RXTE GOF	***Plotting with PLT*** ***Recipes from the RXTE Cook Book***	***RXTE FAQ***

Introduction

In the course of your RXTE data reduction and analysis, you'll be plotting files using PGPLOT routines packaged together in a Fortran subroutine called PLT. This is the case whether you run the general plotting ftool fplot or the data analysis programs xspec and xronos. Here, we explain how to exploit the power of PLT to produce attractive plots tailored - we hope - to your needs.

This isn't really a set of recipes. Rather, it's a primer divided into sections that cover different aspects of PLT:

1. **Basic PLT Commands** lists the most common commands entered at the `PLT>` prompt to change the appearance of a plot.
2. **Multi-Panel Plots, Vectors and Errors** discusses how PLT deals with multi-y plots and lists more PLT commands, some of which can also be used with single-panel plots.
3. **More PLT Commands** continues the listing of PLT commands.
4. **QDP, the *Quick and Dandy Plotter*, and PCO files** describes QDP, a program for plotting ASCII files. It also explains PCO (Plot Command) files and some general issues about how PLT interprets data.
5. **Fitting in PLT** explains the modest, but useful, model-fitting capabilities of PLT.
6. **PLT in Xspec: Some Idiosyncrasies**
7. **PLT in Fplot: Some Idiosyncrasies**

Plotting Basics

- **Setting the plot device:** PLT plotting devices are specified with stings like `/ps`, e.g.:

```
/xw  for XWindows
/xt  for Xterm
/vt  for Versaterm
/ps  for PostScript
```

  Setting the plot device initially occurs outside PLT and depends on the program driving it:

  - **fplot:** Specify the device at the prompt, e.g.:

```
Device: /XWindow, /XTerm, /TK, /PS, etc[/XW] /xw
```

- **xspec:** Specify the device with the `cpd` command at the prompt, e.g.:

```
XSPEC> cpd /xw
```

- **xronos:** Specify the device at the prompt, e.g.:

```
Enter PGPLOT device[/XW] /xt
```

To see the full list of available devices type `?`.

Two additional options are worth mentioning here. First, if you want to enter some PLT commands before sending the plot to a device, you can enter `/null` or `>` as the device. Second, if you want to change the device once inside PLT, use the "device" command (abbreviated to `d`) at the prompt:

```
PLT> d /vt
```

- **Getting help:** Like all good interactive programs, PLT has on-line help. To access it, type "help" at the `PLT>` prompt.

- **Rescaling:** After setting the plot device, the plot will appear. If you want to rescale the x-axis to run, say, between values of 100 and 250:

```
PLT> r x 100 250
```

Rescaling the y-axis works in the same way; e.g. `r y 0.8 1.2` will redraw the y-axis to run between 0.8 and 1. And if you want to combine both rescalings in one, command:

```
PLT> r 100 250 0.8 1.2
```

- **Labelling:** To change the existing labels or to add new ones, the "label" command (abbreviated to `la`) is used. For example, here's the command for putting a label on the x-axis saying "Time (MET in seconds)":

```
PLT> la x Time (MET in seconds)
```

You can put labels at these other locations in the same way:

```
top .. at the top of the plot, above the plot window
otop . over the top label
y .... on the y-axis
oy ... over the y-axis label
ox ... below the x-axis label
f .... at the default filename location (above the top left corner of
          the plot window)
```

- **Logarithmic scales:** The commands `log x` and `log y` will make the x and y-axes logarithmic. To restore the linear scale, type `log off`. Make sure, before using this command, that all you points are greater than zero. If not, you should rescale before the log conversion.

- **Removing the time label:** By default, PLT puts the time at which the plot was made on the bottom right-hand corner. To remove it:

```
PLT> t off
```

The command `t on` will restore it.

- **Rotating the numbers on the y-axis:** The command `la ro` will make the numbers on the y-axis read horizontally rather than vertically.

