GNUPLOT Quick Reference

(Copyright(c) Alex Woo 1992 June 1)

Starting GNUPLOT

to enter GNUPLOT	`gnuplot`
to enter batch GNUPLOT	`gnuplot macro_file`
to pipe commands to GNUPLOT	`application | gnuplot`

see below for environment variables you might want to change before entering GNUPLOT.

Exiting GNUPLOT

exit GNUPLOT	`quit`

All GNUPLOT commands can be abbreviated to the first few unique letters, usually three characters. This reference uses the complete name for clarity.

Getting Help

introductory help	`help plot`
help on a topic	`help <topic>`
list of all help available	`help or ?`
show current environment	`show all`

Command-line Editing

The UNIX, MS-DOS and VMS versions of GNUPLOT support command-line editing and a command history. EMACS style editing is supported.

Line Editing:

move back a single character	`^ B`
move forward a single character	`^ F`
moves to the beginning of the line	`^ A`
moves to the end of the line	`^ E`
delete the previous character	`^ H and DEL`
deletes the current character	`^ D`
deletes to the end of line	`^ K`
redraws line in case it gets trashed	`^ L,^ R`
deletes the entire line	`^ U`
deletes the last word	`^ W`

History:

moves back through history	`^ P`
moves forward through history	`^ N`

The following arrow keys may be used on the MS-DOS version if READLINE is used.

IBM PC Arrow Keys:

Left Arrow	`same as ^ B`
Right Arrow	`same as ^ F`
Ctrl Left Arrow	`same as ^ A`
Ctrl Right Arrow	`same as ^ E`
Up Arrow	`same as ^ P`
Down Arrow	`same as ^ N`

Graphics Devices

All screen graphics devices are specified by names and options. This information can be read from a startup file (.gnuplot in UNIX). If you change the graphics device, you must replot with the **`replot`** command.

get a list of valid devices	`set terminal [options]`

Graphics Terminals:

AED 512 Terminal	`set term aed512`
AED 767 Terminal	`set term aed767`
Amiga	`set term amiga`
Adobe Illustrator 3.0 Format	`set term aifm`
Apollo graphics primitive, rescalable	`set term apollo`
Atari ST	`set term atari`
BBN Bitgraph Terminal	`set term bitgraph`
SCO CGI Driver	`set term cgi`
Apollo graphics primitive, fixed window	`set term gpr`
SGI GL window	`set term iris4d [8 24]`
MS-DOS Kermit Tek4010 term - color	`set term kc_tek40xx`
MS-DOS Kermit Tek4010 term - mono	`set term km_tek40xx`
NeXTstep window system	`set term next`
REGIS graphics language	`set term regis`
Selanar Tek Terminal	`set term selanar`
SunView window system	`set term sun`
Tektronix 4106, 4107, 4109 & 420X	`set term tek40D10x`
Tektronix 4010; most TEK emulators	`set term tek40xx`
VAX UIS window system	`set term VMS`
VT-like tek40xx terminal emulator	`set term vttek`
UNIX plotting (not always supplied)	`set term unixplot`
AT&T 3b1 or 7300 UNIXPC	`set term unixpc`
X11 default display device	`set term x11`
X11 multicolor point default device	`set term X11`

Turbo C PC Graphics Modes:

Hercules	`set term hercules`
Color Graphics Adaptor	`set term cga`
Monochrome CGA	`set term mcga`
Extended Graphics Adaptor	`set term ega`
VGA	`set term vga`
Monochrome VGA	`set term vgamono`
Super VGA - requires SVGA driver	`set term svga`
AT&T 6300 Micro	`set term att`

MS Windows 3.x and OS/2 Presentation Manager are also supported.

