PLOTXY: A VERSATILE PLOT PROGRAM

Robert Parker and Loren Shure

INTRODUCTION

Plotxy is a program for generating graphs from data files with a minimum amount of fuss. In the simplest case, data points have been generated as x-y pairs in an ASCII file called *xydata* perhaps written with a formatted Fortran write statement or created with an editor, such vi. The program reads the file, and plots y as a function of x, interpolating with straight lines; axes are automatically assigned with reasonable limits and annotations. All this can be done with the three commands

```
file xydata
read
plot
stop
```

Here a data file with the name *xydata* has been read format-free (that is with * instead of a format in Fortran). A PostScript file named *mypost* has been written; its contents may be displayed to the terminal with *pageview*, or *ghostscript* or sent to a printer for hardcopy. All printers at IGPP (UCSD) are PostScript printers. As documented in the rest of this writeup, it is possible to add embellishments of considerable complexity: for example many series on one plot, plotting of symbols at the points, cubic spline interpolation of continuous curves, filled-in areas, logarithmic scales, error bars, titles, axis labels and many other decoartive elements.

The program operates from the command line in a simple language. You are not prompted, but instead you enter instructions to tell the program which options are needed. Almost every option has a default value, so that if nothing is mentioned about a particular parameter the default is taken: for example, the default plotting scales are linear in x and y. Once a particular option has been invoked, it remains in force until altered. (Exception: **fill**).

BASICS

Upon execution the program prints the prompt:

```
Enter commands for graph 1
```

Now you must type commands selected from the catalog below; each command begins in the 1st position of a new line; it may be followed by some literal or numerical parameters, which must be separated from the command word by a space. Any command may be abbreviated by its first four characters. Alternatively, particularly with complex graphs, you should prepare a file of commands to run by redirecting the standard input as follows:

```
plotxy < myfile
```

where *myfile* contains the list of commands for the program.

To read data from a file there are several commands defining the attributes of the data to be input; an obvious example is the name of the disk file, defined by **file**; other input attributes are things like whether this data set is to be connected with a smooth interpolating curve (**smooth**) or to be plotted as individual points (**symbol**). Having set up all the necessary specifications, you perform the actual input with the **read** command. A second data series may be read by using **read** again. The numbers may come from the same file or, by resetting the input file name, a different one. All the parameters remain in force from the previous read unless they are specifically altered. Each of the data series with its different properties is accumulated in memory ready to be plotted.

Plotting is accomplished with **plot**. If this is the first call to **plot**, all data series read up to this point will appear on one graph. Actually all that is done by this command is to create the file *mypost* which must be displayed or sent to a printer after *plotxy* has halted. Some parameters apply to the whole plot, and may be set at any point before **plot**. For example, there are the x and y axis labels (**xlabel**, **ylabel**) and the plot title (**title**). The size of the plot defaults to about 5 across by 6 high, with extreme values slightly larger than those found in the input series. These things can be overridden using **xlimit** and **ylimit**. When all the data are strictly positive, logarithmic scales can be set with **logxy**.

Before typing **plot** it is sometimes helpful to inspect the series and the way in which the data are going to be interpreted. This is done with **status**, which provides a synopsis of each data series and most of the current parameter settings.

Additional graphs may be created by reading in new data and invoking **plot** again as often as necessary. To terminate the program type **stop**. The program halts and the plotfile can be displayed or printed.

COMMAND CATALOG

The commands are given their full English names here (though only four characters are needed). Commands may be given in upper or lower case letters. The characters following the first blank after the command word itself are termed the 'command field'. A command line may be 120 characters long. Parameters that may be omitted in the command field are enclosed in square brackets. The parameters may be numbers, file names or text; it should be obvious from the context which is appropriate. A list separated by slashes denotes a set of possible alternative items. A line beginning with one or more spaces is ignored and may be used as a comment. In mathematical formulas an asterisk denotes multiplication and double asterisk exponentiation.

affine a b c d

Transforms the x and y coordinates of the next and subsequent data series to be read according to new(x) = a*x + b, new(y) = c*y + d. This is an affine transformation.
Blank command field a = 1, b = 0, c = 1, d = 0

background color

Color the rectangular region framed by the current axes uniformly with the specified color; see command **color** for how to name the color, and **palette** on how to customize it. The background is opaque, so it will cover previous plots if the current graph lies on top of earlier material; this makes the color white useful. Setting color to *none* returns to a transparent plot background.
Blank command field retain previous setting.

cancel [n]

Removes the last n data series read into memory. If no series have been read in, do nothing. If n is greater than the number of series currently present, all the series are removed without complaint.
Blank command field n = 1

character h [angle]

