Contents

Part I.
Introduction to gnuplot

1. Introduction to gnuplot

1.1. Basic Use

Since gnuplot has been ported to various operating systems, its usage slightly depends on the platform. Here we describe an introduction to gnuplot for the case of UNIX an X11. Basically its usage is common to those systems, so that this tutorial may be helpful for the other operating systems.
First of all executing gnuplot: Gnuplot displays a banner and credit, then shows a gnuplot command line prompt `gnuplot>` . Gnuplot is a command line driven plotting tool. You give commands here to make your figure.

```
% gnuplot

        G N U P L O T
        Version 4.0 patchlevel 0
        last modified Thu Apr 15 14:44:22 CEST 2004
        System: Linux 2.4.23

        Copyright (C) 1986 - 1993, 1998, 2004
        Thomas Williams, Colin Kelley and many others

        This is gnuplot version 4.0.  Please refer to the documentation
        for command syntax changes.  The old syntax will be accepted
        throughout the 4.0 series, but all save files use the new syntax.

        Type ‘help‘ to access the on-line reference manual.
        The gnuplot FAQ is available from
                http://www.gnuplot.info/faq/

        Send comments and requests for help to
                <gnuplot-info@lists.sourceforge.net>
        Send bugs, suggestions and mods to
                <gnuplot-bugs@lists.sourceforge.net>


Terminal type set to 'x11'
gnuplot>
```

At the `gnuplot>` prompt you can use the following commands:

- commands to quit, read a command file, and save parameters
- commands to plot
- commands to change parameters

- commands to execute the shell
- commands to define a function, substitute variables, and calculate

Actually there are more commands which cannot be categorized into the items above, so that it is hard to explain everything here. See gnuplot online help by help command. Here we explain the simplest way to draw a graph with gnuplot.

1.2. Commands to Quit, Read a Command File, and Save Parameters

`exit` or `quit` command terminates gnuplot. Once you quit gnuplot, all of setting you made will be lost. To `save` the current setting, use save command followed by a file name in which parameters and functions you defined are stored. The file name is quoted by a single or double quotation. The file name is arbitrary, but if the same name exists in the current directory, gnuplot overwrites internal parameters in that file without any warnings.

```
gnuplot> save "savefile.plt"
```

The saved file is a usual text file. You can edit the contents with a text editor. To draw a graph again with this file, use the `load "savefile.plt"` command at the gnuplot command-line, or execute gnuplot and give the data-file name as a command line option.
Inside gnuplot

```
gnuplot> load "savefile.plt"
```

Outside gnuplot (shell command line)

```
% gnuplot savefile.plt
```

The difference of those two methods is: with the `load` command you go back to the `gnuplot>` command prompt after gnuplot read the file, then you enter the usual interactive mode. If you give the data-file name as a command line option (the second case), gnuplot ends after it reads the file, and you come back to shell. This is a batch mode.

1.3. Commands to plot

There are two basic commands to plot a graph: `plot` and `splot`. The former is used for a 2-dimensional graph, and the latter is for a 3-dim. Gnuplot makes a graph of any kinds of functions or numerical data stored in a file. To plot a function, use the `plot/splot` command with a range of X-axis (or X and Y ranges for 3-dim. plot) and the function. You can omit the range parameters. Here is an example of plotting y=sin(x), which you may often see at many gnuplot tutorials.

```
gnuplot> plot sin(x)
```

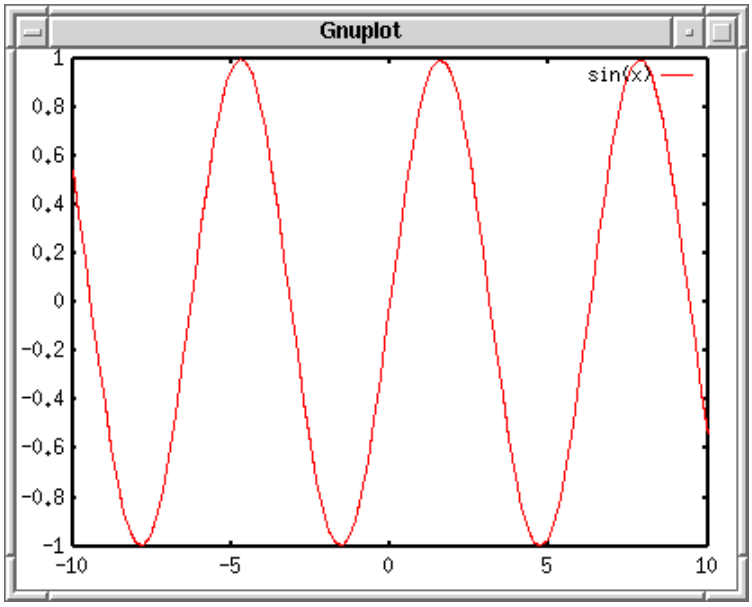

Figure 1: First plot

This is the 2-dimensional graph gnuplot generates. The frame is drawn by a thick line, which is called "border". The X and Y axes have graduation called "major tics", and numeric labels are written at the tics location. The intervals between each major tic can be divided by minor tics. You can draw names of X and Y axes. The X-axis name – "xlabel" – is shown below the x-axis border, while the position of "ylabel" depends on your terminal. If your terminal can rotate letters, the 90-degree rotated ylabel should go to the left of y-axis, otherwise it is shown at the top of y-axis.
If ranges of X and Y axes are not specified, gnuplot determines appropriate values for those automatically. The example above you can see the default X range which is -10 to +10, and the Y range was automatically determined. To set the X range 0 to 5, `[0:5]`.

```
gnuplot> plot [0:5] sin(x)
```

1.4. Commands to change variables and parameters

There are a number of parameters which change your plot appearance. You can change them by the `set` command. See online help.

```
gnuplot> help set
 The ‘set‘ command can be used to sets _lots_ of options.  No screen is
 drawn, however, until a ‘plot‘, ‘splot‘, or ‘replot‘ command is given.
```

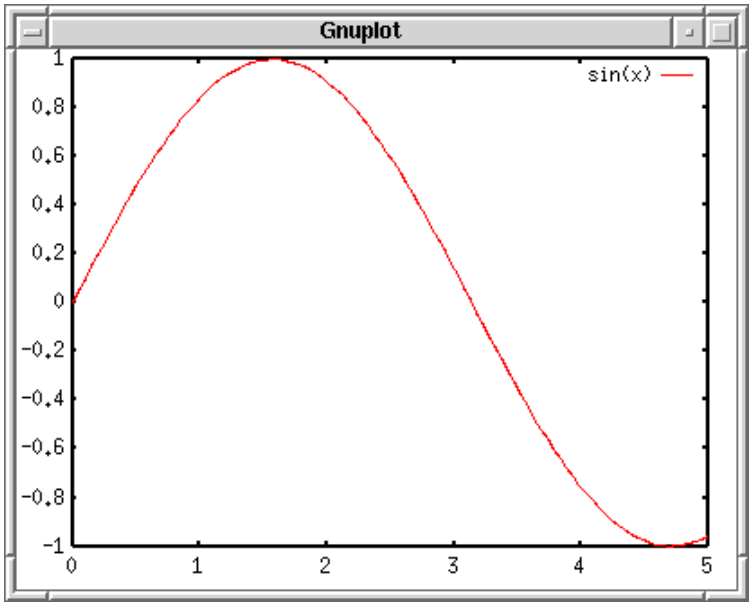

Figure 2: Changing plot interval

```
 The ‘show‘ command shows their settings;  ‘show all‘ shows all the
 settings.

 If a variable contains time/date data, ‘show‘ will display it according to
 the format currently defined by ‘set timefmt‘, even if that was not in effect
 when the variable was initially defined.

Subtopics available for set:
      angles          arrow           autoscale       bar
      bmargin         border          boxwidth        clabel
      clip            cntrparam       contour         data
      dgrid3d         dummy           encoding        format
.....
      zero            zeroaxis        zlabel          zmtics
      zrange          ztics
```

Here are several examples to change the parameters. Firstly insert some text into the xlabel and ylabel. The text should be quoted by a single or double quotation. Next, specify the range of X and Y axes. As explained above the X range can be changed if you specify that at plotting. Alternatively you can change them by the "xrange" and "yrange" parameters.

```
gnuplot> set xlabel "X-AXIS"
```

```
gnuplot> set ylabel "Y-AXIS"
gnuplot> set xrange [0:5]
gnuplot> set yrange [-2:2]
gnuplot> plot sin(x)
```

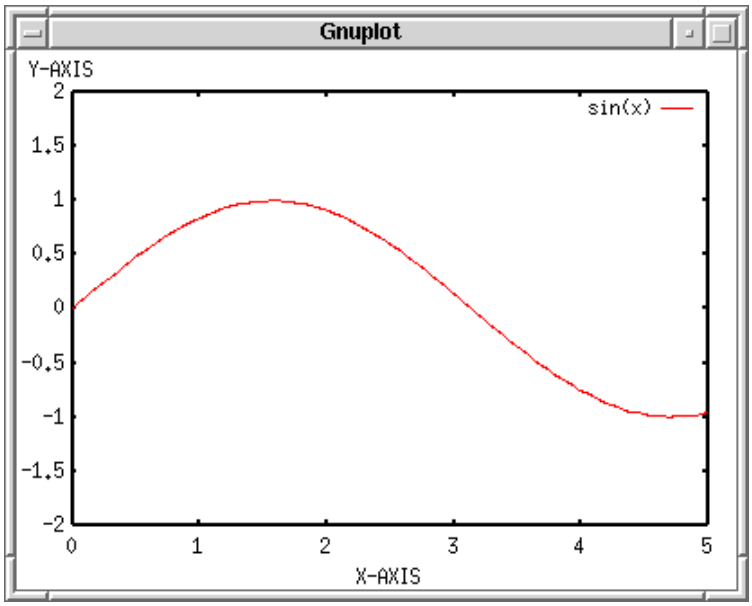

Figure 3: Changing plot interval

If a terminal is not specified, gnuplot makes a graph on your screen. The `set terminal` command changes the destination of your plot into a postscript file or printer, etc. Gnuplot produces a variety of graph by means of various drivers, so that it is independent of the platforms (but quality of the drawing still depends of the terminal). The terminals which your gnuplot can handle can be displayed by the `set terminal` command.

```
gnuplot> set terminal

Available terminal types:
          unknown  Unknown terminal type - not a plotting device
            table  Dump ASCII table of X Y [Z] values to output
            linux  Linux PC with (s)vgalib
              ....
             tpic  TPIC -- LaTeX picture environment with tpic \specials
         pstricks  LaTeX picture environment with PSTricks macros
          texdraw  LaTeX texdraw environment
               mf  Metafont plotting standard
```

```
gnuplot> set terminal postscript
Terminal type set to 'postscript'
Options are 'landscape noenhanced monochrome dashed defaultplex "Helvetica" 14'
```

Gnuplot produces a graph in a Postscript format when `set terminal postscript` command is given. If an output direction is not specified, the produced Postscript data flow on your screen. The `set output` command changes the destination of output.

```
gnuplot> set output "plot.ps"
gnuplot> plot sin(x)
```

1.5. Commands about Shell

You can escape to an interactive shell temporary in which any shell commands can be used. To spawn a shell, use `shell` command. To return to gnuplot, use `exit`. A single shell command can be executed with the ! character at the beginning of the command, like `! ls -a`.
Gnuplot supports 'pwd' and 'cd' commands with which you can display your working directory, or change the directory. The working directory is your current directory when gnuplot is invoked. To change the current directory, `cd "../other/dir"` . You need a quotation mark.

1.6. Commands for Substitution, Function Definition, Calculations, etc.

You can use gnuplot as a simple calculator. To substitute a value into a variable, just type "variable = value" at the gnuplot command line. To see the value of variable, use print command.

```
gnuplot> a=10
gnuplot> print a
10
```

"Variable = expression" substitutes a calculated value into the variable. Double precision is used for the calculation except for integer.

```
gnuplot> a=1+2*sqrt(3)
gnuplot> print log(a)
1.49606798806764
```

The defined variable can be used for the `plot` command. Gnuplot holds the circular constant in "pi". To draw a graph of a*sin(x) from -2pi to +2pi, where a=0.5:

```
gnuplot> set xrange [-2*pi:2*pi]
gnuplot> a=0.5
gnuplot> plot a*sin(x)
```

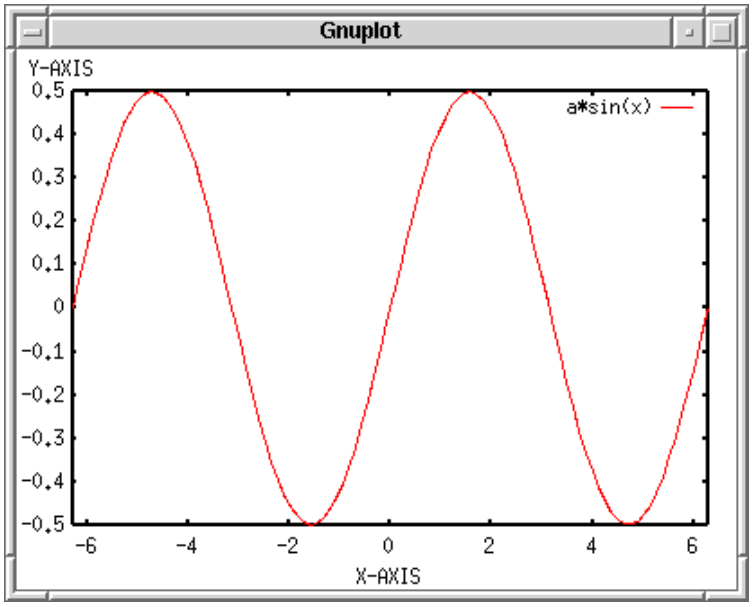

Figure 4: Variables in function

You can define your own function. For example, to make a function of f(x)=sin(x)*cos(x), it is defined as:

```
gnuplot> f(x)=sin(x)*cos(x)
```

The function defined above can be referred to as "f(x)". You can also include a user-defined variable in your function.

```
gnuplot> f(x)=a*sin(x)*cos(x)
```

This function contains a variable "a" which is defined by user, and the function varies with this parameter.

2. Use for a Numerical Calculation - Plots of Calculated Results

2.1. Preparation of Data File

Here is an example for plotting of numerical data which are generated by numerical calculations. To do it, the calculated results should be stored in a data-file that is a usual text file. The following programs calculate a Pade approximation of y=exp(-x). In C:

```
#include <stdio.h>
#include <math.h>

int main(void);
int main(void)
{
 int i;
 double x,y,z1,z2,d;

 d = 0.1;
 x = 0.0;
 for(i=0;i<50;i++){
   x += d;
   y  = exp(-x);
   z1 = (6 - 2*x)/(6 + 4*x + x*x);
   z2 = (6 - 4*x + x*x)/(6 + 2*x);
   printf("% 6.2f % 11.4e % 11.4e % 11.4e\n",
              x,y,z1,z2);
 }
 return 0;
}
```

In Fortran:

```
      INTEGER I
      REAL*8  X,Y,Z1,Z2,D
      D = 0.1
      X = 0.0
      DO 10 I=1,50
          X = X+D;
          Y  = EXP(-X)
          Z1 = (6 - 2*X)/(6 + 4*X + X*X)
          Z2 = (6 - 4*X + X*X)/(6 + 2*X)
          WRITE(6,20) X,Y,Z1,Z2
   10 CONTINUE
   20 FORMAT(F6.2,3(1X,1PE11.4))
```

```
      STOP
      END
```

Those programs give the following output. The first column is X-coordinate, the second is a direct calculation of EXP(-X), the third and forth columns are the Pade approximation with different orders. As you can see, this approximation is only valid for small X values.

```
  0.10  9.0484E-01  9.0484E-01  9.0484E-01
  0.20  8.1873E-01  8.1871E-01  8.1875E-01
  0.30  7.4082E-01  7.4074E-01  7.4091E-01
  0.40  6.7032E-01  6.7010E-01  6.7059E-01
  0.50  6.0653E-01  6.0606E-01  6.0714E-01
                    ....
  4.60  1.0052E-02 -7.0237E-02  5.7632E-01
  4.70  9.0953E-03 -7.2510E-02  6.0325E-01
  4.80  8.2297E-03 -7.4627E-02  6.3077E-01
  4.90  7.4466E-03 -7.6597E-02  6.5886E-01
  5.00  6.7379E-03 -7.8431E-02  6.8750E-01
```

2.2. Plot (X,Y) Pairs

The data file "output.dat" contains the calculated values above. Each X-value has 3 different Y values in this case. To draw the second column (calculated EXP(-X) values) as a function of the first column, use `using` keyword at plotting.

```
gnuplot> plot "output.dat" using 1:2 with lines
```

A style of graph is specified by the `with` keyword. In the example above, `lines` style is given, with which each data point is connected by small lines. There are several kinds of line-styles those are numbered by 1, 2, 3... To change the line style, `with lines` is followed by the line-style number. If the number is not given, gnuplot assigns numbers automatically from 1 to a certain maximal number.
There are several plot-styles — draw symbols, connect with lines, connect with step-like lines, draw bars, etc.

