus, the strings `1 2 3` , `1,2,3` , `1, 2, 3` , and `1 , 2 , 3` are all parsed as three arguments. Sometimes it is necessary to leave a place-holder that indicates an argument should be skipped. This is done by entering two adjacent commas. The string `1,,3` is parsed as three arguments with the second argument being null. Null arguments are often used to indicate that the current value should not be changed. If it is necessary to enter any special character as part of an argument, the argument should be enclosed in quotation marks. The string `1,"2,3,4",5` would be parsed as three arguments and the second argument would be the string `2,3,4` .

PLT allows you to embed the simple mathematical operators, `+` , `-` , `*` , and `/` into numbers. Thus, the argument `2*3` would be parsed as 6. The numeric expression is evaluated from left to right; hence, the argument `1+2/4` is parsed 3/4 or 0.75. This syntax can be useful in QDP files. For example, suppose column 1 is the time in seconds, and you wish to plot time in hours. This can be done with a global edit that appends the string `/3600.` to the numbers in column 1.

The character `#` is used to denote a number. When you see this character, you should not type `#` , but rather replace it with a number. Likewise, the character `$` denotes a string. Optional arguments are enclosed in square brackets `[...]` . If an argument must be one of several discrete choices, the choices will be listed separated by vertical lines `|` .

1.4 Questions

Please address suggestions for improvements, or reports of software bugs, to the author:

Allyn Tennant, ES-65
NASA MSFC
Huntsville, AL 35812
USA

Telephone: 205 544-3424
FAX: 205 544-7754

SPAN: SSL::TENNANT or 7207::TENNANT
Internet: tennant%ssl.span@fedex.msfc.nasa.gov

Please address requests for copies of this manual or the software to:

COSMIC
The University of Georgia
382 East Broad Street
Athens, GA 30602
USA

Telephone: 404 542-3265

1.5 Acknowledgements

These days most non-trivial software packages have evolved over a long period of time and PLT is no exception.

The first program to bear the name of QDP was written in the late 1970's by Andy Szymkowiak for use by the X-ray group on a PDP 11/70 at Goddard Space Flight Center. Although I don't think that a single line of code has survived from that original version, I am grateful to Andy for that version, and hence for the basic idea of an interactive graphics program.

The QDP/PLT development flourished during my years at the Institute of Astronomy, Cambridge, U.K. I am grateful to Andy Fabian for being able to fund my stay there, and also for providing the stimulating environment where such working software could be developed. It is important that PLT was developed not as a software project, but rather to meet real needs in the analysis of data.

Now that I am at Marshall Space Flight Center, I would like to thank Martin Weisskopf for his continuing support of these efforts.

I am grateful to Tim Pearson for providing and for continuing to support the PGPLOT graphics package. PGPLOT is flexible, easy to use, portable and device independent.

Numerous other people have made contributions to PLT, ranging from simple comments, such as "it doesn't work when I do this", to actually providing the code for new features. Some of these people are mentioned in the on-line help file under the "history" subtopic. I would like to say thank you to all the people who have offered comments.

Chapter 2

Basics

2.1 QDP files

The quickest and most convenient way to use PLT is with a QDP file. A QDP file is an ASCII text file that contains a rectangular array of data. Since QDP files are ASCII they are easy to create and highly portable to different computer systems. All QDP files must contain a two dimensional array of data. The row-column location of a number in the file determines the row-column index in the data array passed to PLT. It is possible, but not necessary, to include QDP and/or PLT commands at the top of the QDP file. These commands often serve to document the data. All the QDP program does is to read the file, and to pass the information to the PLT subroutine.

In order to try out the examples in this chapter and the next, you should first create a "`DEMO.QDP`" file that contains the following:

```
1   1  16
2   4   9
3   9   4
4  15   1           ! Yes 15 and NOT 16
```

(`XANADU:[PLOT.QDP]DEMO.QDP` contains a pre-typed version of this file.) This example file contains no QDP or PLT commands. The QDP default is to assume that each column of numbers is a separate vector.

This example illustrates that QDP files can contain comments. Comments begin with the comment character `!` and continue to the end of the line. The above example contains the comment "`Yes 15 and NOT 16`". Comments are completely ignored. This documentation will often include a comment with the example commands. When trying out the command, you do not need to type the comment; however, if you do type it, then no harm will be done.

The QDP data lines are free format and the numbers can be separated by spaces, a comma, or tabs. Every row should contain the same number of columns; however, if some data are missing, you can enter the word `NO` instead of an actual number. QDP translates the `NO` into the PLT no-data flag; which will be ignored by PLT.

2.2 Plot the file

Once you have created a version of `DEMO.QDP`, you can run QDP by typing:

```
$ QDP DEMO
```