Change the height of the lettering in titles, labels, notes and axis numbering to h inches. The new value applies to the subsequent text to be read, so that different height letters can appear in the title, the axis notations, etc. The value of h just

before **plot** determines the height of the axis numerals; if this is zero, axis numerals and tick marks are both suppressed (also see **frame**). The optional parameter angle specifies the angle at which text will be plotted in the next **note**. Letter height can also be controlled by the a special text phrase (see **LETTERING**).
Default h = 0.11, angle = 0

color color
color n

On those devices capable of color, this command sets a new color from the list: black, red, blue, green, brown, orange, yellow, purple, gray, white, lightgray. White is useful for drawing on top of dark curves. Colors may be abbreviated to their first three letters, or referred to by number: 0 or 1 black, 2 red, 3 blue, 4 green, 5 brown, 6 orange, 7 yellow, 8 purple, 9 gray, 10 white, 11 lightgray. This describes the PostScript implementation, but there may be variations depending on hardware. See **palette** for a way to customize the color selection. The color of a plotted data series is the one set at the time of the associated **read** command; similarly with labels, notes, etc. Also the axes and the frame are drawn in the color specified at the time **plot** is called.
Default black

dash [s1 s2]

Plot the next data series to be read in as a dashed line, with visible segments s1 inches long and missing segments s2 inches long. **dash** is an input attribute applying to the next **read** command. To return to an unbroken curve, set s1 or s2 to zero.
Default s1 = 0, s2 = 0
Blank command field s1 = 0.2, s2 = 0.1

file filename

Defines the file name of the diskfile from which data are to be read; or the symbol * which implies read from the console. The name must consist of 64 or fewer characters. After this command the next **read** statement will begin at the beginning of the file.
Default = *
Blank command field = Rewind current file

fill

In the PostScript model only, fills a closed polygon based on the next input series with the current color. A closed curve is created by joining the first and last points of the series with a straight line. **Fill**, unlike most commands, does not remain in force, but applies only to the next **read**. This command is incompatible with **mode**s 3 and 30. Polygonal symbols (e.g. hexagon) are drawn filled when this command is on. If parts of the curve go outside the plot window, the results are unpredictable. Also a polygon with a dashed boundary cannot be filled.

format format specifier

Defines a Fortran format for reading the next data series from a diskfile. The format specifier may be

(1) a normal Fortran format specifier enclosed in parentheses, for example
format (2g12.4)

(2) the single character * meaning ‘format free’ reading;

(3) the single character *b* meaning a binary read.

In each case the data are read with a single Fortran read statement of the appropriate type. Usually, if the numbers can be unambiguously read by a person without the need to skip certain columns or other tricks, there is no need to use an explicit format – the default * works well. In type (1) never use an I format because values are stored as REAL variables; thus a number written with I4 must be read with an F4.0 format. Always remember the space after the word **format** and the parentheses in mode (1).
Default = *
Blank command field = *

frame *+box –box grid –grid none*
frame *–xnum –ynum –xaxis –yaxis*
frame *top/bottom left/right*
frame *+xeven +yeven*

If *+box* is present, two more sides are added to the axes to complete a rectangular box around the plot; *–box* removes the box. *grid* includes a lightly dashed grid corresponding to the axis tick marks as well as adding a box; *grid-solid* draws solid gray grid lines rather than dashed ones; *–box* cancels the grid and the surrounding box and returns to the normal situation with two orthogonal axes. –grid removes the grid. The option *none* completely strips the graph of axes, the box and the grid.

The second form of the command allows you to delete the numerical annotation from either axis: with *–xnum* the numbers are removed from the x-axis; with *–ynum* from the y-axis. The tick marks remain; they may be removed by setting the character height to zero immediately before plotting. Numbers can be restored with *+xnum* and *+ynum*. Similarly a complete axis may be removed with *–xaxis* or *–yaxis*, and restored later with plus signs. When contradictory options are issued (*–xaxis* and *+xaxis* both present, for example) the program takes the option omitting the least amount of information.

The third form allows the user to draw an axis on the "wrong" side of the graph: at the right or at the top; the labels move with the axes. Obviously *bottom* and *left* restore the axes to their conventional locations.