2.3. Draw Several Lines

Next, the third and fourth columns are plotted on the same figure. To draw several lines simultaneously, repeat the data specification like – `plot "A" using 1:2 with line, "B" using 1:2 with points, ...` Sometimes such a command line becomes very long. If a line is ended with a character '**'* the next line is regarded as a continuous line. Don't put any letters after the back-slash.

```
gnuplot> plot "output.dat" using 1:2 with lines, \
> "output.dat" using 1:3 with lines,\
> "output.dat" using 1:4 with lines
```

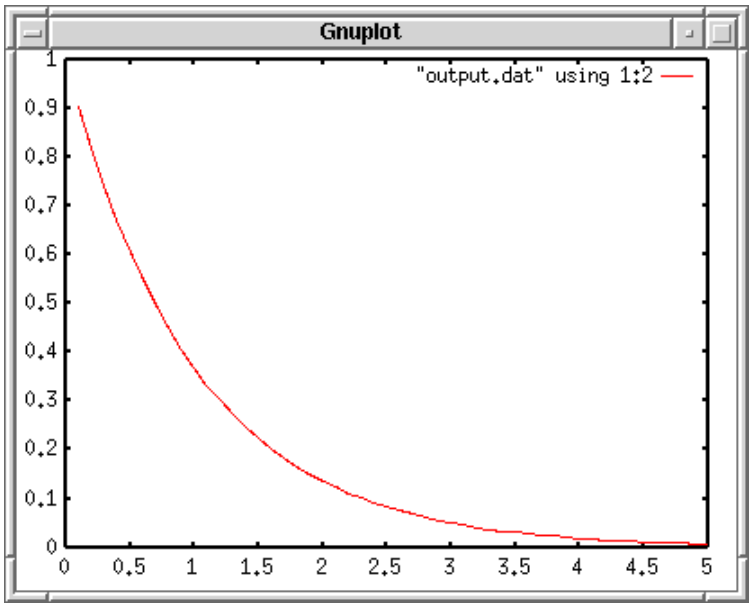

Figure 5: Pade approximation

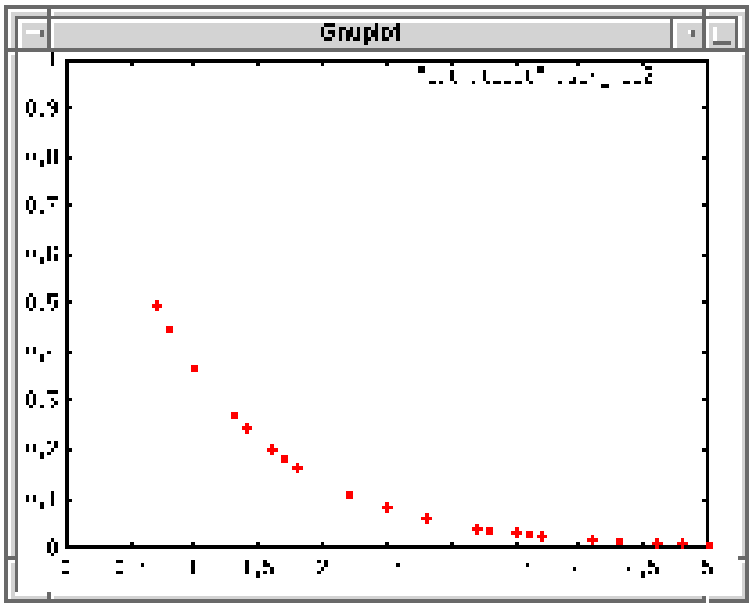

Figure 6: Pade approximation with points

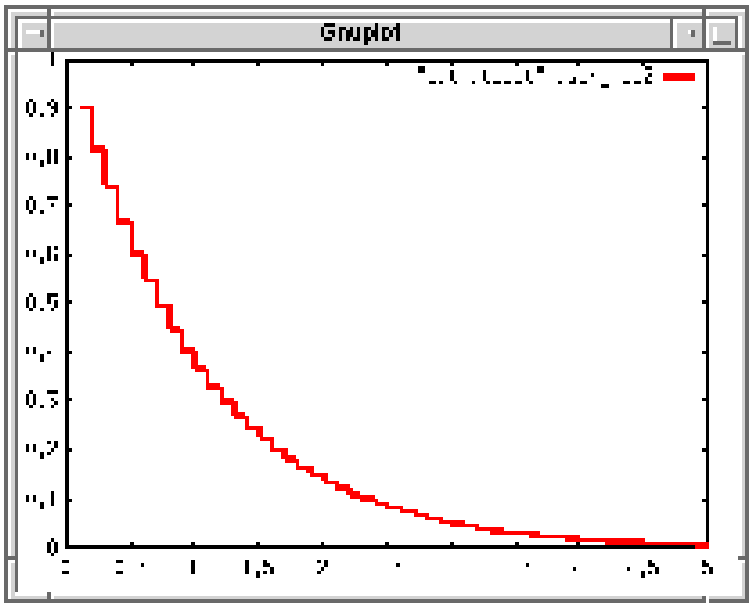

Figure 7: Pade approximation with steps

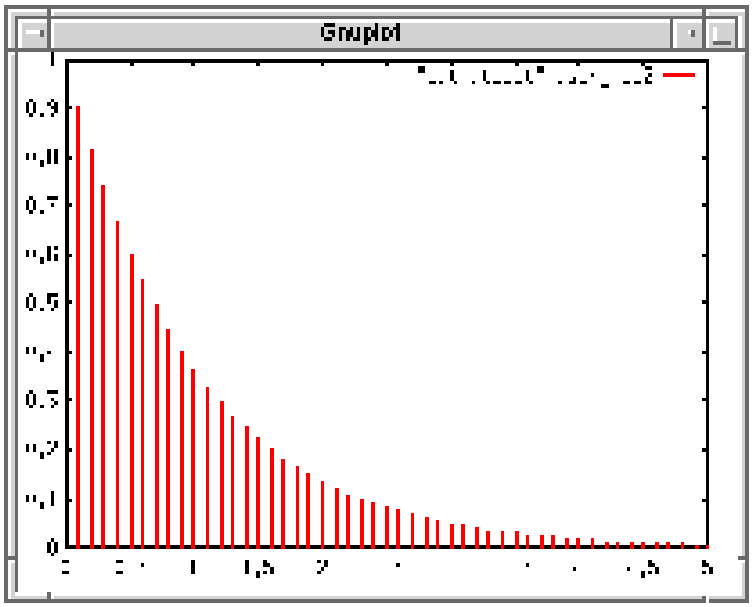

Figure 8: Pade approximation with impulses

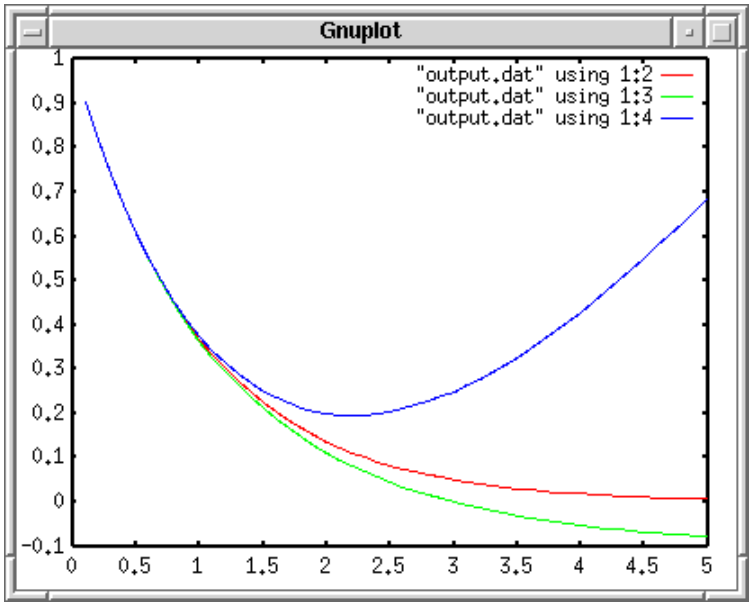

Figure 9: plotcalc2

Different functions in one graph The second line shown by green color is negative at the large X values, so that the Y-axis scale was enlarged to -0.1
In the figure legend, the data-file name and the column numbers those were used for plotting were indicated. The red line is obtained by an analytical function, so let's change the first line of the legend into "Analytical". The next green line is a result of Pade approximation with L=1 and M=2, so that "L=1, M=2" should be displayed. The blue line is also the result of "L=2, M=2" Pade approximation.

```
gnuplot> plot "output.dat" using 1:2 title "Analytical" with lines, \
> "output.dat" using 1:3 title "L=1, M=2" with lines,\
> "output.dat" using 1:4 title "L=2, M=1" with lines
```

Adjusting labels of functions

2.4. Insert a Graph Title and Axis Names

Now, let's insert X and Y axis names. The name of X axis is just "x", while the Y axis is "y=exp(-x)". To set those names, use the `set xlabel` and `set ylabel` commands. In addition, you can put a title of this figure, "Pade approximation", by the `set title` command. With the `replot` command, you can omit a long plot command.

```
gnuplot> set xlabel "x"
gnuplot> set ylabel "y=exp(-x)"
gnuplot> set title "Pade approximation"
```

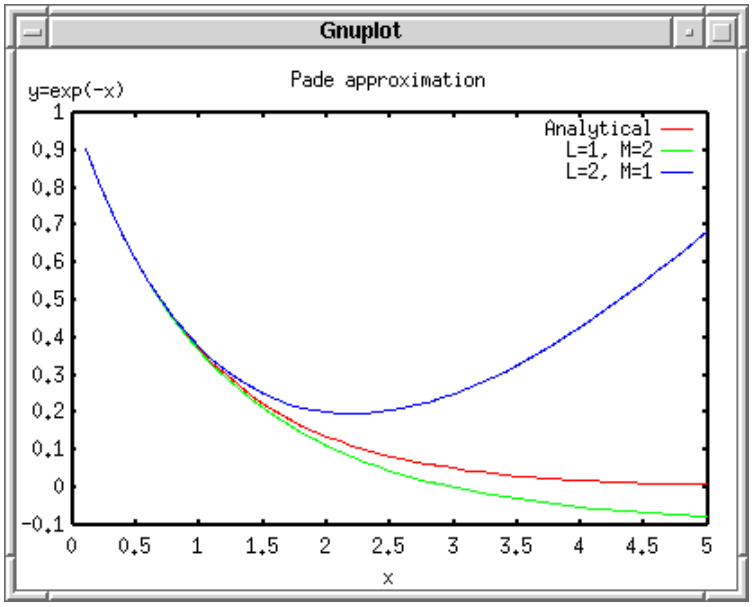

Figure 10: plotcalc4

```
gnuplot> replot
```

The figure size is changed when the title and axis names are given. Gnuplot calculates the figure size automatically so as to fit everything in a screen, then the graph itself becomes smaller when the axis names or figure title are given.
The Y axis name "y=exp(-x)" does not go to the left side of the Y-axis, but it is displayed at the top. This is because gnuplot cannot rotate a text when the X Window system is used. With the Postscript terminal you can get a rotated Y-axis name which locates at an appropriate position. If your gnuplot is newer than ver.3.8, the Y-axis name should be rotated on your screen too.

2.5. Change the X and Y Axes Range

Now let's change the X and Y ranges. Set the Y-range [0,1], and the X-range [0,2].

```
gnuplot> set xrange [0:2]
gnuplot> set yrange [0:1]
gnuplot> replot
```

2.6. Put Graduations

The graduations on the X-axis starts with 0, and the interval is 0.5. To change the graduations (tics), use `set x|ytics` The tics can be controlled by three optional

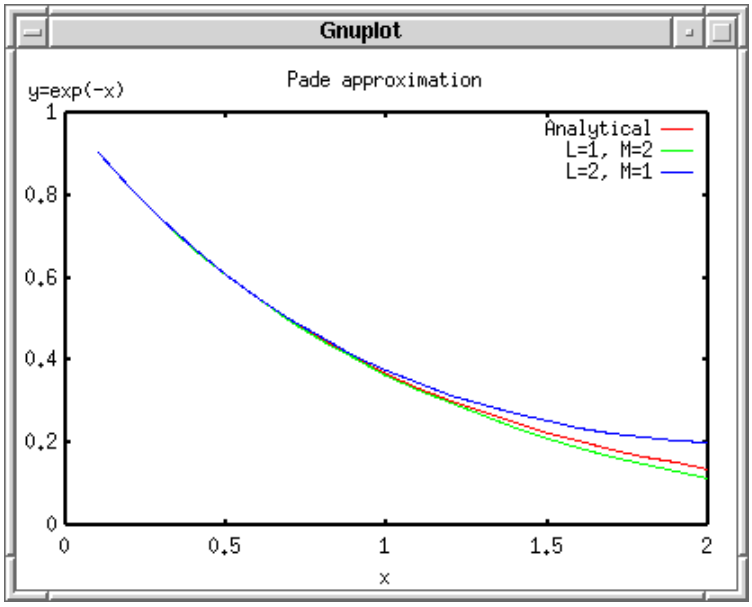

Figure 11: plotcalc5

numbers. If there is a one number after the `set tics` command, like `set xtics 10` , this number "10" is regarded as an increment. If there are two figures, the first one is the initial value, and the second is the increment. If three, the last one becomes the final value.
You can also divide the interval which is given as an increment by the `set tics` command, and draw small tics inside the interval with the `set mx|ytics n` command, where n is the number of division.

```
gnuplot> set xtics 1
gnuplot> set mxtics 5
gnuplot> set ytics 0.5
gnuplot> set mytics 5
gnuplot> replot
```

2.7. Done!

Then we make the graph in a Postscript format, and print it out. Firstly change the terminal into "postscript", and specify a name of output file. Before quit gnuplot, save your parameters and other setup into a file (output.plt)

```
gnuplot> set term postscript
gnuplot> set output "output.ps"
gnuplot> replot
```

```
gnuplot> save "output.plt"
gnuplot> quit
```

The file 'output.ps' can be browsed with a Postscript viewer like 'gv' or 'ghostview', and you can print this file with a Postscript printer. The following image is our Postscript data 'output.ps' displayed with ghostview. When you plotted the graph

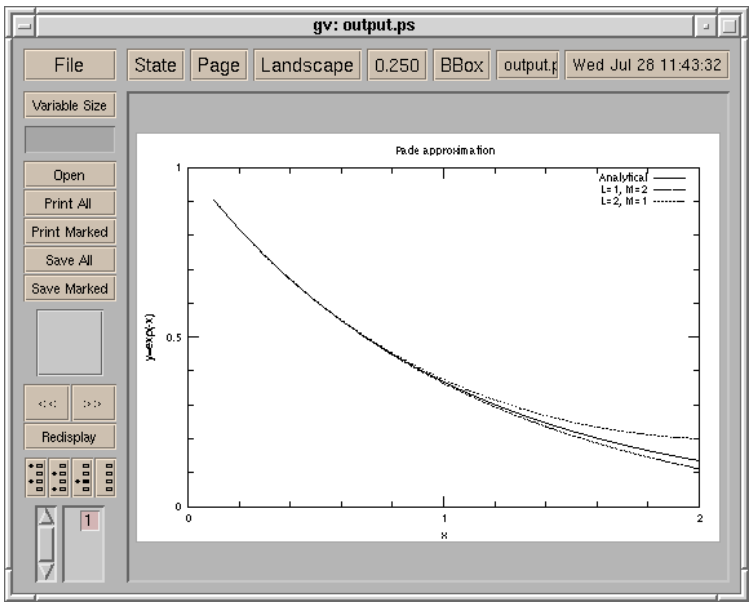

Figure 12: Postscript plotting

on your screen, the first line was a red line. However it turned into the solid line here. The blue and green lines became the dashed lines. (Though it is hard to see them in a small size image.) As you can see above, line and symbol types depend on the terminal you use. In order to know which line number corresponds to the solid, dashed, dotted, etc. try `test`. For example, on the X window system, you get:

```
gnuplot> set term x11
gnuplot> test
```

