Appendix B: `plot`

`plot` is a general purpose, advanced plotting program. `plot` can be frustrating to use because it assumes you know what you want to do and actually mean what you say—so if you don't know what you're doing and don't know what you say means to the machine, you can draw a mess fast. Of course, after you know the program you should be able to draw a beautiful graph as fast as you drew that mess.

The most unusual feature of `plot` is the way you give values to the parameters needed to draw a graph, e.g., the size of the graph, maximum and minimum values, etc. Instead of asking you if you want to change each of those 39 parameters, `plot` allows you to set them in any order, at any time you want. For example, if you want to change the minimum value of x to 0, you type

```
* SET XMIN=0
```

(Don't forget to hit the *Return* or *Enter* key after each line. Note: you cannot edit these lines using arrow keys: you must backspace!)

So the first thing you need to know is the name of the parameters. You will find a complete list at the end of this appendix but let me mention three important sets of parameters now. The first set determines the way the axes look: `XMIN, XMAX, XSCALE` and similarly for y. `XMIN` and `XMAX` should be self explanatory. `XSCALE` determines the nature of the scale:

`XSCALE=1` is a linear (normal) scale;

`XSCALE=2` is a logarithmic scale (distance in plot is proportional to $\log x$);

`XSCALE=3` is an inverse scale (distance in plot is proportional to $\frac{1}{x}$); and

`XSCALE=4` is a probability scale (distance in plot is related to $\text{erf}^{-1}(x)$).

(Note: you will rarely actually set `XMIN` and `XMAX` as there are autoscale commands, but you will often be setting the type of scale.)

The second set determines how the program finds the proper data to plot. Basically to get numbers (x, y, and errors) into the program you need to make a file with the editor. The first line of this file should be a one-line description of the data in the file (i.e., `free fall experiment - trial #2`). The following lines should consist of data—an x value, some spaces (or tabs or commas), the corresponding y value, some spaces, and the corresponding absolute error in y. Thus when looked at as a whole, the first column will consist of x values, the second column the corresponding y values, etc. If you called this file `plot.dat`

you have made a file exactly in the default manner. The program will be able to `READ` this file without telling it anything. If you put the x values in column 2, you will have to inform the program by typing

```
* SET XCOL=2
```

That is, `XCOL` is a variable that tells the program which column contains the x values. Similarly, `YCOL, YECOL` (y-error) and `XECOL` (x-error) are variable names. If you have no errors, tell the program by typing

```
* SET YECOL=0
```

If you didn't call your file `plot.dat`, you must tell the program by typing

```
* SET FILE=filename
```

Note the above `SET`ing does not actually change the data inside the program. The data inside the program is only changed when you:

```
* READ
```

Finally the third set determines how functions are plotted.

```
* SET F(X)=SIN(PI*X)
* SET PMIN=-1 PMAX=1
* FCURVE
```

`FCURVE` will plot the function in the domain `PMIN` to `PMAX` (the "p" is for parameter—remember `xmin` and `xmax` have another meaning). Note that the border parameters (`ymin`, `ymax`, etc.) must be appropriately[1] set before the function curve is added to the plot. `F(X)` can be set using the usual computer syntax[2] for algebra. There are eleven built-in functions that correspond to the usual *WAPP*$^+$ functions with the usual parameters `A`, `B`, and `C`. These can be plotted by `SET`ting the function number `NFUNCT` to a value between 1 and 11. `NFUNCT=0` (the default) plots the function `F(X)`.

You use `SET` to tell the program on how you want your data displayed. To actually display data you need action commands. The sequence:

```
* READ
* SCALE
* BORDER
* DPOINT
```

[1] `FSCALE` will find appropriate `ymin`, `ymax`, etc., based on the function `f(x)` and domain `PMIN` to `PMAX`, but more commonly the border parameters — including `PMIN` and `PMAX` — are set to fit the data with `SCALE`

[2] Note that you must use `*` for multiplication not just juxtaposition.

tells the program to read the data (from the previously set `FILE`); `SCALE` tells the program to find appropriate `XMIN`, etc. given the data contained in the program; `BORDER` tells the program to display the graph outline you have described; `DPOINT` tells the program to plot the data points. Type `FCURVE` to plot your function. Another useful command is `HELP`—it types out a listing of the commands and the variables.