The last form controls the tick mark style: normally numbered ticks are larger than those without numerical labels. The desirability of this arrangement is not always unambiguous. The height of the ticks on an axis can be made even with +xeven for the x axis, +yeven for the y axis. The default is restored with –xeven, –yeven.
Default *–box*
Blank command field *+box*

help

Lists the four-letter abbreviation of all the commands. This may remind you of a name you have forgotten. Some systems also allow access to the documentation file.

landscape

In the PostScript version, causes the output to be rotated by 90 degrees clockwise, that is, landscape mode. The command must be issued before the first **plot** command in order to be effective. The orientation state cannot be changed back to the conventional (portrait) style unless a new plotfile is created with **output**. Then, if **landscape** is used a second time, before the first plot on the new file, the state is reversed from its current one.

logxy [style]

logxy [n]

Specifies the type of scales for the next plot with one of the following styles: *linlin, loglin, linlog, loglog,* where the first syllable describes the x axis, the second the y axis, so that *loglin* means the x axis is logarithmic and the y axis linear, and so on. The style may also be specified by an integer: *0 linlin, 1 loglin, 2 linlog, 3 loglog.*

Somewhat obscurely, style may also be *equilin* (or 4). Then a linear scaling is performed, but instead of scaling x and y separately, the data are scaled equally and sized to insure the data area is contained within the plot window. Thus the original proportions of the data are preserved, circles remain circles, etc. Usually x1, x2, y1, y2, in **xlim** and **ylim** will be unset (or zero) but if not, the selected portion of the x-y plane will be drawn in proper proportion within the area of the graph.
Default *linlin*
Blank command field *loglog*

mode n [col1 col2 col3]

Defines how the input data are grouped in the next and subsequent **read** commands. The integer n may be 1, −1, 2, 3, −3, 4 or 10, −10, 20, 30, −30: mode 1 implies data are simply consecutive y values with x uniformly increasing, beginning at x = 1 and unit increment (these x values may be modified with **affine**); mode = −1 means that x values are read and that y increases uniformly from 1 with unit increment; mode = 2, the default, means data are x, y pairs; 3 means data are x, y, z triples in which z is taken to be the uncertainty in y (an error bar is plotted between y−z and y+z). mode = −3 means the third member of data group is the uncertainty in the x-value. A symbol may be plotted at the actual value of y itself if **symbol** is set. If s, the symbol height, is set greater than or equal to 10, a symbol is drawn with height z instead of the bar. Also see NOTES 5. When mode is 4 the x and y data must be read by separate **read** commands, the x series being input first. The series length is that of the x series.

In mode 10 it is assumed the input file is in the form of a table, and a single column is picked to be plotted against uniform x; the selected column is the second argument, col1. With −10 x is plotted against uniform y. Similarly, mode = 20 is like mode 2, allowing you to read x from col1 and y from col2, two columns from the table. An example is mode 20 5 2, which means column 5 is used for x and column 2 for y in the x-y series. Modes 30 and −30 are similar, picking the three columns from the table in the obvious way. In these modes any supplied **format** is ignored and replaced by ∗. Chaos results if there are too few numbers in any row of the table.
Default n = 2
Blank field after n = 10, 20, 30, col1 = 1, col2 = 2, col3 = 3

nocomment

In normal operation command lines can be annotated by comments, indicated by the % sign. Everything after a % on the line is ignored. But users may wish to use the symbol % in labels, notes, file names and so on. Including this command causes the commenting facility to be permanently turned off from that point on, thus restoring the use of % to the user.

note (x y [in] [c] [r]) text
note (p q x y [in] [c] [r] [f]) text
note (+) text
note (v) text
note filename

In the first two options above, *plotxy* reads the characters of text to be plotted on the graph at the coordinates x, y (which do **not** participate in the current **affine** transformation). The text may be up to 80 characters in length. If the optional *in* appears, the coordinates (x, y) refer to the bottom left corner of the first text character, measured in inches from the intersection of the axes; otherwise x, y are in the units of the graph. Normally, the coordinates x, y refer to the bottom leftmost point of the first character of the text, but if the letter *c* appears, the text is centered above the point (x, y); and if *r* appears, the base of the end of the line is placed there (right justification). Notice the parentheses surrounding the coordinates are mandatory. The height of the plotted characters is the value h in the most recent **character** command. Similarly the angle the text makes with horizontal is the one previously set in **character**. To see how many separate notes are allowed in your version of *plotxy*, enter **status** (50 is common). To clear the notes, enter **note** and a completely blank command field (this means omitting the coordinates, of course); this must be done explicitly for new graphs to begin a fresh set of notes, otherwise the notes from the previous plot will be carried forward onto the current one.

If no coordinates are supplied, but a single + appears instead, the text is appended to the material of the previous note, thus overcoming the limitation on charcter count in a single note.

If a single v appears, place the new material vertically under the previous note, left justified.

If four coordinates instead of two appear in the parentheses, an arrow is drawn with its tip at p, q and its tail tastefully near the text, which is plotted as before with x, y as specified by r, c, or left justified by default. The text is always horizontal with this option. The letter f following the coordinates causes the arrow head to be filled in. If no text accompanies a four-coordinate note, the arrow tail goes exactly to the second position.