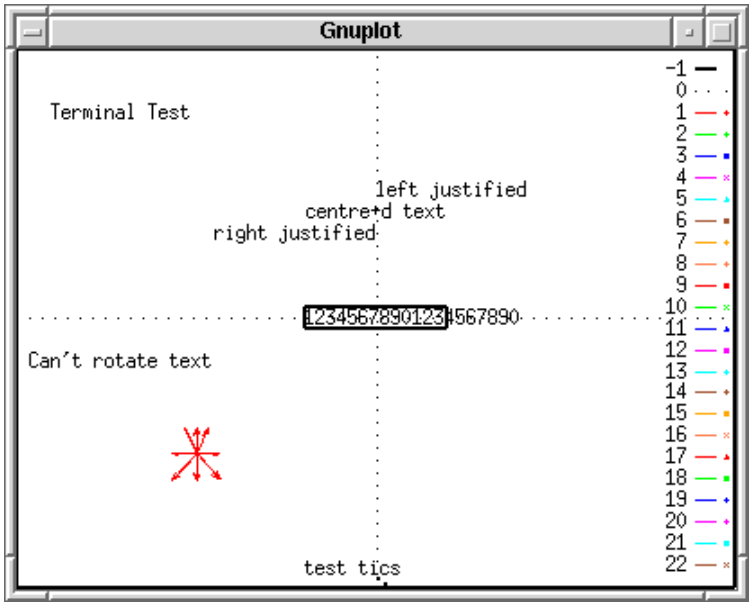

Figure 13: Testing terminal capabilities

3. Use for Experimental Data - Plots of Experimental Data

3.1. Preparation of Data File

Here we describe how to plot experimental data in a data file. There are three data sets in our example, and those data are written sequentially in the file. Each data block is separated by 2 blank lines. The experimental data point (X,Y,Z) is energy points, measured values, and uncertainties in Y. The uncertainties are given as absolute errors, in other words, Z has the same dimension as Y values. A line which begins with "#" is regarded as a comment line, and ignored.

```
2.1500E-02 1.3060E+00 5.3098E-02
2.3900E-02 1.2220E+00 4.7043E-02
2.6800E-02 1.3430E+00 4.9854E-02
2.9700E-02 1.2580E+00 4.5860E-02
3.2500E-02 1.2430E+00 4.4506E-02
              ....
9.4500E-01 1.2290E+00 3.7317E-02
1.0350E+00 1.2630E+00 4.1449E-02
1.1330E+00 1.2670E+00 4.2289E-02
```

```
2.4000E-02 1.2970E+00 3.1387E-02
4.0000E-02 1.3060E+00 2.8993E-02
6.0000E-02 1.2960E+00 2.8382E-02
8.0000E-02 1.3300E+00 2.8728E-02
              ....
7.0000E+00 1.2210E+00 2.5031E-02
7.2000E+00 1.1990E+00 2.5299E-02
7.4000E+00 1.1860E+00 2.5618E-02


2.2500E-02 1.3310E+00 3.4606E-02
2.7500E-02 1.3370E+00 2.4066E-02
3.5000E-02 1.3440E+00 2.6880E-02
              ....
1.8936E+01 1.0080E+00 2.9232E-02
2.0064E+01 9.6300E-01 2.9853E-02
2.1296E+01 1.0310E+00 3.1961E-02
```

3.2. Plot Data

Those three experimental data sets can be approximated by y = -0.01687 x + 1.3512. This function is plotted with the data which are stored in a file, plotexp.dat. To access each data block in the datafile, use the `index` keyword to specify the block to be drawn. You can specify the first block by `index 0:0` or `index 0`. The third block is pointed by `index 2`. To combine the first and second blocks, use `index 0:1`.
To plot data with error bars, use `with yerrorbars`. This requires the error data at the third column in the file, and the columns are specified by `using 1:2:3` . If an error is given in percent (%), `using 1:2:(2*3/100.0)` converts them into an absolute error.

```
gnuplot> plot "plotexp.dat" index 0:0 using 1:2:3 with yerrorbars,\
> "plotexp.dat" index 1:1 using 1:2:3 with yerrorbars,\
> "plotexp.dat" index 2:2 using 1:2:3 with yerrorbars
```

When your command line is too long, put "ät the end of line, then the next line is treated as a continued line. Gnuplot recognizes short keywords, for example "w" is "with", "i" is "index", and so on. In addition, you can omit a file name if the same file is used, just like the example above. In the next two lines after the first "plot" command line, a short form "" can be used instead of "plotexp.dat".

3.3. Make a Legend

Each experimental data set was measured by three different experimentalists. Their names are, "A. Smith" for the fist one, "B. Smith" for the second, and "C. Smith" for the third, and they carried out the experiments in 1992, 1993, and 1999.

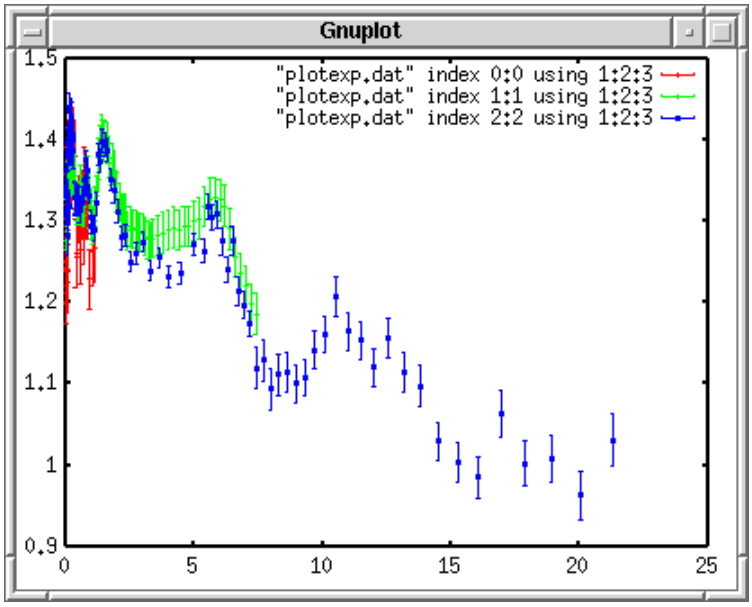

Figure 14: plotexp1

```
gnuplot> plot "plotexp.dat" ind 0:0 usi 1:2:3 ti "A. Smith (1992)" w yerr,\
> "plotexp.dat" ind 1:1 usi 1:2:3 ti "B. Smith (1993)" w yerr,\
> "plotexp.dat" ind 2:2 usi 1:2:3 ti "C. Smith (1999)" w yerr
```

3.4. Draw a Calculated Line

To overlay a function and experimental data, the fitted linear function, f(x) = -0.01687*x +1.3512, is defined first.

```
gnuplot> f(x)= -0.01687*x + 1.3512
gnuplot> plot f(x) with lines, \
> "plotexp.dat" ind 0:0 usi 1:2:3 ti "A. Smith (1992)" w yerr,\
> "plotexp.dat" ind 1:1 usi 1:2:3 ti "B. Smith (1993)" w yerr,\
> "plotexp.dat" ind 2:2 usi 1:2:3 ti "C. Smith (1999)" w yerr
```

In the above case, color number used for each measurement is shifted because the linear function was inserted at the top.

3.5. Change the Line Style

Those experimental data are distinguishable by colors when the graph is on your screen. However, to convert it into Postscript you may see a strange symbol as below. In the Postscript terminal the line number 1 is solid, but the number 2,3, and 4 are

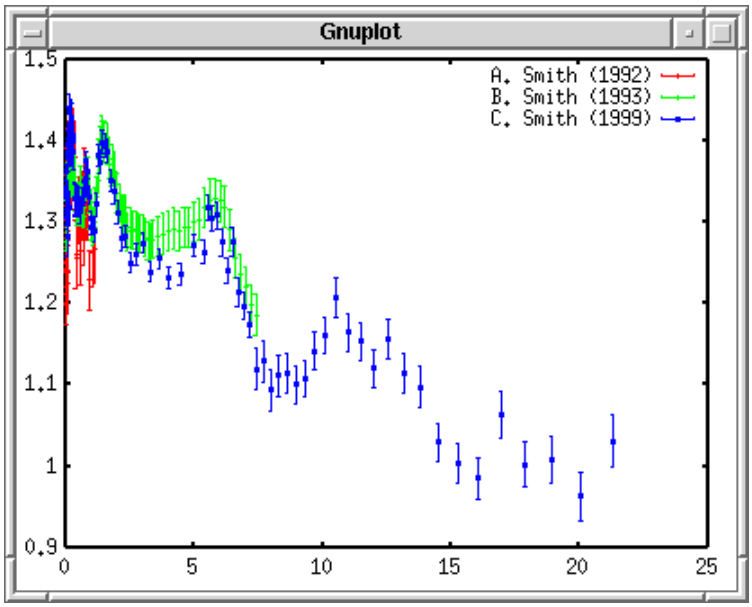

Figure 15: plotexp2

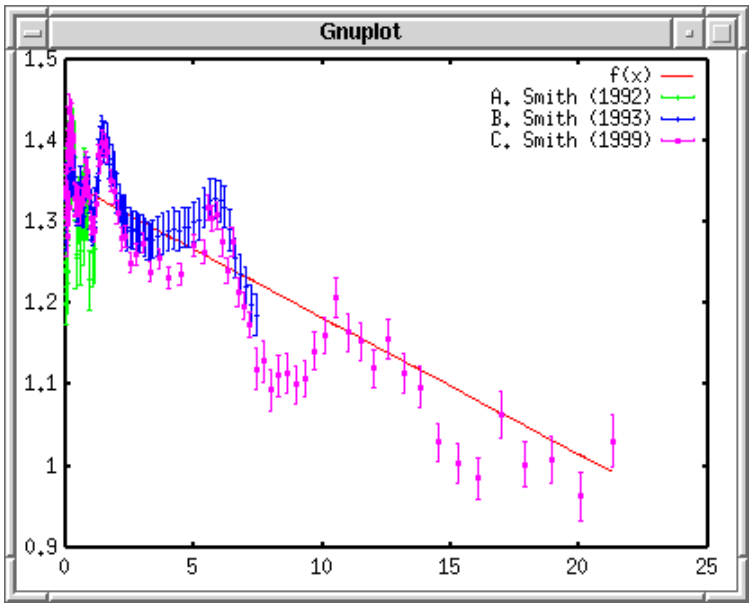

Figure 16: plotexp3

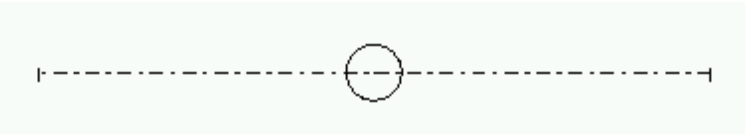

Figure 17: errorbar

dashed, dotted lines. So that you get a dashed error bar ! I suppose nobody likes such a funny symbol. Let's define all the lines solid, and each measurement has a distinct symbol.

```
gnuplot> set linestyle 1 lt 1 lw 3
gnuplot> set linestyle 2 lt 1 pt 7
gnuplot> set linestyle 3 lt 1 pt 8
gnuplot> set linestyle 4 lt 1 pt 9
```

If your gnuplot is newer than ver.3.8:

```
gnuplot> set style line 1 lt 1 lw 3
gnuplot> set style line 2 lt 1 pt 7
gnuplot> set style line 3 lt 1 pt 8
gnuplot> set style line 4 lt 1 pt 9
```

The first line defines the linestyle No.1 as the solid line with width of 3. The second to fourth lines define the linestyles those are used for experimental data. The line kind is solid, but the symbols of No.7, 8, and 9 are used.

```
gnuplot> f(x)= -0.01687*x + 1.3512
gnuplot> plot f(x) notitle with lines linestyle 1, \
> "plotexp.dat" ind 0:0 usi 1:2:3 ti "A. Smith (1992)" w yerr linestyle 2,\
> "plotexp.dat" ind 1:1 usi 1:2:3 ti "B. Smith (1993)" w yerr linestyle 3,\
> "plotexp.dat" ind 2:2 usi 1:2:3 ti "C. Smith (1999)" w yerr linestyle 4
```

3.6. Insert Axis Names

In the above graph, a legend for the fitted line was removed with the `notitle` keyword. Although it is difficult to see this figure since all data points and the line is red, this inconvenience will disappear when it is printed out.
Now, make labels for the X and Y axes. The name of X axis is "Energy [MeV]", while the Y axis is "Cross Section [b]". To set those names, use `set xlabel` and `set ylabel` . The `replot` command invokes the "plot" command you typed before, so that you don't need to type the long command again.

```
gnuplot> set xlabel "Energy [MeV]"
gnuplot> set ylabel "Cross Section [b]"
gnuplot> replot
```

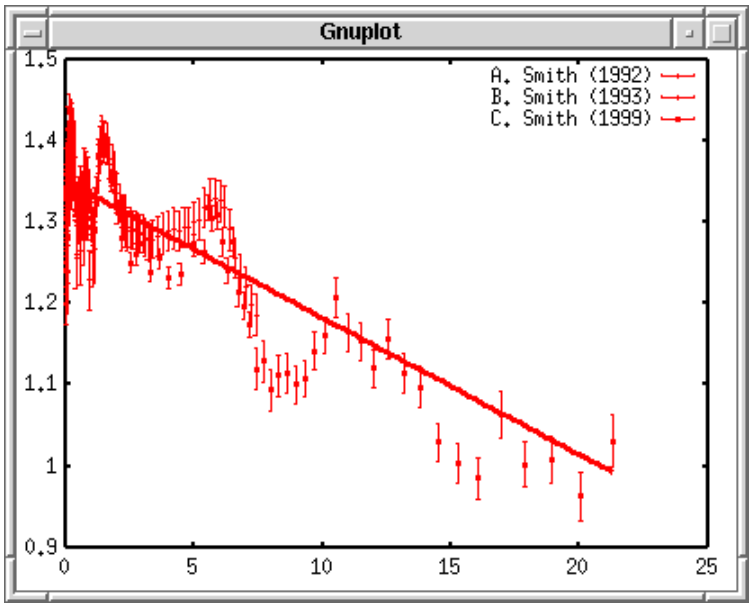

Figure 18: plotexp4

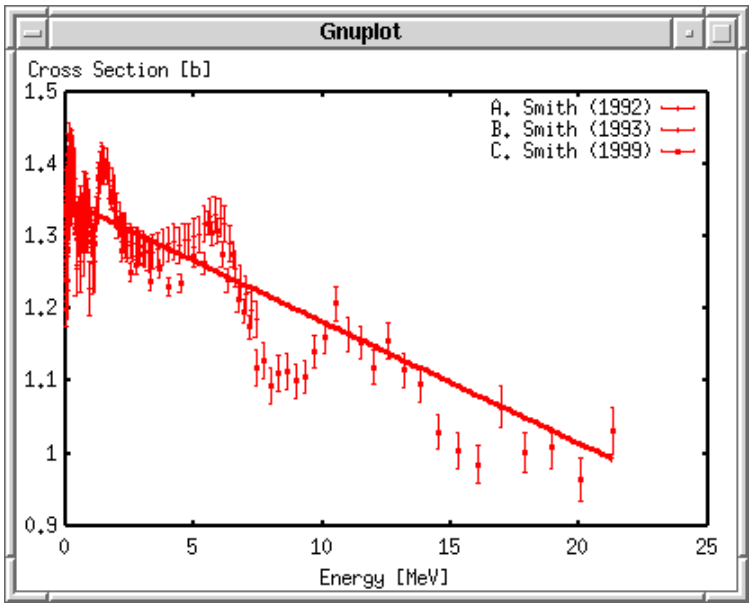

Figure 19: plotexp5

3.7. Adjust the Range

Then, adjust the ranges for X and Y axes. For Y axis, setting the minimal value of 0 and the maximal of 2 works fine. There are too many measured data points near X=0. We take logarithm to magnify there. The least X value is set to 0.01, while the largest value is 20. The `set logscale x|y` command controls the logarithm.

```
gnuplot> set xrange [0.01:20]
gnuplot> set yrange [0:2]
gnuplot> set logscale x
gnuplot> replot
```

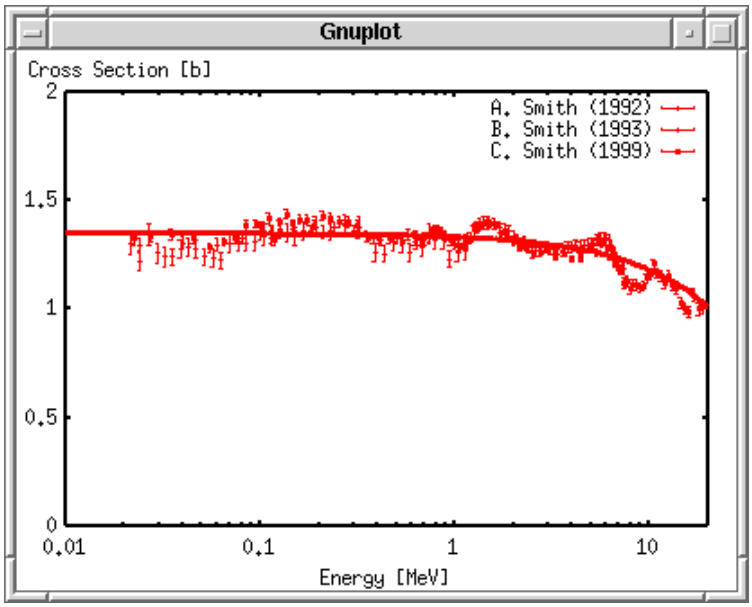

Figure 20: plotexp6

3.8. Put Graduations

Finally, put graduations on the axes. The X axis is log, so you can leave it. The Y axis has a tic with an interval of 0.5. Let's set the interval 1, and divide it by 10. In addition, we draw a grid in the graph. The grid is shown at the major tics where you see figures.

```
gnuplot> set ytics 1
gnuplot> set mytics 10
gnuplot> set grid
gnuplot> replot
```

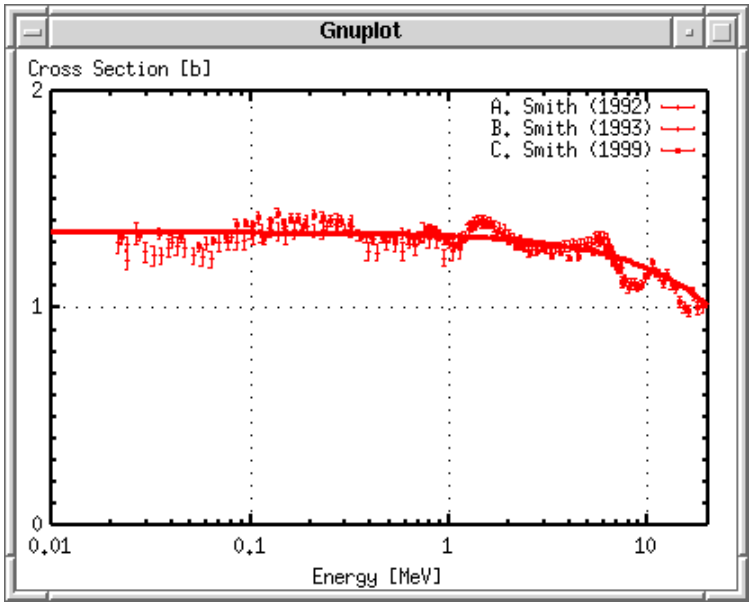

Figure 21: plotexp7

3.9. Generate a Postscript File

Now we have completed. With the Postscript driver, specify an output file name, and `replot`, you get a Postscript graph. Before quit gnuplot, save everything in a file

```
gnuplot> set term postscript
gnuplot> set output "plotexp.ps"
gnuplot> replot
gnuplot> save "plotexp.plt"
gnuplot> quit
```

A Postscript printer can generate a printed graph. Postscript browsers like ghostscript or gv, can display the content. The image below was made with gv. It is not clear to see because the image size is small. Anyway, the No.7 symbol is the filled circle, No.8 is the open triangle, and No.9 is the filled triangle. The kind of symbols and lines depends on the terminal. There are a number of symbols when a Postscript terminal is used, but a limited number of those are practically used for a plot of experimental data, those are circle, triangle, square, and so on (both filled and open). Here is the style numbers of those symbols.