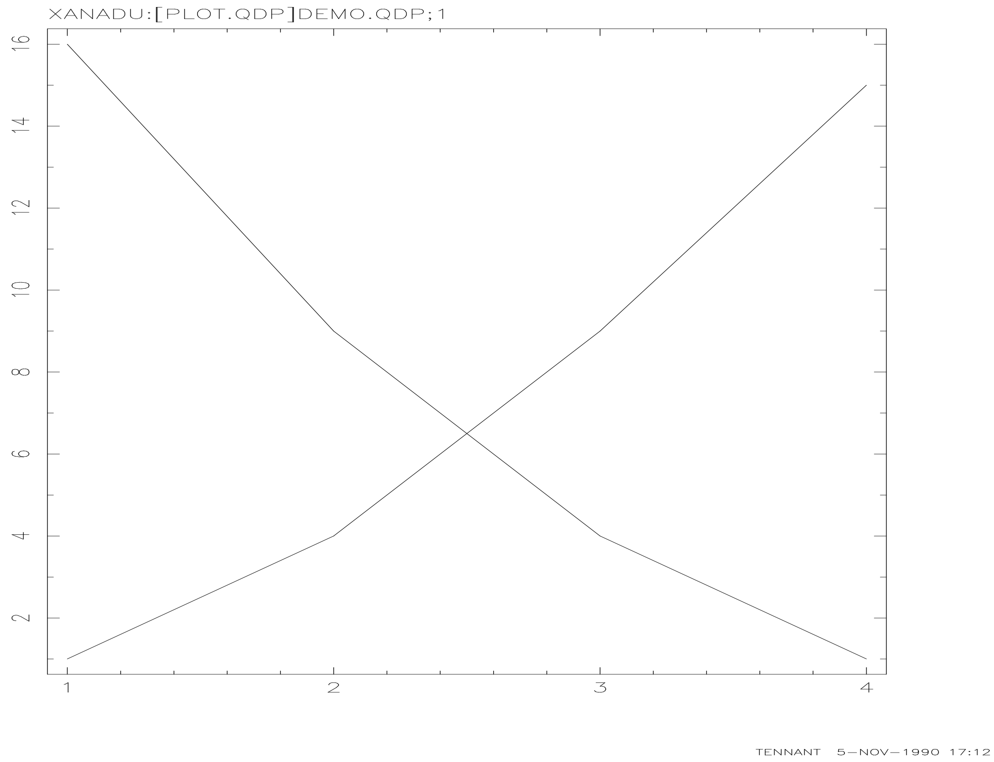

Figure 2.1: The default appearance of the `DEMO.QDP` file.

(If QDP fails to run, then you might need to define the QDP symbol as described in Appendix D, **Installation Guide**.) It is not necessary to enter the `.QDP` extension as the QDP program assumes that as the default. When the program starts, you will receive the following message:

```
To produce plot, please enter
PGPLOT file/type:
```

You should enter the PGPLOT specification for the device on which you wish to plot. If you do not know the device name, then enter `?` and all the device types supported by your local version of PGPLOT will be listed. If your terminal supports Tektronix graphics, then enter `/TE` to make the plot appear on your terminal. You might also try `/RE` for Retrographics. Most Tektronix emulators support the Retrographics extensions that allow the software to toggle between text and graphics modes.

A graph containing two lines will now be drawn as illustrated in Figure 2.1. Since the file contained three columns of numbers, the default mode assumes there are three plot groups. The first plot group determines the x coordinate. The next two columns are plotted as two lines. On a color display the first line will be red and the second green, which are the default colors for plot groups 2 and 3. The name of the QDP file appears in the top left of the plot and your userid, current date, and time appear in the bottom right of the plot. The `PLT>` prompt will now appear. In the following sections you will see how to enter various commands to change the default plot. The most useful command for beginners is the `HElp` command which can be used to get instructions on how to use any command.

Another useful command is `EXit` that will get you out of the PLT subroutine. If you are in QDP, control will be returned to the operating system; in other programs, control will be returned to the calling program.

2.3 Rescaling

PLT chooses a default scale that makes all the data visible. The examples in this section will illustrate the `Rescale` command that can be used to change this scale. Using the plot created in the previous section, enter

```
PLT> R X 0 5
```

When you enter this command, the graph is redrawn with 0.0 on the left side and 5.0 on the right side of the viewport. Likewise `R Y 0 20` will put 0.0 on the bottom and 20.0 on the top. If you wish to change one number without the other, you can skip the field with commas or terminate the line before changing the default. For example, both

```
PLT> R Y 0 20    ! Set both lower and upper limit
```

and

```
PLT> R Y,,20     ! Set upper limit to 20, leaving lower unaffected
PLT> R Y 0       ! Set lower limit to 0, leaving upper unaffected
```

will produce the same effect. If you wish to change both the `X` and `Y` limits, then use

```
PLT> R 1 5 1 16  ! Set X-range to 1 to 5, and Y-range 1 to 16
```

If you wish to go back to the default scale, then use `R` with no arguments:

```
PLT> R Y         ! Will reset Y limits to default
PLT> R           ! Will reset both X and Y limits to default
```

At any time you can find out what the current scale limits are with

```
PLT> R ?
 Current Gap=   .025
Window   XLAB      XMIN        XMAX         YLAB      YMIN        YMAX
   1 :            .9250    ,  4.075     :              .6250    ,  16.38
PLT>
```

This produces a table of the current scaling parameters. The current gap is the default size of the gap between the edge of the data and the edge of the plot. For the default scale, the difference between the minimum/maximum and the data minimum/data maximum is due to the gap. With a gap of zero, the default minimum/maximum value will exactly match the data minimum/data maximum. The default is to plot all plot groups into just one window hence only one row appears in the table corresponding to that window. The columns labeled `XLAB` and `YLAB` contain the current x and y labels, which are currently blank.

If you want to see what the data minimum and maximum values are you should use

```
PLT> SHow Group

Grp  Wind    Label      XData Min     XData Max     YData Min     YData Max
  1    -1              :  1.000     ,   4.000     :  1.000     ,   4.000
  2     1              :  1.000     ,   4.000     :  1.000     ,   15.00
  3     1              :  1.000     ,   4.000     :  1.000     ,   16.00
```

The three rows correspond to the three plot groups. The column labeled `Wind` contains the window in which the group is currently being plotted, and a negative number indicates that the group is not actually plotted. In this example, group 1 is used to determine the x coordinate and so is not actually plotted. The columns labeled `YData Min` and `YData Max` contain the actual data minimum and maximum of that plot group.

2.4 Making a hardcopy

PLT makes a hardcopy by using the same PGPLOT routines but routed to a different graphics device. Thus the command does not make a hardcopy of what is currently on your screen, but rather, what would be plotted if you reissued the `Plot` command. The `HArd ?` command will display the name of your current default hardcopy device. It is possible to override this default when you enter the `HArd` command, thus `HArd /VPS` would make a vertical (portrait) mode Postscript file no matter what the default is. If you would like a default different from what is set up on your system, then you should define the logical name, or on UNIX, the environment variable, `PLT_HARDCOPY` to contain the default you want. Let's assume the default is OK. So, merely enter

```
PLT> CSize 1.3    ! To increase the character size a bit
PLT> FOnt Roman   ! To use the nice looking Roman font
PLT> HArdcopy     ! To make a hardcopy file
```

PGPLOT would have now made a file in your current directory. You should consult your PGPLOT manual for the rules on how to print this file. On many systems it is possible to use the `@HARD` command that will both create a file and then spool the file to the printer.