Hardcopy Devices:

Unknown - not a plotting device	`set term unknown`
Dump ASCII table of X Y [Z] values	`set term table`
printer or glass dumb terminal	`set term dumb`
Roland DXY800A plotter	`set term dxy800a`

Dot Matrix Printers

Epson-style 60-dot per inch printers	`set term epson_60dpi`
Epson LX-800, Star NL-10	`set term epson_lx800`
NX-1000, PROPRINTER	`set term epson_lx800`
NEC printer CP6, Epson LQ-800	`set term nec_cp6 [monochrome color draft]`
Star Color Printer	`set term starc`
Tandy DMP-130 60-dot per inch	`set term tandy_60dpi`
Vectrix 384 & Tandy color printer	`set term vx384`

Laser Printers

Talaris EXCL language	`set term excl`
Imagen laser printer	`set term imagen`
LN03-Plus in EGM mode	`set term ln03`
PostScript graphics language	`set term post [mode color 'font' size]`
CorelDraw EPS	`set term corel [mode color 'font' size]`
Prescribe - for the Kyocera Laser Printer	`set term prescribe`
Kyocera Laser Printer with Courier font	`set term kyo`
QMS/QUIC Laser (also Talaris 1200)	`set term qms`

Metafiles

AutoCAD DXF (120x80 default)	`set term dxf`
FIG graphics language: SunView or X	`set term fig`
FIG graphics language: Large Graph	`set term bfig`
SCO hardcopy CGI	`set term hcgi`
Frame Maker MIF 3.0	`set term mif [pentype curvetype help]`
Portable bitmap	`set term pbm [fontsize color]`
Uniplex Redwood Graphics Interface Protocol	`set term rgip`
TGIF language	`set term tgif`

HP Devices

HP2623A and maybe others	`set term hp2623A`
HP2648 and HP2647	`set term hp2648`
HP7580, & probably other HPs (4 pens)	`set term hp7580B`
HP7475 & lots of others (6 pens)	`set term hpgl`
HP Laserjet series II & clones	`set term hpljii [75 100 150 300]`
HP DeskJet 500	`set term hpdj [75 100 150 300]`
HP PaintJet & HP3630	`set term hppj [FNT5X9 FNT9X17 FNT13x25]`
HP laserjet III (HPGL plot vectors)	`set term pcl5 [mode font fontsize ]`

TeX picture environments

LaTeX picture environment	`set term latex`
EEPIC – extended LaTeX picture	`set term eepic`
LaTeX picture with emTeX specials	`set term emtex`
PSTricks macros for TeX or LaTeX	`set term pstricks`
TPIC specials for TeX or LaTeX	`set term tpic`
MetaFont font generation input	`set term mf`

Files

plot a data file	`plot 'fspec'`
load in a macro file	`load 'fspec'`
save command buffer to a macro file	`save 'fspec'`
save settings for later reuse	`save set 'fpec'`

PLOT & SPLOT commands

plot and **splot** are the primary commands **plot** is used to plot 2-d functions and data, while **splot** plots 3-d surfaces and data.

Syntax:

plot {ranges} <function> {title}{style} {, <function> {title}{style}...}

splot {ranges} <function> {title}{style} {, <function> {title}{style}...}

where <function> is either a mathematical expression, the name of a data file enclosed in quotes, or a pair (**plot**) or triple (**splot**) of mathematical expressions in the case of parametric functions. User-defined functions and variables may also be defined here. Examples will be given below.

Plotting Data

Discrete data contained in a file can displayed by specifying the name of the data file (enclosed in quotes) on the **plot** or **splot** command line. Data files should contain one data point per line. Lines beginning with # (or ! on VMS) will be treated as comments and ignored. For **plot**s, each data point represents an (x,y) pair. For **splot**s, each point is an (x,y,z) triple. For **plot**s with error bars (see **plot errorbars**), each data point is either (x,y,ydelta), (x,y,ylow,yhigh), (x,y,xlow,xhigh), (x,y,xdelta,ydelta), or (x,y,xlow,xhigh,ylow,yhigh). In all cases, the numbers on each line of a data file must be separated by blank space. This blank space divides each line into columns.

For **plot**s the x value may be omitted, and for **splot**s the x and y values may be omitted. In either case the omitted values are assigned the current coordinate number. Coordinate numbers start at 0 and are incremented for each data point read.