The following option allows a large number of short notes to be read from a file. When a filename follows the command, coordinates and notes (one set per line) are read from the named diskfile to its end. Up to 300 such notes can normally be read (Use **status** to check this). Each note in the notefile is at most 16 characters long and the whole series comprises horizontal text, homogeneous in character size and color specified at the time when the file is read. Font changes can be induced in the usual way by phrases in the notes themselves. The coordinates are in graph units only and must *not* be enclosed in parentheses. The notes are all left justified. These notes are canceled by **note** with a blank command field just like those specified as commands. If it is desired to begin the note with a space, terminate the coordinates with a comma; spaces after the comma are part of the note.
Blank field = delete all old notes

output filename

Defines the name of the plot file to be filename, which must be composed of 64 or fewer characters. Every time **output** is issued, the currently opened plot file is closed and a new one opened ready to receive further plots. Suppose you make a mistake; if you enter the command **output** without a filename the current plotfile is erased. Subsequent output is sent to the current file which has been re-initialized.
Default = *mypost*
Blank command field: restart plot in current file

palette n h s b

Change the color designated n to a new one with hue h, saturation s, and brightness b. For n in the range 1 to 11, existing colors will be changed (but not renamed!); for n between 11 and 20, you will create a new color with the corresponding number n, but without a name. The command may be repeated with different n to load a selection of colors. **palette** sets the colors for the next graph to be drawn, not the next data series, so that resetting the same color several times will result in only one color being chosen, the last one set. This is a PostScript command only. The three HSB color coordinates (h, s, b) are defined in any book on PostScript: Roughly, h goes continuously around the color circle: from 0 (red), to 0.333 (green) to 0.667 (blue) back to 1 (red again); s sets the amount of white mixed in: it ranges from 0 (all white) to 1 (no white – full saturation); b is brightness: from 0 (black) to 1 (maximum brightness).

plot [x0 y0] [abs]

plot centered

Creates the next complete graph containing all the data series currently in memory. Unless a **save** command has been used, the plotted series are the ones read in since the last **plot** command or, if this is the first such command, all the series. A plot file is generated usually named *mypost*; you can set a different name with **output**.

If x0 and y0 are specified, the new graph is plotted with its origin at those coordinates in inches relative to the previous plot origin. The plot origin for these purposes is the place where the annotated axes cross, not the point (0, 0). The optional word *abs* means that x0 and y0 refer absolutely to the bottom left corner of the page. A graph can be conveniently *centered* on the (8.5 by 11 inch) page by employing the second form of the command.

If the first graph is plotted without the x0 y0 arguments, it is positioned at the bottom of the page; subsequent graphs without x0 and y0 are plotted above the earlier ones.

All graphics will be drawn on the same physical page. To get a second page you must make another file with output.

read [n]

Performs the reading of the file according to the specifications in force at this point; see **mode**. Each **read** instruction is performed with a single Fortran READ statement with an implied DO; this means many data may appear on a single line in ASCII files. With binary files only one binary record is read with every **read** command. The integer n is the number of points to be read from the file, but if n is absent the file is read to the end of file (eof). When the eof is not reached, the file remains open and ready for further reading beginning at the next unread record (that is, the next line in ASCII files); if the eof was reached, another read on this file will begin at the beginning. Usually, up to 100 separate data series may be present at any one time, but use **status** to confirm this. Note n must be explicit if **file** = *

save

Normally when new points are read after a **plot** command, the data values for the previous graph are erased. To prevent this, the command **save** must be entered before the next **read** statement; then the old and the new data are plotted together on the new graph. If no new data are to be read in, there is no need to use **save** since the earlier data are retained for plotting in this case.

smooth [*natural akima off*]

Decides whether continuous curves of y against x are interpolated with straight lines, **smooth** *off*, or natural cubic splines, *natural*, or Akima splines with *akima*. Akima splines are piecewise cubic curves with less overshoot than natural splines.

When splines are used, the series is taken to be a single-valued function of x and the actual x values are re-ordered to be increasing by the program if necessary. Also see NOTES 2. This command is an input attribute, applying to subsequent **read** commands, not to the whole plot. **smooth** cancels a **symbol** command and vice versa. **smooth** *off* reverts to **symbol** mode if that was the previous style of plotting, with the same symbol number and height as before. Note the automatic plot limits use the original data series, not the smoothed values, so that sometimes pieces of a **smooth**ed curve may be lost off the top or bottom of a graph even when you have let the program find its own limits. The command field may be abbreviated to the first two letters.
Default = *off*
Blank command field = *natural*

skip [n]

Skips the next n records in the current data file. With ASCII files this means skipping n lines. The command examines the current read **format** to determine whether the current file is binary or ASCII. Skipping can also be performed by including slashes in a format specification but this is usually less convenient.
Blank command field n = 1

status

Lists a synopsis of the current data series (their lengths, extreme values and other attributes), the plot and reading parameters, and the number of words available for further data series. Also lists some program array limitations of the current version of *plotxy*.