The default PLT font is the Simple font because it plots the fastest. When you are making a hardcopy, speed is less important than quality. Therefore, you are encouraged to use the Roman font, which will give a more professional look to your hardcopy. As most journals greatly reduce the size of figures before printing, you should increase the character size. In the above example, `CSize 1.3` makes the character size a factor of 1.3 times larger than the default. The default line width is one, which is the thinest possible line. On some laser printers, this is too thin, and therefore, you should increase the line width, using the `LWidth` command. Using `LWidth 7` is not unreasonable for publication quality on some printers. In general the default hardcopy plot will fill the page on which it is being plotted. If a viewgraph was made of a full page plot, the projected size would overfill most screens. Therefore, it is useful to decrease the default size of the plot a bit. This can be done with the `Viewport` command. The default viewport is `.1 .1` which means the box containing the graph extends from 0.10 to 0.90 of the total physical plotting area. To make a plot half the size, use `View .3 .3`. `View .2 .2` will result in a good size for most viewgraphs.

Chapter 3

Aesthetics

3.1 Labels

This chapter describes the various options available to change the appearance of the plot. One of the most common things to do is to add labels, using the `LAbel` command. To put the label "Time (sec)" on the x-axis, "Distance" on the y-axis, and "My data" at the top of the plot,

```
PLT> LA X Time (sec)
PLT> LA Y Distance
PLT> LA T My data
PLT> P
```

You will notice that only certain PLT commands cause the graphics display to be updated. This allows you to enter several commands quickly without having to wait for the screen to be redrawn after each command. Whenever you want to see what the current graph looks like, you should enter the `Plot` command or just `P`.

There are also *Outer* labels (called `OX`, `OY`, `OT`) that can be used. These outer labels provide a simple way to create labels that need to lie on two lines. For example, the commands

```
PLT> LA X Universal
PLT> LA OX Time (sec)
PLT> P
```

would label the x-axis with two lines of text with the word "Universal" being written above the words "Time (sec)". Now what do you think the following command will do?

```
PLT> LA OT Fun! Fun! Fun!
PLT> P
```

If you try this you will find that only the word "Fun" appears. This is because `!` is the PLT comment character. If you wish to enter a PLT command that contains the comment character, then you must enclose the entire argument in quotation marks:

```
PLT> LA OT "Fun! Fun! Fun!"
PLT> P
```

To remove any label, enter the command with no text; thus,

```
PLT> LA OT
PLT> P
```

will remove the text "Fun! Fun! Fun!" from the graph. The name of the QDP file appears in the `File` position; thus the command `LA F` will remove this name. The time-stamp that appears at the bottom of the plot can be removed with the `Time OFf` command and, of course, `Time ON` will turn it back on. In general, you should leave the file name and time-stamp in place, as this information is very useful on a hardcopy. Sometimes, when working with a slow plotting device, you will want to speed things up by not plotting any labels. This can be done the `LAbel OFf` command. Of course, you should issue the `LAbel ON` command before making a hardcopy.

Text is drawn with PGPLOT; so the standard PGPLOT escape sequences are used. Hence, the commands

```
PLT> LA T \gx\u2
PLT> P
```

will label the top of the graph with χ^2. The default font is the PGPLOT *Normal font*, which draws rather quickly. For journal quality text, you should override the default font with the `FOnt Roman` command. This will cause all text, including the numeric labels on the axes, to be written in the nicer looking, but slower plotting, Roman font. Use `FOnt ?` to get a list of possible fonts.

It is also possible to place a *numbered label* anywhere in the plot. To see this, try

```
PLT> LA 1 Pos 2 4 LIne -45 "Point at (2,4)"
PLT> P
```

The above command plots LAbel 1 at Position (2,4) with a LIne extending at an angle of -45° to the x-axis and with the text message "Point at (2,4)". Each attribute can be set individually. Hence, if you decide you don't like the line extending downwards, you could change the angle with

```
PLT> LA 1 LIne 135
PLT> P
```

This leaves the pointing position and text unaffected, but resets the angle of the line (and also the justification of the string).

3.2 Vertical plots

When you first plotted the `DEMO.QDP` file, two plot groups were plotted on the same panel. It is possible to plot each plot group on a separate panel in a vertical stack. To see this, try

```
$ QDP DEMO
(enter device type)
PLT> Plot Vertical
PLT> Plot
```

The `Plot Vertical` command resets the internal parameters so that each visible plot group will be plotted in a separate panel in a vertical stack. Nothing is replotted until the `Plot` command alone is issued. This allows you to reset other parameters, without having to wait for a new graph to be drawn.

The y-scale can be adjusted in each panel separately. Hence,

```
PLT> R Y2 0 10
PLT> R Y3 0 50
PLT> P
```

will set the y-range of the top panel to be 0 to 10 and of the bottom panel to be 0 to 50. The `R ?` command can be used at any time to display the current ranges. At this point it would be wise to label each plot group. So enter

```
PLT> LA G1 x-axis
PLT> LA G2 group 2
PLT> LA G3 group 3
PLT> P
PLT> R ?
 Current Gap=   .025
Window   XLAB      XMIN        XMAX         YLAB      YMIN        YMAX
   2 : x-axis      .9250    ,  4.075     : group 2     .0000    ,  10.00
   3 : x-axis      .9250    ,  4.075     : group 3     .0000    ,  50.00
PLT>
```

The `R ?` prints out the beginning of each label and therefore, with a good set of labels, it is easy to keep track of what is plotted where.

At this point it is worth pointing out the difference between plot groups, and the rescale parameters. A plot group is a group of associated data points that cannot be displayed in different panels. The `Rescale` command affects the scale of the designated panel. Thus, `R Y2 0 10` will set the y-scale in the second panel to range from 0 to 10. For maximum compatibility with previous versions of PLT, the `Plot Vert` command plots group 2 on panel 2. The command `LAbel G1` will associate a label with a plot group. Thus if you enter the commands `Xaxis 2`, `Plot Vert`, and then `Plot`, you will find that plot group 2 now determines the x-axis and hence the label "group 2" is now used as the x-label. Plot group 1 is now plotted in the top panel, with the same label "x-axis" which, of course, this is no longer correct.