- **Changing the window size:** The rectangular window containing the plot itself is known as the *viewport* and can be changed in size and shape. The viewport configuration is given by the coordinates of its bottom left and top right corners - in dimensionless units such that the length and breadth of the screen are both unity. The default is (0.1, 0.1, 0.9, 0.9) which provides a large viewport with space left for the labels. So, to change to a taller-looking window, you could type:

```
PLT> vp 0.25 0.1 0.75 0.9
```

- **Making a hardcopy:** The best way to make a hardcopy in PLT is to use the "hardcopy" command (abbreviated to `h`). This command temporarily changes your plot device to PostScript and allows you to create a file corresponding to whatever the current plot looks like. For example, to make the PostScript file `hard_pulse.ps`:

```
PLT> h hard_pulse.ps/ps
```

Please note the `/ps` which must be appended, without a space, to the name of the PostScript file. Alternatives to /ps comprise:

```
/PS     (PostScript file, landscape orientation)
/VPS    (PostScript file, portrait orientation)
/CPS    (Colour PostScript file, landscape orientation)
/VCPS   (Colour PostScript file, portrait orientation)
```

- **Thickening the width of lines:** A useful command to apply before creating a hardcopy is `lw`, short for "lwidth". It thickens all the lines and characters in the plot to increase its visibility - for, say, transparencies or figures that you intend to reduce in size for publication.

```
PLT> lw 3
```

for example, will make the lines three times thicker than the minimum, which is the default.

- **Changing the character size:** The simple command `cs` will change the size of all characters by whatever factor follows it. For example:

```
PLT> cs 1.5
```

will increase the size of text by 50 percent.

Multi-Panel Plots, Vectors and Errors

PLT can plot up to nine quantities simultaneously against a tenth - either all in one large panel or in separate horizontal panels. The relevant PLT term here is *vector* which refers either to a single array of data or to a combination of data array and associated error array. By default, the first vector is used as the x-axis and the others are plotted against it.

- **Multi-panel plotting:** To obtain a multi-panel plot, the command `p v`, for "plot vertical", is used - automatically, if you're using PLT inside fplot or xronos. When plotting in panels, the default is to use the first vector as the x-axis and to place the second vector in the top panel, the third vector in the next-to-top panel, and so on.

  To obtain a single-panel plot of all the vectors, the command `p o`, for plot overlay, is used.

- **Rescaling and labelling individual panels:** The vector number is used to identify the particular y-axis. For example, to change the next-to-top y-axis to run from 80 to 120:

  ```
  PLT> r y3 80 120
  ```

  Note that these commands do not cause an automatic replot, unlike the case for `r y` and `r x`. To see the changes, just type `p`. You can rescale the x-axis either with `r x` or with `r y1`.

  Labelling works in the same way. For example, to label the y-axis of the top panel with "Elevation Angle":

  ```
  PLT> r y2 Elevation Angle
  ```

- **Adding or removing error bars:** Error bars, if present, can be turned on or off with the commands `e on` and `e off`, respectively. The vector number is included, so, to stop plotting the error bars in the top panel:

  ```
  PLT> e off 2
  ```

- **Line plots:** The points in a given panel can be connected with lines. In fact, this display mode is the default for vectors without error bars. More than one line style is available, identified by numbers and displayed on screen with the "ls ?" command:

  ```
  PLT> ls ?
   PGPLOT line styles are:
   1=Solid, 2=Dash, 3=Dash-dot, 4=Dot, 5=Dash-dot-dot-dot
  ```

  To make the top panel have a dotted line connecting the points, for example, type:

  ```
  PLT> ls 3 on 2
  ```

- **Plotting in histogram style:** To produce a stepped, histogram-style plot, use the command `li st`, short for "line step", as in this example to make the top panel into a histogram:

  ```
  PLT> li st on 2
  ```

- **Symbols on the data points:** By default, data points are displayed either as crosses (if they have error bars) or as a line (if they don't). A third alternative is to use one of several symbols or *markers* as they're known in PLT. To see the list of markers, type `ma ?` and to put, say, marker no.14 on vector 4, type:

```
PLT> ma 14 on 4
```

- **Changing color:** The "color" command, abbreviated to `co`, works in the same way as the marker command. To see a list of available colors, type `co ?` and to change the color of the second vector from its default (red) to dark blue (color no.4), type:

  ```
  PLT> co 4 on 2
  ```

- **How to delete a panel from a plot:** Two steps are involved. First, indicate which vector you want removed from the plot with the `co off` (color off) command; second, issue a `p v` command to tell PLT to redistribute the panels (without this command, PLT will plot an empty panel). For example:

  ```
  PLT> co off 3
  PLT> p v
  PLT> p
  ```

  The final plot command is necessary because, by itself, `p v` doesn't send a plot to the device.

- **Changing the x-axis:** The command to change the x-axis from its default (vector-1) to, say, vector-3, is `x 3`. After this command, you'll almost always have to rescale the x-axis. And if you don't want the first vector to appear, you should color it zero.