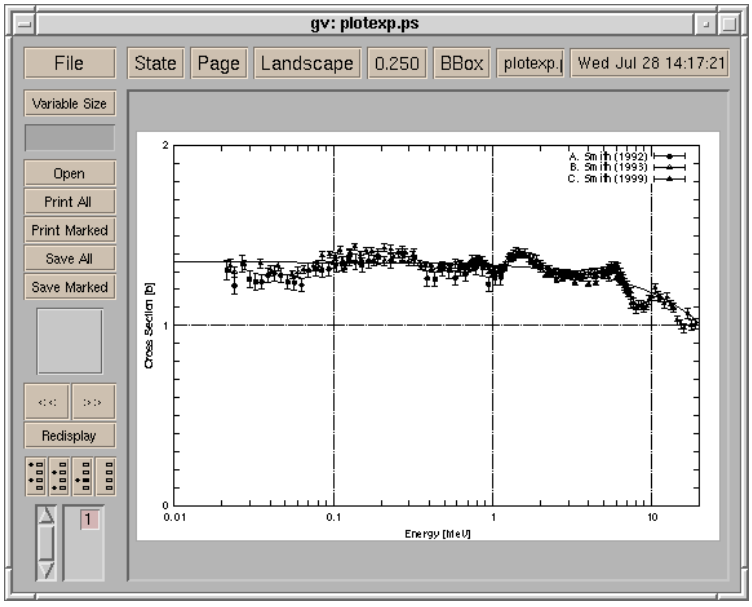

Figure 22: plotexp8

1 +
2 cross
3 *
4 (o) square
5 (f) square
6 (o) circle
7 (f) circle
8 (o) triangle
9 (f) triangle
10 (o) inverse-triangle
11 (f) inverse-triangle
12 (o) diamond
13 (f) diamond

4. Functions

4.1. User-Defined Functions

If you can write an equation, gnuplot can calculate it. I mean, functions to be plotted are expressed in a simple functional form, for example, f(x)=a*x+b, and they do not contain a complicated integration / differentiation which needs a numerical calculation. Gnuplot has many basic functions such as sine, cosine, Bessel, Gamma, etc, and it can plot them as well as combinations of those functions.
Maybe gnuplot users are not so interested in plotting functions, except for your Math homework (or some special purpose which I don't know), but this is extremely useful for a parameter-fitting to experimental data, namely a least-squares fitting. Gnuplot can solve not only linear least-squares problems but also non-linear ones. In this section, we learn how we can use a user-defined function, and how to fit this to experimental data.
The function we are thinking here is the Lorentzian plus background term of $\frac{1}{\sqrt{(x)}}$. This equation contains four parameters, a, b, c, and d, those are determined by experimental data.

$$f(x) = \frac{c}{(x-a)^2+b} + \frac{d}{\sqrt{x}}$$

Definition of a function is similar to Fortran or C programming language. In the example below, $f(x) = \frac{c}{((x-a)*(x-a)+b)} + \frac{d}{\sqrt{(x)}}$ defines our equation. Square of (x-a) term also can be written in a Fortran manner, (x-a)**2. The parameters, a,b,c, and d are arbitrary.

```
gnuplot> a=0.25
gnuplot> b=0.02
gnuplot> c=0.05
gnuplot> d=0.1
gnuplot> f(x)=c/((x-a)*(x-a)+b)+d/sqrt(x)
gnuplot> set xrange [0:1]
gnuplot> set yrange [0:4]
gnuplot> plot f(x)
```

4.2. Values of the function

Since gnuplot calculates user-defined functions numerically to draw a graph on your screen, you can look at the calculated numbers with the `print` command. The value of function depends on the parameters (a,b,c,d, and e). Gnuplot calculates functions in a double-precision.

```
gnuplot> print f(0.25)
2.7
```

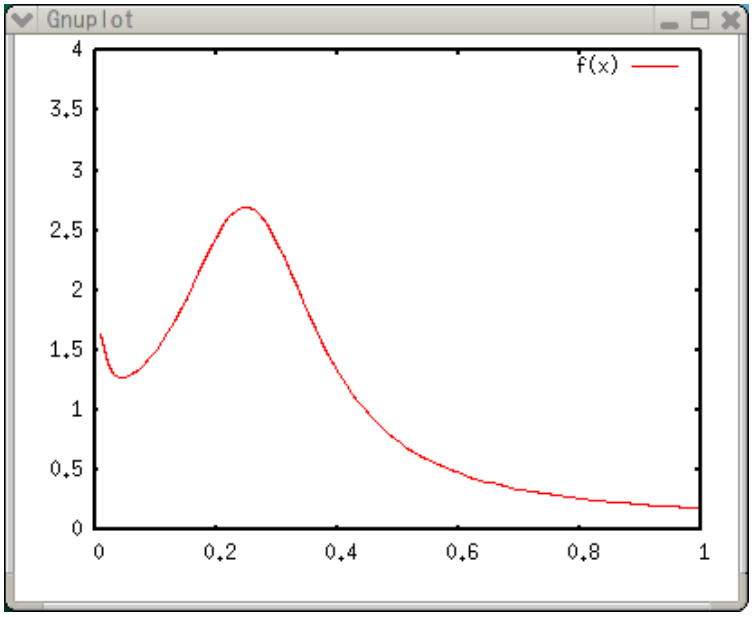

Figure 23: plotfunc1

```
gnuplot> print f(0.4)
1.33458447124371
gnuplot> a=0.4
gnuplot> print f(0.4)
2.65811388300842
```

To obtain the calculated values in a tabulated format, which can be processed with other applications such as a spread-sheet, use a special terminal table . To write the result, specify a file name with `set output` .

```
gnuplot> set term table
Terminal type set to 'table'
gnuplot> plot f(x)
#Curve 0, 100 points
#x y type
0 0 u
0.010101 1.63972 i
0.020202 1.39031 i
0.030303 1.30688 i
   ....

0.979798 0.191506 i
0.989899 0.188622 i
```

```
1 0.185837 i

gnuplot> set output "calc.plt"
gnuplot> replot
```

4.3. Search for the parameters with a Least-Squares Method

We now fit the function to experimental data, and obtain best values of the parameters, a, b, c, and d. The experimental data are stored in a file "exp.dat".

```
 2.5000E-03 3.0420E+00 6.47E-01
 3.5000E-03 2.5700E+00 4.37E-01
 4.5000E-03 2.3020E+00 2.53E-01
   ...

 7.0000E-01 2.7420E-01 2.14E-03
 7.5000E-01 2.5680E-01 1.81E-03
 8.0000E-01 2.4630E-01 1.59E-03
```

The data file consists of 3 columns, each of which is (x,y,z) data pair, the Z-data stand for an absolute uncertainty of the data Y. It means Z has the same dimension as Y. For example (in the case above), if your Y value is 3.04 cm, its uncertainty should be 0.647cm. The inverse of Z becomes a weight of this data point at the data fitting. If no uncertainty is given, the weights for all data points are the same.
First of all, we show the data and function in one plot. As shown in Experimental Data or Numerical Calculation section, the axis names, ranges, tics, are adjusted properly.

```
gnuplot> set xlabel "Energy [MeV]"
gnuplot> set ylabel "Cross Section [b]"
gnuplot> set xtics 0.1
gnuplot> set ytics 0.5
gnuplot> plot f(x) title "Lorentzian",\
> "exp.dat" using 1:2:3 title "experiment" with yerrors
```

As I said that the parameters, a,b,c, and d, are arbitrary, but they were roughly determined for this experimental data. The parameter 'a' is a position of Lorentzian peak, and which is, say, about 0.25 from this plotting. For the parameter 'b', its square-root corresponds to the width of this peak, and I estimated that b is about 0.02.
It is quite easy to do a least-squares fitting with gnuplot. Just use the fit command, and add parameters to be searched for by the via option. When your fitting function is strongly non-linear, you need to be careful about the initial values of your parameters. Here we used the values above as the initial parameters for this fitting.

```
gnuplot> fit f(x) "exp.dat" using 1:2:3 via a,b,c,d
```

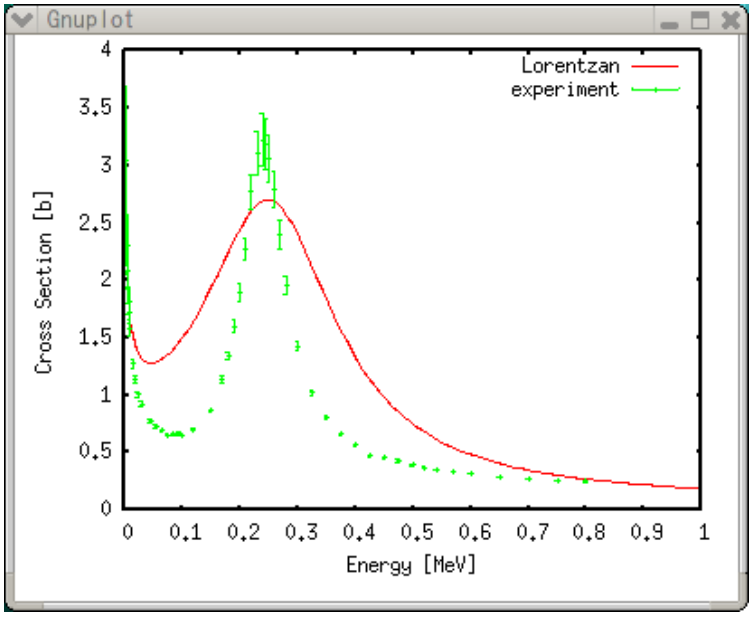

Figure 24: plotfunc2

```
Iteration 0
WSSR        : 96618.1           delta(WSSR)/WSSR   : 0
delta(WSSR) : 0                 limit for stopping : 1e-05
lambda     : 1150.73

initial set of free parameter values

    ...

After 17 iterations the fit converged.
final sum of squares of residuals : 3341.93
rel. change during last iteration : -5.29173e-06

degrees of freedom (ndf) : 47
rms of residuals      (stdfit) = sqrt(WSSR/ndf)      : 8.43237
variance of residuals (reduced chisquare) = WSSR/ndf : 71.1049

Final set of parameters            Asymptotic Standard Error
=======================            ==========================

a                = 0.26191         +/- 0.005759     (2.199%)
b                = 0.00251445      +/- 0.0008358    (33.24%)
```

```
c               = 0.00541346       +/- 0.0009206    (17.01%)
d               = 0.182469         +/- 0.007329     (4.016%)


correlation matrix of the fit parameters:

               a      b      c      d
a               1.000
b               0.042  1.000
c              -0.229  0.783  1.000
d               0.210 -0.538 -0.768  1.000
gnuplot> replot
```

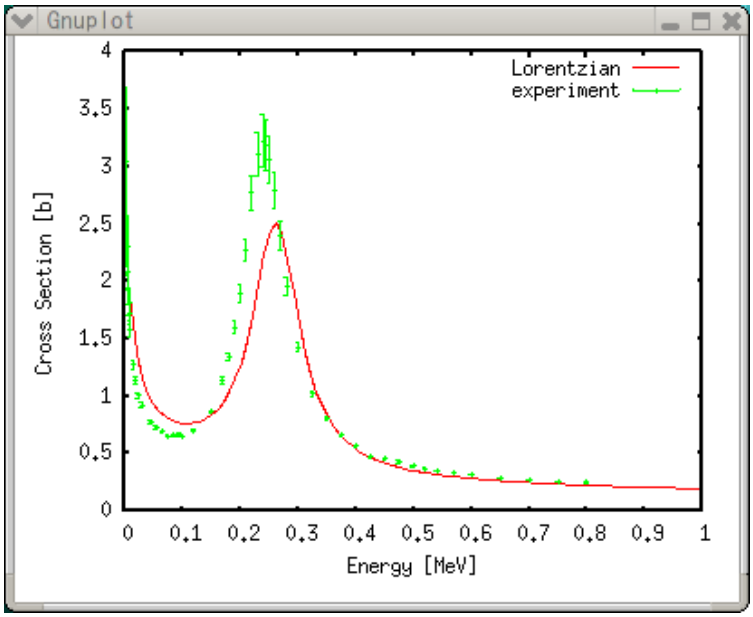

Figure 25: plotfunc3

As shown above, the fitted function is not so good except for the region where the data are off-resonance. This is because our fitting function was not the best choice. Its peak shape looks fine with the Lorentzian, so we modify `d/sqrt(x)` by introducing a new parameter 'e' and express it as `d*x**e` . The initial value of 'e' is -0.5.

```
gnuplot> e=-0.5
gnuplot> f(x)=c/((x-a)*(x-a)+b)+d*x**e
gnuplot> fit f(x) "exp.dat" using 1:2:3 via a,b,c,d,e

     ...
```

```
Final set of parameters            Asymptotic Standard Error
=======================            ==========================

a                = 0.25029         +/- 0.002106     (0.8412%)
b                = 0.00197707      +/- 0.0002747    (13.89%)
c                = 0.00550098      +/- 0.0003662    (6.657%)
d                = 0.21537         +/- 0.003743     (1.738%)
e                = -0.358371       +/- 0.0115       (3.208%)


correlation matrix of the fit parameters:

                a      b      c      d      e
a                1.000
b                0.021  1.000
c               -0.078  0.788  1.000
d               -0.110 -0.384 -0.500  1.000
e               -0.304  0.198  0.335  0.381  1.000
gnuplot> replot
```

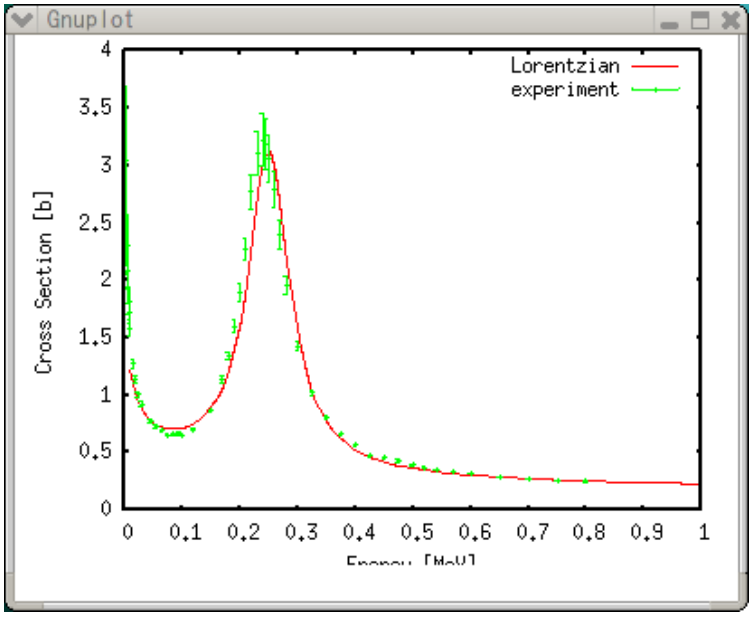

Figure 26: plotfunc4

Still its not perfect. Fixing `a=0.24` and search only for `via b,c,d,e` gives you better looks, but its chi-square is larger.