To undo the effects of the `Plot Vertical` command, you should enter

```
PLT> Plot Overlay
PLT> P
```

The y-axis label is the label of the first plot group to be plotted in that panel. Since now more than one group appears in the panel, this is now longer most appropriate. To override the y-axis label in a given panel, use the `LAbel Y` command. In other words the `LA Y` command can be used to denote all the y plot groups in a given panel, whereas `G1`, `G2`, *etc.* will associate a label with the specified plot group.

3.3 Colors, lines, and markers

The default mode of PLT is to plot group 1 with color index 1, group 2 with color index 2, *etc.* The `COlor ?` command can be used to generate a list of the default colors used to plot each color index. The command `COlor 3 ON 2` will cause color index 3 to be used with group 2 is plotted. With the PGPLOT default colors, this means that group 2 will now be plotted in green. It is important to realize that the `COlor` command changes the color index and only indirectly, the color.

Due to historical accident the `COlor` command can be used to prevent plot groups from being plotted. This is because color 0 corresponds to no-color or invisible. Thus `COlor 0 ON 2` will suppress the plotting of group 2. A cleaner way to do this is with the command `COlor OFf 2`. A `COlor OFf` command followed by a `COlor ON` command will restore the original color index. The `R Y` (with no arguments) command only uses plot groups that are visible to determine the default scale. For example, assume you are working with 6 plot groups and the values of groups 1 to 5 all lie in the range of 0.0 to 1.0, whereas the values in group 6 all lie near 100,000. For this example, the commands

```
PLT> COlor OFF 6
PLT> R Y
```

would redraw the graph, and the default y-range will lie between 0.0 and 1.0.

In the above examples, PLT drew a line between the points being plotted. If you wish to display the plot with markers, then you should turn on the plotting of markers with

```
PLT> MArk ON
PLT> P
```

For this example, the line connecting the various points disappears and only markers will be drawn. PLT draws a line when (a) all attributes (`LIne`, `MArker`, and `Errors`) are `OFF` or (b) the line attribute is `ON`. Thus if you want both the connecting line and markers to appear, then you need to turn on the `LIne` attribute with

```
PLT> LIne ON
PLT> P
```

The command `MArker Size 2` can be used to make the markers twice as big. The default marker style for all plot groups is type 2 as this marker plots very quickly. To change the style of the marker, try `MArker 9 ON 2` to use marker style 9 when plotting group 2. The command `MArker ?` will display a table of marker styles.

3.4 Log scale

It is also possible to plot the data on a log scale. To do this, type

```
PLT> LOg Y
PLT> P
```

Use `LOg X` to use a log scale on the x-axis. The `LOg OFf` command will turn off the log scale on both the x- and y- axes. Note: Using `LOg` does not cause the data to be altered, only the appearance of the plot changes. If the lower limit of the scale being logged is negative or zero, then PLT will rescan the data searching for the smallest positive value, and make that the lower limit.

You should also be aware of the fact that the size of the gap, created by the `GAp` command is affected by log scale. Thus for a non-zero gap, the sequence `R X` followed by `LOg X` produces a different range than `LOg X` followed by `R X`. In the first case, the data minimum and maximum values are found, and then gap added in linear space. Applying the `LOg` command does not change this scale. In the second case, the scale is first logged, then the data minimum and maximum values are found. At this point the correct gap for a log plot is added.

Chapter 4

Fitting

4.1 Errors

This section describes a QDP file that contains errors and how to control the plotting of those errors. The next two sections describe how to define a model, find the best fitting parameter values, and then estimate the uncertainties on the parameter values. Although the examples will be based on the QDP file containing errors, it is possible (and sometimes better) to fit data without errors.

You should now create a `DEMO1.QDP` file that contains the following:

```
READ Serr 1 2
LAbel X Time
LAbel Y Distance
 1.0  .25   1.24  .3
 1.5  .25   1.86  .3
 2.0  .25   3.76  .3
 4.0 1.75  16.43  .3
 7.0 1.25  49.06  .3
```

The first line in this file is not a PLT command but rather a QDP command. The QDP `READ` command is used by QDP to tell PLT which vectors contain errors. In this case the READ command tells QDP that vectors 1 and 2 will have symmetric errors; hence, columns 1 and 2 contain data and errors for vector 1, and columns 3 and 4 contain data and errors for vector 2. Following the QDP command are two PLT commands that will be passed to the PLT program and executed before the graph is drawn. Including PLT commands in the QDP file provides a way to override built-in defaults and/or to add labels to the graph. Data lines occur after all the command lines. To read and plot this file, use

```
$ QDP DEMO1
(enter device type)
PLT>
```

The graph which looks like Figure 4.1 should now appear. When you plot data containing errors, the error attribute is `ON`, and the errors plotted. As described in Section 3.3, the line will no longer appear connecting the data. If you want to see that line, then should use `LIne ON` to explicitly switch on the line. To suppress plotting of the errors use,

```
PLT> Error OFf
```