Surface Plotting

Implicitly, there are two types of 3-d datafiles. If all the isolines are of the same length, the data is assumed to be a grid data, i.e., the data has a grid topology. Cross isolines in the other parametric direction (the ith cross isoline passes thru the ith point of all the provided isolines) will also be drawn for grid data. (Note contouring is available for grid data only.) If all the isolines are not of the same length, no cross isolines will be drawn and contouring that data is impossible.

For splot if 3-d datafile and using format (see **splot datafile using**) specify only z (height field), a non parametric mode must be specified. If, on the other hand, x, y, and z are all specified, a parametric mode should be selected (see **set parametric**) since data is defining a parametric surface.

example of plotting a 3-d data	`set parametric;splot 'glass.dat'`
example of plotting explicit	`set noparametric;splot 'datafile.dat'`

Using Pipes

On some computer systems with a popen function (UNIX), the datafile can be piped through a shell command by starting the file name with a '<'. For example:

pop(x) = 103*exp(x/10) plot "< awk '{ print $1-1965 $2 }' population.dat", pop(x)

would plot the same information as the first population example but with years since 1965 as the x axis.

Similarly, output can be piped to another application, e.g.

set out "|lpr -Pmy_laser_printer"

Plot Data Using

The format of data within a file can be selected with the **using** option. An explicit scanf string can be used, or simpler column choices can be made.

```
plot "datafile"      { using {<ycol> |
                     <xcol>:<ycol> |
                     <xcol>:<ycol>:<ydelta> |
                     <xcol>:<ycol>:<width> |
                     <xcol>:<ycol>:<xdelta> |
                     <xcol>:<ycol>:<ylo>:<yhi> |
                     <xcol>:<ycol>:<xlo>:<xhi> |
                     <xcol>:<ycol>:<xdelta>:<ydelta> |
                     <xcol>:<ycol>:<ydelta>:<width> |
                     <xcol>:<ycol>:<ylo>:<yhi>:<width> |
                     <xc>:<yc>:<xlo>:<xhi>:<ylo>:<yhi>}
                     {"<scanf string>"}}...
splot "datafile"     { using {<xcol>:<ycol>:<zcol>}
                     {" <scanf string> "}}...
```

<xcol>, <ycol>, and <zcol> explicitly select the columns to plot from a space or tab separated multicolumn data file. If only <ycol> is selected for **plot**, <xcol> defaults to 1. If only <zcol> is selected for **splot**, then only that column is read from the file. An <xcol> of 0 forces <ycol> to be plotted versus its coordinate number. <xcol>, <ycol>, and <zcol> can be entered as constants or expressions.

If errorbars (see also **plot errorbars**) are used for **plot**s, xdelta or ydelta (for example, a +/- error) should be provided as the third column, or (x,y)low and (x,y)high as third and fourth columns. These columns must follow the x and y columns. If errorbars in both directions are wanted then xdelta and ydelta should be in the third and fourth columns, respectively, or xlow, xhigh, ylow, yhigh should be in the third, fourth, fifth, and sixth columns, respectively.

Scanf strings override any <xcol>:<ycol>(:<zcol>) choices, except for ordering of input, e.g.,

```
plot "datafile"      using 2:1 "%f%*f%f"
```

causes the first column to be y and the third column to be x.

If the scanf string is omitted, the default is generated based on the <xcol>:<ycol>(:<zcol>) choices. If the **using** option is omitted, "%f%f" is used for **plot** ("%f%f%f%f" or "%f%f%f%f%f%f" for **errorbar plot**s) and "%f%f%f" is used for **splot**.

```
plot "MyData"        using "%*f%f%*20[^\n]%f" w lines
```

Data are read from the file "MyData" using the format "%*f%f%*20[^\n]%f". The meaning of this format is: "%*f" ignore the first number, "%f" then read in the second and assign to x, "%*20[^\n]" then ignore 20 non-newline characters, "%f" then read in the y value.