4.4. Output in the Postscript format

As shown in Experimental Data section we generate a Postscript figure. The data points are drawn with solid line and seventh symbol (circle), and the function is shown with the thick dashed line.

```
gnuplot> set linestyle 1 lt 1 pt 7
gnuplot> set linestyle 2 lt 2 lw 3
gnuplot> set size 0.6,0.6
gnuplot> set term postscript eps enhanced color
Terminal type set to 'postscript'
Options are 'eps enhanced color dashed defaultplex "Helvetica" 14'
gnuplot> set output "exp.ps"
gnuplot> plot"exp.dat" using 1:2:3 title "experiment" with yerrors ls 1,\
>        f(x) title "Lorentzian" with line ls 2
```

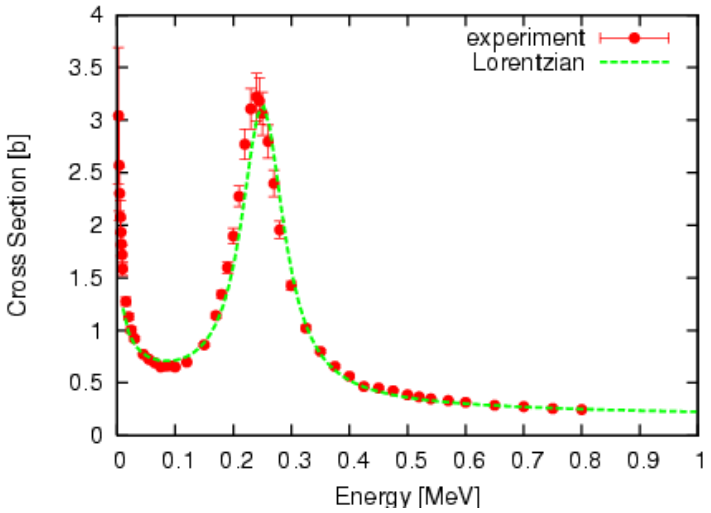

Figure 27: plotfunc5

5. Life with gnuplot

5.1. Make a Batch Process

Here we explain how to finish your graph which was prepared as described before. A plot-file "output.plt" is a saved file at the last part of "Numerical Calculations" section. This file contains parameters and functions, some of those are default ones while the other part is defined by user. In the following list we eliminated the default lines, and "set terminal" and "plot" lines are folded.

```
#!/usr/local/bin/gnuplot -persist
# set terminal postscript landscape noenhanced monochrome \
#              dashed defaultplex "Helvetica" 14
# set output 'output.ps'
set xlabel "x" 0.000000,0.000000  ""
set ylabel "y=exp(-x)" 0.000000,0.000000  ""
set title "Pade approximation" 0.000000,0.000000  ""
set xrange [ 0 : 2 ] noreverse nowriteback
set yrange [ 0 : 1 ] noreverse nowriteback
set mxtics 5.000000
set mytics 5.000000
set xtics border mirror norotate 1
set ytics border mirror norotate 0.5
plot "output.dat" using 1:2 title "Analytical" w l, \
     "output.dat" using 1:3 title "L=1, M=2" w l, \
     "output.dat" using 1:4 title "L=2, M=1" w l
#    EOF
```

At the top of this file you may see the part where an output device (Postscript) and file-name are defined. You can draw the same graph with this file by using the `load` command. When you make another calculation, and the result is stored in "output.dat", you can reused this plot-file for the new plotting.

```
gnuplot> load "output.plt"
```

Alternatively, you can give this file name as a command-line option when gnuplot is invoked. In this way gnuplot exits after the last line of the file is read. A graph on your screen disappears at the same time. To keep the graph on your screen, insert `pause -1` at the end of this file. The graph stays there till you hit a 'return' key.

```
% gnuplot output.plt
```

A command-line option `-persist` also keeps the graph on your screen. With this method, gnuplot itself exits but the graph window settles down.

```
% gnuplot -persist output.plt
```

As can be see at the first line of the plot-file, this option is already included as a shell script option. This option can be used by giving permission of execution to this file, as follows.

```
% chmod +x  output.plt
% ./output.plt
```

With this manner you have to close the window every time.
You can continue your work interactively at the gnuplot command-line once the plot-file is loaded. However, a batch-mode is more convenient. Firstly, insert `pause -1` at the end of the file, then

- Edit the plot-file with a text editor
- Browse the graph on a screen

Repeat this sequence until you get a satisfactory figure. If you are aiming at a Postscript file, your sequence may become:

- Edit the plot-file with a text editor
- Output the graph in a Postscript data
- Preview it with gv (ghostview)

In this case, a comment sign '#' at the beginning of `set terminal postscript` and `set output 'output.ps'` lines should be deleted. You don't need `pause` because no drawing appears on your screen.

5.2. Draw Many Figures

One nice thing to use gnuplot in your scientific activity is that we can reuse the plot-file to make a similar figure. We often draw very similar figures those have same axis names, same ranges, etc. but the numerical data inside the graph are different. With gnuplot you can do this by using only one plot-file. To do it, change the data file name at `plot "datafile"` or use the same data file name but the file is overridden by various data.
Gnuplot batch mode is very useful when you want to make a large number of figures at one time. If you are doing this with something another software, you have to struggle with your computer until midnight. Why don't you make your life easier with gnuplot and some UNIX commands to generate many figures within a few seconds.
Here we think about a case that you have many data files — calc1.dat, calc2.dat, calc3.dat — in your directory, and each file contains simple (X,Y) data. Firstly, we make a graph of the data file, calc1.dat . Here we define the figure title, name of axes, X and Y ranges, and graduations, just like we have done in the numerical calculation section.

```
set terminal png
set output "calc1.png"
set xlabel "Energy [MeV]"
set ylabel "Cross Section [b]"
set title "(n,2n) reaction"
set xrange [ 0 : 20 ]
set yrange [ 0 : 2 ]
set mxtics 5
set mytics 5
set xtics 5
set ytics 0.5
plot "calc1.dat" using 1:2 notitle w l
```

A PNG image file is generated by gnuplot, when you "feed" this data.plt file to gnuplot. If you need an EPS file, use `set terminal postscript` , and change the name of output file properly, like calc1.eps.

```
% gnuplot < data.plt
```

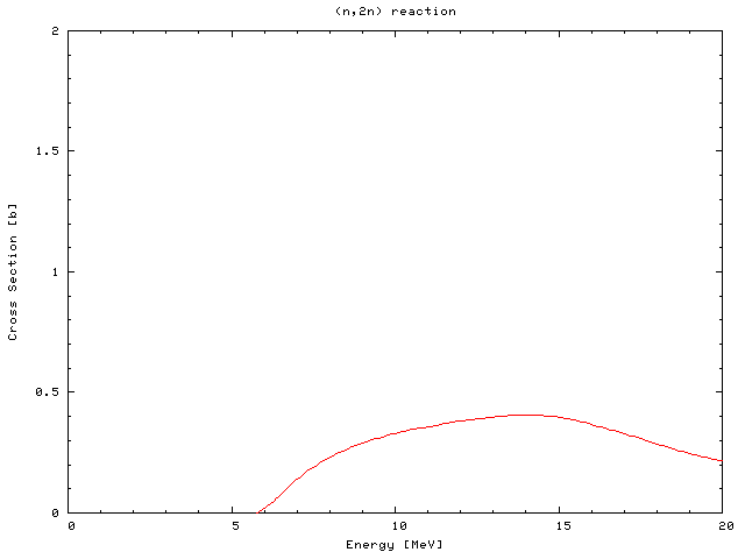

Figure 28: calc1

Now we reuse this to make a figure of another data file. To do it, we have to rewrite the file names "calc1.dat" and "calc1.png" in that file. Unfortunately you cannot use strings-type data in a gnuplot command, we replace the text "calc1" into "calc2" with the UNIX command, sed. Alternatively, those file names are give as command line options of a shell script.
The following command changes the text "calc1" in a file into "calc2".

```
% sed "s/calc1/calc2/g" data.plt | gnuplot
```

This one-line command makes a new figure, calc2.png. You repeat this for all your data files. When the number of data is not so large, you can use a shell-loop ('foreach' of csh, or 'for' of sh).

- csh, tcsh

```
% foreach i (calc2 calc3 calc4 cal5)
 foreach? sed "s/calc1/$i/g" data.plt | gnuplot
 foreach? end
```

- sh, bash

```
$ for i in calc2 calc3 calc4 cal5 ; do
 > sed "s/calc1/$i/g" data.plt | gnuplot
 > done
```

When you have many files, you can remove the extension '.dat' of the file name with sed like below. Be careful that there are quotation and back-quotation marks. The inside sed command is to remove '.dat' extention of the variable $i , and the outside is to convert the strings "calc1."

```
$ for i in *.dat ; do
> sed "s/calc1/‘echo $i | sed "s/\.dat$//"‘/g" data.plt | gnuplot
> done
```

You can also make a shell script to draw figures one-by-one. Once you make such a script, you can reuse this for other jobs. The script contains many lines corresponding to the number of your data files.

```
#!/bin/sh
sed "s/calc1/calc2/g" data.plt | gnuplot
sed "s/calc1/calc3/g" data.plt | gnuplot
sed "s/calc1/calc4/g" data.plt | gnuplot
sed "s/calc1/calc5/g" data.plt | gnuplot
 ...
```

To make this you can use the UNIX commands, 'ls' and 'awk'. The following one-line command shows you the list of commands with which you can make many figures at one time. To make a shell-script file, re-direct this output to a file (command `> script.sh`).

```
% ls *.dat | awk '{printf("sed \"s/calc1/%s/g\" data.plt | gnuplot\n",$1)}'
```

You can write gnuplot commands directly in a shell script, and your file name is given as a command line option. To do this, use a "here document" of the UNIX shells. In this example, we make an EPS file "calc1.eps" from the data file "calc1.dat."

```
#!/bin/sh
gnuplot << EOF
set terminal postscript eps color enhanced
set output "$1.eps"
set xlabel "Energy [MeV]"
set ylabel "Cross Section [b]"
set title "(n,2n) reaction"
set xrange [ 0 : 20 ]
set yrange [ 0 : 2 ]
set mxtics 5
set mytics 5
set xtics 5
set ytics 0.5
plot "$1.dat" using 1:2 notitle w l
EOF
```

As you can see in the example above, the names of data and EPS files (without extensions) are replaced by a variable $1. The real names of those variables are given as the command line option to this shell script plot.sh. This script makes the EPS file "calc1.eps" from the data file "calc1.dat" when the command is invoked like below. With this method you can make a figure of your data files which are in a different directory, and the result is stored in the same directory.

```
 $ ./plot.sh calc1
```

To change the names of data file and EPS, you replace one of the $1 variables in the script above into $2, and give two command line options to this script.

Finally the next shell script makes it possible to generate many figures within a second.

```
#!/bin/sh
./plot.sh  calc1
./plot.sh  calc2
./plot.sh  calc3
./plot.sh  calc4
 ...
```

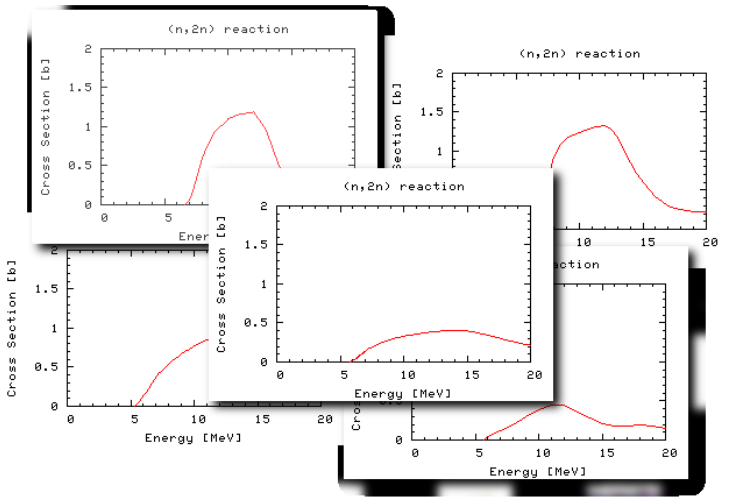

Figure 29: manyfigure

6. Plot Style

Here you can see what kinds of graph gnuplot can draw. To specify the style of graph, `with` style. To connect the data points with lines, use `with lines`. The `with points` option places symbols at the data points.

- Draw lines, dots, and symbols
- Draw a bar-graph
- Draw symbols with error bars
- Draw vectors
- Others (candlesticks, financebars)

Those styles can be easily combined. For example, to draw a line graph and a bar graph at the same time:

```
gnuplot> plot "file.dat" with boxes, "" notitle with lines
```

6.1. Draw Lines, Dots, and Symbols

6.1.1. lines

It connects each data point with lines. Suitable to smoothly varying data.

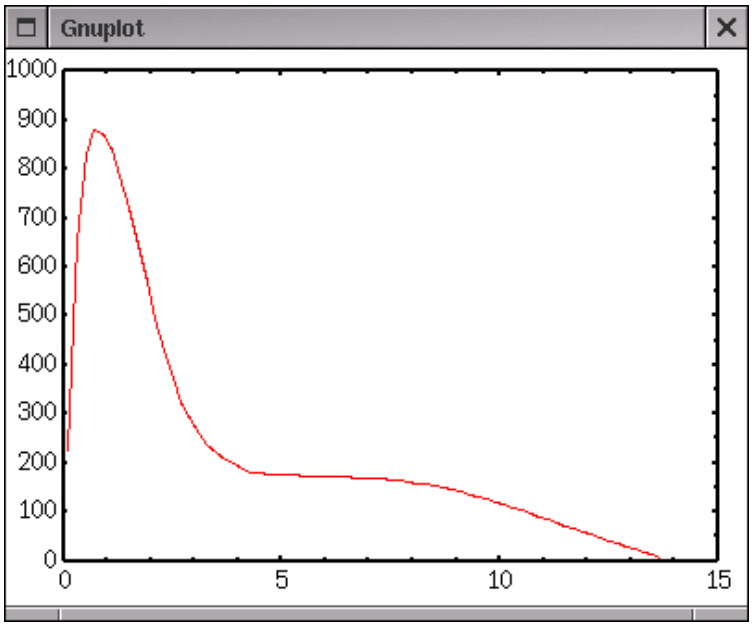

Figure 30: with lines

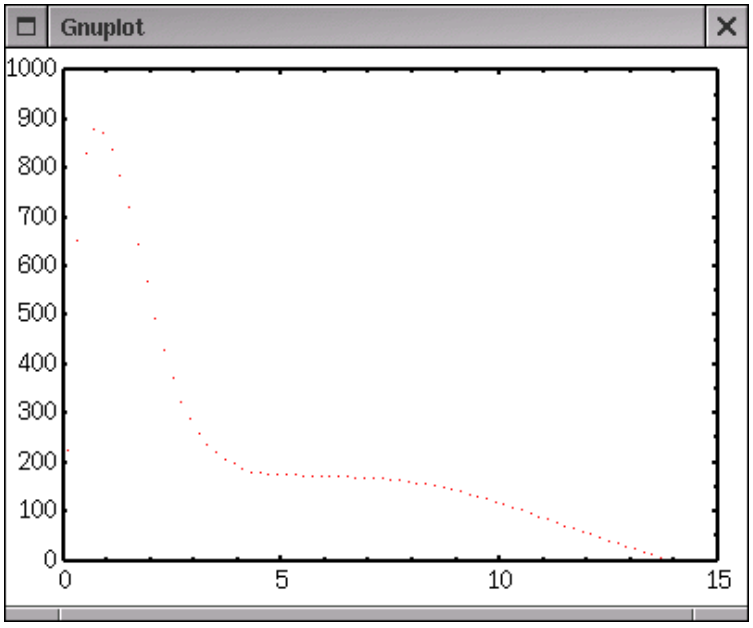

Figure 31: with dots

6.1.2. dots

It displays dots, can be used when there many data points, but hard to see though.