stop

Closes the output file and brings program to an orderly halt. This must always be the last command of any run from a terminal, otherwise part of your plot will be lost. **Stop** may be omitted if *plotxy* is running from a script, since the end-of-file tells the program input is finished.

symbol n [s]
symbol name [s]

Defines the next input series to be a set of discrete points with symbols rather than a curve. The kind of symbol is defined by the integer n in the first style:

0 square	8 upward arrow	16 small circle
1 triangle	9 hourglass	17 circle
2 octagon	10 campstool	18 large circle
3 diamond	11 hexagon	19 small filled circle
4 plus	12 Y	20 small filled square
5 asterisk	13 vertical bar	21 small filled triangle
6 cross	14 star of David	
7 barred square	15 dot	

Some of the symbols can be referred to by name instead of the integer code: *asterisk, circle, cross, diamond, dot, hexagon, octagon, plus, square, star, triangle* and the solid symbols specified by prefixing *filled* or *solid* in front of *circle, square* and *triangle*.

The approximate height each of symbol is s inches. Small symbols (16, 19, 20 and 21) are about half the nominal height, while 18 is about twice the stated height. Negative s causes the symbol to be drawn upside down, useful for triangle and arrow. If the size s is omitted, the value from the previous **symbol** command is inherited. Size can be variable, set through the third column in **mode** input; see below.

To cause one of these symbols to be drawn in a text string (a **note** for example) just enclose the symbol number plus 2000 in backslashes, for example, \2019\.

To plot a series of a particular symbol with varying sizes, you must combine **symbol** with **mode** 3 or 30; in this mode three values are read for each input point. Then, provided that s in the **symbol** command is set greater than or equal to 10, the three numbers are treated as x, y, h, where h is the symbol height in inches.
Default n = −1, s = 0.15

title text

Specifies a title for the plot. This may be up to 115 characters in length. A blank command field cancels the previous title and leaves the next plot untitled. The character font of the title is assumed by all the lettering of the graph unless explicitly reset. If the title comprises a font-setting phrase, that sets the font for the rest of the graph and the graph is untitled.
Default text = blanks

xlabel text

Specifies a label to be written centered below the x axis. See **title** for other details.
Default text = blanks

xlimit xlength [x1 x2]
xlimit xlength [x1 x2 dx]

Defines the length of the x axis, xlength, in inches and the lower and upper limits of x: x1, x2. All plotted points lie inside (x1, x2); those outside are omitted from the plot. If x1 = x2 = 0, or if these values are omitted in the command, the x extremes are chosen to encompass the values in the data series. If x2 is less than x1 the data and axes are plotted reversed, that is, with x decreasing to the right, between the given limits. Reversed logarithmic axes are not permitted. This is a plot attribute, governing the behavior when **plot** is invoked. If the fourth argument, dx, is supplied, tick marks and numbers are written at integer multiples of dx, provided this results in a reasonable quantity of them; otherwise, the defaults are invoked. With log axes, dx has no effect except, when it is set negative, tick marks between powers of ten are suppressed.

The default behavior for plot size is somewhat complex: if the x and y variables cover similar intervals (within a factor of two), the default height and width are arranged to be in the proportions of the data, thus giving the same scales for the x and y variables. Otherwise a height of 6 inches and width of 5 is taken.
Default xlength = 5, x1 = x2 = 0

ylabel text

Same as **xlabel** but for the y axis.
Default text = blanks

ylimit ylength [y1 y2 [dy]]

Same as **xlimit** but for the y axis.
Default ylength = 6, y1 = y2 = 0

weight w [wlines]

On PostScript versions this command controls the weight of plotted lines by giving the integer w, the line thickness in one-thousandths of an inch. Optionally, lines

and points on the graph can be given a different (usually heavier) weight from text and other material; this is set in wlines. Invoking **weight** with only one parameter implicitly sets wlines=w for subsequent input data.
Default w = 6, wlines = w

DEFAULT VALUES

See NOTES 10 for the way to change these.

affine 1, 0, 1, 0	**output** *mypost*
character 0.11, 0	**smooth** *off*
dash 0, 0	**symbol** −1, 0
file *	**title** blanks
format *	**xlabel** blanks
frame *off*	**xlimit** 5, 0, 0
logxy 0	**ylabel** blanks
mode 2	**ylimit** 6, 0, 0
weight 6, 6	

LETTERING

Plotxy provides a variety of fonts in which the title, labels, notes may be written as well as the ability to include mathematical material and Greek letters. The names of the fonts are *simplex, complex, italic, duplex*; the default is *complex.* To get any of the others in a text string enclose the first three letters of the font name in backslashes (e.g. \ita\ or \dup\) ahead of the text. The font remains in force until explicitly changed. To obtain a uniform font throughout the graph and its labels include a font-setting phrase (e.g.\ita\) at the beginning of **title**. If you want to vary the fonts within one plot you can specify the desired font changes in the text; font changes may appear at any point in a piece of text. There is also a math font obtained by: \$\. In it letters are set in italics, while all other characters retain the previous font setting. The math state toggles to and fro, so that \$\x=y\$\ will be set in italics in a line set in some other font. Note: *plotxy* draws its own letters as a set of lines, and does not use PostScript fonts.