Plot With Errorbars

Error bars are supported for 2-d data file plots by reading one to four additional columns specifying ydelta, ylow and yhigh, xdelta, xlow and xhigh, xdelta and ydelta, or xlow, xhigh, ylow, and yhigh respectively. No support exists for error bars for **splot**s.

In the default situation, GNUPLOT expects to see three to six numbers on each line of the data file, either (x, y, ydelta), (x, y, ylow, yhigh), (x, y, xdelta), (x, y, xlow, xhigh), (x, y, xdelta, ydelta), or (x, y, xlow, xhigh, ylow, yhigh). The x coordinate must be specified. The order of the numbers must be exactly as given above. Data files in this format can easily be plotted with error bars:

plot "data.dat" with errorbars (or yerrorbars)

plot "data.dat" with xerrorbars

plot "data.dat" with xyerrorbars

The error bar is a line plotted from (x, ylow) to (x, yhigh) or (xlow, y) to (xhigh, y). If ydelta is specified instead of ylow and yhigh, ylow=y-ydelta and yhigh=y+ydelta are derived. The values for xlow and xhigh are derived similarly from xdelta. If there are only two numbers on the line, yhigh and ylow are both set to y and xhigh and xlow are both set to x. To get lines plotted between the data points, **plot** the data file twice, once with errorbars and once with lines.

If x or y autoscaling is on, the x or y range will be adjusted to fit the error bars.

Boxes may be drawn with y error bars using the **boxerrorbars** style. The width of the box may be either set with the "set boxwidth" command, given in one of the data columns, or calculated automatically so each box touches the adjacent boxes. Boxes may be drawn instead of the cross drawn for the **xyerrorbars** style by using the **boxxyerrorbars** style.

x,y,ylow & yhigh from columns 1,2,3,4	`plot "data.dat" us 1:2:3:4 w errorbars`
x from third, y from second, xdelta from 6	`plot "data.dat" using 3:2:6 w xerrorbars`
x,y,xdelta & ydelta from columns 1,2,3,4	`plot "data.dat" us 1:2:3:4 w xyerrorbars`

Plot Ranges

The optional range specifies the region of the plot that will be displayed.

Ranges may be provided on the **plot** and **splot** command line and affect only that plot, or in the **set xrange**, **set yrange**, etc., commands, to change the default ranges for future plots.

```
[{<dummy-var>=}{<xmin>:<xmax>}]      { [{<ymin>:<ymax>}] }
```

where <dummy-var> is the independent variable (the defaults are x and y, but this may be changed with **set dummy**) and the min and max terms can be constant expressions.

Both the min and max terms are optional. The ':' is also optional if neither a min nor a max term is specified. This allows '[ ]' to be used as a null range specification.

Specifying a range in the **plot** command line turns autoscaling for that axis off for that plot. Using one of the **set** range commands turns autoscaling off for that axis for future plots, unless changed later. (See **set autoscale**).

This uses the current ranges	`plot cos(x)`
This sets the x range only	`plot [-10:30] sin(pi*x)/(pi*x)`
This sets both the x and y ranges	`plot [-pi:pi] [-3:3] tan(x), 1/x`
sets only y range, & turns off autoscaling on both axes	`plot [ ] [-2:sin(5)*-8] sin(x)**besj0(x)`
This sets xmax and ymin only	`plot [:200] [-pi:]  exp(sin(x))`
This sets the x, y, and z ranges	`splot [0:3] [1:4] [-1:1] x*y`

Plot With Style

Plots may be displayed in one of twelve styles: **lines**, **points**, **linespoints**, **impulses**, **dots**, **steps**, **errorbars** (or **yerrorbars**), **xerrorbars**, **xyerrorbars**, **boxes**, **boxerrorbars**, or **boxxyerrorbars**. The **lines** style connects adjacent points with lines. The **points** style displays a small symbol at each point. The **linespoints** style does both **lines** and **points**. The **impulses** style displays a vertical line from the x axis (or from the grid base for **splot**) to each point. The **dots** style plots a tiny dot at each point; this is useful for scatter plots with many points. The **steps** style is used for drawing stairstep-like functions. The **boxes** style may be used for barcharts.