Actually all this `SET`ting and commanding can be done together as shown below: (note one command per line, but you may define several variables)

```
* READ YECOL=0, FILE=freefall1.dat
* SCALE
* BORDER TITLE='Free Fall', XLABEL='Time (s)', YLABEL='Z (m)'
* DPOINT
* FCURVE PMIN=0, PMAX=.65, NFUNCT=10, A=0, B=0, C=4.9
```

Two final comments: The equal signs and commas in the above example make the reading of the line easier—but they make the typing of the line harder. You may substitute spaces for them to make typing easier. Similarly, one gets tired of typing in the full command in capitals—lowercase and any unambiguous abbreviations may also be used. Thus instead of the above you may type:

```
* re yec 0 fi freefall1.dat
* sc
* bo tit 'FREE FALL' xl 'TIME (s)' yl 'Z (m)'
* dp
* fc pmin 0 pmax .65 nfu 10 a 0 b 0 c 4.9
```

Getting Hardcopy Plots

Now that you have a plot displayed on the terminal, you should want a copy of the screen to put in your lab notebook. The command `PCOPY` will make a copy of the displayed plot to the file specified by `PFILE`. If `PFILE=" "` (as it is by default), the output goes to the printer.

Special Commands

Start—in a terminal type: `plot`
$*linux* command — do any *linux* command from inside `plot`
@*filename* — execute indirectly a file of `plot` commands
Ctrl D — to exit program
Examples:

```
* $ls *.dat
* @fet.plt
```

Report problems/suggestions to Tom Kirkman.

`plot` Commands

(unambiguous abbreviations and lowercase allowed)

Syntax:

```
* COMMAND
* COMMAND VARIABLENAME = value, VARIABLENAME = value ...
```

Examples:

```
* BORDER XMIN=0, XMAX=1, YMIN=-1, YMAX=1, TITLE='now is the time'
* bo xmi 0 xma 1 ymi -1 yma 1 ti 'now is "the time"'
```

`BORDER` draws a border for a graph

`CLEAR` clears the screen

`CROSSH` digitizing mode—using the mouse move the 'crosshair' to the desired point. Hold the Shift key and click; the x and y value at crosshair is reported.

`DCURVE` connects with a `LINE` `NPOINT` data points, starting with `IBEGIN`.

`DPOINT` plots `NPOINT` individual data points starting with `IBEGIN`; point symbol determined by `POINT`, error bars by `XECOL` and `YECOL`

`FCURVE` plots the function `NFUNCT` for p from `PMIN` to `PMAX` in `NSTEP` steps. $p = x$ for `NFUNCT`s 0–11, `NFUNCT`=0 displays `F(X)`, link in funct(x,y,p,nfunct) for `NFUNCT` $>$ 11

`FSCALE` finds appropriate mins and maxs for x and y axes for `FCURVE`

`HELP` displays these messages

`LABEL` writes `LABEL` to the right of most recent `CROSSH`air

`PCOPY` copies plot to printer (lpr -Plp) or to `PFILE`

`PRINTD` displays data stored in program

`READ` reads data from `FILE`— see `XCOL, YCOL, XECOL, YECOL, ROW, IBEGIN, NPOINT`

`SCALE` finds appropriate min and max for x and y axes from `NPOINT` data pairs starting at `IBEGIN` (automatic 5% excess unless scale=0). Room for error bars included if x & y `ECOL` not zero

`SET` allows you to set variables values without doing any action

`SHOW` shows variable values

`plot` Variables

(Default settings are indicated in parentheses.)