You may also get Greek letters by enclosing their names in backslashes, as \GAMMA\ or \lambda\; upper case Greek appears when the English name is upper case. The name of a Greek letter can be abbreviated to its fewest unambiguous leading letters: thus \s\ specifies sigma, but you need \ome\ for omega. The default theta and phi characters are the flowery script versions: to get the plain ones type: \rgth\ and \rgph\, for regular theta and phi. Superscripts are possible with the construct \sup{...}, so that x-squared is rendered x\sup{2}. Similarly with subscripts one writes, for example, g\sub{ij}. As mentioned in **symbol** you may plot a special graphics symbol by enclosing the symbol integer plus 2000 in backslashes. The code \bs\ suppresses the character advance so that characters may be superimposed. Characters may be embellished with a hat \ˆ\, or a tilde \˜\, or a an overbar \−\. The code follows the decorated letter: x\−\ for example.

There are some characters with no keyboard equivalent. Some of them can be specified by four-letter codes enclosed in backslashes: the normal rule is to use the first four letters of the name. Here are the special characters and their codes: gradient \grad\; integral \inte\; infinity \infi\; times symbol \time\; partial derivative \part\; degree symbol \degr\; square root \sqrt\.

There are other, perhaps more obscure, symbols that you can access only with numerical codes: a 4-digit key number enclosed in backslashes. Here is a table of the codes for these special symbols; every symbol in the graphics character set has such a code but only 1387 to 1431 are listed because all those without keyboard equivalents are contained in this list.

1387 partial d	1403 summation
1388 del	1404 regular theta
1389 member of	1405 {
1390 less or equal	1406 }
1391 greater or equal	1407 @
1392 proportional	1408 hat
1393 integral	1409 [
1394 circuit int	1410]
1395 infinity	1411 #
1396 + or –	1412 paragraph
1397 – or +	1413 dagger
1398 times	1425 tall <
1399 division	1426 tall >
1400 product	1429 degree
1401 times dot	1430 tends to
1402 radical	1431 regular phi

Finally, another use for the four-digit code is to specify text size. The height of the text following the phrase \0025\ is changed from its current value to 0.25 inches; any number less than 500 is interpreted as the letter height in hundredths of an inch, but remember there must be four digits between the backslashes.

NOTES

1. *Plotxy* is reasonably graceful with error conditions: explanatory messages are issued in most circumstances. The messages are usually in two parts: what went wrong and what the program has done about it. For example, if unintelligible data are encountered, the program will reject the whole series and then issue a warning. Attempting to plot negative data on a log scale does not cause a crash – an error message is printed and the offending scale is made linear instead of logarithmic.

2. When **smooth** is used, the data series is re-ordered to make x increasing; if consecutive x-values are then identical, this implies a discontinuity in the function, which cannot be smoothed. *Plotxy* issues a warning and plots the re-ordered series unsmoothed, that is, with points joined by straight-line segments.

3. If you have explicitly set plot extremes with **xlimit** and **ylimit** and a data value falls outside the window, when either of the splines has been selected, the program draws a piece of curve between the last captured point(s) and the edge of the graph in the direction of the invisible point. When **smooth** is *off* the virtual "pen" is simply lifted until onscale data are encountered. This allows you to insert breaks in your data records by inserting large values; but remember to set limits explicitly before **plot**.

4. To plot a line with symbols at the observations, just read the data file twice, once with an interpolating line (**smooth** or not, as desired) and then with **symbol**. If you want lines with both long and short dashes this too can be done by reading the data twice each time with different, but compatible dash sizes, so that when superposed the desired effect is achieved. This is possible because *plotxy* is careful about dashed lines, always starting at the beginning of a drawn section. Not every combination of long and short dashes can be built this way, although a surprising variety can. For example, if you use

```
dash  a b
read
dash  c d
```

```
  read
```

then one way to obtain a repeating pattern is to set c=(2∗a−(n−1)∗b)/(n+1) and d=(n∗b−a)/(n+1), where n is an integer such that a/b < n < 1+2∗a/b.

5. The size of the feet on the ends of an error bar can be controlled: it is the same as the symbol height, when symbols are also plotted at the point. A negative or zero-height symbol eliminates the foot; negative height symbols will be plotted, but upside down which is evident only in a few cases.

