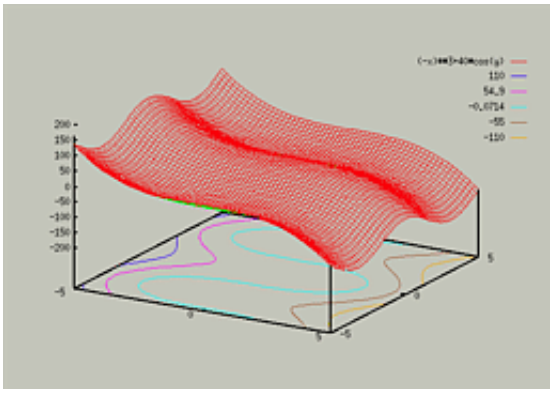
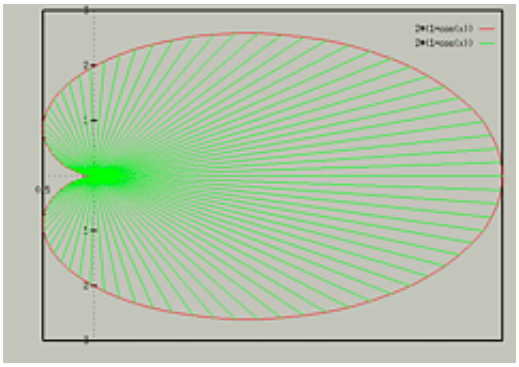


Introduction to GnuPlot

College of Natural Sciences Computing Laboratories
University of Northern Iowa
Cedar Falls, IA 50614



Gnuplot is a command-line driven program for producing 2D and 3D plots. Although it doesn't have as many features as some of the commerical mathematical software available, it isn't as complex to use as packages such as Mathematica or Matlab. It is ideal for a users you only need a plot of a graph, and who don't want to learn a major tool.

The features and topics on Gnuplot covered in this document include :

If you have questions at any time, you can access the on-line help by typing **help** within Gnuplot. Gnuplot has a very good help system.

Starting and Quitting Gnuplot

To access Gnuplot, you probably want to be logged into one of the Suns in Wright 339. It is possible to run GnuPlot on any Xserver, but that is beyond the scope of this document. When you are logged into a Sun, you can do one of the following:

- At the prompt on one of the suns, type **gnuplot**
- Pick GnuPlot from the Wright Hall Menu that comes up when you hold the Right mouse button down.

To quit gnuplot, type **exit**, or hit Control-D.

Specifying Functions in Gnuplot

In 2D plots, the variable to use is **x**, in 3D plots, use **x** and **y**.

Multiplication is denoted by *, and division by /. Exponents are denoted by**, and all multiplication must be explicit. That is, *3x* would generate and error, you would want to use *3*x*.

For example, the polynomial $3x^4 + 4x - 2/3$ would be the following in Gnuplot:

- **3*x**4 + 4*x - 2/3**

Gnuplot also has a number of predefined functions. These are called by putting the arguments in parenthesis, i.e., **sin(x)**. These include the following:

- The standard trig functions, **sin, cos, and tan** Note, to use the constant pi can be referenced by just using **pi**.
- The inverse trig functions , **asin, acos, and atan**.
- The hyperbolic trig functions, **sinh, cosh, and tanh**.
- The exp and log function. exp raises *e* to the power of its argument. For example, $4e^{2x}$ would be **4* exp(2*x)** in Gnuplot. log returns the natural log (base *e*) of it's argument. This corresponds to *ln* in normal math notation.
- For information on other functions, type **help functions** in Gnuplot.

The following are other examples of functions. Make sure you understand what functions are being described.

- **(x**2 - 4)/(x +2)**
- **x**2 + y**2 - (x*y)**(2/3)**
- **log(exp(x))** (Hint, this simplifies to *x*)
- **sin(x*y)**

  3*x2 + 6/(1*y**2)**

A Simple Example

For a quick example, let's say you wanted a graph of *sin(x)*. You would then type the following at the **gnuplot>** prompt:

- plot sin(x)

This will pull up another window with the graph you wanted. If you made a mistake or want to slightly change what you typed, hit the **UP ARROW**. This will pull up what you typed. If you have typed a couple of lines, you can keep hitting the arrow button. The you can use the left and right arrow keys to change what you typed. As a quick example, try changing what you typed to

  plot sin(2*x)

Graphing Two Dimensional Functions

The command used for two-dimensional plots is **plot**. The simplest use of it is:

- plot <function>

For example, to plot x^3, type

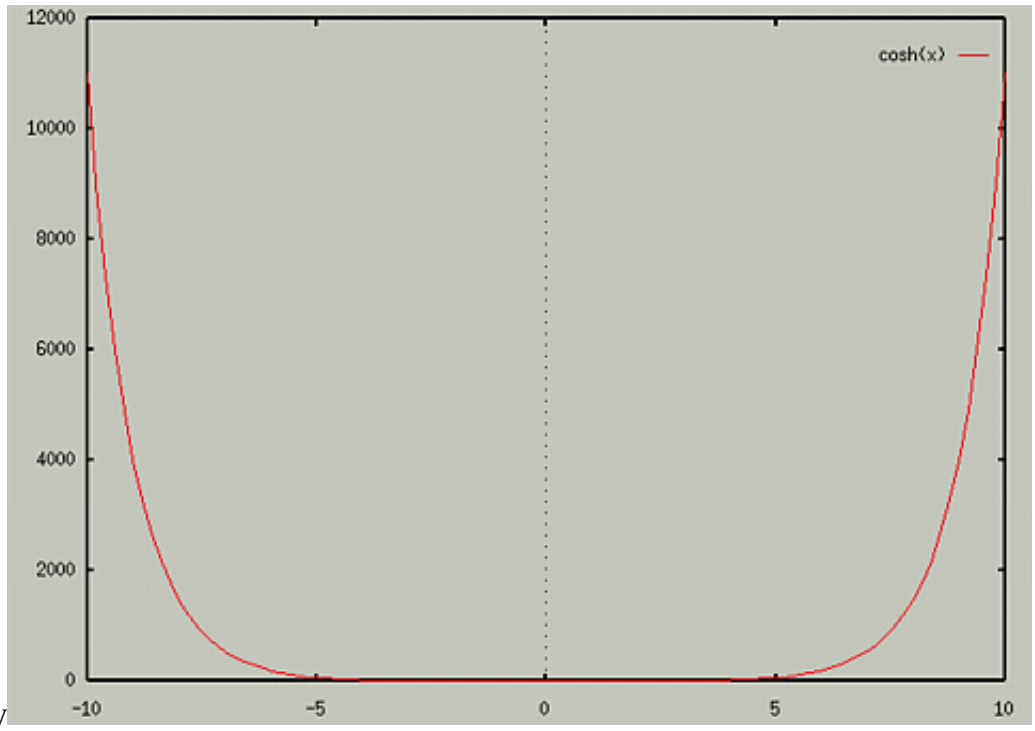
- **plot x**3**

You should see something like the following pop up in the graphics screen.

For another example, to see what the hyperbolic cosine function looks like, type

- **plot cosh(x)**

This should display



Note that in this picture the scale isn't the best for finding a lot of information about the graph. It tells us that the function gets very large as x gets larger. But if we want to see what the function looks like closer to 0, we need to

change the scale. Gnuplot uses an autoscale mechanicism. It sets the scale so that the graph will fit on the screen. To change the scale, use one of the following forms of **plot**:

- **plot [x1:x2] [y1:y2] <function>**
- **plot [x1:x2] <function>** (To just set the x range)
- **plot [] [y1:y2] <function>** (To just set the y range).