The **errorbars** style is only relevant to 2-d data file plotting. It is treated like **points** for **splot**s and function **plot**s. For data **plot**s, **errorbars** is like **points**, except that a vertical error bar is also drawn: for each point (x,y), a line is drawn from (x,ylow) to (x,yhigh). A tic mark is placed at the ends of the error bar. The ylow and yhigh values are read from the data file's columns, as specified with the **using** option to plot. The **xerrorbars** style is similar except that it draws a horizontal error bar from xlow to xhigh. The **xyerrorbars** or **boxxyerrorbars** style is used for data with errors in both x and y. A barchart style may be used in conjunction with y error bars through the use of **boxerrorbars**. The See **plot errorbars** for more information.

Default styles are chosen with the **set function style** and **set data style** commands.

By default, each function and data file will use a different line type and point type, up to the maximum number of available types. All terminal drivers support at least six different point types, and re-use them, in order, if more than six are required. The LaTeX driver supplies an additional six point types (all variants of a circle), and thus will only repeat after twelve curves are plotted with points.

If desired, the style and (optionally) the line type and point type used for a curve can be specified.

with <style>	`{<linetype> {<pointtype>}}`

where <style> is either **lines**, **points**, **linespoints**, **impulses**, **dots**, **steps**, **errorbars** (or **yerrorbars**), **xerrorbars**, **xyerrorbars**, **boxes**, **boxerrorbars**, **boxxyerrorbars**.

The <linetype> & <pointtype> are positive integer constants or expressions and specify the line type and point type to be used for the plot. Line type 1 is the first line type used by default, line type 2 is the second line type used by default, etc.

plots sin(x) with impulses	`plot sin(x) with impulses`
plots x*y with points, x**2 + y**2 default	`splot x*y w points, x**2 + y**2`
plots tan(x) with default function style	`plot [ ] [-2:5] tan(x)`
plots "data.1" with lines	`plot "data.1" with l`
plots "leastsq.dat" with impulses	`plot 'leastsq.dat' w i`
plots "exper.dat" with errorbars & lines connecting points	`plot 'exper.dat' w l, 'exper.dat' w err`

Here 'exper.dat' should have three or four data columns.

plots x**2 + y**2 and x**2 - y**2 with the same line type	`splot x**2 + y**2 w l 1, x**2 - y**2 w l 1`
plots sin(x) and cos(x) with linespoints, using the same line type but different point types	`plot sin(x) w linesp 1 3, \ cos(x) w linesp 1 4`
plots file "data" with points style 3	`plot "data" with points 1 3`

Note that the line style must be specified when specifying the point style, even when it is irrelevant. Here the line style is 1 and the point style is 3, and the line style is irrelevant.

See **set style** to change the default styles.

Plot Title

A title of each plot appears in the key. By default the title is the function or file name as it appears on the plot command line. The title can be changed by using the **title** option. This option should precede any **with** option.

title "<title>"

where <title> is the new title of the plot and must be enclosed in quotes. The quotes will not be shown in the key.

plots y=x with the title 'x'	`plot x`
plots the "glass.dat" file with the title 'revolution surface'	`splot "glass.dat" tit 'revolution surface'`
plots x squared with title "x^2" and "data.1" with title 'measured data'	`plot x**2 t "x^2", \ "data.1" t 'measured data'`