6.1.3. points

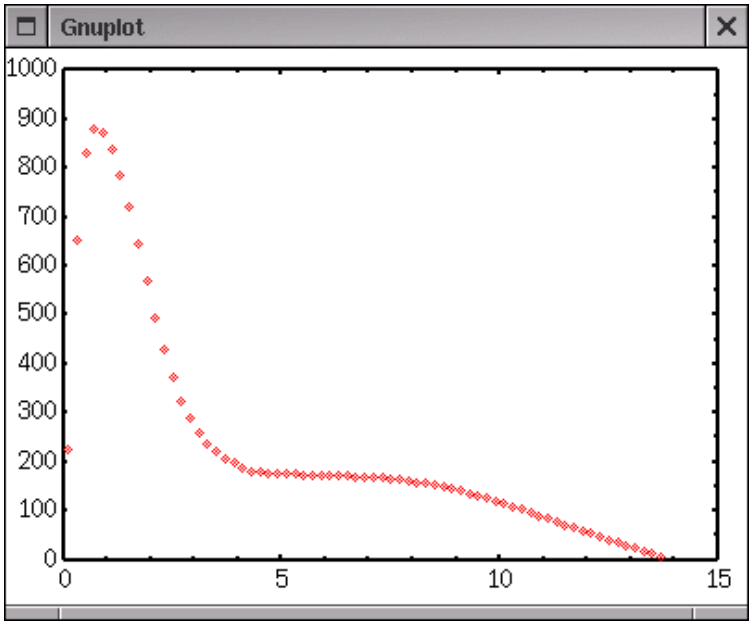

Figure 32: with points

Symbols are shown at the data point location, can be used to plot experimental data. The number of different kinds of symbol depends on your terminal, but there are at least 6. The number '-1' is a dot. The size of symbol can be changed by the `set pointsize` command.

6.1.4. linespoints

Draw lines and symbols at the same time.

6.1.5. impulses

Draw vertical lines from each data point to X-axis. This is a bar-graph without width.

6.1.6. steps fsteps histeps

Three kinds of histogram. The difference of those is a definition of starting and ending points, which you can see below. Gnuplot can draw a histogram, but it cannot calculate

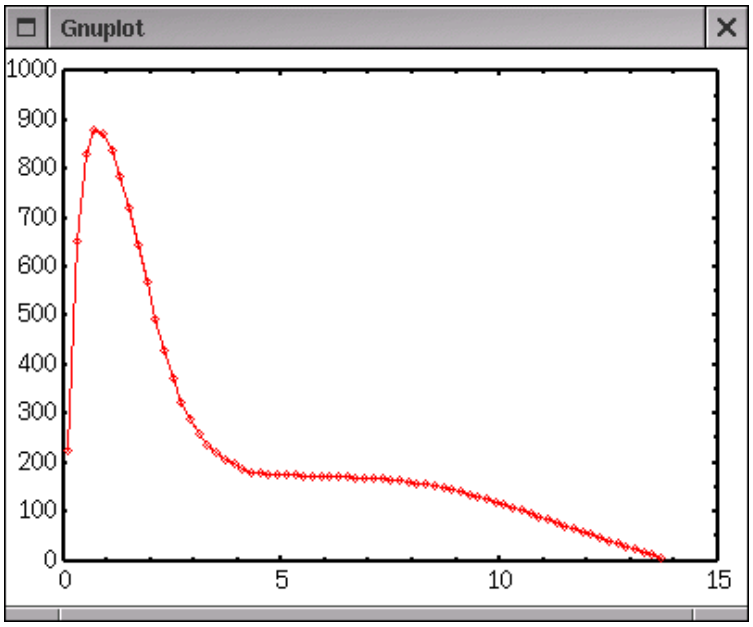

Figure 33: with linespoints

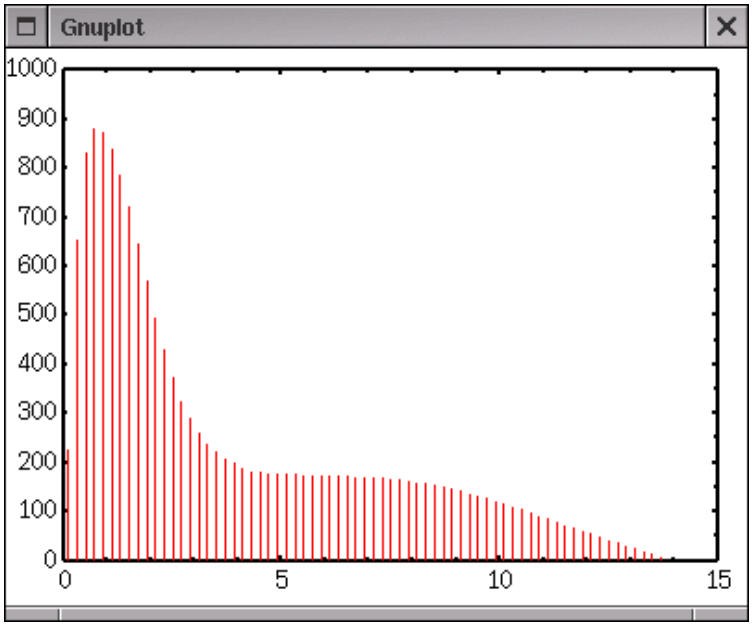

Figure 34: with impulses

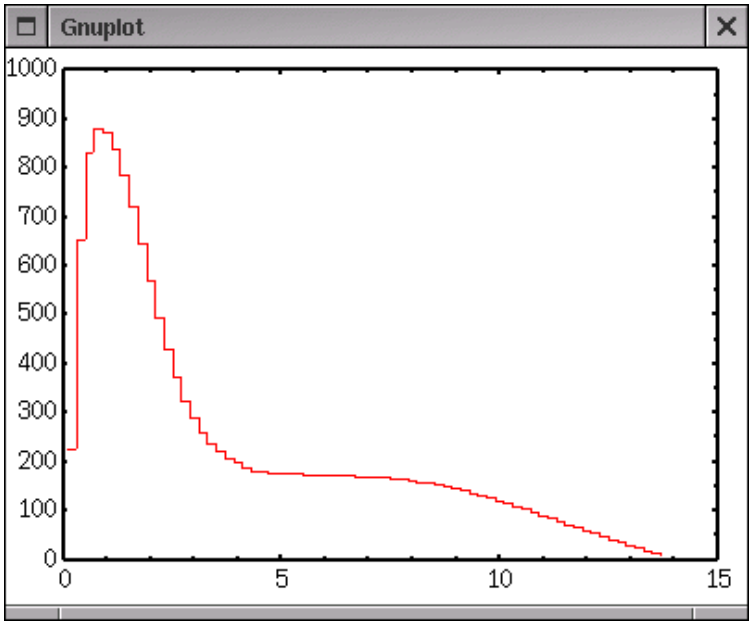

Figure 35: with steps

the histogram data themselves.
Here you can see the difference of 3 histograms.
The data position is shown by a blue box. With `steps`, the data point defines an initial position, while the data points are treated as the final position with `fsteps`. With `histeps` a graph becomes a kind of bar-graph but the bars stick together.

6.2. Draw a Bar-Graph

6.2.1. boxes

This bar graph resembles the histograms above except for vertical lines. Inside the each bar is empty, cannot be filled with color. If you want a color-filled bar-graph, exports the produced graph into some drawing softwares line Tgif, and paint them.
A width of each bar is calculated automatically so that each box touches the adjucent box. To control the width, use `set boxwidth` width. For this case the width of all boxes become width . The width can be given by a data file, the third column (otherwise you can specify the column by `using X:Y:Z` where Z becomes the width). In the example below, the data points are shown by blue symbols.

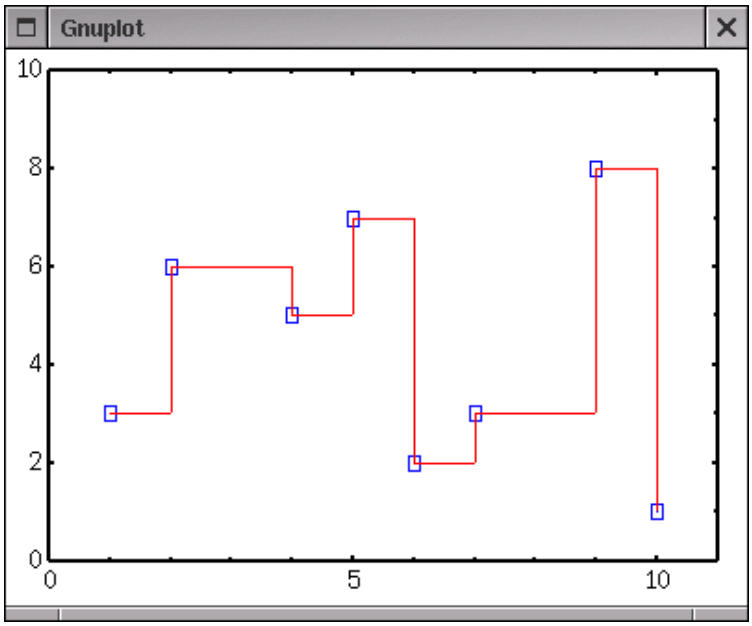

Figure 36: with steps

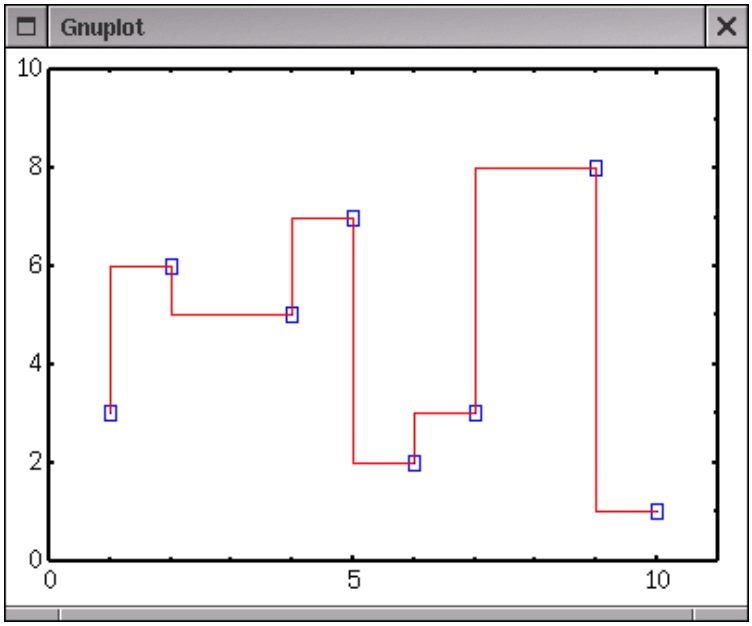

Figure 37: with fsteps

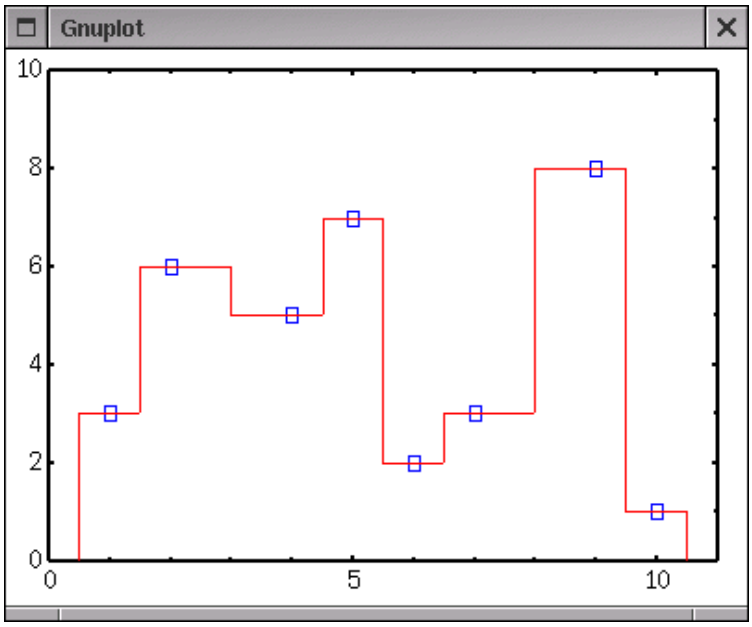

Figure 38: with histeps

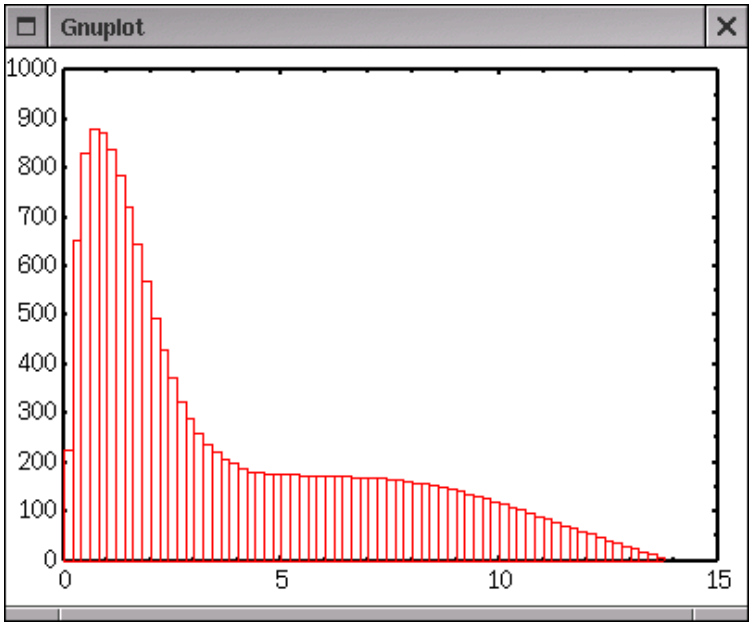

Figure 39: with boxes

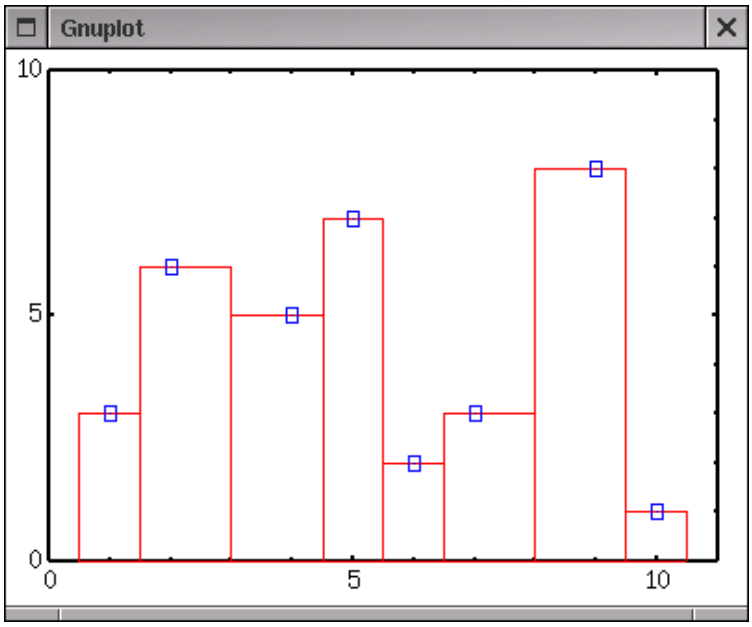

Figure 40: Widths are calculated automatically

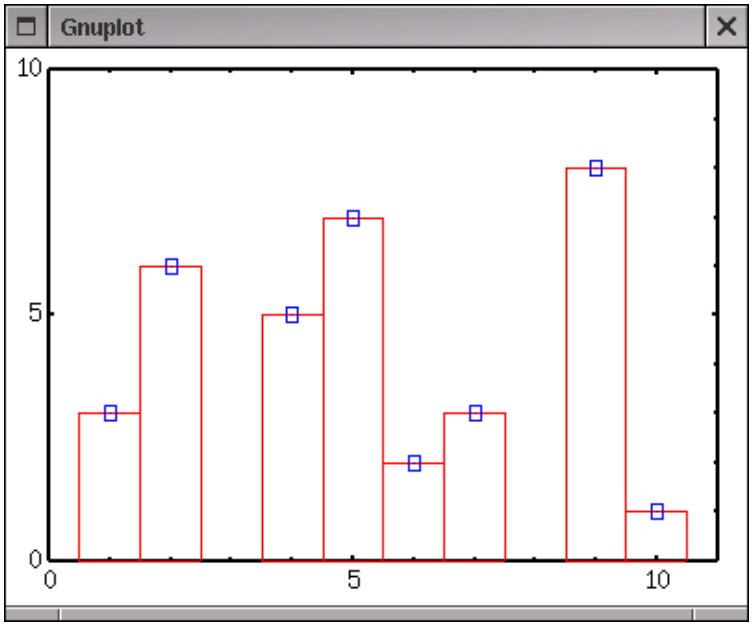

Figure 41: Set the width to 1

6.2.2. default

6.2.3. set boxwidth 1

6.2.4. data file

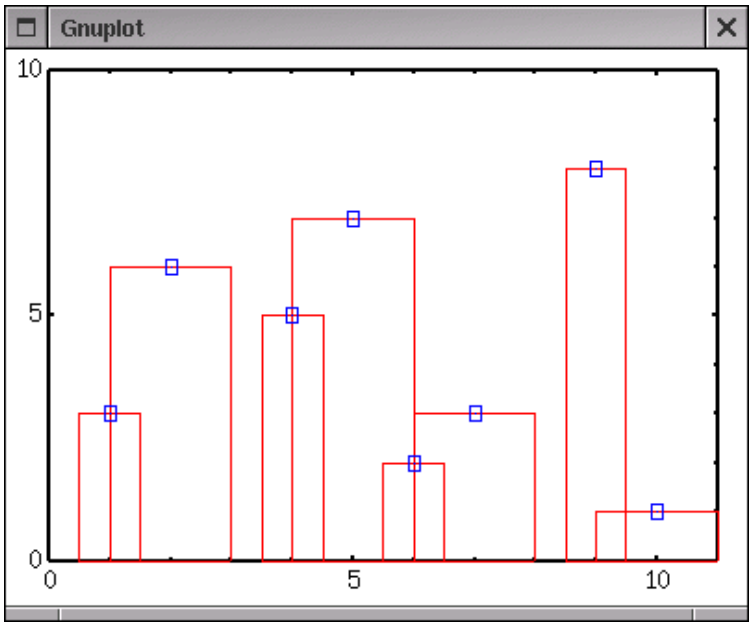

Figure 42: data file

When your data file has the third column, that is used for the bar-width. In the right figure, widths of 1,2,1,2... are given in the file

6.3. Draw Symbols with Error Bars

6.3.1. yerrorbars

Same as 'points' but Y values have errors. The length of error bar is given in a file. If the data file has 3 columns, the third column is used as the Y-error, Y plus/minus dY. It there are 4 columns, the third and fourth columns are used, likel, Y plus dY1 minus dY2.