QDP, the Quick and Dandy Plotter, and PCO files

As mentioned above, PLT is a Fortran subroutine called by programs like xspec and fplot which pass it arrays to plot. Another program that uses PLT is QDP, the Quick and Dandy Plotter. QDP is a handy program that drives PLT to plot ASCII files. As such, QDP can be thought of as the ASCII analogue of fplot which only plots FITS files.

- **Getting started:** QDP is part of the Xanadu package and is invoked by the command `qdp`.

- **Input files:** QDP takes as input ASCII files containing the data to be plotted in columns, like this short example:

  ```
  1.23 0.5 104 0.567 51.6 10.7
  2.04 0.4 112 0.411 58.8 10.6
  3.94 0.5 178 0.149 96.4 12.1
  5.12 0.6 181 0.148 17.6  8.9
  6.55 0.6 259 0.299 12.3 10.7
  7.90 0.5 411 0.515 19.6 10.7
  ```

  The default filename extension is `.qdp`.

  To tell PLT which columns are errors and which are data, the header commands `read serr` and `read terr` are used: `serr` for single-sided errors (one column of errors); `terr` for twin-sided errors (two columns of errors). **Note that xronos, xspec and fplot take care of this**

automatically: you only need to insert these header commands into QDP files you create yourself.

Thus for the sample QDP file above, the following header commands can all be used and will have different effects:

- `read serr 1 2 3`

  Will create a three-vector plot, with the third and fifth columns being plotted against the first column. The errors for the first, third and fifth columns being the second, fourth and sixth columns, respectively.

- ```
  read serr 2
  read terr 3
  ```

  Will create a three-vector plot, with the second and fourth columns being plotted against the first column. This time, the first column has no error bars; the second column has errors in the third column; while the fourth column has errors in the fifth and sixth columns.

- Omitting any header command will create a six-vector plot with columns 2-6 being plotted against the first, with none having errors.

- **Linear x-axes:** If you want to create a plot with a regularly spaced x-axis, you don't need to include a whole column of x values: the header command `x l` (short for "xaxis linear") can be used instead. For example, if you have 20 values of X-ray flux that you'd like to plot, with their errors, against orbital phase, you can create a QDP file by simply prefacing the data with the header-commands:

  ```
  x l -0.3107 0.05
  read serr 1
  12.39 4.0
  11.09 3.8
   7.79 3.4
    .    .
    .    .
    .    .
  ```

  where, in this example, the phase of the first bin is -0.3107, and the step size of 0.05, i.e. 1/20, is chosen because the data were taken at regular phase intervals spanning one complete orbit.

- **PCO (plot command) files:** Any of the commands that control the appearance of a plot can be put into a plot command (PCO) file. This option saves time when you have a series of plots that you'd like to look the same. Here's a sample of a PCO file called `burst.pco`:

  ```
  la otop The Bursting Pulsar GRO J1744-27
  la top RXTE PCA
  la f PCUs 0-3, all layers combined
  la y1 Time (seconds)
  la oy1 16-s bins
  la y2 Counts/s
  t off
  la ro
  ```
  ```

```
r y1 0 10000
```

PCO files, which have the default filename extension `.pco`, are invoked in PLT with an @ sign:

```
PLT> @burst
```

The same command can also be placed in a QDP file.

Note that the programs like xspec and xronos that call PLT use their own internal PCO files. For example, when you plot a spectrum in xspec, PLT is called with a PCO file that labels the x-axis with "Energy (keV)".

More PLT Commands

- **Changing character fonts:** PLT has four fonts: Normal (the default), Roman (a serifed font like Times), Italic (a sloping version of Roman) and Script (a cursive font). To change the overall font, i.e. for all the characters, use the `font` command, for example:

  ```
  PLT> font roman
  ```

  If you want a few characters to be in a different font, then use the `\f` flag followed by the first letter of the font name. For example, suppose all your characters are in Roman font, but you'd like the name of the spacecraft, as it appears on the filename label, to be in Italic. The corresponding commands would be:

  ```
  PLT> font roman
  PLT> la f \fiRossi X-ray Timing Explorer
  ```

- **Greek letters:** The Greek font associates a Roman letter (or letters) with a Greek letter. It is case-sensitive and is activated with the `\g` flag. In the following example, the first command includes a capital gamma in the y3 label, while the second includes a lower-case rho in the y4 label:

  ```
  PLT> la y3 \gG
  PLT> la y4 \gr \fn(cm/3)
  ```

  Note the use of the `\fn` flag to reset the font (to Normal) for the rest of the label.

- **Subscripts and superscripts:** You can include subscripts and superscripts in your labels with the "down" and "up" flags `\d` and `\u`, respectively. Note they work relatively, e.g. using a down flag after an up flag will cause the text to appear at the usual level rather than as a subscript. An example makes this clearer:

  ```
  PLT> la y \fiF\fn\d2-10 \u(erg cm\u-2\d s\u-1\d keV\u-1\d)
  ```

  roughly corresponds to the TeX:

  ```
  $F_{\rm 2--10}$ (erg cm$^{-1}$ s$^{-1}$ keV$^{-1}$)
  ```

- **Putting labels anywhere inside the window:** Labels can be put anywhere inside the plotting window - not just outside it in the "standard" positions. Here's an example:

  ```
  PLT> la 1 p 6.4 12 "Iron Fluorescence"
  ```

  The label command is followed first by a position number, in this case 1, which serves to identify it as a "position" label (as opposed to one of the standard labels) and to distinguish it from other position labels. The next part of the string, "p 6.4 12", contains the location of the center of the label: the first number is the x-coordinate *in the same units as the x-axis* and the second number is the y-coordinate *in the same units as the y-axis*. Finally, comes the label itself enclosed in double quotation marks.

  There are other options: for help, type "help la #" at the `PLT>` prompt.

- **Grid lines:** The command `gr on` will expand the major tick marks on the x and y-axes into a grid covering the window. The issue of tick marks is fairly involved: for more information, type "help gr" at the `PLT>` prompt.

- **Writing out a QDP file:** The command `wd` - short for "write data" - will create a QDP file corresponding to the data you're plotting. Obviously, there's little point if your data started off in a QDP file, but it can be useful tool in programs like xspec which process input files considerably before actually plotting them. The motivation of using the `wd` command could be to get a file for input into your own plotting program, or, say, to obtain in a simple, direct way, an array containing the effective area of a detector (via xspec's "iplot efficiency" command).

- **Changing background and frame colors:** If you're using a color graphics device such as `/xw`, you can change the color of background with the command `scr` (set color representation). The first argument is an index to identify the element of the plot:

  ```
  0  for the background
  1  for the frame and labels
  2  for the data themselves
  ```

  The next three arguments are the normalized intensities (to unity) of the red, green and blue cathode ray tubes. For example, to change the background color to a deep purple:

  ```
  PLT> sc 0 0.6 0 0.6
  ```

- **Spawning out of PLT:** To get a shell prompt without leaving PLT, type `$`. For example, the two commands:

  ```
  PLT> h hard_pulse.ps/ps
  PLT> $ lpr hard_pulse.ps
  ```

  will first create a PostScript file, then send it to the printer. If you want to execute more than one shell command, type $ without anything after it.

- **Even more PLT commands:** To see all the PLT commands:

  ```
  PLT> help ?
  ```

This will provide an entrance point for more information.

Fitting in PLT

PLT contains a chi-squared minimization routine and a suite of simple models that can be used to fit your data. Models are defined with the `mo` command (`mo ?` will list the choices) and can be built up of more than one component. For example:

```
   PLT> mo const gauss
```

will define a model comprising a constant plus a Gaussian. After defining a model, you'll be prompted for the initial parameter values, like this:

```
PLT> mo co gaus
  1 CO: VAL(  1.000    ), SIG( 0.0000E+00), PLO( 0.0000E+00), PHI( 0.0000E+00)?
 10
  2 GC: VAL(  1.000    ), SIG( 0.0000E+00), PLO( 0.0000E+00), PHI( 0.0000E+00)?
 1700
  3 GW: VAL(  1.000    ), SIG( 0.0000E+00), PLO( 0.0000E+00), PHI( 0.0000E+00)?
 300
  4 GN: VAL(  1.000    ), SIG( 0.0000E+00), PLO( 0.0000E+00), PHI( 0.0000E+00)?
 10
```

where the user input in this example is emboldened. Here, only the initial values of the parameters have been specified, but it's also possible to fix the parameters and to set upper and lower limits. This is done by entering values for the other fields associated with each parameter. Setting `SIG` to -1 will fix a parameter, while setting `PLO` and `PHI` will constrain the range of the parameter during the fits to be no less than `PLO` and no greater than `PHI`.

To initiate a fit, type "fit" followed by the number of the vector you want to fit. [Note that the fit will include only those x-values in the current window, i.e. the full range unless you rescaled the x-axis before the fit.] The screen output will look like something this:

```
PLT> fit 2
 Fitting group   2,  from  0.000E+00 to  3.000E+03
 Fitting    187 points in a band of    187.
   10.00000       1700.000       300.0000       10.00000
  ( -3)   W-VAR=5.6675E+04
  ( -3)   W-VAR=3.6336E+04
  ( -4)   W-VAR= 7212.
  ( -5)   W-VAR= 2779.
  ( -6)   W-VAR= 283.4
  ( -7)   W-VAR= 32.80
  ( -8)   W-VAR= 32.74
  -15.75922       1678.088       1096.758       92.64162
```

Here, the initial parameter values are shown, along with the fit iterations, then the final, best-fitting parameter values.

Confidence ranges are obtained with the "fit err" command which takes, as its argument, the model parameter number. For example:

```
PLT> fit err 3
```

will calculate the 90-percent confidence range of the third fit parameter.

If you want to reset a parameter value, then use the "newpar" command.

Other fitting commands: "fit sub" will subtract the current model from the data to produce a new column of data; "wm" (for "write model") will create an ASCII file containing the current model and its parameters. You can read it back in with the "mo @" command, e.g.:

```
PLT> wm cdm.mod
  .
  .
  .
PLT> mo @cdm.mod
```

PLT in Xspec: Some Idiosyncrasies

The spectral fitting program xspec uses internal, "pre-programmed" PCO files extensively. Whenever you type "plot data resids", for example, PLT is instructed to create two windows, one with the current model and data, the other with the residuals. In both cases, the x-axis, if energy not channels, is obtained from the `EBOUNDS` extension of the response file. Descriptive labels are added, including the path to the current data file in the filename position. Other things can be plotted; to see the list, type:

```
XSPEC> plot ?
```

If you want to alter the appearance of an xspec plot, you should use the `iplot` command which will send a plot to the current device (set with the `cpd` command) and then give you a `PLT>` prompt at which you can enter any of the commands above. For "double" plots (e.g. `iplot ldata chi`) PLT is called twice and overlays two windows on top of one another. By default, the top window (usually the one with the data and model) is window-1, while the second (often with the residuals) is window-2. To work in window-1, type

```
PLT> win 1
```

before changing anything. Not specifying the window can sometimes lead to puzzling consequences. For example, if you label the top "Thermal model omitted" but don’t see anything, it might be because the label, being associated with the residuals window, is actually covered by the data window.

PLT in Fplot: Some Idiosyncrasies

Like xspec, the plotting ftool fplot uses internal, "pre-programmed" PCO files, though not as extensively. Here’s a typical run:

```
kaah[45]%
Name of FITS file and [ext#] data3_hard.lc
Name of X Axis Parameter[error] TIME
```

```
Name of Y Axis Parameter[error] up to 8 allowed RATE[ERROR]
Lists of rows[-]
Device: /XWindow, /XTerm, /TK, /PS, etc[?] /xw
Any legal PLT command[]
```

And here's some explanation:

1. **fplot offset=yes:** You should explicitly set the hidden parameter `offset` to be "yes" if you want the first x-value to be subtracted from all the x-values to produce an x-axis starting at zero (the default is "no"). This subtraction will occur before PLT is called.

2. **data3_hard.lc:** Self-explanatory.

3. **TIME:** The column in the FITS file to be used as the x-axis. As we saw in the section on QDP, PLT needs to know whether the second column it encounters is either the x-error array (if there is one) or the first y-array. This is done by enclosing in brackets the name of the column containing the x-error array. Note that fplot is case-sensitive, so you might want to check the column names beforehand with the flcol ftool.

4. **RATE[ERROR]:** In this example, the y-array `RATE` does have an associated error array, called `ERROR`.

5. **Lists of rows[-]:** This allows part of the file to be plotted, if desired. The removal of array elements occurs before PLT is called.

6. **Device: /XWindow, /XTerm, /TK, /PS, etc[?] /xw:** Self-explanatory.

7. **Any legal PLT command[]:** An option for passing a command to PLT before the plot appears. Note that because of the way fplot is written, using this option overwrites fplot's internal PCO file which takes labels for the axes from header information in the file itself.

If you have a question about RXTE, please send email to one of our help desks. This page is maintained by the RXTE GOF and was last modified on Friday, 19-Jun-1998 17:59:57 EDT.

XTE Project Scientist Jean Swank, swank@pcasun1.gsfc.nasa.gov
Technical Rep: Sherri Calvo, sherri.calvo@gsfc.nasa.gov