Set-Show Commands

all commands below begin with set	set
set mapping of polar angles	angles [degrees|radians]
arrows from point to	arrow [<tag>][from <sx>,<sy>,<sz>] [to <ex>,<ey>,<ez>][nohead]
force autoscaling of an axis	autoscale [<axes>]
enter/exit parametric mode	[no]parametric
display border	[no]border
clip points/line near boundaries	[no]clip <clip-type>
specify parameters for contour plots	cntrparam [spline][points][order][levels]
enable splot contour plots	[no]contour [base|surface|both]
default plotting style for data	data style <style-choice>
specify dummy variable	dummy <dummy1>,<dummy2>...
tic-mark label format specification	format [<axes>]["format-string"]
function plotting style	function style <style-choice>
draw a grid at major tick marks & minor tics (optional)	[no]grid [mxgrid OR mygrid]
enables hiddenline removal	[no]hidden3d
specify number of isolines	isosamples <expression>
enables key of curves in plot	key <x>,<y>,<z>
logscaling of an axes (optionally giving base)	logscale <axes> [<base>]
mapping 3D coordinates	mapping [cartesian|spherical|cylindrical]
offsets from center of graph	offsets <left>,<right>,<top>,<bottom>
mapping 2D coordinates	[no]polar
set radial range	rrange [<rmin>:<rmax>]
set sampling rate of functions	samples <expression>
set scaling factors of plot	size <xsize>,<ysize>
control display of isolines of surface	[no]surface
control graphics device	terminal <device>
change direction of tics	tics <direction>
adjust relative height of vertical axis	ticslevel <level>
adjust size of tick marks	ticscale [<size>]
turn on time/date stamp	[no]time
set centered plot title	title "title-text" <xoff>,<yoff>
set parametric range	trange [<tmin>:<tmax>]
set surface parametric ranges	urange or vrange
sets the view point for **splot**	view <rot_x>,<rot_z>,<scale>,<scale_z>
sets x-axis label	xlabel "<label>" <xoff>,<yoff>
set horizontal range	xrange [<xmin>:<xmax>]
change horizontal tics	xtics <start>,<incr>,<end>, "<label>" <pos>
adjust number of minor tick marks	[no]mxtics OR [no]mytics [<freq>]
draw x-axis	[no]xzeroaxis
sets y-axis label	ylabel "<label>" <xoff>,<yoff>
set vertical range	yrange [<ymin>:<ymax>]
change vertical tics	ytics <start>,<incr>,<end>, "<label>" <pos>
draw y-axis	[no]yzeroaxis
set default threshold for values near 0	zero <expression>
draw axes	[no]zeroaxis
sets z-axis label	zlabel "<label>" <xoff>,<yoff>
set vertical range	zrange [<zmin>:<zmax>]
change vertical tics	ztics <start>,<incr>,<end>, "<label>" <pos>
draw z-axis	[no]zzeroaxis

Contour Plots

Enable contour drawing for surfaces. This option is available for **splot** only.

Syntax: set contour { base | surface | both } set nocontour

If no option is provided to **set contour**, the default is **base**. The three options specify where to draw the contours: **base** draws the contours on the grid base where the x/ytics are placed, **surface** draws the contours on the surfaces themselves, and **both** draws the contours on both the base and the surface.

See also **set cntrparam** for the parameters that affect the drawing of contours.

Contour Parameters

Sets the different parameters for the contouring plot (see also **contour**).

```
set cntrparam   {{ linear | cubicspline | bspline }|
                points <n> |
                order <n> |
                levels { [ auto ] <n> |
                discrete <z1> <z2> ... |
                incr <start> <increment> [ <n> ] }}
```

5 automatic levels	set cntrparam levels auto 5
3 discrete levels at 10%, 37% and 90%	set cntrp levels discrete .1 1/exp(1) .9
5 incremental levels at 0, .1, .2, .3 and .4	set cntrparam levels incremental 0 .1 5
sets n = 10 retaining current setting of auto, incr., or discr.	set cntrparam levels 10
set start = 100 and increment = 50, retaining old n	set cntrparam levels incremental 100 50

This command controls the way contours are plotted. <n> should be an integral constant expression and <z1>, <z2> any constant expressions. The parameters are:

linear, **cubicspline**, **bspline** - Controls type of approximation or interpolation. If **linear**, then the contours are drawn piecewise linear, as extracted from the surface directly. If **cubicspline**, then piecewise linear contours are interpolated to form a somewhat smoother contours, but which may undulate. The third option is the uniform **bspline**, which only approximates the piecewise linear data but is guaranteed to be smoother.