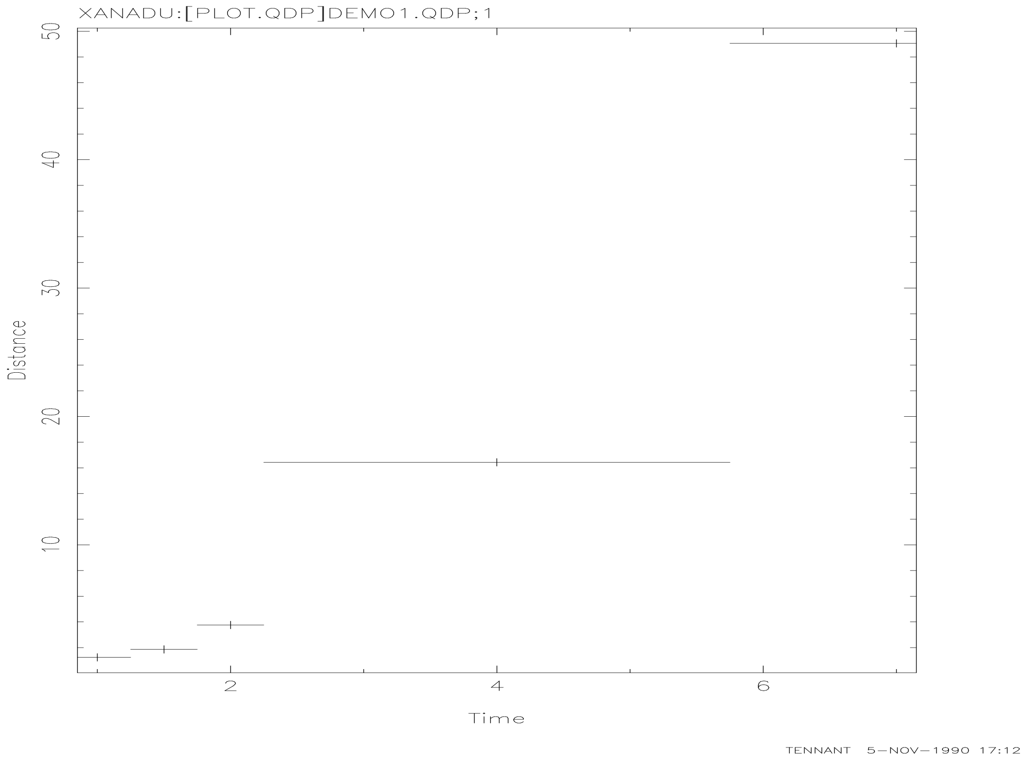

Figure 4.1: The default appearance of the DEMO1.QDP file.

```
PLT> P
```

This disables plotting of the error bars. Once again the line connecting the points appears. You should now enter

```
PLT> LIne Step
PLT> P
```

to produce a stepped-line plot. Using `Error ON` at this time will cause both the stepped line and the errors to be plotted. This is because the `LIne Step` command sets an internal flag that a line should be plotted and the `Error ON` command sets another flag to plot errors. The command `LIne OFf` will turn off the plotting of the (stepped) line. The plotting of a line, errors and markers can all be turned on or off for each vector independently. See the `LIne`, `Error` and `MArker` commands, in Appendix B, **PLT Command Summary**, for more information.

4.2 Fitting

This section requires the DEMO1.QDP file described in the previous section. Before you can fit data, you must first define a model. First, read in the data and define a constant model with

```
$ QDP DEMO1
(enter device type)
PLT> MOdel CONS
```

At this point, you will be prompted for the default initial value for the constant. Enter $\boxed{\text{Return}}$ to use the default, and at the `PLT>` prompt, type `Fit`.[1] When `Fit` runs, it first tells you which plot group is being fitted and the range over which data are being fitted. It is important to realize that if you have used `R X` to rescale the x-axis so that some points are outside the range plotted, then these points would not be included in the fit. You cannot exclude points using the `R Y` command. (This is intended to prevent cheating.) You will next see the message "Fitting 5 points in a band of 5". This informs you that there are 5 points in the current x-range. In order to execute faster, the FIT routine resets the minimum and maximum of the array, to achieve the smallest range possible that includes all points in the x-range, and so the "band of 5" output informs you how big this minimum range is. Next the FIT routine prints the current parameter values (1.00000 in this case). The program then prints the current value of the weighted variance `W-VAR`. If you have errors on your data, the weighted variance is χ^2; for no errors, `W-VAR` is just the variance. The number in `()` is $\log\lambda$ and is for the expert's use.

The *CURFIT* routine will terminate when the change in χ^2 or, for an unweighted fit the relative change in the variance, is less than 0.05. If this condition has not been met after 10 iterations, you will be prompted "`Continue fitting? (n)`". Answer `Y` to continue or `N` to terminate. If you are are fitting in background or batch mode, then you should always leave a blank line after the `Fit` command. Thus if `Fit` does not terminate, the "`Continue fitting? (n)`" question will read the blank line with the default answer of "no" and terminate. If the fit does terminate, PLT will see the blank line, and ignore it. When *CURFIT* terminates, the current parameter values are again printed and this model is drawn on the current plot.

For the above, the total variance is 18323 and hence a CONS does not look like a very good model. Let's try a more complicated model with

```
PLT> MOdel CO LI QU
```

to include constant, linear and quadratic components. Again you can default on all the initial values. When you type `Fit`, you should find that `W-VAR` has decreased to 4.23. This is clearly a better fit.

To generate a list of all possible built-in components, use the `MOdel ?` command. To obtain a description of what a component does, use the `HElp MOdel` command followed by the component name. If you can not construct your model from the built-in components, then you can create additional components. A COD file can be used to define a sophisticated component. COD files are ASCII text files that contain functions written in a Forth-like computer language. Chapter 7 and Appendix A describe COD in some detail. If your component is too complicated for COD, or you don't like using COD, then you can create a Fortran function that can be used as a new component. The next chapter will describe how to create this function `UFNY`, and the supporting routines to replace the built-in `DEMO` component.

It is possible to save the current model to a disk file using the `WModel` command. For example,

```
PLT> WModel DEMO1
```

[1]The *FIT* subroutine minimizes χ^2 using a modified version of Bevington's *CURFIT* subroutine. Bevington's book *Data Reduction and Error Analysis for the Physical Sciences*, published by McGraw-Hill in 1969, is an excellent introduction to statistics. Anyone interested in a detailed understanding of how CURFIT works should consult this book.

will create a `DEMO1.MOD` file. To read this model back into PLT use the command `MOdel @DEMO1`. Model files can be printed out to make a hardcopy of the current parameter values. If you do not enter a file name with the `WModel` command, then the model is written to your current terminal screen.

4.3 Parameter uncertainties

The `Uncertainty` command can be used to estimate the uncertainties in the parameter values.[2] To try this, you should first fit the data in the `DEMO1.QDP` file to a `CO LI QU` model, as described in the previous section. Now enter the command

```
PLT> Uncertain 1
```

The program will now change the value of parameter 1 by a small amount and recompute χ^2. At each step, the delta parameter value and the $\Delta\chi^2$ are printed out. For complicated models it may take many steps before the desired value of $\Delta\chi^2$ is found. The program considers both positive and negative delta parameter values. The default value of $\Delta\chi^2$ is 2.7 which corresponds to the 90% confidence range for a single parameter.