Note, that if you set the range for the x values on one plot, they will stay that way for future plots. To set the default x range back to [-10,10], type the following:

- set xrange [-10:10]

Also, if you set the range for the y values, they will stay the same for future plots as well. You can set GnuPlot back to autoscaling the y axis by typing the following:

- set autoscale y

For more information on the **set xrange** and **set autoscale** commands, use the help command. To see what cosh looks like from *x=[-5, 5]*, type

- **plot [-5:5] cosh(x)**

Or, to see what cosh looks like from *y=[0, 10]*, type

plot [] [0,10] cosh(x)

Graphing Three Dimensional Functions

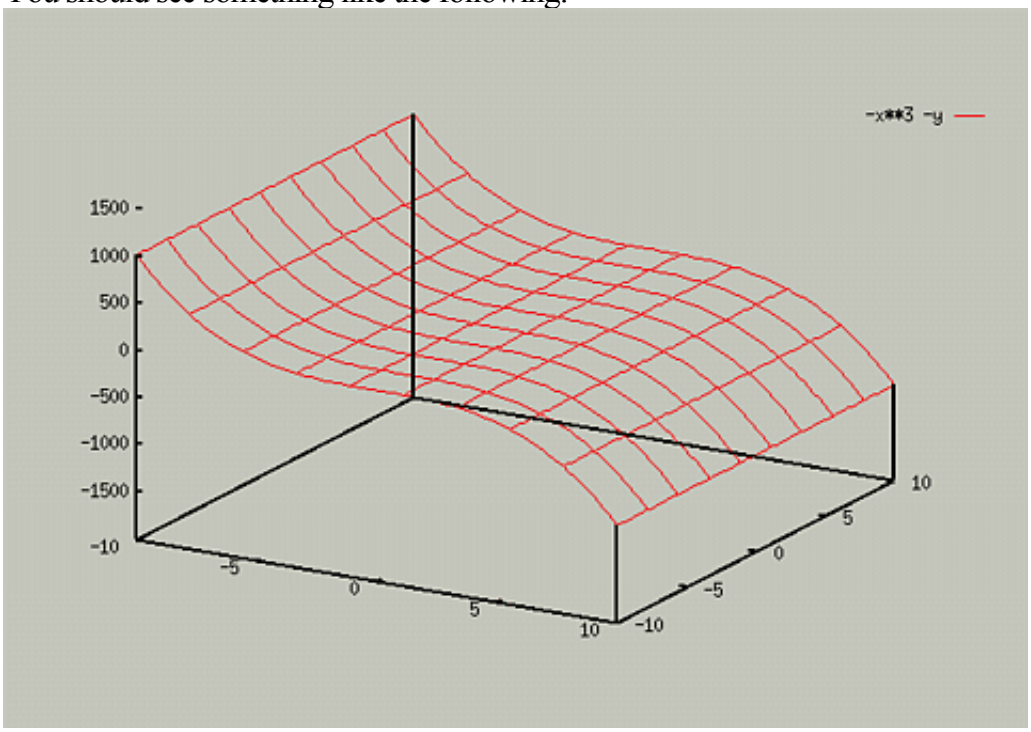
The command used for three-dimensional plots is **splot**. The simplest use of it is:

- splot <function>

For example, to plot the function $z = -x^3 - y$, you would type the following:

- splot -x**3 -y

You should see something like the following:



You can change the scale in a similar manner to **plot**:

- **splot [x1:x2] [y1:y2] [z1:z2] <function>**
- **splot [x1:x2] <function>** (To just set the x range)
- **splot [] [y1:y2] <function>** (To just set the y range).
- **splot [] [] [z1:z2] <function>** (To just set the y range).

Or, you can use the following commands to change the default ranges for subsequent graphs:

- **set xrange [x1:x2]**
- **set yrange [y1:y2]**
- **set zrange [z1:z2]**

To change back to the default x and y ranges, type the following:

- **set xrange [-10:10]**
- **set yrange [-10:10]**

It is important to note that if you set the range for the zaxis in one plot, it will remain at that range for subsequent plots. If you want to turn GnuPlot's autoscaling back on, type:

- set autoscale z

For more information on autoscaling, type **help autoscale**. Also, if you get an error about an invalid range, you need to change the ranges you are using.

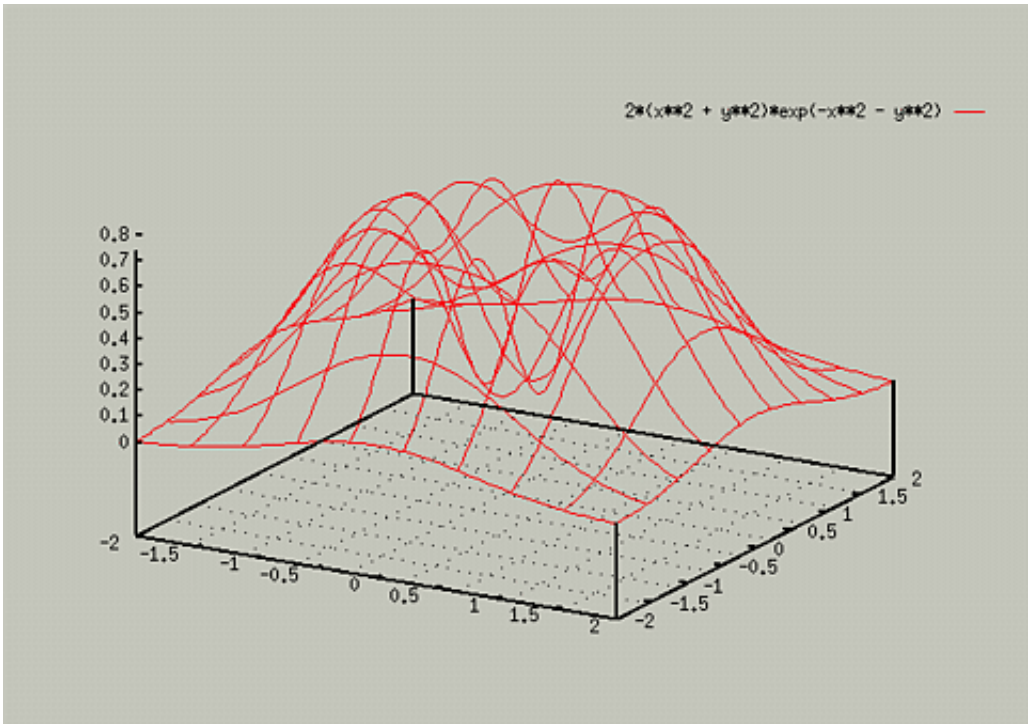
Hidden 3D view

Sometimes, 3D graphs can be hard to interpret. For instance, the graph of

- $z = 2(x^2 + y^2)exp(-x^2 - y^2)$

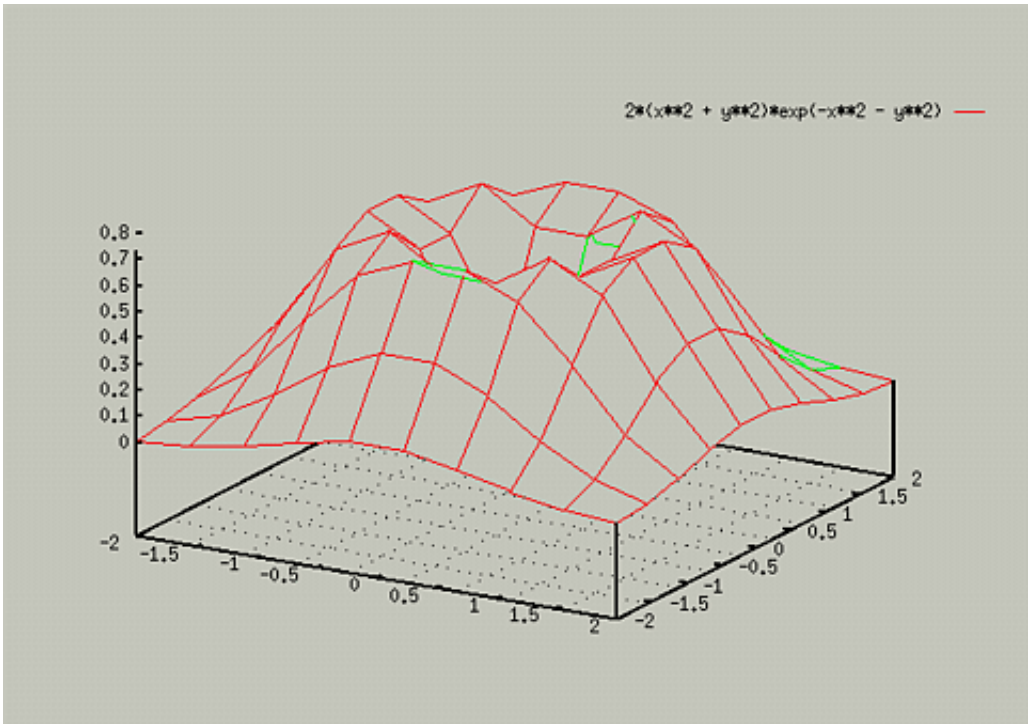
could be seen by typing the following:

- splot [-2:2] [-2:2] 2*(x**2 + y**2)*exp(-x**2 - y**2)



Note that in this graph, the set grid command was used. To make viewing this easier, you can use the **set hidden3d** command. For instance, if you typed the following, you would get:

- set hidden3d
- replot



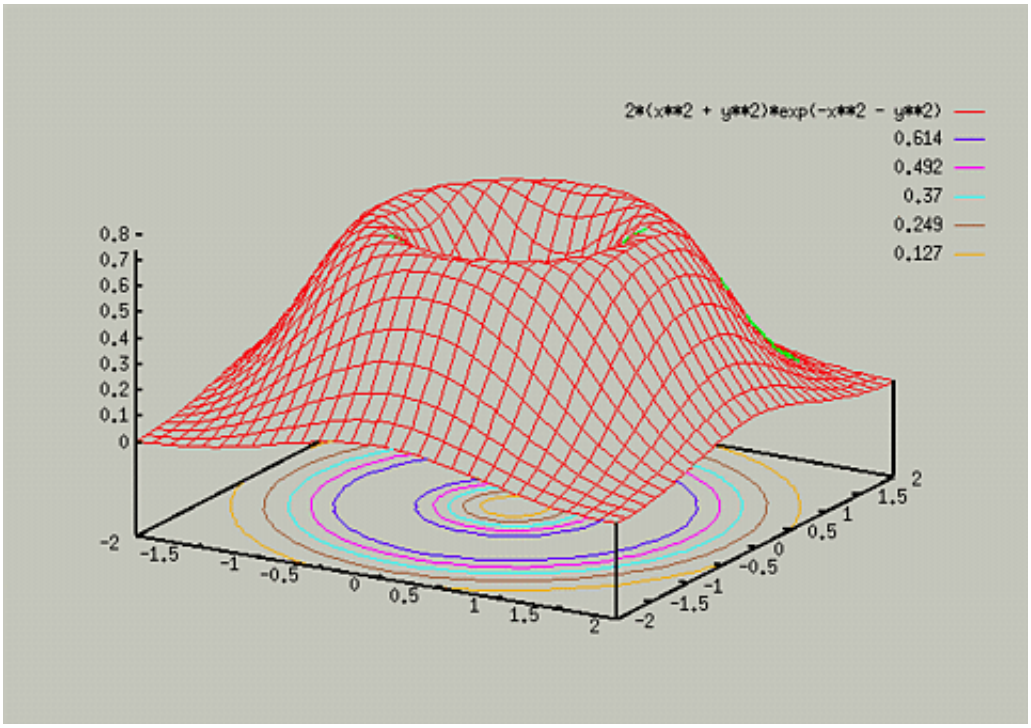
Increasing Precision of 3D plots

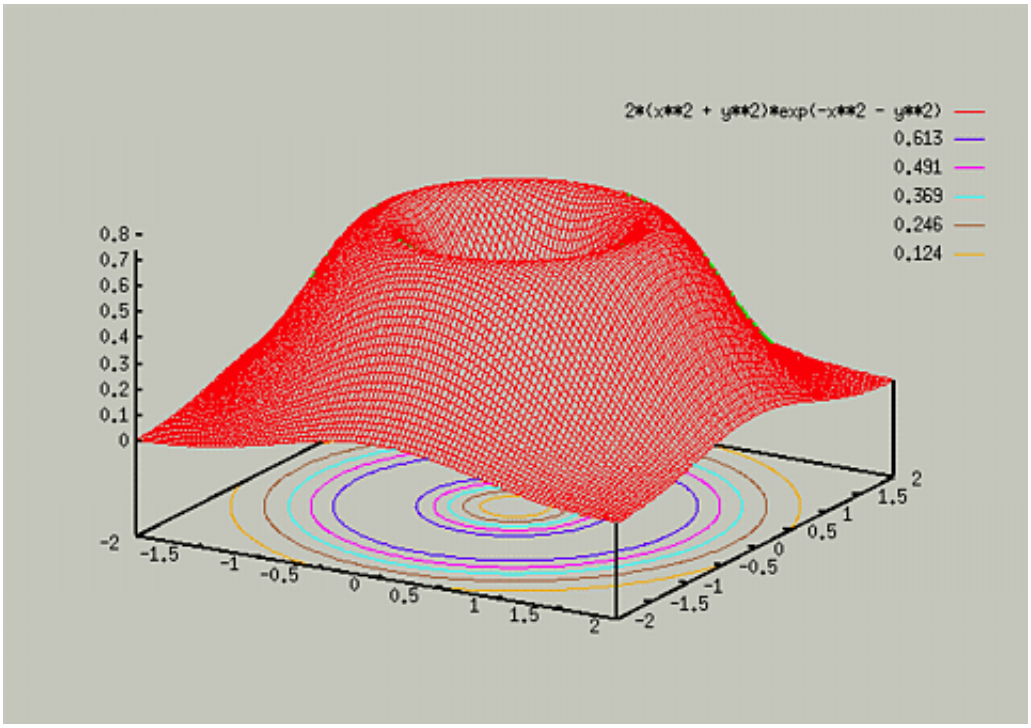
When you look at the previous graph, it seems that the resolution of the graph is not very high. To decrease the amount of choppiness in the graph, you need to increase the precision used. This can be done for 2D plots by the **set samples** command, and for 3D plots by **set isosamples** command. You will almost always have to use this for 3D plots, because the default sampling rate is very low. The syntax of the command is:

- set isosamples x_rate, y_rate

The default is for both rates to be set to 10. For example, typing the following yields:

- set isosamples 30, 30
- splot [-2:2] [-2:2] 2*(x**2 + y**2)*exp(-x**2 - y**2)
- set isosamples 100, 100
- replot





It is important to note that the higher you set the isosamples, the longer it will take to create the graph. In most cases, you won't want to set the sample size larger thant 100.

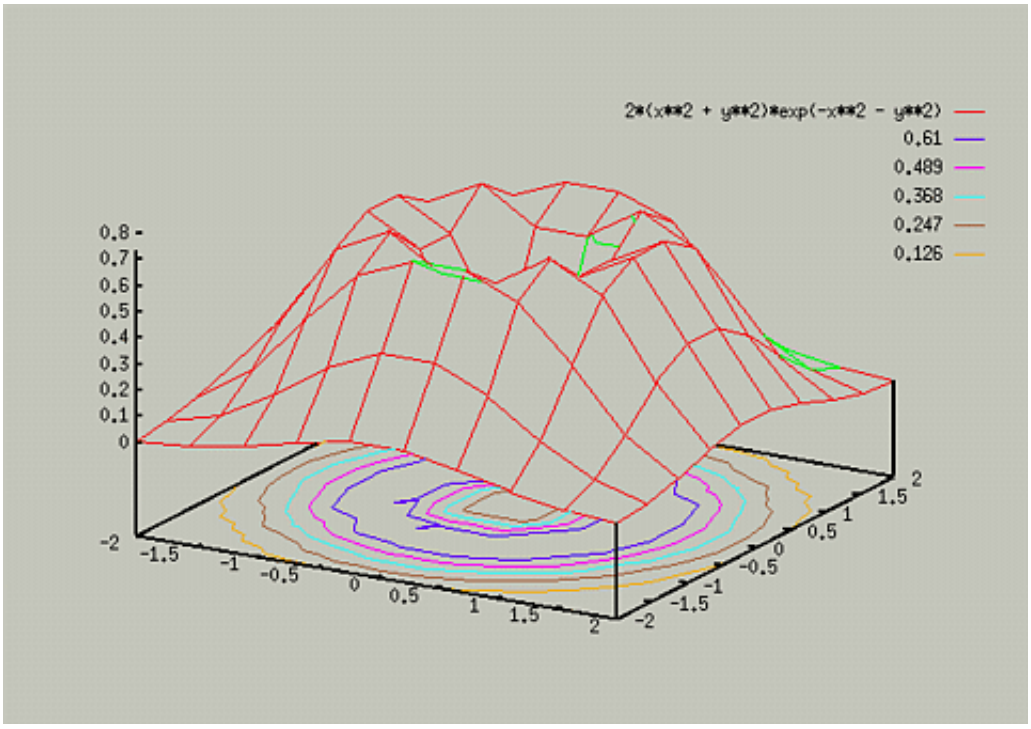
Adding Contour Lines

The following commands relate to contour lines in 3D plots:

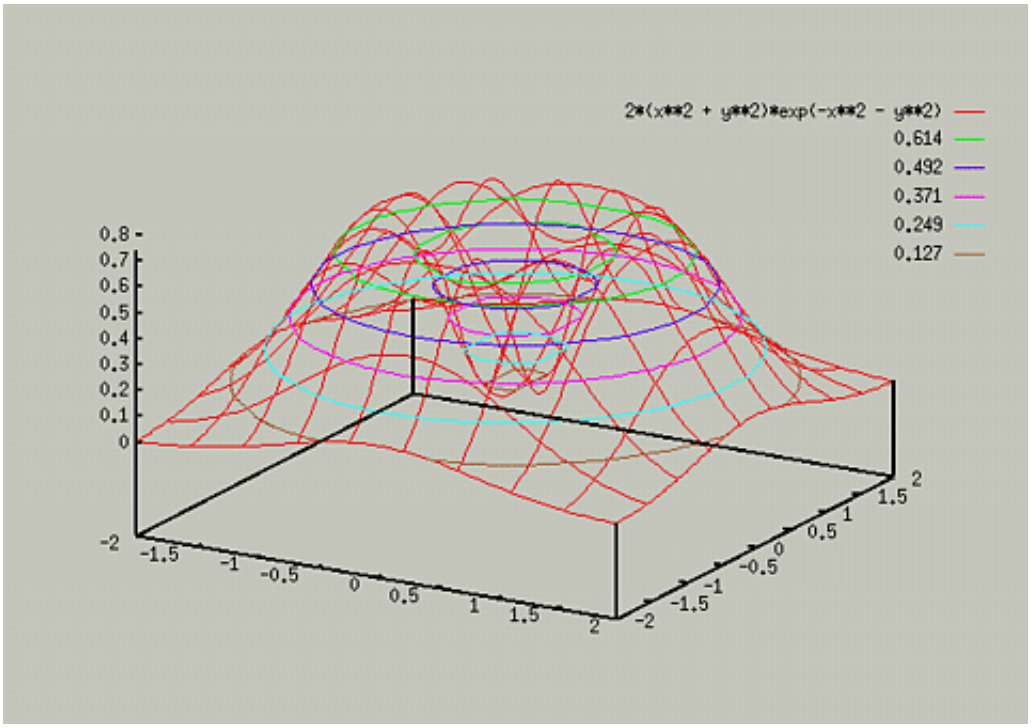
- **set contour base** - Draw the contour lines along the base of the diagram
- **set contour surface** - Draws the lines along the 3D surface
- **set contour both** - Draws lines on both surface and base
- **set nocontour** - Turns off contour lines