points - Eventually all drawings are done with piecewise linear strokes. This number controls the number of points used to approximate a curve. Relevant for **cubicspline** and **bspline** modes only.

order - Order of the bspline approximation to be used. The bigger this order is, the smoother the resulting contour. (Of course, higher order bspline curves will move further away from the original piecewise linear data.) This option is relevant for **bspline** mode only. Allowed values are integers in the range from 2 (linear) to 10.

levels - Number of contour levels, 'n'. Selection of the levels is controlled by 'auto' (default), 'discrete', and 'incremental'. For 'auto', if the surface is bounded by zmin and zmax then contours will be generated from zmin+dz to zmax-dz in steps of size dz, where dz = (zmax - zmin) / (levels + 1). For 'discrete', contours will be generated at z = z1, z2 ... as specified. The number of discrete levels is limited to MAX_DISCRETE_LEVELS, defined in plot.h to be 30. If 'incremental', contours are generated at <n> values of z beginning at <start> and increasing by <increment>.

Specifying Labels

Arbitrary labels can be placed on the plot using the **set label** command. If the z coordinate is given on a **plot** it is ignored; if it is missing on a **splot** it is assumed to be 0.

set label {<tag>}{" <label text> "}	{at <x>,<y> {,<z>}} {<justification>}
set nolabel {<tag>}	
show label	

The text defaults to "", and the position to 0,0,0. The <x>, <y>, and <z> values are in the graph's coordinate system. The tag is an integer that is used to identify the label. If no <tag> is given, the lowest unused tag value is assigned automatically. The tag can be used to delete or change a specific label. To change any attribute of an existing label, use the **set label** command with the appropriate tag, and specify the parts of the label to be changed.

By default, the text is placed flush left against the point x,y,z. To adjust the way the label is positioned with respect to the point x,y,z, add the parameter <justification>, which may be **left**, **right** or **center**, indicating that the point is to be at the left, right or center of the text. Labels outside the plotted boundaries are permitted but may interfere with axes labels or other text.

label at (1,2) to "y=x"	`set label "y=x" at 1,2`
label "y=x^2" w right of the text at (2,3,4), & tag the label number 3	`set label 3 "y=x^2" at 2,3,4 right`
change preceding label to center justification	`set label 3 center`
delete label number 2	`set nolabel 2`
delete all labels	`set nolabel`
show all labels (in tag order)	`show label`

(The EEPIC, Imagen, LaTeX, and TPIC drivers allow \\ in a string to specify a newline.)

Miscellaneous Commands

For further information on these commands, print out a copy of the GNUPLOT manual.

change working directory	`cd`
erase current screen or device	`clear`
exit GNUPLOT	`exit or quit or EOF`
display text and wait	`pause <time> ["<string>"]`
print the value of <expression>	`print <expression>`
print working directory	`pwd`
repeat last **plot** or **splot**	`replot`
spawn an interactive shell	`! (UNIX) or $ (VMS)`

Environment Variables

A number of shell environment variables are understood by GNUPLOT. None of these are required, but may be useful.

If GNUTERM is defined, it is used as the name of the terminal type to be used. This overrides any terminal type sensed by GNUPLOT on start up, but is itself overridden by the .gnuplot (or equivalent) start-up file (see **start-up**), and of course by later explicit changes.

On Unix, AmigaOS, and MS-DOS, GNUHELP may be defined to be the pathname of the HELP file (gnuplot.gih).

On VMS, the symbol GNUPLOT$HELP should be defined as the name of the help library for GNUPLOT.

On Unix, HOME is used as the name of a directory to search for a .gnuplot file if none is found in the current directory. On AmigaOS and MS-DOS, GNUPLOT is used. On VMS, SYS$LOGIN: is used. See help start-up.

On Unix, PAGER is used as an output filter for help messages.

On Unix and AmigaOS, SHELL is used for the **shell** command. On MS-DOS, COMSPEC is used for the **shell** command.