With the `DEMO1.QDP` file you will find that both parameter 1 and 2 are consistent with zero. To see whether they can be eliminated, try the following:

```
PLT> Newpar 1,0,-1
PLT> Newpar 2,0,-1
PLT> Fit
```

The first command resets both the `VAL` and `SIG` terms of parameter 1 to be 0.0 and -1.0, respectively. A `SIG` of -1.0 means that the parameter is frozen and hence not allowed to change. Note: The command `Newpar 1,,-1` would have frozen parameter 1 at its current value. The second line freezes the value of parameter 2 to be zero. The results of the `Fit` reveal that χ^2 has increased by 0.55 and the F-statistic or a likelihood ratio tells us that these two components were not required by the model.

The `Uncertainty` command is fairly robust but on occasion can have difficulties. Sometimes, `Uncertainty` will find a new minimum value of χ^2. This causes the search to be stopped and the parameter values to be reset. At this point, you should re-issue the `Fit` command to locate precisely the new minimum. Sometimes, `Uncertainty` will be unable to locate the requested value of $\Delta\chi^2$ after 10 tries. At this point the message `UNCERT--Give up.` is printed. It will be up to you to decide whether the error has been correctly calculated. Finally, the `Uncertainty` command uses the `SIG` value to estimate the location of the error. If this number is greatly in error, then `Uncertainty` will be starting its search in the wrong place. If this occurs, then it is sometimes possible to adjust `SIG` to be a more accurate estimate, before issuing the `Uncertainty` command. It is also possible that the `SIG` is inaccurate because the true minimum has not been found and further fitting is needed.

[2] Uncertainties in parameter values are estimated using the method described in "Parameter Estimation in X-ray Astronomy", by M. Lampton, B. Margon, and S. Bowyer, in *The Astrophysical Journal* (1976) Vol. 208, p. 177.

Chapter 5

Miscellaneous

5.1 PLT command files

PLT commands can also be entered *via* a command file. For example, if you often enter the sequence of commands `FOnt Roman` followed by `CSize 1.3`, then you could create a file called `NICE.PCO` that contains the lines

```
FOnt Roman
CSize 1.3
```

To execute these commands inside PLT, all you need to type is

```
PLT> @NICE
```

A default file extension `.PCO` is assumed. Thus command files provide a way to enter several and/or complicated commands easily.

Note that the reference to a command file is a legal PLT command that can appear in a QDP file. Since this is a PLT command, QDP itself will not open and read the command file. Hence, QDP commands and data lines cannot be entered *via* a command file. Command files serve two important uses in QDP files. First, they provide a way to enter the same set of commands to several files. Second, for long data files, editing the QDP file can be tedious. Hence, you can edit it once to enter a reference to the command file. Thereafter, whenever you want to change the PLT command list, you need only to edit the command file.

PLT searches up to three different directories for the specified indirect command. The current directory is always searched first. If the file is not found in the current directory then PLT tries to translate the logical name (under VMS) or environment variable (under UNIX or DOS) called MY_XCOMS. If MY_XCOMS has been defined, then PLT searches the specified directory. If the file still has not been found then PLT searches the `XANADU:[LIB.XCOMS]` directory. This three level search allows you to create system-wide files, user-wide files, and directory specific files. For example, many locations create a file `HARD.PCO` in the `XANADU:[LIB.XCOMS]` directory that (1) creates a hardcopy file and (2) spools the file to the printer (done using the `$` command to spawn a job to spool the plot to the printer). This then allows all users on that system to use

```
PLT> @HARD
```

to immediately print a hardcopy. If you do not like something about the existing `@HARD` command then you can easily create a new private version of this command. First copy the file to one of your own directories, modify the file, and define MY_XCOMS to contain the name of the directory containing the new version. Once this has been done, PLT will find and run your version of the command instead of the system installed version.

It is possible to use parameters with indirect command files. The parameter values are entered on the same line that opened the indirect command file. Thus,

```
PLT> @test one two three
```

would cause PLT to open and read the `TEST.PCO` file with three parameters "`one`", "`two`", and "`three`". If `n` is a number then the sequence `%n%` will be replaced with the nth parameter. For the above example, `%1%` will be replaced with '`one`', `%2%` with '`two`', *etc.* The following illustrates a possible indirect file that could use up to three parameters:

```
LABel X %1%
LABel Y %2%
LABel T %3%
```

If you fail to enter all three parameters, then `%n%` will be replaced with a null string for the unentered parameters.

It is possible for one indirect file to call another indirect file and pass in parameters. Thus,

```
@deeper first %2% %3%
```

is a valid line in an indirect command file. In this example, the first parameter is "`first`", whereas the next two parameters will be set equal to parameters 2 and 3 of the current script. Also quotes can be used to denotes a single parameter with embedded spaces, or other 'magic' characters. Thus the line,

```
@file "This is all one" two three
```

contains three parameters, and the first parameter is the string '`This is all one`'.

5.2 Version control

New features are constantly being added to PLT, and so it is important to keep track of these changes. There are three places where changes are noted. First, the PLT `VErsion` command can be used to identify the date of the most recent change to the version of PLT linked into the program you are using. If certain commands do not appear to work then you should check this number. Often you will find that the program has not been linked for a while and as a result the command that you are trying to use was added after the last link.

The second place in which version numbers are recorded is the on-line help file. The `HElp VErsion` command will list all recently-added new features and when they were made. A serious attempt is made to ensure that on-line help is updated as the software is modified. For best results on your system, you should also update the on-line help every time you update PLT itself. However, there is no requirement for these two version numbers to match. Thus, when you install a new version of PLT it is not necessary to immediately relink all software that uses it.

Finally the printed manual is updated about once a year. Therefore, it can be slightly out of date.

Chapter 6

Fortran interface

6.1 Programming PLT

After using the QDP/PLT software for a while, some people would like to see more sophisticated features such as the ability to read binary files, or to add different vectors together. Although the author is always willing to take suggestions (and even to implement some of these suggestions), the PLT design goal is to implement new features in as general manner as possible. Thus if you need to read a particular file format, or to manipulate data in a particular manner, you should implement your own front end to the PLT subroutine. This is simple to do since the QDP program cleanly separates reading of the file from actually calling the PLT routine.