Note that the **set contour surface** option is not available with the **set hidden3d** option. Typing the following commands yields the following two graphs:

- set hidden3d
- set contour base
- splot [-2:2] [-2:2] 2*(x**2 + y**2)*exp(-x**2 - y**2)



- set nohidden3d
- set contour surface
- replot



If only the contours lines are desired, the **set nosurface** command will turn off drawing the surface. For more information on this, type **help set surface**. The drawing of contour lines is highly custimizable, for information type **help set cntrparam**.

For example, if you only want the contour lines for *z=.2, .4, .6*, you could enter the following:

set cntrparam levels discrete .2, .4, .6

Changing View of Graph

Many times, you will want to change the view of a graph. To do this, you need to use the **set view** command. This command is fairly unwieldy. The syntax for the command is one of the following:

- **set view *horizontal_angle*,*vertical_angle***
- **set view *horizontal_angle*,*vertical_angle*,*zoom***
- **set view ,,*zoom***

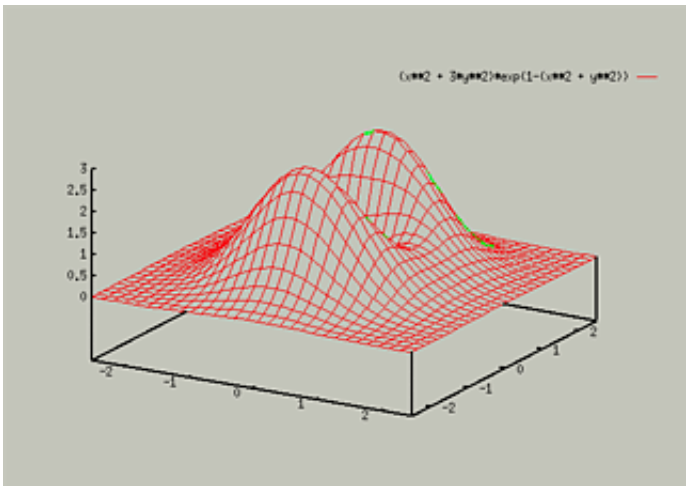
The first argument is the angle between the viewpoint and the horizontal axis of the screen. The second is the angle between the axis perpendicular to the screen. The third number if the zoom factor. The default values are 60, 30, 1. To change back to the default view, type:

- set view 60,30,1

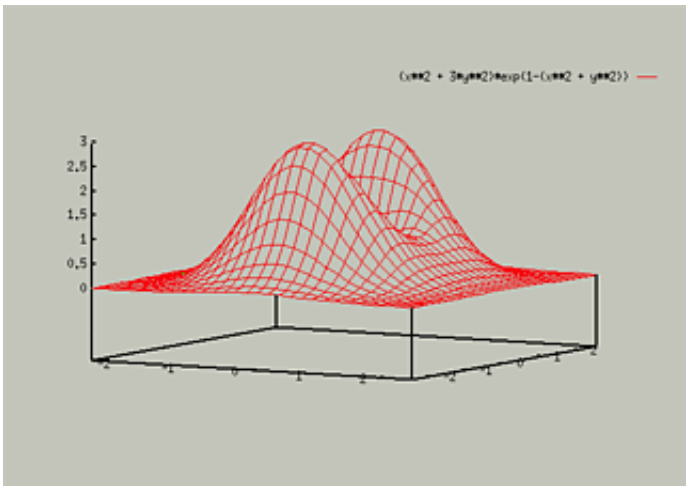
For instance, to see how changing the horizontal angle changes the graph,consider the following:

- set hidden3d
- set isosamples 30

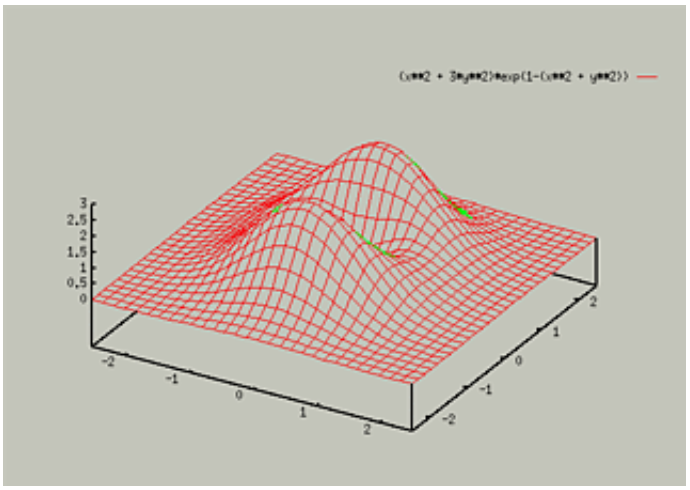
- splot [-2.5:2.5] [-2.5:2.5] (x**2 + 3*y**2)*exp(1-(x**2 + y**2))



- set view 80,30
- replot

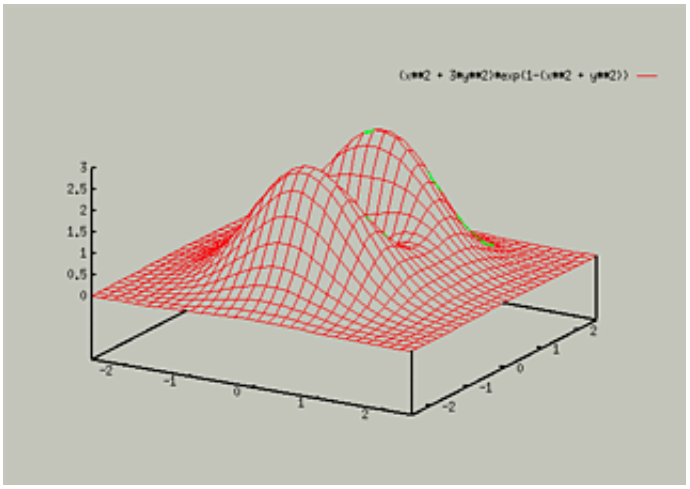


- set view 40,30
- replot

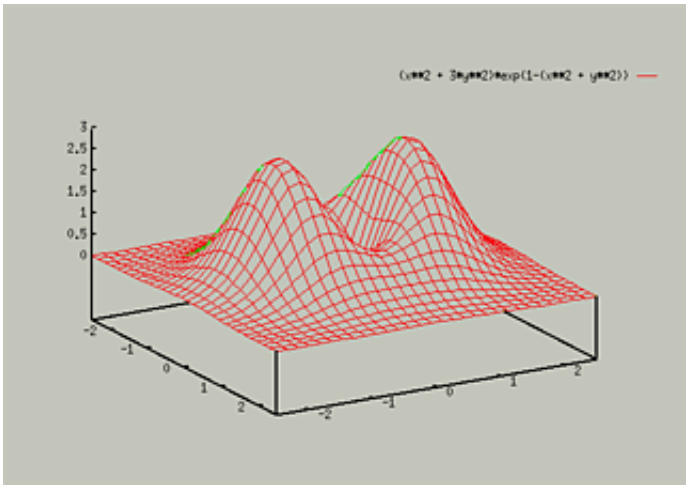


To see what happens when you change the vertical angle, consider the following:

- set view 60,30
- splot [-2.5:2.5] [-2.5:2.5] (x**2 + 3*y**2)*exp(1-(x**2 + y**2))