6. Artistic users seem to want to vary the font and size of every notation and label. This is straightforward: any piece of text may be preceded by a font phrase; the height may be defined in the text as described in the previous section or by the **character** command immediately before the text is entered (and similarly with **color**). The only writing that can not be specified in this way is the numerical annotation of the axes. To control its size and color set these parameters immediately before **plot**; to arrange a special font put the font phrase you desire *at the end of* the title text. If you want the two axes in different colors with different numeral sizes and fonts, this can be done too, but discovering how is left as an exercise. (Hint: use frame and plot 0 0)

7. Reversing x and y axes (as when coordinates are specified as latitude and longitude) is easily done with **mode** 20:

```
  mode 20 2 1
  read
```

Modes 10, 20, 30 are designed to read from tables in which the columns contain numbers only, not other kinds of material, such as code words. If you want to extract values from a table some of whose columns are not numbers, those columns can be skipped if you use an explicit **format** statement.

8. Command lines can be commented by appending the % (percent) symbol; see examples. In any of the input modes that treat the data file as a table (modes 10 20 30 −30), lines beginning with % will be treated as comments and skipped without causing an error. Comment skipping can be cancelled: see **nocomment**.

9. *Plotxy* is written entirely with single-precision (real*4) variables. If data are to be plotted in which the only variations occur in the seventh (or later) significant figure, there will be inaccuracies. The program can be recompiled in double precision, with a compiler flag −r8 under Sun compilers, or by inserting an implicit declaration in each subroutine and function. Alternatively, a constant value can be subtracted from the data values before plotting to map them into a more manageable interval.

10. The preset defaults may not be to the user's taste. Finding and replacing them in the code would be a tedious undertaking. Instead, the last subroutine, 'dfault', provides a facility whereby a number of commands are executed internally by *plotxy* before the program reads from the command line. Preferred defaults, such as font, letter height, framin, etc can be set there, and will over-ride the defaults given in the documentation.

EXAMPLES

Here are two examples of quite presentable plots made with relatively little effort. The first is a reproduction of some graphs of Bessel functions found in Abramowitz and Stegun's Handbook of Mathematical functions. A rather sparse table has been entered equally spaced directly into the input file and the values are spline smoothed to add authority. All modern operating systems allow you to prepare an input file (a script in Unix terminology) and submit it to a program as if it were entered interactively; it is very handy then to put the data in the file together with the plot commands. Notice comments can been inserted by beginning a command line with a blank or a %; notice trailing comments are allowed too.

```
     A diagram from Chap 9 of Handbook of Mathematical Functions
smooth    %  Cubic spline interpolation
affine 1 –1 1 0
mode 1
     First 16 values of J0 for x=0, 1, 2, .. 15
read 16
1.000  .762  .224 –.260 –.397 –.178  .151  .300  .172 –.090
–.246 –.171  .048  .207  .171 –.014
     Now 16 values of J1
read 16
0.000  .440  .577  .339 –.066 –.328 –.277 –.005  .235  .245
 .043 –.177 –.233 –.070  .133  .205
     Next 15 values of Y0 for x=1, 2, .. 15  dashed
affine 1 0 1 0
dash .05 .07
read 15
 .088  .510  .377 –.017 –.309 –.288 –.026  .224  .250  .056
–.169 –.225 –.078  .127  .205
     Finally values of Y1
read 15
–.781 –.107  .325  .398  .148 –.175 –.303 –.158  .104  .249
 .164 –.057 –.210 –.167  .021

xlim 3.5 0 16
ylim 3 0 0

title \du\
xlab \co\FIGURE 9.1: \$|\J\sub{0}(x), Y\sub{0}(x),
note (1, 0.8)J\sub{0}
note (1.8, –0.5)Y\sub{1}
note (2, 0.577, 3, 0.8)J\sub{1}
note (3, 0.377, 5, 0.6)Y\sub{0}

plot 1 6.6
```

In the above listing the **xlab** command line has been truncated to fit on the page. Several different fonts have been used; notice the axis numerals are in the default front, complex. The picture has been placed at the top of the page leaving room for the second example below it.