This chapter describes how to call the RDQDP and PLT subroutines. Also listed is the complete source code for the QDP program. Although QDP can be used as an example of how to call PLT, it is perhaps too simple. Therefore the DEMO Fortran program more clearly shows how to do this. Finally there are instructions on how to create your own user-defined function that, when linked with the QDP/PLT software, can be used with the `Model` and `Fit` commands.

Although PLT uses several other internal routines, you are discouraged from directly using these routines in your code. This is because PLT continues to evolve, and there is no way that the author can add the functionality required without the ability to modify the internal interfaces. There are no plans to modify the calling sequence for all the routines described in this chapter.

6.2 Subroutine RDQDP

The calling sequence for the RDQDP subroutine is:

```
      SUBROUTINE RDQDP(ICHAT, LUNIN, CNAM, Y, MXPTS, IERY, MXVEC,
     :   NROW, NPTS, NVEC, CMD, MXCMD, NCMD, IER)
      INTEGER   MXPTS, MXVEC, MXCMD
      CHARACTER CNAM*(*), CMD(MXCMD)*(*)
      REAL      Y(MXPTS)
      INTEGER   IERY(MXVEC)
      INTEGER   ICHAT, LUNIN, NROW, NPTS, NVEC, NCMD, IER
C---
C Opens and reads a QDP file.
C---
C ICHAT     I    >10 means print comment lines, >0 print row/col info.
C LUNIN     I    <>0 means file already open on LUN.
C CNAM      I/O  File name.
C Y           O  The data array
C MXPTS     I    The actual size of the Y array.
C IERY        O  The PLT error flag array
C MXVEC     I    The actual size of the IERY array
C NROW        O  Maximum number of rows that the file could contain.
C NPTS,NVEC   O  Needed by PLT
C CMD         O  Command array (MXCMD input dimension).
C NCMD        O  Number of commands read
C IER         O  =-1 if user entered EOF, =0 file read, =1 no file read.
```

There are several ways to specify a file to be read by RDQDP. RDQDP will go through the following steps to determine what file to read. Once a file has been determined the remaining steps will be skipped. Specifically RDQDP will do the following:

- If the variable `LUNIN` is non-zero then RDQDP will assume that the input file has already been opened and is attached to the specified unit number.
- If the variable `CNAM` is non-blank, then RDQDP opens a file with the specified name.
- At this point it is necessary to obtain a file name from an external source and the parser is called to handle this. If this is the first time the parser has been called in the current program, an attempt will be made to read the command line. If a QDP file name is found on the command line, then that file is opened.
- If no file name could be found, or the file could not be opened, then RDQDP will prompt the user for an input file name. If the user enters an end-of-file (`^Z` under VMS, or `/*` under all systems), then RDQDP will exit with `IER`=-1. If the user enters a blank line for the file name, RDQDP will exit `IER`>0. Of course, if the user enters a valid file name, then that file is opened, and `IER` will return a value of 0.

Once a file has been opened, and if `MXCMD`>1, RDQDP will add a `LAbel F` command to the `CMD` array that contains the name of the file actually opened. Of course, if the file contains a `LAbel F` command, then that command will overwrite the label that RDQDP creates.

`ICHAT` is the 'chatter' flag. If `ICHAT`>10 then RDQDP will display lines that have `!` in the first column, on your terminal screen. Displaying these lines, provides a useful way to confirm that RDQDP has opened the correct file. Such comment lines are completely ignored, and the comments will be removed from any other line containing a comment. RDQDP examines the beginning of each line and if the line contains a QDP command, RDQDP proceeds to interpret the command and set the appropriate variables to be passed to PLT. If the line starts with a PLT command and `NCMD`<`MXCMD`, then RDQDP will increment `NCMD` and add the line to the `CMD` array. For lines containing data, RDQDP interprets the line into real numbers and stores these numbers in the `Y` array.

The RDQDP routine does not open and read any indirect command files, but just stores the command in the `CMD` array. Therefore you cannot use an indirect command file to contain the data array. Since the calling program determines the size of the `CMD` array it is sometimes useful to store all PLT commands in an indirect command file and to add one line to the QDP file to read the indirect file. When PLT reads an indirect file it will accept command lines up to 250 characters long, and there is no limit to the number of lines that can be read.

When RDQDP reads the first data line, it determines the number of columns in that line. Based on the number of columns, and the size of `MXPTS` passed in, RDQDP calculates the maximum number of rows that would fit into the `Y` array. If the `ICHAT`> 0, RDQDP will then display on the terminal the number of columns, the numbers of vectors (calculated from the data from in any `READ` lines), and the maximum number of rows.

6.3 Subroutine PLT

The calling sequence for the PLT subroutine is:

```
      SUBROUTINE PLT(Y, IERY, MXROW, NPTS, NVEC, CMD, NCMD, IER)
      REAL      Y(*)
      INTEGER   IERY(*), MXROW, NPTS, NVEC, NCMD, IER
      CHARACTER CMD(*)*(*)
C---
C General plot subroutine.
C---
C Y(*)      I    The data array.  The array should be dimensioned
C                Y(MXROW,MXCOL) where MXROW and MXCOL are the actual
C                sizes of the arrays in the calling program.
C                MXCOL=NVEC+NSERR+2*NTERR  where NSERR is the number
C                of vectors that have symmetric errors and NTERR
C                is the number of vectors that have two-sided errors.
C IERY(*)   I    =-1 plot errors as SQRT(Y)
C                = 0 no errors.
C                =+1 explicit symmetric errors.
C                =+2 for two-sided errors
C MXROW     I    The actual first dimension of the Y array.
C NPTS      I    The number of points to plot (NPTS<=MXROW).
C NVEC      I    The number of vectors to be plotted.
C CMD(*)    I    Array of commands.
C NCMD      I    Number of commands.
C IER         O  Error flag, =-1 if user entered EOF, =0 otherwise.
```

It is important to remember that the variable NVEC does not refer to the number of columns of data but rather the number of vectors. Each vector must have one entry in the IERY array that describes the type of error on that vector. Depending on the type of error, each vector can be composed of one, two, or three columns of data. To calculate the number of columns needed by the vectors, let NSERR be the number of vectors with symmetric errors (IERY(I)=1) and NTERR the number with two sided errors (IERY(I)=2). The total number of columns MXCOL will be given by MXCOL=NVEC+NSERR+2*NTERR.

