

GNUPLOT Quick Reference

(Copyright(c) Alex Woo 1992 June 1)

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

Graphics Terminals:

set terminal [options]

to enter GNUPLOT

to enter batch GNUPLOT

to pipe commands to GNUPLOT

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

Exiting GNUPLOT

exit GNUPLOT
quit

All GNUPILOT 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 on a topic
list of all help available
show current environment

Command-line Editing

The UNIX, MS-DOS and VMS versions of GNUPILOT 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

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

Hardcopy Devices:

Unknown - not a plotting device
Dump ASCII table of X Y [Z] values
Printer or glass dumb terminal
Roland DXY800A plotter
Dot Matrix Printers
Epson-style 60-dot per inch printers
Epson LX 800, Star NI-10 NX-1000, PROPRINTER NEC printer CP6, Epson LQ-800 Star Color Printer
Tandy DMP-130 60-dot per inch Vectrix 384 & Tandy color printer
Laser Printers

Talaris EXCL language
Imagen laser printer

LN03-Plus in EGM mode

PostScript graphics language

CorelDraw EPS
Prescribe - for the Kyocera Laser Printer

Kyocera Laser Printer with Courier font

QMS/QUIC Laser (also Talaris 1200)

Metafies

AutoCAD DXF (120x80 default)

FIG graphics language: SunView or X
FIG graphics language: Large Graph

SCO hardcopy CGI

Frame Maker MIF 3.0

Portable bitmap

Uniplex Redwood Graphics Interface Proto-

col

TGIF language

HP Devices

HP2623A and maybe others

HP2648 and HP2647

HP7580, & probably other HPs (4 pens)

HP7475 & lots of others (6 pens)

HP LaserJet series II & clones

HP DeskJet 500

HP PaintJet & HP3630

HP laserjet III (HPGL plot vectors)

TeX picture environments

EEPiC – extended LaTeX picture

LaTeX picture with emTeX specials

PSTricks macros for TeX or LaTeX

TPiC specials for TeX or LaTeX

MetaFont font generation input

```
set term dxf
set term fig
set term bfig
set term hfig
set term mif [pentype curvetype help]
set term pbm [fontsize color]
set term rgip
```

```
set term tgif
```

```
set term hp2623A
```

```
set term hp2648
```

```
set term hp7580B
```

```
set term hpgl
```

```
set term hpljii [75 100 150 300]
```

```
set term hpdj [75 100 150 300]
```

```
set term hppj [FNT5X9 FNT9X17 FNT13X25]
```

```
set term pc15 [mode font fontsize ]
```

```
set term excl
set term imgen
set term ln03
set term post [mode color 'font' size]
set term corel [mode color 'font' size]
set term prescribe
set term kyo
set term qms
```

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 **plots**, each data point represents an (x,y) pair. For **splots**, each point is an (x,y,z) triple. For **plots** with error bars (see **plot errorbars**), each data point is either (x,y,delta), (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 **plots** the x value may be omitted, and for **splots** 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 **plot** 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 nparametric;splot 'datafile.dat'
```

Using Pipes

On some computer systems with a **open** 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 "|lp -P my_laser_printer"
```

Files

plot a data file
load in a macro file

save command buffer to a macro file
save settings for later reuse

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

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.

```
{ using "<ycol>" | <xcol>:<ycol> | <xcol>:<ycol>:<ydelta> | <xcol>:<ycol>:<width> | <xcol>:<ycol>:<xdelta> | <xcol>:<ycol>:<ylow>:<yhigh> | <xcol>:<ycol>:<xlo>:<xhi> | <xcol>:<ycol>:<xdelta>:<ydelta> | <xcol>:<ycol>:<ycol>:<xlo>:<xhi> | <xcol>:<ycol>:<xdelta>:<ylow>:<yhigh> | <xcol>:<ycol>:<xlo>:<xhi>:<ylow>:<yhigh> | <xc>:<yc>:<xlo>:<xhi>:<ylow>:<yhigh> }... {<scanf string>}... { 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 **plots**, **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> or "%f%f%f" or "%f%f%f%f" for **plot** (or "%f%f%f%f" for **splot**) and "%f%f%f" is used for **splot**.

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

Data are read from the file "MyData" using the format "%*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.

Specifying a range in the **plot** command line 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, &