On AmigaOS, GNUFONT is used for the screen font. For example: "setenv GNUFONT sapphire/14".

On MS-DOS, if the BGI interface is used, the variable **BGI** is used to point to the full path to the BGI drivers directory. Furthermore SVGA is used to name the Super VGA BGI driver in 800x600 res., and its mode of operation as 'Name.Mode'. For example, if the Super VGA driver is C:\TC\BGI\SVGADRV.BGI and mode 3 is used for 800x600 res., then: 'set BGI=C:\TC\BGI' and 'set SVGA=SVGADRV.3'.

Expressions

In general, any mathematical expression accepted by C, FORTRAN, Pascal, or BASIC is valid. The precedence of these operators is determined by the specifications of the C programming language. White space (spaces and tabs) is ignored inside expressions.

Complex constants may be expressed as {<real>,<imag>}, where <real> and <imag> must be numerical constants. For example, $\{3, 2\}$ represents $3 + 2\mathbf{i}$ and $\{0, 1\}$ represents $\mathbf{i}$ itself. The curly braces are explicitly required here.

Functions

The functions in GNUPLOT are the same as the corresponding functions in the Unix math library, except that all functions accept integer, real, and complex arguments, unless otherwise noted. The **sgn** function is also supported, as in BASIC.

Function	Arguments	Returns
abs(x)	any	absolute value of `x`, $\|x\|$; same type
abs(x)	complex	length of `x`, $\sqrt{\mathrm{real}(x)\hat{}2+\mathrm{imag}(x)\hat{}2}$
acos(x)	any	$cos\hat{}-1x$ (inverse cosine) in radians
arg(x)	complex	the phase of x in radians
asin(x)	any	$sin\hat{}-1x$ (inverse sin) in radians
atan(x)	any	$tan\hat{}-1x$ (inverse tangent) in radians
besj0(x)	radians	j_0 Bessel function of x
besj1(x)	radians	j_1 Bessel function of x
besy0(x)	radians	y_0 Bessel function of x
besy1(x)	radians	y_1 Bessel function of x
ceil(x)	any	$\lceil x \rceil$, smallest integer not less than x (real part)
cos(x)	radians	$cos\,x$, cosine of x
cosh(x)	radians	$cosh\,x$, hyperbolic cosine of x
erf(x)	any	Erf(real(x)), error function of real(x)
erfc(x)	any	Erfc(real(x)), 1.0 - error function of real(x)
exp(x)	any	$e\hat{}x$, exponential function of x
floor(x)	any	$\lfloor x \rfloor$, largest integer not greater than x (real part)
gamma(x)	any	Gamma(real(x)), gamma function of real(x)
ibeta(p,q,x)	any	Ibeta(real(p,q,x)), ibeta function of real(p,q,x)
igamma(a,x)	any	Igamma(real(a,x)), igamma function of real(a,x)
imag(x)	complex	imaginary part of x as a real number
int(x)	real	integer part of x, truncated toward zero
lgamma(x)	any	Lgamma(real(x)), lgamma function of real(x)
log(x)	any	log_ex, natural logarithm (base e) of x
log10(x)	any	log_10x, logarithm (base 10) of x
rand(x)	any	Rand(real(x)), pseudo random number generator
real(x)	any	real part of x
sgn(x)	any	1 if $x > 0$, -1 if $x < 0$, 0 if $x = 0$. imag(x) ignored
sin(x)	radians	$sin\,x$, sine of x
sinh(x)	radians	$sinh\,x$, hyperbolic sine x
sqrt(x)	any	$\sqrt{x}$, square root of x
tan(x)	radians	$tan\,x$, tangent of x
tanh(x)	radians	$tanh\,x$, hyperbolic tangent of x

Operators

The operators in GNUPLOT are the same as the corresponding operators in the C programming language, except that all operators accept integer, real, and complex arguments, unless otherwise noted. The ** operator (exponentiation) is supported, as in FORTRAN.

Parentheses may be used to change order of evaluation.