`A` used to give values to function parameters (see `NFUNCT`) (0)

`B` same
(Note: `A` and `B` are used in `WINDOW(X)`: `WINDOW(X)=1` iff $a \leq x \leq b$, else zero)

`C` same

`F(X)` function: use `A, ABS, ACOS, ASIN, ATAN, B, C, COS, COSH, EXP, INORM, K, K1-K9, LOG, LOG10, NORM, P0-P9, PI, SIN, SINH, SQRT, TAN, TANH, WINDOW, X,` any real number, operations `+, -, *, /, **,` or `^` (default: 8^{th} degree polynomial)

`FILE` `READ` seeks data from a file with this name (plot.dat)

`FONT` character size— 1:small, 2:standard, 3:big, 4:huge; 3&4 plotter only (2)

`IBEGIN` first array element used during `READ, DPOINT, DCURVE, SCALE, PRINT` (1)

`LABEL` string to be placed in plot using `CROSSH` and `LABEL` (LABEL)

`LANG` graphics language (1)

`LINE` pattern— 1:solid, 2:dotted, 3:dashed, 4:long-dash, 5:DASH-dot, 6:DASH-dash, 7:DASH-dash-dash (plotter only) (1)

`NFUNCT` 0:`F(X)`, 1:$y = a + bx$, 2:$y = ax^b$, 3:$y = a\exp(bx)$, 4:$y = a + b\ln(x)$, 5:$y = a + b/x$, 6:$1/y = a + bx$, 7:$1/y = a + b/x$, 8:$y = a\exp(b/x)$, 9:$1/y = a + b\ln(x)$, 10:$y = a + bx + cx^2$, 11:$y = ax^b$ >11:link funct(x,y,p,nfunct) (0)

`NPOINT` number of points used during `READ, PRINT, DPOINT, DCURVE, SCALE` (512)

`NSTEP` number of subdivisions of p range — used during `FCURVE` (100)

`PATSIZ` spacing of `LINE` pattern in fraction of screen width (.04)

`PEN` plotter pen number (1)

`POINT` type of data symbol 0:dot, 1:circle, 2:square, 3:diamond, 4:triangle up, 5:triangle down; `POINT`<0 for filled symbol (2)

`POISIZ` size of symbol in fraction of screen width (.004)

`PMAX` maximum value of p — used during `FCURVE` (p equals x for `NFUNCT`=0–11) (1)

`PMIN` minimum value of p — used during `FCURVE` (0)

`ROW` first row of data in `FILE`— used during `READ` (2)

`TERM` 1:visual 550, 2:selanar, 3:else (2)

`TICSIZ` size of axes ticks in fraction of screen (.015)

`TITLE` title of graph—put on during `BORDER` (GRAPH)

`XCOL` column of x data in `FILE`— used during `READ` (1)

`XECOL` column of x-error data in `FILE`— used during `READ, SCALE, DPOINT; XECOL`=0 suppresses x-error bars (0)

`XLABEL` label on x axis— put on during `BORDER` (X)

`XMAX` largest x value in plot (1)

`XMIN` smallest x value in plot (0)

`XOFF` offset of the start of the x axis— unit: fraction of screen (.15)

`XSCALE` type of x scale— 1:linear, 2:logarithm, 3:inverse, 4:probability (1)

`XSIZE` size of x axis— unit: fraction of screen (`XSIZE` + `XOFF` $<$ 1.) (.8)

`YCOL` column of y data in `FILE`— used during `READ` (2)

`YECOL` column of y-error data in `FILE`— used during `READ, SCALE, DPOINT; YECOL`=0 suppresses y-error bars (3)

`YLABEL` label on y axis— put on during `BORDER` (Y)

`YMAX` largest y value in plot (1)

`YMIN` smallest y value in the plot (-1)

`YOFF` offset of the start of the y axis— unit: fraction of screen (.1)

`YSCALE` type of scale— 1:linear, 2:logarithm, 3:inverse, 4:probability (1)

`YSIZE` size of y axis—unit: fraction of screen (`YSIZE` + `YOFF` $<$ .75) (.6)

> If there were some machine, somewhere, that could spit out universes with randomly chosen values for their fundamental constants, then for every universe like ours it would produce 10^{229} duds.
>
> Though I haven't sat down and run the numbers on it, to me this seems comparable to the probability of making a Unix computer do something useful by logging into a tty and typing in command lines when you have forgotten all of the little options and keywords.
>
> *In the Beginning... Was the Command Line* by Neal Stephenson (1999)