```
plot [-2:sin(5)*8] sin(x)**besj0(x)
```

This sets x,y and ymin only

```
plot [:200] [-pi:] exp(sin(x))
```

This sets the x, y, and z ranges

```
splot [0:3] [1:4] [-1:] x*y
```

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 **splots**.

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 x|low and x|high 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,ymin & ymax from columns 1,2,3,4           plot "data.dat" us 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 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 staircase-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 **splots** and function plots. For data plots, **errorbars** is like **points**, except that a vertical error bar is also drawn: for each point (x,y), a line is drawn from (x,y_{low}) to (x,y_{high}) . A tic mark is placed at the ends of the error bar. The y_{low} and y_{high} 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 x_{low} to x_{high} . 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**. 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
plots x*y with points, x**2 + y**2 default
plots tan(x) with default function style
plots "data.1" with lines
plots "leastsq.dat" with impulses
plots "exper.dat" with errorbars &
lines connecting points
```

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

```
plots x**2 + y**2 and x**2 - y**2 with the splot x**2 + y**2 w l 1, x**2 - y**2 w l 1
same line type
plots sin(x) and cos(x) with linespoints, using plot sin(x) w linesp 1 3, \
the same line type but different point types
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 title 'x'
plots the "glass.dat" file
plots x squared with title "x^2" and "data.1"
plots x**2 t "x^2", \
with title 'measured data'
plot x
splot "glass.dat" tit 'revolution surface'
plot x**2 t "x^2", \
"data.1" t 'measured data'
```

Set-Show Commands

all commands below begin with set
 set mapping of polar angles
 arrows from point to
 force autoscaling of an axis
 enter/exit parametric mode
 display border
 clip points/line near boundaries
 specify parameters for contour plots
 enable splot contour plots
 default plotting style for data
 specify dummy variable
 tic-mark label format specification
 function plotting style
 draw a grid at major tick marks & minor ticks
 (optional)
 enables hiddenline removal
 specify number of isolines
 enables key of curves in plot
 logscaling of an axes (optionally giving base)
 mapping 3D coordinates
 offsets from center of graph
 mapping 2D coordinates
 set radial range
 set sampling rate of functions
 set scaling factors of plot
 control display of isolines of surface
 control graphics device
 change direction of tics
 adjust relative height of vertical axis
 adjust size of tick marks
 turn on time/date stamp
 set centered plot title
 set parametric range
 set surface parametric ranges
 sets the view point for splot
 sets x-axis label
 set horizontal range
 change horizontal tics
 adjust number of minor tick marks
 draw x-axis
 sets y-axis label
 set vertical range
 change vertical tics
 draw y-axis
 set default threshold for values near 0
 draw axes
 sets z-axis label
 set vertical range
 change vertical tics
 draw z-axis