The variable MXROW contains the physical first dimension of the Y array. Thus the calling program should dimension Y to be (MXROW,MXCOL) or the Fortran equivalent (MXROW*MXCOL). The variable NPTS contains the number of rows that contain valid data. All rows from NPTS+1 to MXROW will be ignored. When PLT starts it will execute NCMD lines from the CMD array. Any valid PLT command can be entered into this array. For example, one line could contain a reference to an indirect command file, and this would cause PLT to execute all commands found in this file. If the command list contains the EXit command, then PLT will exit when this command executes and any commands following the EXit will be ignored. Since PLT does not actually plot any data until all the commands are executed, it is a good idea to precede an EXit with a Plot command, since that will force a plot to be produced.

If PLT exits normally, *i.e.*, with the the EXit command, then IER is set to zero. If the user enters an end-of-file then PLT exits with IER<0.

6.4 The QDP program

The complete source code for the QDP program is:

```
C Program QDP, the Quick and Dandy Plotter.
C Reads and plots a QDP file.
C---
C [AFT]
C---
      INTEGER   MXPTS, MXVEC, MXCMD
      PARAMETER (MXPTS=131072)
      PARAMETER (MXVEC=64)
      PARAMETER (MXCMD=50)
C
      CHARACTER CMD(MXCMD)*100
      CHARACTER CNAM*72
      REAL      Y(MXPTS)
      INTEGER   IERY(MXVEC)
      INTEGER   ICHAT, IER, LUN, NCMD, NPTS, NROW, NVEC
C---
  100 CNAM=' '
      ICHAT=0
      LUN=0
      CALL RDQDP(ICHAT, LUN, CNAM, Y, MXPTS, IERY, MXVEC,
     :   NROW, NPTS, NVEC, CMD, MXCMD, NCMD, IER)
      IF(IER.NE.0) GOTO 900
      CALL PLT(Y,IERY,NROW,NPTS,NVEC,CMD,NCMD,IER)
      IF(IER.LT.0) GOTO 100
C---
  900 CONTINUE
      END
```

The QDP program calls the RDQDP subroutine to read the QDP file, and then passes the data read to PLT. The parameter statements show that this version can read a file containing up to 131,072 numbers, up to 64 different vectors, and up to 50 PLT command lines. RDQDP sets the size of the array dimensions to make maximum use of the data array. For example, if you read a file containing two columns, then you could read up to 65536 rows of data. If the file contains 64 vectors and none of the vectors contains errors, *i.e.*, there are 64 columns of numbers, then the maximum number of rows will be 2048. If, however, all 64 vectors contain two-sided errors, then only 512 rows can be read. Each PLT command line can be at most 100 characters long.

QDP sets both `CNAM=' '` and `LUN=0` to force RDQDP to prompt for a QDP file name. If the file is opened, then RDQDP reads the file, and initializes all the variables needed by the PLT routine. If RDQDP has set `IER`=0 then some data has been read and hence the PLT routine is called.

PLT interprets the PLT commands and plots the data. If the user enters an end-of-file character at the `PLT>` prompt, PLT exits with `IER`=-1. This causes the QDP program to loop back and call RDQDP again. For normal exits, `IER`=0, the QDP program quietly exits.

6.5 The DEMO program

The file `XANADU:[PLOT.QDP]DEMO.FOR` contains a simple Fortran program that creates the necessary arrays and then calls the PLT subroutine. The complete source code for the `DEMO.FOR` program is:

```
C---
C DEMO.FOR demonstrates how to call PLT from a Fortran program.
C---
C [AFT]
C---
      INTEGER   MXROW, MXCOL, MXVEC, MXCMD
      PARAMETER (MXROW=200, MXCOL=3, MXVEC=2, MXCMD=10)
      CHARACTER CMD(MXCMD)*72
      REAL      Y(MXROW, MXCOL)
      INTEGER   IERY(MXVEC), NVEC, NPTS, NCMD, IER
      INTEGER   I
C---
C Create two vectors.  The first vector will contain the X locations
C and a symmetric error (with constant value of 0.5).  The second
C vector will contain X*X and no error.
      NVEC=2
      IERY(1)=1
      IERY(2)=0
      NPTS=100
      DO 190 I=1,NPTS
         Y(I,1)=I
         Y(I,2)=.5
         Y(I,3)=Y(I,1)*Y(I,1)
  190 CONTINUE
C---
C Now add a couple of commands, to make plot look nicer.
      CMD(1)='LAB X Time (sec)'
      CMD(2)='LAB Y Distance (m)'
      CMD(3)='LAB T Made with DEMO.FOR'
      CMD(4)='LINE STEP 2'
      NCMD=4
C---
C Call the PLT subroutine.
      CALL PLT(Y, IERY, MXROW, NPTS, NVEC, CMD, NCMD, IER)
      END
```

`DEMO.FOR` was written as an example to show how the various parameters are initialized before calling PLT. In the program, `NVEC=2` tells PLT to expect two vectors; `IERY(1)=1` tells PLT that the first vector contains symmetric errors and, hence, is composed of two columns; and `IERY(2)=0` tells PLT that the second vector does not have errors. The `DO 190` loop fills 100 points of these two vectors. The `Y` array is large enough to contain up to 200 points. The first column of the `Y` array contains the x-values, which run from 1 to 100; the second column contains the errors (constant value

0.5); and the third column contains x^2. After the `Y` array is initialized, the `CMD` array is initialized with four PLT commands. The first three commands define labels, and the last command creates a stepped-line plot.

The file `XANADU:[PLOT.QDP]DEMO.COM` will compile and link the `DEMO.FOR` program on a VMS system. This file can be used as an example for linking other routines that call PLT. It is necessary to link with both the XANLIB library and the PGPLOT graphics library.