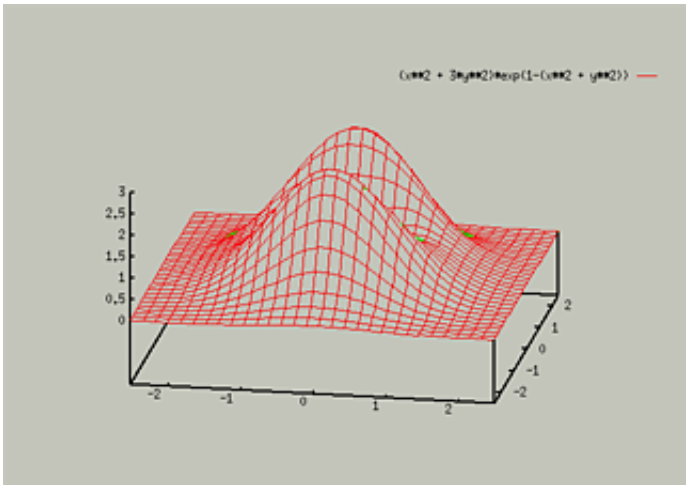


- set view 60,60
- replot



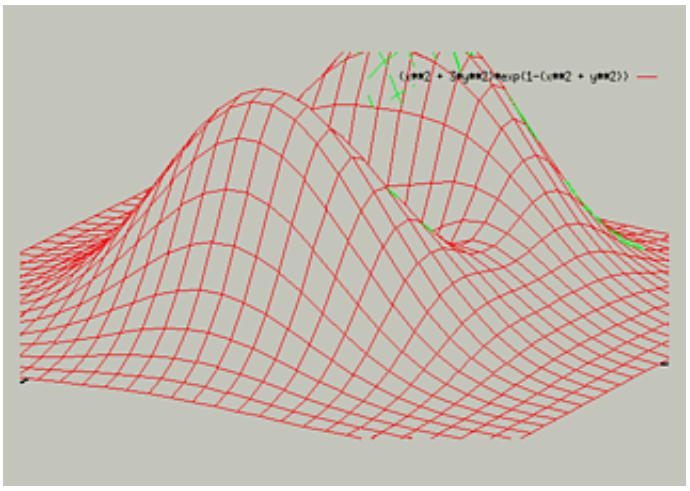
- set view 60,10

  replot



To see an example of zooming, consider

- set view 60,30
- set view ,,2
- replot



Parametric Equations

GnuPlot includes support for Parametric Equations. To switch to parametric mode, type the following:

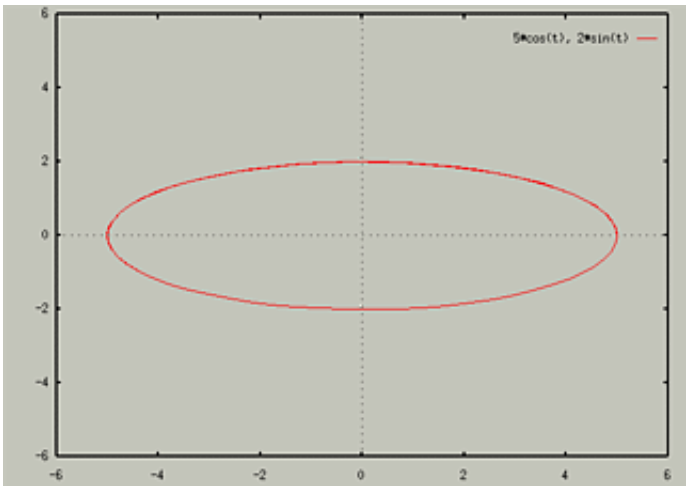
- **set parametric**

To go back to standard Cartesian mode, type:

- **set noparametric**

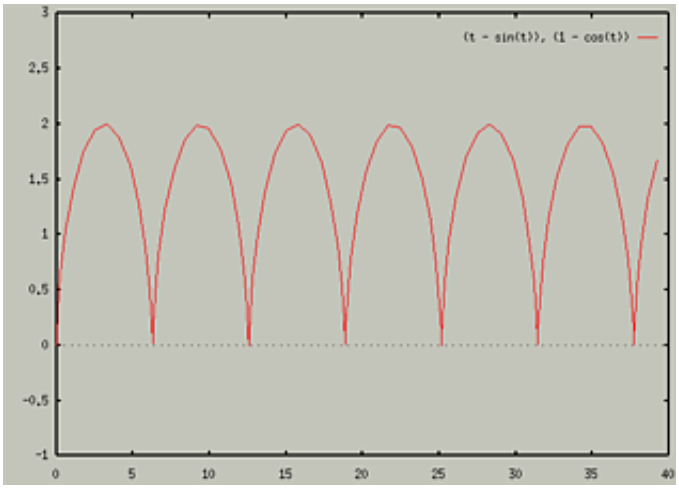
For example, to see the plot of the set of equations $x = 5 \cos t$ and $y = 2 \sin t$, you could type the following:

- **set parametric**
- **set xrange [-6:6]**
- **set yrange [-6:6]**
- **set trange[0:10]**
- **set isosamples 60**
- **plot 5*cos(t), 2*sin(t)**



The xrange and yrange values specify the range of the graph displayed. The **set trange** command sets the range that is used to compute the parametric equations. If you use brackets to set a range within the plot command, it will refer to the trange. The first part of the plot command before the comma is the *x* equation, and the second part is the *y* equation.
For another example, consider trying to plot the cycloid $x = t - \sin t$ and $y = 1 - \cos t$.

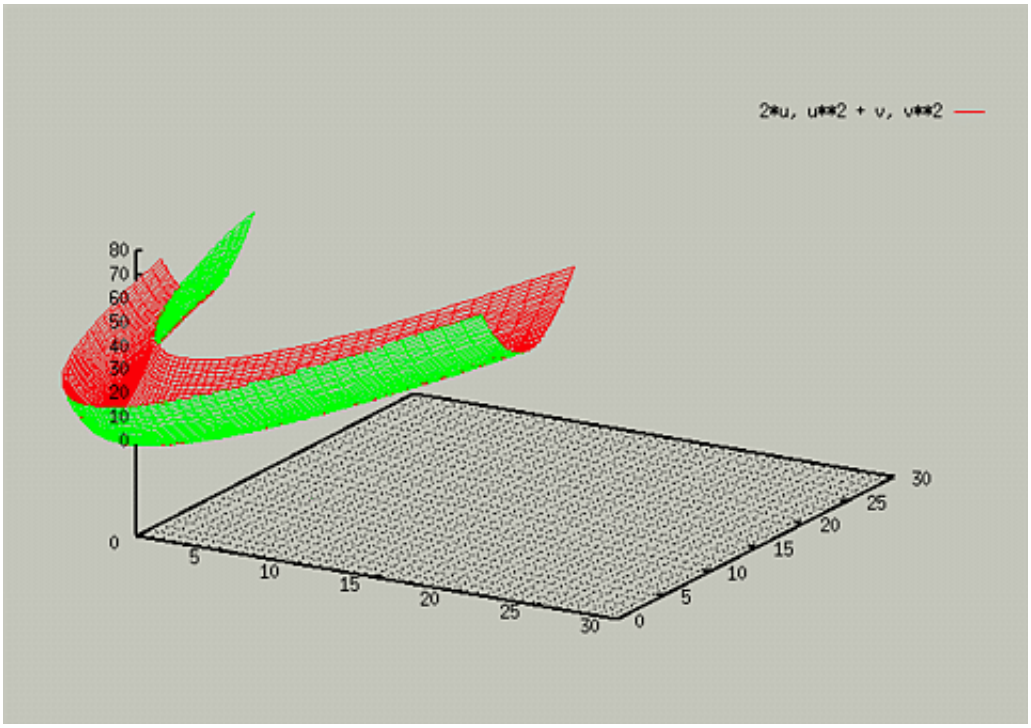
- **set xrange [0:40]**
- **set yrange[-1:3]**
- **set trange [0:40]**
- **plot t-sin(t), 1-cos(t)**



