```
 G N U P L O T
 Macintosh version 3.7
 patchlevel 1
 last modified Fri Oct 22 18:00:00 BST 1999

 Copyright(C) 1986 - 1993, 1998, 1999
 Thomas Williams, Colin Kelley and many others

 Type `help` to access the on-line reference manual
 The gnuplot FAQ is available from
 <http://www.ucc.ie/gnuplot/gnuplot-faq.html>

 Send comments and requests for help to <info-gnuplot@dartmouth.edu>
 Send bugs, suggestions and mods to <bug-gnuplot@dartmouth.edu>


Terminal type set to 'pict'
gnuplot> help splot using
Sorry, no help for 'splot using'
gnuplot> help splot
 `splot` is the command for drawing 3-d plots (well, actually projections on
 a 2-d surface, but you knew that).  It can create a plot from functions or
 a data file in a manner very similar to the `plot` command.

 See `plot` for features common to the `plot` command; only differences are
 discussed in detail here.  Note specifically that the `binary` and `matrix`
 options (discussed under "datafile-modifiers") are not available for `plot`.

 Syntax:
       splot {<ranges>}
             <function> | "<datafile>" {datafile-modifiers}}
             {<title-spec>} {with <style>}
             {, {definitions,} <function> ...}

 where either a <function> or the name of a data file enclosed in quotes is
 supplied.  The function can be a mathematical expression, or a triple of
 mathematical expressions in parametric mode.

 By default `splot` draws the xy plane completely below the plotted data.
 The offset between the lowest ztic and the xy plane can be changed by `set
 ticslevel`.  The orientation of a `splot` projection is controlled by
 `set view`.  See `set view` and `set ticslevel` for more information.
Press return for more:
 The syntax for setting ranges on the `splot` command is the same as for
 `plot`.  In non-parametric mode, the order in which ranges must be given is
 `xrange`, `yrange`, and `zrange`.  In parametric mode, the order is `urange`,
 `vrange`, `xrange`, `yrange`, and `zrange`.
```

The `title` option is the same as in `plot`. The operation of `with` is also the same as in `plot`, except that the plotting styles available to `splot` are limited to `lines`, `points`, `linespoints`, `dots`, and `impulses`; the error-bar capabilities of `plot` are not available for `splot`.

The datafile options have more differences.

Subtopics available for splot:

binary	data-file	datafile	errorbars
example	grid_data	matrix	parametric
ranges	style	title	with

Subtopic of splot: style

Functions and data may be displayed in one of a large number of styles. The `with` keyword provides the means of selection.

Syntax:

```
      with <style> { {linestyle | ls <line_style>}
                   | {{linetype | lt <line_type>}
                      {linewidth | lw <line_width>}
                      {pointtype | pt <point_type>}
                      {pointsize | ps <point_size>}} }
```

where <style> is either `lines`, `points`, `linespoints`, `impulses`, `dots`, `steps`, `fsteps`, `histeps`, `errorbars`, `xerrorbars`, `yerrorbars`, `xyerrorbars`, `boxes`, `boxerrorbars`, `boxxyerrorbars`, `financebars`, `candlesticks` or `vector`. Some of these styles require additional information. See `set style <style>` for details of each style.

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
Press return for more: order, if more are required. The LaTeX driver supplie point types (all variants of a circle), and thus will only repeat after 12 curves are plotted with points. The PostScript drivers (`postscript`) supplies a total of 64.

If you wish to choose the line or point type for a single plot, <line_type> and <point_type> may be specified. These are positive integer constants (or expressions) that specify the line type and point type to be used for the plot. Use `test` to display the types available for your terminal.

You may also scale the line width and point size for a plot by using <line_width> and <point_size>, which are specified relative to the default values for each terminal. The pointsize may also be altered globally---see `set pointsize` for details. But note that both <point_size> as set here and

as set by `set pointsize` multiply the default point size---their effects are not cumulative. That is, `set pointsize 2; plot x w p ps 3` will use points three times default size, not six.

If you have defined specific line type/width and point type/size combinations with `set linestyle`, one of these may be selected by setting <line_style> to the index of the desired style.

Press return for more: The keywords may be abbreviated as indicated.

Note that the `linewidth` and `pointsize` options are not supported by all terminals.

Examples:

This plots sin(x) with impulses:
```
      plot sin(x) with impulses
```

This plots x with points, x**2 with the default:
```
      plot x*y w points, x**2 + y**2
```

This plots tan(x) with the default function style, file "data.1" with lines:
```
      plot [ ] [-2:5] tan(x), 'data.1' with l
```

This plots "leastsq.dat" with impulses:
```
      plot 'leastsq.dat' w i
```

This plots the data file "population" with boxes:
```
      plot 'population' with boxes
```

Press return for more: This plots "exper.dat" with errorbars and lines connec (errorbars require three or four columns):
```
      plot 'exper.dat' w lines, 'exper.dat' notitle w errorbars
```

This plots sin(x) and cos(x) with linespoints, using the same line type but different point types:
```
      plot sin(x) with linesp lt 1 pt 3, cos(x) with linesp lt 1 pt 4
```

This plots file "data" with points of type 3 and twice usual size:
```
      plot 'data' with points pointtype 3 pointsize 2
```

This plots two data sets with lines differing only by weight:
```
      plot 'd1' t "good" w l lt 2 lw 3, 'd2' t "bad" w l lt 2 lw 1
```

See `set style` to change the default styles.

Subtopic of splot: with
Functions and data may be displayed in one of a large number of styles.
The `with` keyword provides the means of selection.

```
 Syntax:
       with <style> { {linestyle | ls <line_style>}
                    | {{linetype | lt <line_type>}
                       {linewidth | lw <line_width>}
                       {pointtype | pt <point_type>}
                       {pointsize | ps <point_size>}} }

 where <style> is either `lines`, `points`, `linespoints`, `impulses`, `dots`,
 `steps`, `fsteps`, `histeps`, `errorbars`, `xerrorbars`, `yerrorbars`,
 `xyerrorbars`, `boxes`, `boxerrorbars`, `boxxyerrorbars`, `financebars`,
 `candlesticks` or `vector`.  Some of these styles require additional
 information.  See `set style <style>` for details of each style.

 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
Press return for more:
```