Contour Plots

set
angles [*degrees|radians*]
arrow [*tx,gy*] [*from* <*sx*> <*sy*>, <*sz*>]
 [to <*ex*> <*ey*>] [*nohead*]
autoscale [*axes*]
 [*no*] **parametric**
 [*no*] **border**
 [*no*] **clip** <*clip-type*>
cptrparam [*spline*][*points*] [*order*] [*levels*]
 [*no*] **contour** [*base|surface|both*]
data *style* <*style-choice*>
dummy <*dummy1*>, <*dummy2*> ...
format [*axes*]["*format-string*"]
function *style* <*style-choice*>
[no]grid [*mxgrid* OR *mygrid*]
 [*no*] **hidden3d**
isosamples <*expression*>
key <*xz*>, <*y*>, <*z*>
logscale <*axesbase*>]
 mapping [*cartesian|spherical|cylindrical*]
offsets <*left*>, <*right*>, <*top*>, <*bottom*>
 [*no*] **polar**
range [<*xmin*> <*xmax*>]
samples <*expression*>
 size <*xsize*>, <*ysize*>
[no]surface
tics <*direction*>
ticlevel <*level*>
ticscale [<*size*>]
 [*no*] **time**
title "title-text" <*xoff*>, <*yoff*>
trange [<*xmin*> <*xmax*>]
urange or **vrange**
view <*rot_x*>, <*rot_z*>, <*scale*>, <*scale_z*>
xlabel "<*label*>" <*xoff*>, <*yoff*>
xrange [<*xmin*> <*xmax*>]
xtics <*start*>, <*incr*>, <*end*>,
 "<*label*>" <*pos*>
 [*no*] **mxtics** OR [*no*] **mytics** [<*freq*>]
 [*no*] **zeroaxis**
ylabel "<*label*>" <*xoff*>, <*yoff*>
yrange [<*ymin*> <*ymax*>]
ytics <*start*>, <*incr*>, <*end*>,
 "<*label*>" <*pos*>
 [*no*] **zeroaxis**
zero <*expression*>
 [*no*] **zeroaxis**
zlabel "<*label*>" <*xoff*>, <*yoff*>
zrange [<*zmin*> <*zmax*>]
ztics <*start*>, <*incr*>, <*end*>,
 "<*label*>" <*pos*>
 [*no*] **zzeroaxis**

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/y/tics 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 cptrparam** for the parameters that affect the drawing of contours.

Contour Parameters

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

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

5 automatic levels
 3 discrete levels at 10%, 37% and 90%

5 incremental levels at 0, .1, .2, .3 and .4
 set **cptrparam** levels incremental .1 1/exp(1) .9

sets *n* = 10 retaining current setting of auto, set **cptrparam** levels 10

incr., or discr.
 set start = 100 and increment = 50, retaining set **cptrparam** levels incremental 100 50

old *n*

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. 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=xx"  
label "y=xx 2" w right of the text at (2,3,4),  
& tag the label number 3  
change preceding label to center justification  
delete label number 2  
delete all labels  
show label
```

(The EEPIC, Imagen, LaTeX, and TPIIC 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  
erase current screen or device  
exit GNUPLOT  
display text and wait  
print the value of <expression>  
print working directory  
repeat last plot or splot  
spawn an interactive shell
```

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.gh`).

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, `SYSSLOGIN`: 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 $3 + 2i$ and `{0,1}` represents `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
acos(x)	complex	$\text{length of } x, \sqrt{\text{real}(x)^2 + \text{imag}(x)^2}$
arg(x)	any	$\cos^{-1}x$ (inverse cosine) in radians
asin(x)	complex	the phase of x in radians
atan(x)	any	$\sin^{-1}x$ (inverse sin) in radians
atan(x)	any	$\tan^{-1}x$ (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	[x], 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	$\text{Erf}(\text{real}(x))$, error function of $\text{real}(x)$
erfc(x)	any	$\text{Erfc}(\text{real}(x))$, 1.0 - error function of $\text{real}(x)$
exp(x)	any	e^x , exponential function of x
floor(x)	any	[x], largest integer not greater than x (real part)
gamma(x)	any	$\Gamma(\text{real}(x))$, gamma function of $\text{real}(x)$
ibeta(p,q,x)	any	$I_{\beta}(p,q,x)$, ibeta function of $\text{real}(p,q,x)$
igamma(a,x)	any	$I_{\gamma}(p,q,x)$, igamma function of $\text{real}(p,q,x)$
imag(x)	complex	imaginary part of x as a real number
int(x)	real	integer part of x , truncated toward zero
lgamma(x)	any	$\Gamma(\text{real}(x))$, lgamma function of $\text{real}(x)$
log(x)	any	$\log ex$, natural logarithm (base e) of x
log10(x)	any	$\log_{10}x$, logarithm (base 10) of x
rand(x)	any	Rand($\text{real}(x)$), pseudo random number generator
real(x)	any	real part of x
sgn(x)	any	1 if $x > 0$, -1 if $x < 0$, 0 if $x = 0$. $\text{imag}(x)$ 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.