6.3.2. xerrorbars

Same as 'yerrorbars' but the error bars are horizontal.

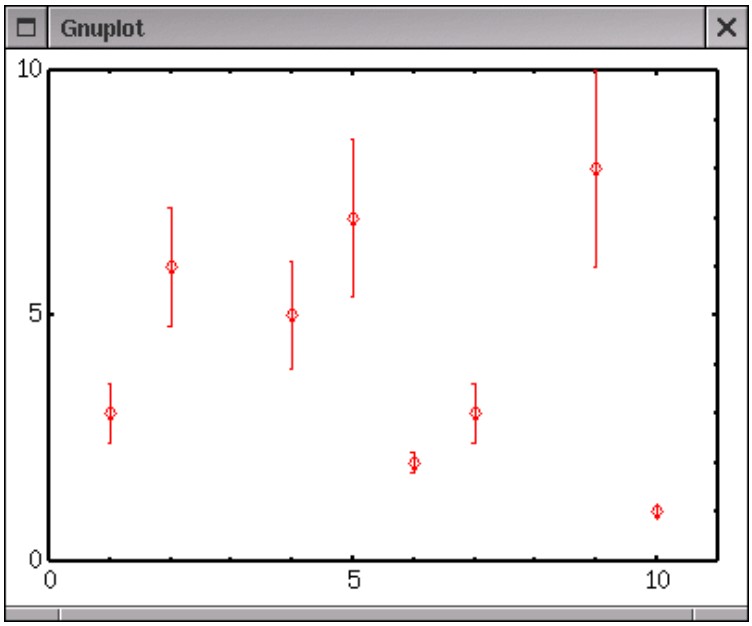

Figure 43: with yerrorbars

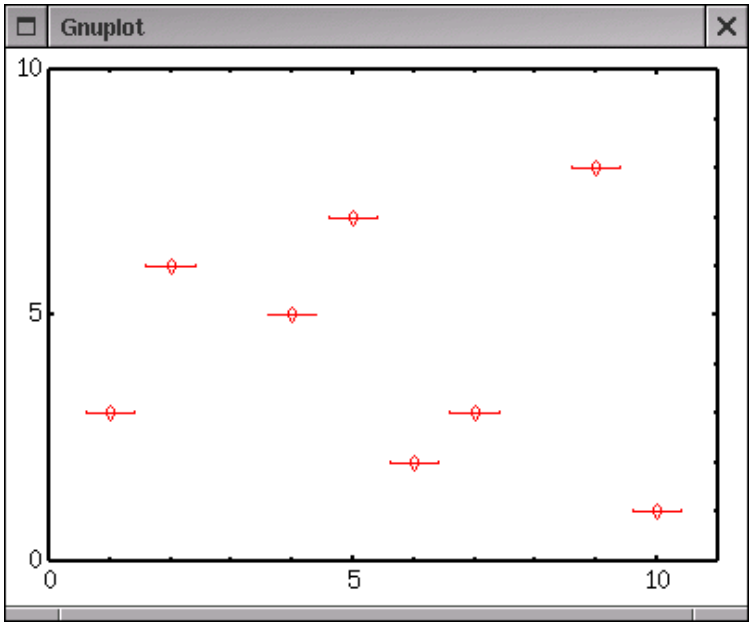

Figure 44: with xerrorbars

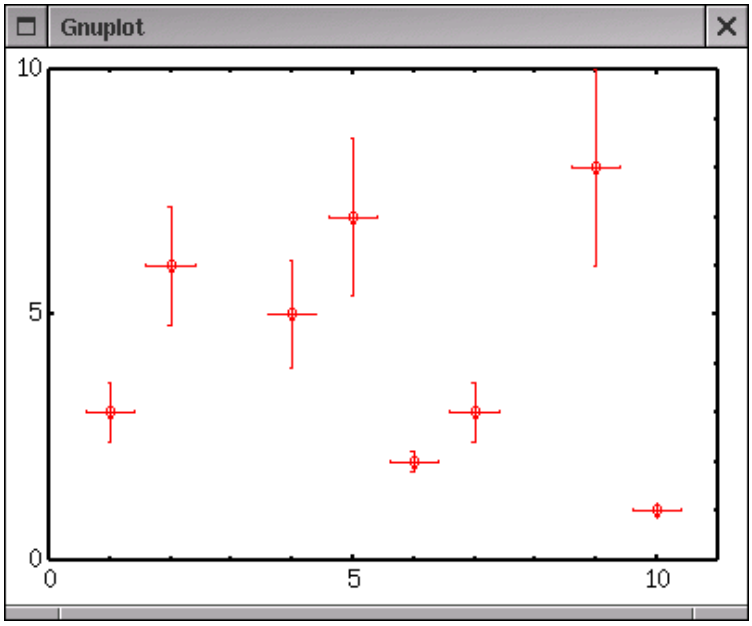

Figure 45: with xyerrorbars

6.3.3. xyerrorbars

Both X and Y values have errors. You need 4 or 6 columns data. See data file how to specify the data and data errors.

6.3.4. vector

Draw vectors. An arrow is plotted from (X,Y) to (X+dX,Y+dY). So that you need 4 columns data in your data file.

6.3.5. Others (financebars, candlesticks)

There are two styles those can be used for financial plot. Those were categorized into "others" here, because the author of this page is not good at finance :-) Maybe those are used for plotting of variation of stok or price. The style "financebars" and "candlesticks" requires 5 columns which are date/time, opening, low, high, and closing prices, respectively. High and low prices are connected with a vertical line, and opening and closing prices are indicated by small horizontal tics. The length of the tics can be changed by `set bar`. Similar to "financebars". A rectangular represents opening and closing prices, and a horizontal line segment shows high and low prices.

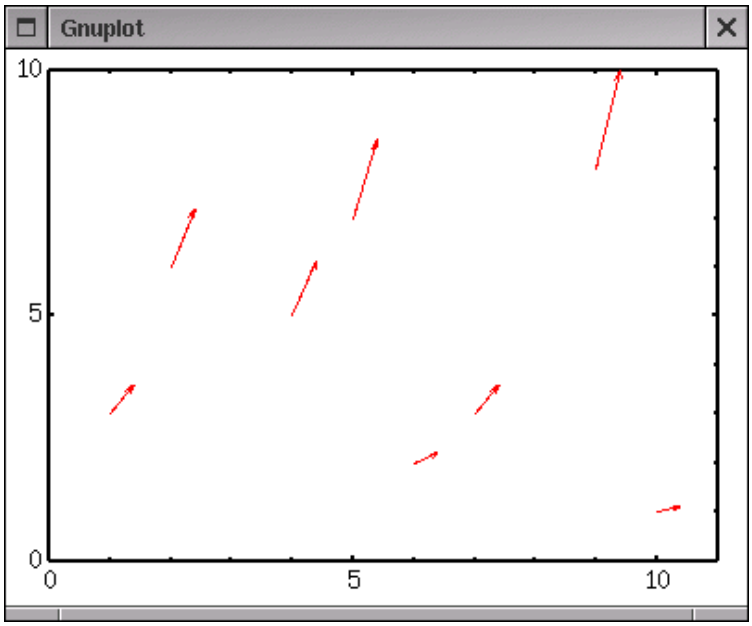

Figure 46: with vector

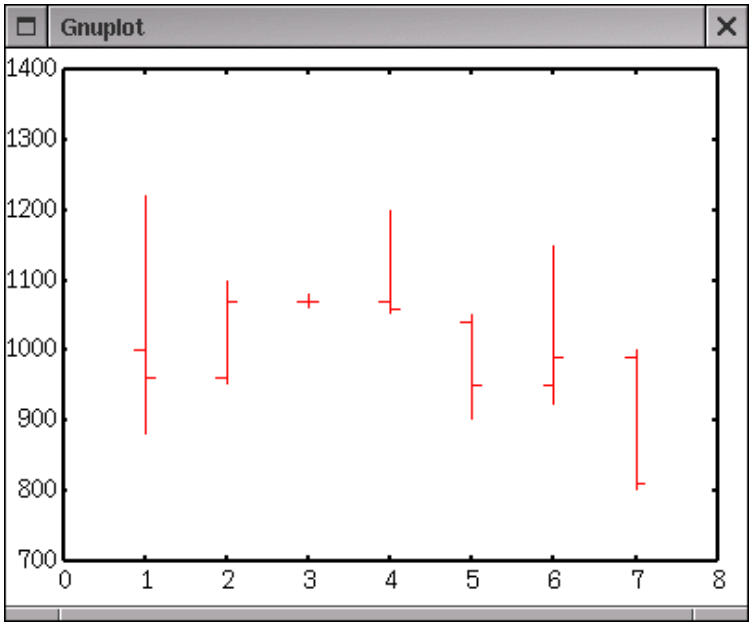

Figure 47: with financebars

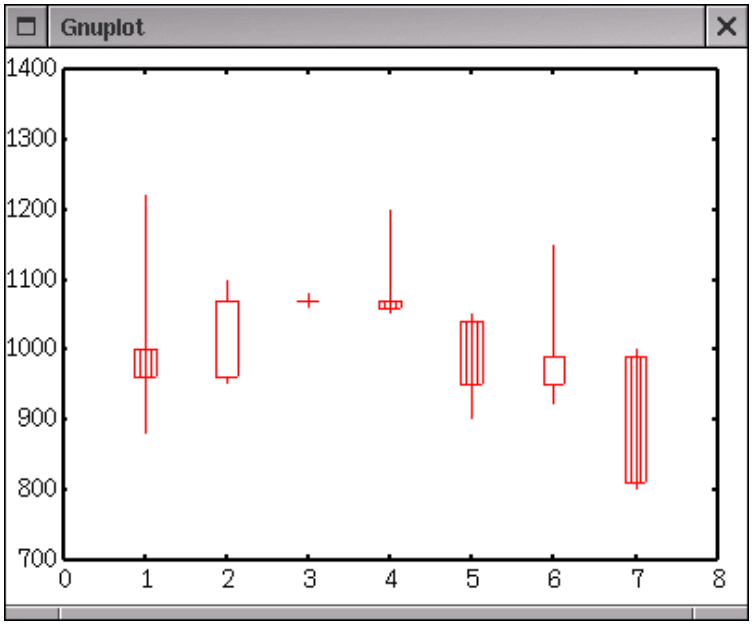

Figure 48: with candlesticks

Part II.
Not so frequently asked questions

7. About Legend

... but gnuplot calls it "key"

7.1. How do I erase a legend?

There are two ways. The first one is,

```
gnuplot> set nokey
```

and the other one is to use the `notitle` keyword at plotting. In the case below, the data file has a legend but the function does not.

```
gnuplot> plot f(x) notitle, "file.dat" title "data"
```

7.2. How do I change the location of a legend?

Usually a legend appears at the top/right corner in the graph. You can change the position with the `set key` command. If you give the command,

```
gnuplot> set key left bottom
```

the legend goes to left/bottom. Available options are, left, right, top, bottom, outside, and below. You can combine some of them. For example, `outside bottom`. It is possible to set the position of legend directly. If you want to move it to the position (X,Y)=(100,100),

```
gnuplot> set key 100,100
```

The coordinate (100,100) is the position of the mid-point between a text and a line/symbol of the first line of the legend. The coordinate is the system defined by the X and Y axes. If you want to place the legend independently of the axes, see coordinate.

7.3. How do I get rid of error bars in a legend?

When one plots data with error bars, the error bar also appears in the legend (see the figure below). However, even if the data have errors in the Y-direction, the error bar in the legend becomes horizontal one, and usually we do not need such an error bar. Here is a simple way to remove the error bar in the legend.
When we plot the next data (test.dat),

```
#  X     Y     Y-error
   1.0   1.2   0.2
   2.0   1.8   0.3
   3.0   1.6   0.2
```

with the following commands,

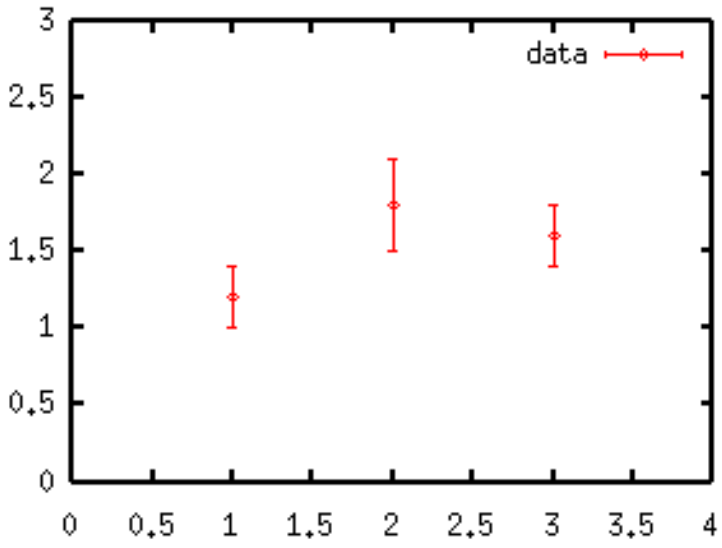

Figure 49: sample2.3a

```
gnuplot> set xrange [0:4]
gnuplot> set yrange [0:3]
gnuplot> plot "test.dat" usi 1:2:3 title "data" w yerrorbars
```

the error bar in the legend becomes like above. It is not a smart way, but

```
gnuplot> set xrange [0:4]
gnuplot> set yrange [0:3]
gnuplot> plot "test.dat" usi 1:2:3 notitle        w yerrorbars 1,\
              "test.dat" usi 1:2     title "data" w points     1
```

works well.

7.4. Location of the text is sometimes strange when Postscript symbols are used in it.

Sometimes gnuplot places a text (in title or legend) at wrong position when a postscript symbol – like {`/Symbol a`} – is used. For example, you may have an extra white-space at the left side even though you want to place the legend just next to the Y-axis with the command `set key left`. This happens because gnuplot does not count {`/Symbol a`} as one character. To adjust the location of the legend containing postscript symbols, you have to give its coordinate directly.

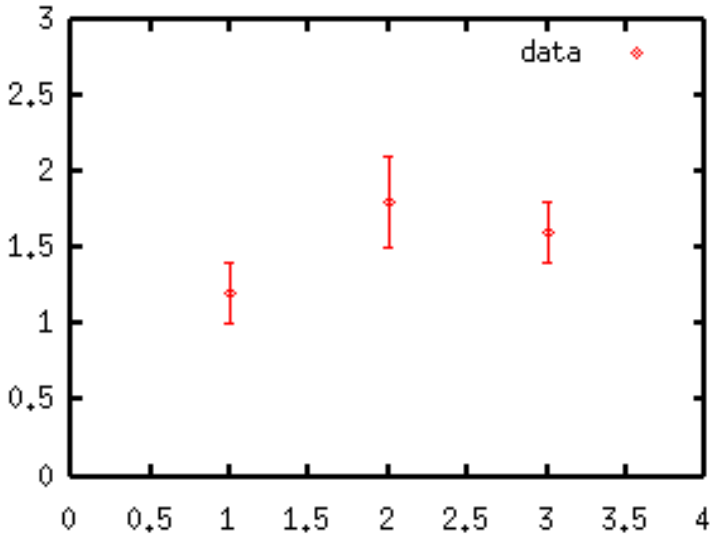

Figure 50: sample2.3b

7.5. Adjust the line-skip

You can change the line-skip in the legend with `set key spacing` command. To make the skip 1.5 times wider, `set key spacing 1.5`.