3D Parametric Plots

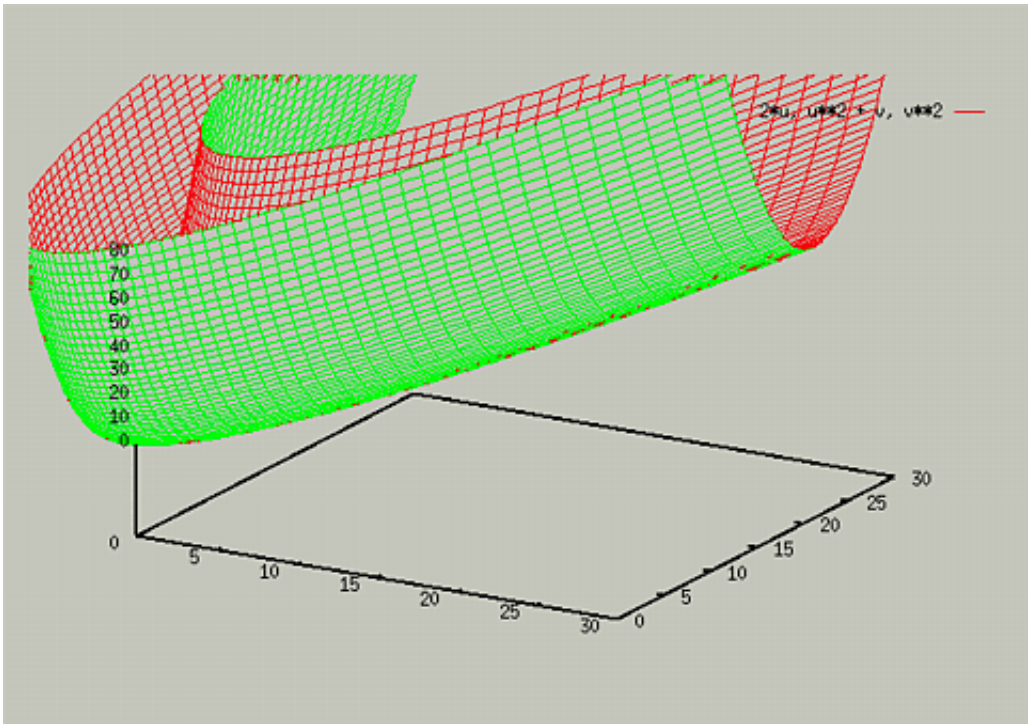
Gnuplot also supports 3D parametric plots. It uses the the variables *u* and *v*. 3D parametttric plots use the xrange, yrange, and zrange values to determine what is displayed on the screen. The **urange** and **vrange** variables can be changed to determine how much of the graph is drawn. For example,:

- **set xrange [0:30]**
- **set yrange [0:30]**
- **set zrange [0:80]**
- **set grid**
- **splot 2*u, u**2 + v, v**2**



If you changed the **urange** and **vrange** parameters, you would get a differant graph.

- **set vrange [-10:10]** - *this makes the graph taller*
- **set urange [-7:7]** - *this makes the graph extend further*
- **set isosamples 80**
- **replot**



Many of the standard 3D features apply to 3D parameter plots (you can use *isosamples, views, and hidden3d*).

Polar Plots

To switch to graphing with polar coordinates, type:

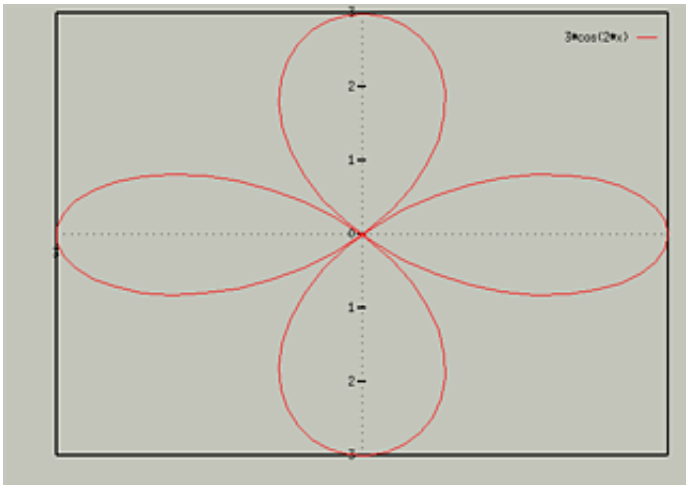
- **set polar**

To switch back to Cartesian coordinates, type:

- **set nopolar**

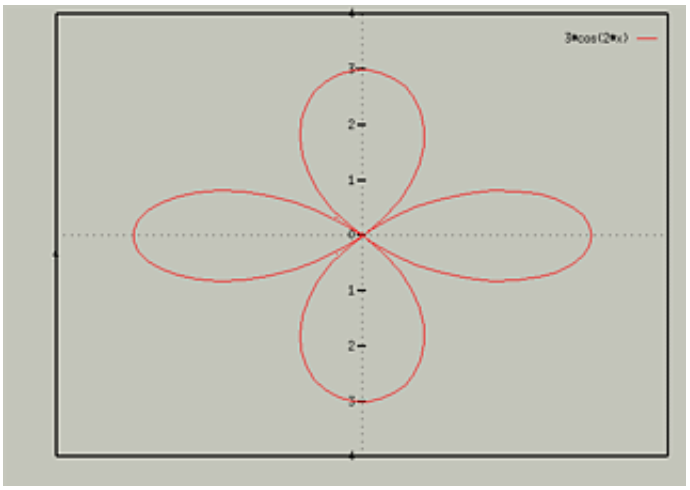
In polar mode, the independent variable *x* corresponds to the angle, or *theta* noramally used for polar plots. Polar plots are always plotted in a rectangular box, to change the size of the graph, use the **set yrange** command. This will change both the horizontal and vertical range. For an example, consider the following:

- **set polar**
- **plot 3*cos(2*x)**



To change the displayed range so their is extra space around the graph, type:

- **set yrange [-4:4]**
- **replot**



Note that this would leave the **yrange** set to [-4:4] for subsequent plots. To turn autoscaling back on, use the following:

- **set autoscale y**

The default for interpretation of the angle is radians. To switch this value, you can use one of the following:

- **set angles degrees**
- **set angles radians**

3D Polar coordiantes

GnuPlot supports the 3D equivalent of polar coordinates, spherical and cylindrical coordinates, are only available for 3D **data** plots. For information on them, see **help set mapping**.

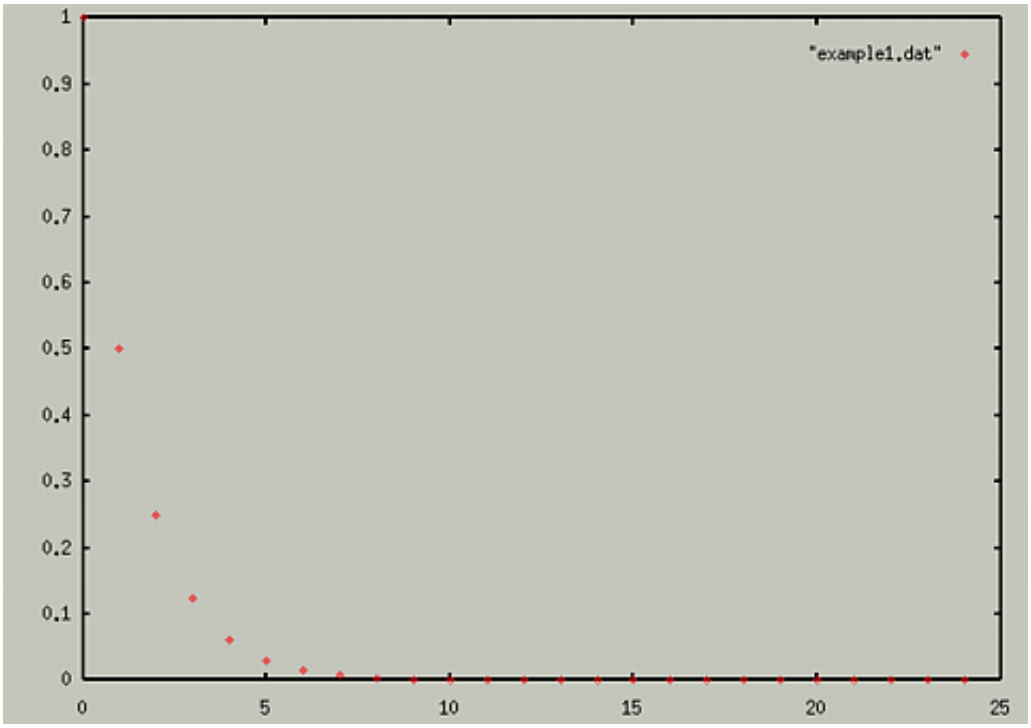
Plotting Data

One of the most powerful features in GnuPlot is how easily Data can be plotted. For example, we have the following data file (in /opt/Gnu/info/gp/example1.dat). The data was generated by applying euler's method to a ODE problem. The first column is the number of subintervals. The second column is the step size (1/(# of subintervals)). The third column is the value obtained, and the fourth column is the absolute error.

```
# Example1.dat
# number of subint.    - width of subinterval, computed value, abs. error
0 1                      5                 0.00673794699908559
1 0.5                    5.0009765625      0.00576138449908559
2 0.25                   5.00317121193893  0.00356673506015159
3 0.125                  5.00478985229103  0.00194809470805701
4 0.0625                 5.00572403277733  0.0010139142217529
5 0.03125                5.00622120456923  0.000516742429855555
6 0.015625               5.00647715291715  0.000260794081935245
7 0.0078125              5.00660694721608  0.000130999783004349
8 0.00390625             5.00667229679632  6.56502027673866e-05
9 0.001953125            5.0067050843668   3.28626322820824e-05
10 0.0009765625          5.0067215063061   1.64406929883398e-05
11 0.00048828125         5.00672972430911  8.22268997247022e-06
12 0.000244140625        5.00673383506831  4.11193077187733e-06
13 0.0001220703125       5.00673589088727  2.05611181058885e-06
14 6.103515625e-05       5.00673691890656  1.02809252577885e-06
15 3.0517578125e-05      5.00673743294363  5.14055459532869e-07
16 1.52587890625e-05     5.00673768996904  2.57030048800289e-07
17 7.62939453125e-06     5.00673781848355  1.28515534214557e-07
18 3.814697265625e-06    5.00673788274127  6.42578132925564e-08
19 1.9073486328125e-06   5.00673791487013  3.21289537197345e-08
20 9.5367431640625e-07   5.00673793093482  1.60642699142954e-08
21 4.76837158203125e-07 5.00673793896686  8.03222910406021e-09
```

