

## Three-Dimensional Plots with Gnuplot

This is a brief introduction by example to making plots of three-dimensional (3-d) surfaces and data with the GNUPLOT plotting program. You'll want to start up GNUPLOT (by typing 'gnuplot') and follow along. The basic command for 3-d plots (that is, projections on a 2-d surface) is 'splot', which shares features and options with the 'plot' command (no error bars, though). You can plot functions or from a datafile.

Start by making a surface plot of a simple function ( $x$  and  $y$  are the default dummy variables and you can set the xrange, yrange, and/or zrange):

```
gnuplot> splot sin(x)*cos(y) with lines
```

It's hard to see what is going on unless you remove hidden lines:

```
gnuplot> set hidden3d
```

```
gnuplot> replot
```

This looks pretty rough, so we check the number of "isosamples" and then increase the sampling rate (type 'help set isosamples' for details):

```
gnuplot> show isosamples
```

```
gnuplot> set isosamples 50,50
```

```
gnuplot> replot
```

Use 'set view' to change how the plot is viewed. The four arguments of this command (separated by commas) specify rotations about the x-axis, the z-axis, overall scaling, and scaling of the z-axis. For example

```
gnuplot> show view # returns the current view
```

```
gnuplot> set view 60, 60, 1, 1.5
```

```
gnuplot> replot
```

Play around with different values to get a feel for changing the view. Change both the angles and the scaling of the axes.

Type **reset** to reset everything to the default values. Now try plotting a defined function (which we call "sinc"), with a grid on the x-y plane:

```
gnuplot> set title "3D gnuplot example"
```

```
gnuplot> sinc(x,y) = sin(sqrt(x**2+y**2))/sqrt(x**2+y**2)
```

```
gnuplot> set grid
```

```
gnuplot> splot [-5:5.01] [-5:5.01] sinc(x,y) with lines
```

The last 'splot' incorporates the xrange and yrange in the plotting command.

The plot seems to float pretty high above the xy-plane. The ‘ticslevel’ sets the height from the bottom of the z-scale above the xy-plane (the default is 0.5). We can also adjust what tics are shown for a given axis (try ‘help set xtics’ to see details):

```
gnuplot> set ticslevel 0.1
gnuplot> set xtics -10,2,10
```

Replot and try eliminating hidden lines and changing the sampling. To dress up the plot some more, we can add axes, with offsets so that they are separated from the graph.

```
gnuplot> set xlabel "X axis" -3,-2 # label x-axis with offsets
gnuplot> set ylabel "Y axis" 3,-2 # label y-axis with offsets
gnuplot> set zlabel "Z axis" -5, 0 # label z-axis with offsets
```

You can also make “parametric” plots, which for 3-d plotting means that the surface is described as  $x = f(u, v)$ ,  $y = g(u, v)$ ,  $z = h(u, v)$ . The ‘set parametric’ command changes plot or splot from normal functions to parametric functions. To change back, use ‘set noparametric’. The plot is made by specifying the ranges of  $u$  and  $v$  and then calling splot with three functions, separated by commas. Exit, start gnuplot again and try this example (we’ll suppress the gnuplot prompts from here on):

```
set title "Torus"
set parametric
set urange [-pi:pi]
set vrange [-pi:pi]
set isosamples 50,20
splot cos(u)+.5*cos(u)*cos(v),sin(u)+.5*sin(u)*cos(v),.5*sin(v) with lines
set hidden3d
replot
```

Finally, we can also plot data. Each line in the data file has either three numbers (the  $x$ ,  $y$ , and  $z$  values), or a single number. To get the isolines drawn correctly, the data should be arranged in “datablocks”, separated by a blank line. Look at 3d\_shape.cpp for an example of generating data files. *Predict what the plot will look like.* Then compile and run to generate the data files and plot them with:

```
set hidden3d
splot "3d_shape1.dat" with lines, "3d_shape2.dat" with lines
replot
```

If only a single value is given on each line, then it is interpreted as a  $z$  value with the datablock number used for  $y$  and the index of the data point in the datablock used for  $x$ . Blank lines separate datablocks in a file. The points in a datablock are treated as a  $y$ -isoline. If the datablocks have the same number of points, the cross-isolines are drawn.