Next we illustrate how a table may be rescanned to pick out different columns for various purposes. The data file *rhodata* contains a table of Wenner array apparent resistivity data at various electrode spacings, together with an estimated uncertainty (column 3) and the fit of a one-dimensional theoretical model (column 4). The file is read in **mode** 30 to get error bars, then reread and smoothed to put the theory on the graph. Log axes are appropriate here for obvious reasons.

```
title \it\
logxy 3
frame

xlim 4 1.0 1000
ylim 3.6 10 2000
ylabel Apparent resistivity  \rho\\sub{a} (\OME\m)
xlabel Electrode separation  r  (meters)

     1st read uses the format to pick measurements and
     errors in mode 30
```

```
  file rhodata
  symbol 19 0.1
  mode 30 1 2 3
  read
      2nd read picks up 1st and 4th column with mode 20
  mode 20 1 4
  dash 0 0
  smooth
  read

  notes
  plot 0 –5.5
  stop
```

Here is the data file *rhodata*

```
          1.52    69.6    10.0    67.9
          4.57    126.3   21.1    126.3
          7.62    207.6   33.6    199.6
          12.1    304.3   47.0    304.2
          18.2    421.2   40.2    431.0
          24.3    508.7   69.2    541.8
          30.4    587.6   52.0    636.4
          42.6    769.8   83.0    778.2
          79.2    987.7   122.0   913.2
          106.7   902.3   150.1   848.7
          137.2   691.2   114.1   715.2
          167.6   543.3   86.16   571.0
          228.6   366.7   101.2   333.1
```

LOCAL IMPLEMENTATION

On the various Unix networks at IGPP, UCSD, *plotxy* is an executable command made available through one of the standard paths like /usr/local/bin. The normal version is the PostScript model, which normally creates the file *mypost* which may be sent to a printer, or to the screen via the utility *pageview*, which opens a window and plots the contents of the file in it.

POSTSCRIPT ANNEX

Most installations of *plotxy* produce PostScript output, although the program has been designed to be independent of any particular graphics implementation. *Plotxy* running in PostScript mode prints a message informing you of the PostScript output; then the default output file is named *mypost*. Because PostScript files are ASCII, you may edit them to produce special effects. To help you, *plotxy* inserts comment lines to indicate which piece of the graph is currently being plotted. Special commands that only work in the PostScript mode are **weight** and **fill**. **weight** command allows different line thickness to be specified.

To fill a region with a solid color, use the **fill** command. Series are drawn in the order they are read, and notes are last. This command works with **smooth**, but not perhaps as one might like, because only y as a function of x is smooth, and the region is completed by joining the first and last points. To draw a filled rounded area (like an ellipse), one should define the boundary densely and turn **smooth** off. Dashed curves will not fill properly.

Because of PostScript limitations, bounding curves for **fill** should comprise fewer than 1001 points.

NEW STUFF

Graphs conveniently centered with 'plot centered'. (11/04)

Alternative default parameters can be conveniently set at compile time: see NOTES 10. (8/04)

weight takes a second parameter, setting the line weight for data in the graph. (5/04)

note allows filled arrow heads; vertical, left justified text with 'note (v)'. (3/04)

New option in **logxy** retains true proportions of original values in graphs: 'logxy equilin'. (6/02)

\^\ puts a hat on the previous character. \~\ puts a tilde on the previous charcater \– \ draws an bar over the previous character. (5/01)

frame +xeven causes tick marks to be of uniform height on the x axis, and similarly for y. (10/00)

note (+) appends further material to a note. (8/00)

color accepts 11-th color, lightgray. Useful for filling areas, but usually too light for lines. (4/00)

plot xo yo abs positions the graph absolutely on the page. (12/99)

background sets colored backgrounds for graphs. (6/99)

Size of feet on error bars can be controlled through symbol height, or the feet eliminated. (6/99) See Note 5.

When tick-mark overcrowding arises, plotxy numerically tags every n-th one, where n is a multiple of 1, 2, or 5 (x 10**n), so that if the tick interval is a simple decimal, the numerical notation will seem rational. Of course, nothing can help those who choose unnatural tick spacing. (11/98)

On axes where the tick marks are too crowded for every one to be identified numerically, the unlucky ones are 62% of the height of the marked ones. (11/98)

frame specifies the surrounding box through the parameters +box –box, instead of with the confusing on/off construction. (5/98)

note has an additional parameter: r for right justification, c for centered notes. (2/98)

Commands can be commented: everything following % is ignored. Data tables (read in mode 10 20 30 –30) may contain comment lines that will be skipped – begin line with %. (6/97)

palette allows the customizing of colors. (6/97)

Most special characters can be plotted with 4-letter codes which are easy to remember – see LETTERING.

frame *grid-solid* allows solid gray dashed grid lines, not dashed ones.

Regular theta and phi can be plotted via \rgth\ and \rgph\.

xlim and **ylim** accept your recommendations on the interval between tick marks.

smooth can generate Akima splines. (4/96)

landscape rotates the figure through 90 degrees. (12/94)

weight allows you the thicken lines in the figure or annotations. (12/93)

fill implements the creation of regions of solid color. (6/92)

symbol accepts most symbol names in place of cryptic numerals. (6/92)

logxy accepts names in place of codes. (6/92)

color accepts color names instead of cryptic numerals.

INDEX