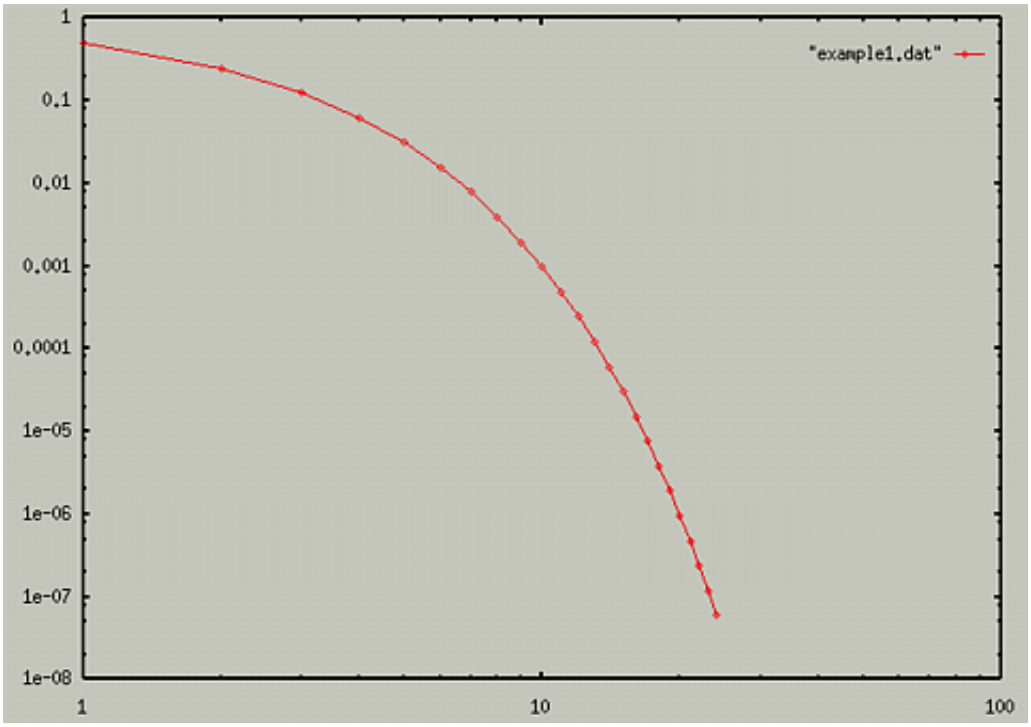
Note that lines beginning with a # are considered comment lines and are ignored. To plot this data, you can type the following: *(if the file isn't in the first level of your home directory, you can specify the full path, or see the directions on Saving and Loading From a file for information on changing directories.)*

- **plot "example1.dat"**



This plots points *(x, y)* where x is from the first column and y is from the second column. To make the graph a little easier to read, you could type the following:(for information on these, see logscale, Line Style)

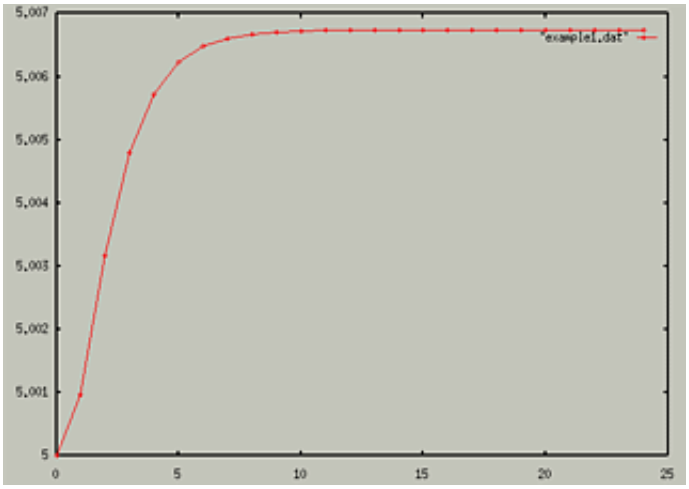
- **set logscale**
- **set data style linespoints**
- **replot**



Changing which columns are plotted

GnuPlot also lets you plot data from differant columns against each other. You add the **using** keywork to a **plot** command. For example, to plot the computed value (column 3) against the number of subintervals (column 1), you could type:

- **set nologscale**
- **plot "example1.dat" using 1:3**



Or to plot the third column against the 4th column,

- **plot "exampl1.dat" using 4:3**

There are many advanced options for plotting columns against each other like this, for more information see **help plot datafile using**.

3D Data Plots

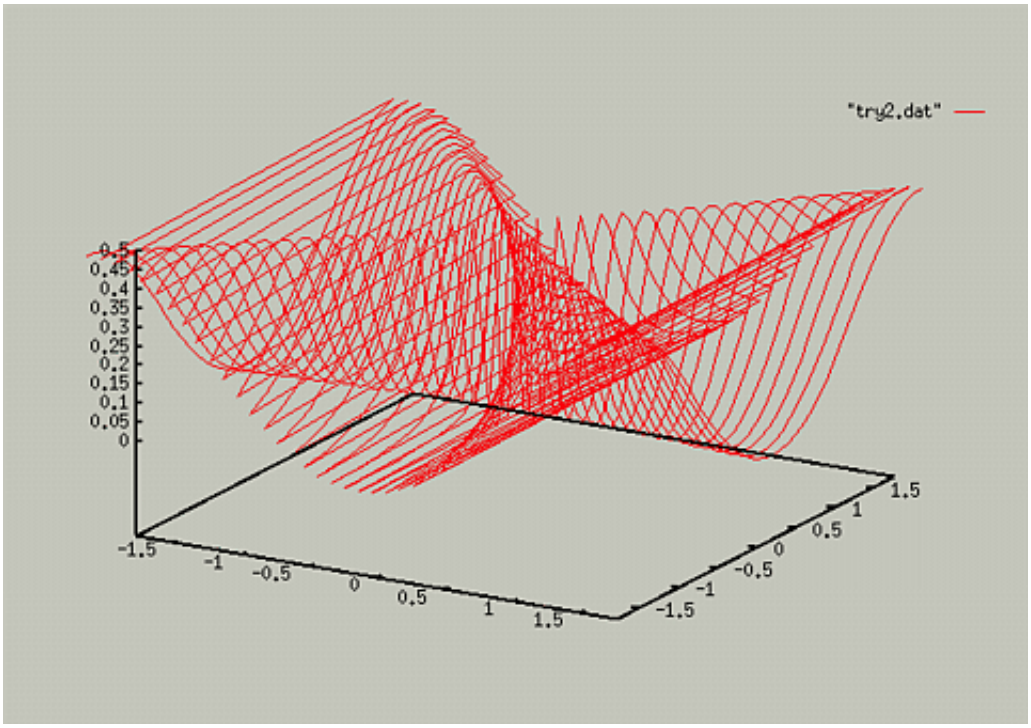
GnuPlot also lets you do 3D plots. The Data File must have three columns, one for the x, the y, and the z values. For instance, if you had a data file like the following:

```
-2.000000       -2.000000       0.500000
-2.000000       -1.500000       0.427300
-2.000000       -1.000000       0.235294
-2.000000       -0.500000       0.062257
-2.000000        0.000000       0.000000
-2.000000        0.500000       0.062257
-2.000000        1.000000       0.235294
-2.000000        1.500000       0.427300
-2.000000        2.000000       0.500000
-1.500000       -2.000000       0.427300
-1.500000       -1.500000       0.500000
-1.500000       -1.000000       0.371134
-1.500000       -0.500000       0.109756
```

```
-1.500000       0.000000       0.000000
-1.500000       0.500000       0.109756
-1.500000       1.000000       0.371134
-1.500000       1.500000       0.500000
-1.500000       2.000000       0.427300
-1.000000      -2.000000       0.235294
-1.000000      -1.500000       0.371134
-1.000000      -1.000000       0.500000
-1.000000      -0.500000       0.235294
...
```

