

GNUPLOT Quick Reference

(Copyright(c) Alex Woo 1992 June 1)

Starting GNUPLOT

to enter GNUPLOT	<code>gnuplot</code>
to enter batch GNUPLOT	<code>gnuplot macro_file</code>
to pipe commands to GNUPLOT	<code>application gnuplot</code>

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

Exiting GNUPLOT

exit GNUPLOT	<code>quit</code>
--------------	-------------------

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

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

History:

moves back through history	<code>^ P</code>
moves forward through history	<code>^ N</code>

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

IBM PC A K

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

Turbo C PC Graphics Modes:

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

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

Hardcopy Devices:

Unknown - not a plotting device	<code>set term unknown</code>
Dump ASCII table of X Y [Z] values	<code>set term table</code>
printer or glass dumb terminal	<code>set term dumb</code>
Roland DXY800A plotter	<code>set term dxv800a</code>

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:

Plotting Data

Discrete data contained in a file can be 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) or (x,y,low,high). 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 **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	<code>set parametric;splot 'glass.dat'</code>
example of plotting explicit	<code>set noparametric;splot 'datafile.dat'</code>

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>:<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 **plots**, ydelta (for example, a +/- error) should be provided as the third column, or ylo and yhi as third and fourth columns. These columns must follow the x and y columns.

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" for **errorbar plots**) 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 or two additional columns specifying ydelta or ylo and yhi respectively. No support exists for x error bars or any error bars for **splots**.

In the default situation, GNUPLOT expects to see three or four numbers on each line of the data file, either (x, y, ydelta) or (x, y, ylo, yhi). 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
```

The error bar is a vertical line plotted from (x, ylo) to (x, yhi). If ydelta is specified instead of ylo and yhi, ylo=y-ydelta and yhi=y+ydelta are derived. If there are only two numbers on the line, yhi and ylo are both set to y. To get lines plotted between the data points, **plot** the data file twice: once with errorbars and once with lines

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, &

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

turns off autoscaling on both axes

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
```


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 plot contour plots
default plotting style for data
specify dummy variable
tic-mark label format specification
function plotting style
draw a grid at tick marks
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
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

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

```
set
angles [degrees|radians]
arrow [<tag>] [from <sx>, <sy>, <sz>]
  [to <ex>, <ey>, <ez>] [nohead]
autoscale [<axes>]
[no]parametric
[no]border
[no]clip <clip-type>
cntrparam [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
[no]hidden3d
isosamples <expression>
key <x>, <y>, <z>
logscale <axes> [<base>]
mapping [cartesian|spherical|cylindrical]
offsets <left>, <right>, <top>, <bottom>
[no]polar
rrange [<rmin>:<rmax>]
samples <expression>
size <xsize>, <ysize>
[no]surface
terminal <device>
tics <direction>
ticslevel <level>
[no]time
title "title-text" <xoff>, <yoff>
trange [<tmin>:<tmax>]
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]xzeroaxis
ylabel "<label>" <xoff>, <yoff>
yrange [<ymin>:<ymax>]
ytics <start>, <incr>, <end>,
"<label>" <pos>
[no]yzeroaxis
zero <expression>
[no]zeroaxis
```

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, `set cntrparam levels 10`
incr., or discr.
set start = 100 and increment = 50, retaining `set cntrparam 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 $z = z1, z2 \dots$ as specified. The number of discrete levels is limited to MAX_DISCRETE_LEVELS, defined in plot.h to be 30. If 'incremental', contours

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), set label 3 "y=x^2" at 2,3,4 right
& tag the label number 3
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, AmigaDOS, 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 AmigaDOS 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 AmigaDOS, SHELL is used for the **shell** command. On MS-DOS, COMSPEC is used for the **shell** command.

On AmigaDOS, 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
abs(x)	complex	length of x , $\sqrt{\text{real}(x)^2 + \text{imag}(x)^2}$
acos(x)	any	$\cos^{-1}x$ (inverse cosine) in radians
arg(x)	complex	the phase of x in radians
asin(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	$\lceil x \rceil$, 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	$\text{Erf}(\text{real}(x))$, 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	$\lfloor x \rfloor$, largest integer not greater than x (real part)
gamma(x)	any	$\text{Gamma}(\text{real}(x))$, gamma function of $\text{real}(x)$
ibeta(p,q,x)	any	$\text{Ibeta}(\text{real}(p, q, x))$, ibeta function of $\text{real}(p, q, x)$
igamma(a,x)	any	$\text{Igamma}(\text{real}(a, x))$, igamma function of $\text{real}(a, x)$
imag(x)	complex	imaginary part of x as a real number
int(x)	real	integer part of x , truncated toward zero
lgamma(x)	any	$\text{Lgamma}(\text{real}(x))$, lgamma function of $\text{real}(x)$
log(x)	any	$\log_e x$, natural logarithm (base e) of x
log10(x)	any	$\log_{10} x$, logarithm (base 10) of x
rand(x)	any	$\text{Rand}(\text{real}(x))$, 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.