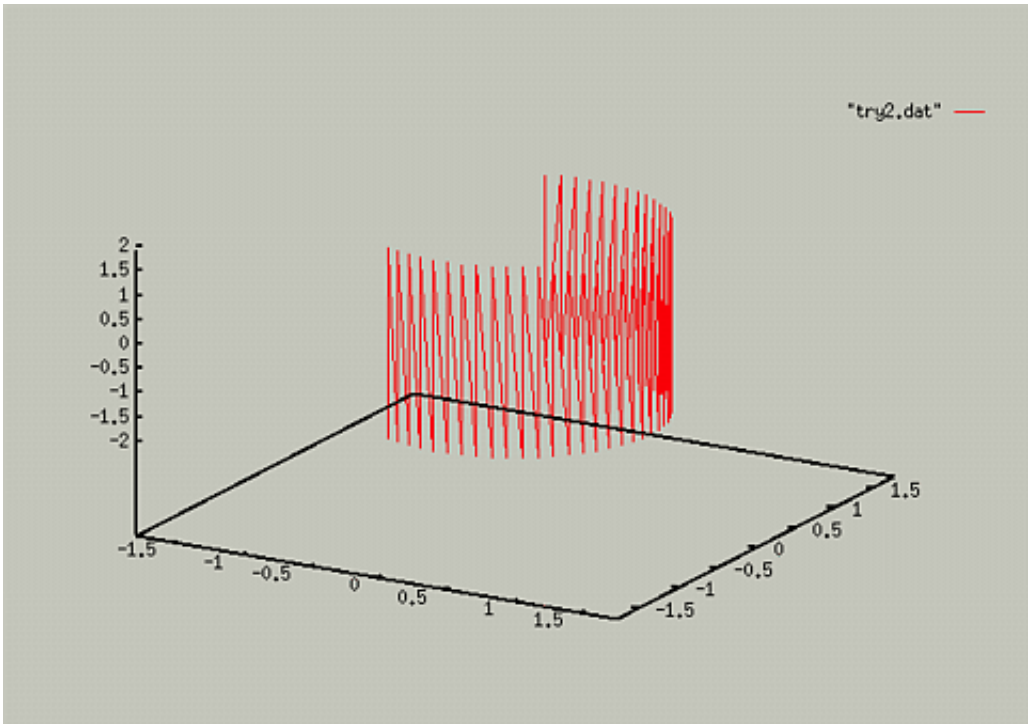
To plot this data, if it were in a file called "try2.dat", you would type:

- **set parametric**
- **set data style lines**
- **splot "try2.dat"**



You can also use cylindrical and spherical coordiantes with 3D data plots. For example:

- **set mapping cylindrical**
- **replot**



For more information, see **help plot datafile**.

Generating Output

The next couple of pages deal with getting output from the graphs you have created. There are several ways you can do this:

- **Print to a Printer** - This would be useful if all you need is a paper copy of the graph you have constructed.
- **Output to a Postscript File** - This would be useful if you would like to print multiple copies, or would like to have the image available to print at a later time without replotting it.
- **Output to a Encapsulated Postscript File** - This would be useful if you would like to include the graphics in a document. Most work processors and desktop publishers (i.e., FrameMaker) import EPS files with the highest quality.

  Output to a Graphics File - This would be useful if you wanted to post the image on the web

Printing to Printer

Once you have the plot you want on the screen, do the following to print the graphic to a printer:

- **set output "| lp"** - This will make the graphic screen disappear
- **set terminal postscript**
- **replot**
- **set output "| lp"** - You can use the up arrow to call this line up again.

After the last command, you should see a message like the following:

- *gnuplot> set output "|lp"*

  request id is lj3sib-961 (standard input)

This means that is was printed.

To go back to having the graphical display, type the following:

- **set terminal x11** - (x11 is the type of windowing system the Suns use)

If you want to print later, you only have to do the last three of the commands, i.e.,

- **set terminal postscript**
- **replot**
- **set output "| lp"** - You can use the up arrow to call this line up again.

Note that the **lp** in the output command is the print command for your machine. Use can use the following for the output command:

- **set output "| lp"** - Prints to the default printer for your machine. For machines in Wright 339, this *lj3sib*.
- **set output "| lp -dlj3sia"** - Prints to the specified printer. *lj3sia* is in Wright 339 as well.

  set output "| lp -n 3" - Prints more than one copy (in this case, 3 copies). This can be combined with the -d option as well.

Outputting to a Postscript File

There are two main uses for graphs in Postscript format. The first is to have a copy you can print later, without using GnuPlot. You need regular Postscript files for this. The second is to include the graph in another document. For this option you need to use Encapsulated Postscript Files. This is a special form of Postscript that can be included in documents. (EPS files don't have the extra lines that are used when a PS file is printed directly to a printer). To generate a postscript file, do the following:

- **set output "filename.ps"** where *filename.ps* is the name of the output file.
- **set terminal postscript**
- **replot**

To print this file from the command line (not in GnuPlot), type

- **lp filename.ps** To use the default printer, or
- **lp -dlj3sib filename.ps** To use the printer lj3sib.

If you print to this file again, it will append the graph to the end of the file as a second page. For instance,

- **set output "file2.ps"**
- **set terminal postscript**
- **splot sin(x*y)**
- **plot x**2**

would create a 2-page postscript file called "file2.ps". The first page would be the plot of $z=sin(xy)$ and the second page would be the plot of $y=x^2$. Usually, you will only want the one graph in each PS file.

When you want to return to printing to the screen, type the following two lines:

- **set output**
- **set terminal x11**

Options for set terminal postscript

There are several options for this command:

- **set terminal postscript color** - This will create a postscript file with color in it. The default is "monochrome", which only produces black and white images.
- **set terminal postscript solid** - This will use solid lines when printing, instead of the default "dashed". This usually creates better looking graphs.

Printing to EPS files

To print to an EPS file, you do the same as for PS files, but use the following **set terminal** command.

- **set output "filename.eps"**
- **set terminal postscript eps**

Make sure you only print one image to each EPS file.

Outputing to Graphics files

You cannot output directly from GnuPlot to Graphics files. You could use the convert command (*convert test.ps test.gif*) if you had printed to a file, or you can use the screen grab option in Image Magick. Do the following to get a graphics files version of a graph.

1. Have the graph you want displayed in a graphics window.
2. Have ImageMagick started. You can do this by typing **display &** in a shelltool or commandtool (not at the gnuplot prompt).
3. If you didn't have Gnuplot already running, choose Load from the menues that pop up when you hold a mouse button down over Image Magick's screen.
4. Choose **Grab**
5. Move the Grab window if you have to, and make sure that the window with your graph in it is unobstructed.
6. Click Okay in the Grab Window. Then click the left mouse button in the window with the Graph you want.

7. After a couple of seconds, a copy of the graph should pop up in GnuPlots' window. You can resize it or apply effects to it within ImageMagick.
8. Choose **Save** from ImageMagick's menu.
9. Choose **format** from the Save screen.
10. Select the graphics type you want (i.e., *.gif* or *.jpeg*.
11. Give the file the name you want, and click on Save.

For more information on ImageMagick, see here for information on the display program or here for information on the convert program.

Saving and Loading from a File

Gnuplot lets you load a series or commands in from a file. This is useful when you are trying to "tweak" a lengthy expression to get the output that you want. If you haven't used an UNIX editor before, you will probably want to start with **PICO**. You can start it within a terminal window by typing **pico**, or you can select it from the Editors menu of the Wright Hall Menu. For information on how to use Pico, see here. For instance, you might need to enter the following lines to get a graph that you want.

- **set time**
- **set title "File Graph"**
- **set hidden3d**
- **set xrange [-5:5]**
- **set yrange [-5:5]**
- **set isosamples 50**
- **# Actually Plot the Graph**
- **splot exp(-x**2 - y**2) -cos(y)-(x**2)/5**

You could put this in your home directory in a file called, for instance, "work.gnu". Then to run these commands, you would type the following in GnuPlot:

- **load 'work.gnu'**

Or if the file is in a differant directory,

- **load 'web/gnuplot/work.gnu'**

This file is on the system, if you want to try it, type

- **load '/opt/Gnu/info/gp/work.gnu'**

The line beginning with the "#" is a comment line. You can put these in your text files as comments to yourself.

Changing Directories

If you want to put the file in another directory than your home directory, you just need to use the **cd** command in GnuPlot. First, to see what directory you are in, type the following:

- **pwd**

You should see something like "/user/bos", where your username is in place of "bos". To change directories, just use the cd command as follows:

- **cd "web/gnuplot"**

Adding Pauses, Looping Animations

To make changing GnuPlot graphics, you can specify multiple plots, and just put **pause** statements between them. For more information, see **help pause**. For an example, type **load '/opt/Gnu/info/gp/work2.gnu'**. Also, if you put a **reread** statement into your text file, it causes GnuPlot to go the beginning of the file, and keep looping. For more information, see **help reread**, and for an example, type **load '/opt/Gnu/info/gp/work3.gnu'**. To exit the loop, hit **Control-C** in the GnuPlot window.

Saving Work

The option that GnuPlot gives for saving work is the **save** command. If you type the following:

- **save "mywork.gnu"**

GnuPlot will then create a text file with the name you gave it that includes **set** commands for every possible setting in GnuPlot. You can then load this file to continue work where you left off.