7.6. Make a frame-box

To make a frame-box around the legend, use `set key box`. The line-kind for the box is the same as the frame of the graph. You can change this by providing the `box` keyword followed by an index of the line-kind. Otherwise you can define the linestyle as:

```
gnuplot> set linestyle 1 lt 2 lw 3
gnuplot> set key box linestyle 1
```

For gnuplot ver.4.0,

```
gnuplot> set style line 1 lt 2 lw 3
gnuplot> set key box linestyle 1
```

8. About Tics

8.1. How do I change an appearance of tics on each axis ?

There are two kinds of tics — the major tics and the minor tics. Numbers (figures) on axes are drawn at the major tics only. At the default the minor tics are not shown except for a logarithmic scale. `set x|ytics` command changes the major tics.

```
gnuplot> set xtics 2
gnuplot> set ytics 0,200
gnuplot> plot x**3
```

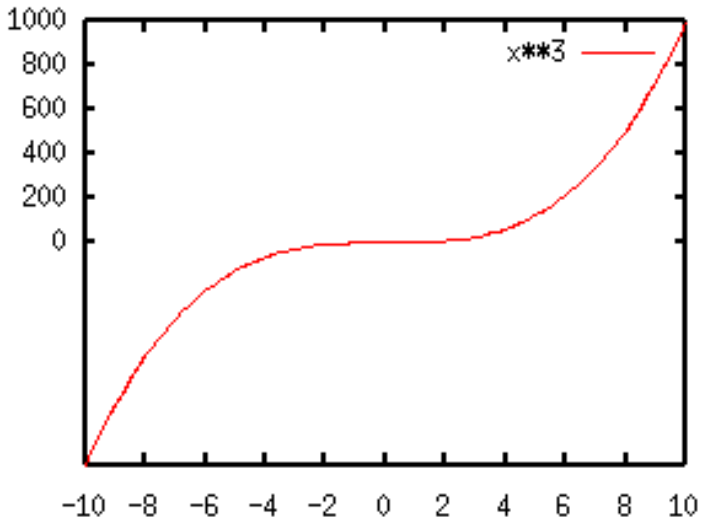

Figure 51: sample3.1a

Only an increment is defined for the X-axis, so that the positions of tics become -10, -8, ... +10. For the Y-axis, the starting value and the increment are both given. Gnuplot makes tics at 0, 200, 400... In this case there is no tic mark where Y is negative. You can also give the last value like, `set ytics 0,200,600`. The minor tics can be controlled by `set mx|ytics` . Intervals between each major tic are divided by this value. The following example is to draw a mid-point between each major tics in the above figure.

```
gnuplot> set mxtics 2
gnuplot> set mytics 2
```

It is possible to place an arbitrary text at the tics position instead of the numbers. you can write "April", "May", "June", and "July" at the positions X=1,2,3, and 4, by the following way.

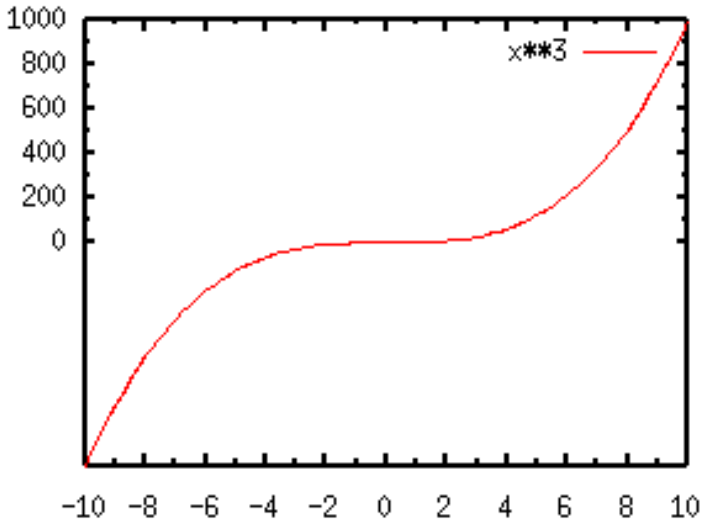

Figure 52: sample3.1b

```
gnuplot> set xtics ("April" 1, "May" 2, "June" 3, "July" 4)
gnuplot> plot "test.dat" using 1:2:3 notitle with boxes,\
              "test.dat" using 1:2   notitle with lines
```

As can be seen, the minor tics are ignored when the major tics are defined in this syntax. The Command `set ticscale n m` changes the length (size) of tics. The major tics are multiplied by the provided value n, while the minor tics are multiplied by m. If m is omitted, the minor tics are half length of the major tics. `set ticscale 1 1` makes the same length of the major and minor tics.
The tics are drawn inwards. To make it outwards use `set tics out`.

8.2. I want to use an exponent instead of a decimal like 0.001 in a log-scale plot.

The default format of the tic-mark labels is "%g". In the log scaling, the labels are written by the "F" format like 0.01, 1000, etc. if the range is more than 0.0001 and less than 100000. If the numbers are outside this range, the format is changed into "E" which gives the numbers such as 1e-05, 1e+06, etc.
To express numbers by power to base 10, change the label format into "$10^{\%L}$" by the command `set format`. "%L" gives the value of power to base 10. The X window terminal cannot show super-scripts, so use the postscript terminal.

```
gnuplot> set format y "10^{%L}"
gnuplot> set terminal postscript eps enhanced
```

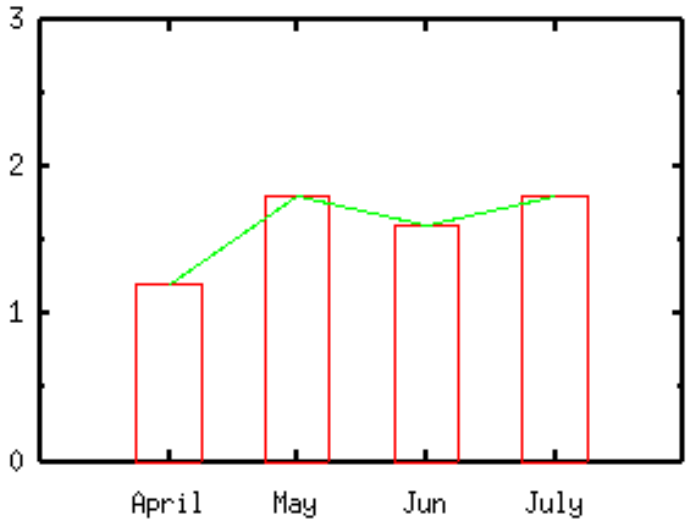

Figure 53: sample3.1c

```
gnuplot> set ylabel "Y-AXIS" 2,0
```

When the format is defined like this, the distance between the Y-axis and its label becomes wider. This can be adjusted by the offset option of the `set ylabel` command. In this case the Y-axis label was moved to right by 2 character-widths.

8.3. How do I change the format of the numbers?

The format of the tic-labels can be changed by `set format` command. The syntax is `set format` + axis name (X, Y, Z, XY, X2, Y2) + format string. This string is similar to the format string in C-language. The next example tells gnuplot to show the tic-labels by ten letters with three digits following the decimal point.

```
gnuplot> set format x "%10.3f"
```

The syntax for the digit is "%" + (total length).(precision). The floating number 6.2 represents that the total length is six and there are two digits following the decimal point, so that tic-labels are shown as "5.00". It is possible to omit the length or precision, like 6 or .2. The default values are used for the omitted number.
The display format is expressed by one letter – 'f', 'e', 'E', 'g', 'x', 'X', 'o', 't', 'l', 's', 'T', 'L', 'S', 'c', and 'P'. The default is "%g". When the tics-labels can be expressed by appropriate length and precision, those are written by "%.0f" format, otherwise "%e" format is used. The next table shows the difference among the format 'f', 'e', 'x', and 'o'. The formats 'e' and 'E' are the same except that the written text is 'e' or 'E'. The format "%O" exists in the gnuplot manual, but it does not work (bug ?)

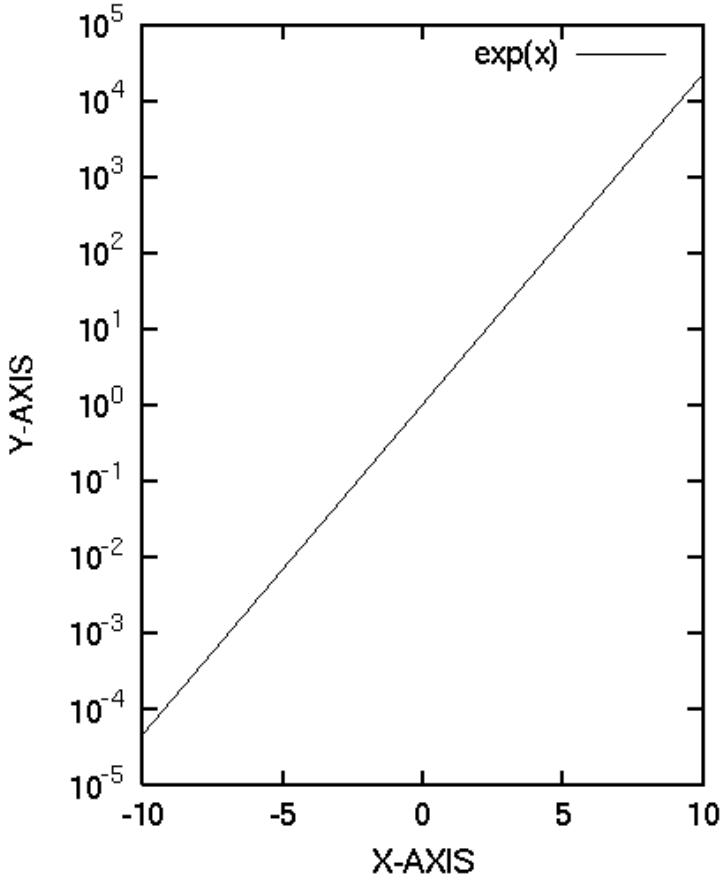

Figure 54: sample3.2

Format	Explanation	Example (underscore means blank)	
f	decimal	%6.3f	6.00
e,E	exponential	%11.4e	5.0000e+01
x,X	hexadecimal	%x	fffffffb
o,O	octal	%o	37777766

The formats, 't', 'l', 'T', and 'L' are related to log-scale plot. Let's draw a function y=exp(-x)+exp(x) in the X range of [-10:10]. Firstly this function is shown in a non-log scale. In order to compare those formats, the tics are shown in three formats at the same time.

```
gnuplot> set format y "%T %t %g"
gnuplot> plot exp(-x)+exp(x)
```

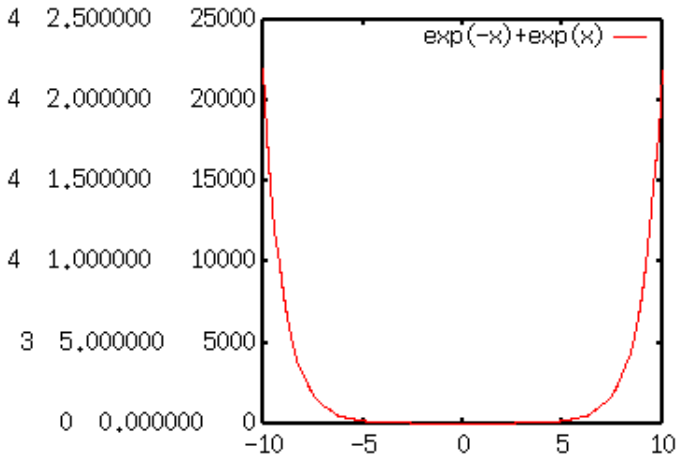

Figure 55: sample3.3a

You can specify the different kinds of format simultaneously just like this case — three numbers are written on the Y-axis. The first one is for "%T", the next is "%t", and the third is "%g", respectively. As you can see, "%t" represents the mantissa to base 10, and "%T" represents the power to base 10. Therefore the number expressed by the "%g" format is A x 10^B, where A is given by "%t" and B is "%T".
Now, the figure in log-scale becomes:

```
gnuplot> set logscale y
gnuplot> replot
```

The numbers expressed by the format "%t" become unity, and "%T" gives log of the Y-axis. So `set format "%T"` displays power to base 10 along the Y-axis. If the

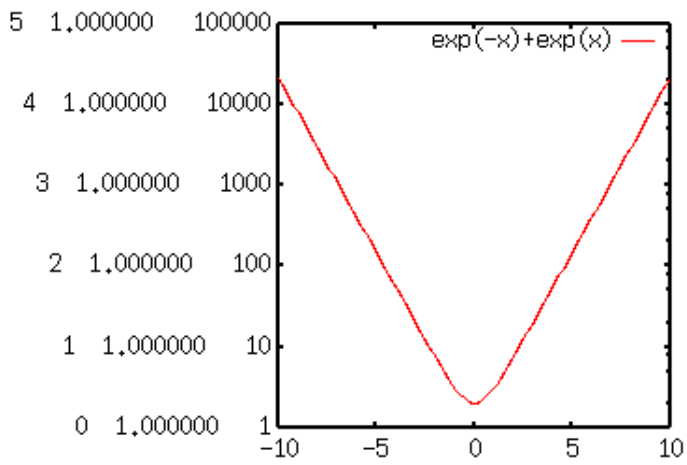

Figure 56: sample3.3b

terminal can display a super-script - like Postscript terminal, the format "$10^{\%T}$" gives an exponential expression.
Usually 't' and 'l', 'T' and 'L' give the same results in a usual log-scaling. If one uses the different base for log-scaling, those give different results. The format 't' and 'T' give mantissa and power to base 10, while 'l' and 'L' give those to an arbitrary base which is defined by `set logscale` axis base command. But base 10 is commonly used for log-scaling, so you can regard those the same.
You can include various letters and texts as well as the numbers in format. For example, set format x "%g km" displays values in 'g' format and unit "km".

8.4. How do I erase numbers ?

This is an application of the above item.

```
gnuplot> set format x ""
```

8.5. I want to make intermediate values in the logarithmic tics.

Generally log-tics appear at the power of 10, (1,10,100...). Sometimes it is helpful to understand graphs if you make tics between them, especially in case the range of axis is not so large (for example, ratio of the max to min is 10 or 100). Gnuplot writes values at the major tics specified by `set x|ytic` only, so you need to define the positions where you want to write values.

```
gnuplot> set logscale y
```

```
gnuplot> set yrange [1:50]
gnuplot> set ytics (1,2,5,10,20,50)
gnuplot> set format y "%2.0t{/Symbol \327}10^{%L}"
gnuplot> set lmargin 8
gnuplot> set terminal postscript eps enhanced
gnuplot> plot exp(x)/x
```

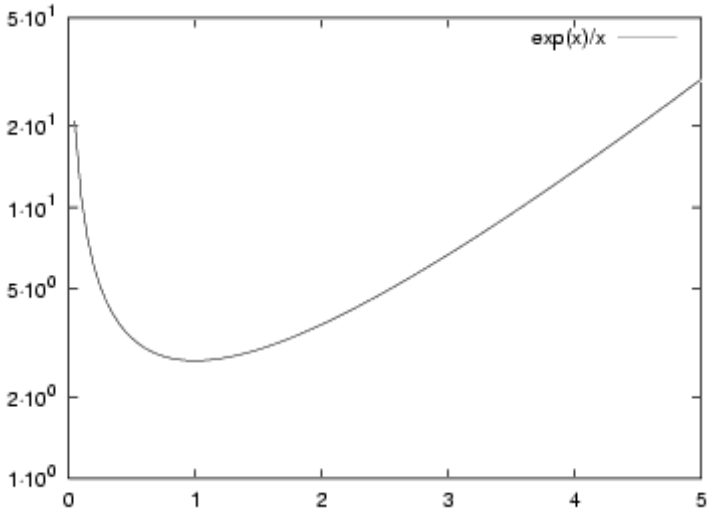

Figure 57: sample3.5

The Postscript symbol in the `set format` line becomes "center dot". To change it into "cross", use {`/Symbol 264`}.
The method explained here does not work on some systems. For example, when the number 50.0 is formatted with "%t" and "%L", some system gives the values 5.0 and 1 (namely 50.0 = 5.0E+01), so it works fine. But other systems give the value of 0.5 and 2 (50.0 = 0.5E+02). In this case the value on the graph axis becomes $0x10^2$. (zero in log-graph ????)

9. About Label

9.1. I want to use super/subscripts in a text

The enhanced postscript terminal can display super- / sub-script in the labels or axis names. To use this termianl one needs "enhanced" option:

```
gnuplot> set terminal postscript enhanced
```

You can write the supersript as X$\hat{2}$, and the subscript is Y_3. To make several letters super- / sub-script, you need brace like Z_64. To use super and subscripts at the same time, try Z@$\hat{2}$_64.
The following is an example to make legends with the superscripts. The functions are y=sin**2(x), y=sin**3(x), and y=sqrt(sin(x)).

```
gnuplot> set terminal postscript eps enhanced
gnuplot> set key spacing 1.3
gnuplot> set xrange [ 0 : pi ]
gnuplot> set yrange [ 0 : 1.5 ]
gnuplot> plot sin(x)**2 ti "sin^2(x)", \
         sin(x)**3      ti "sin^3(x)",\
         sqrt(sin(x))   ti "sin^{1/2}(x)"
```

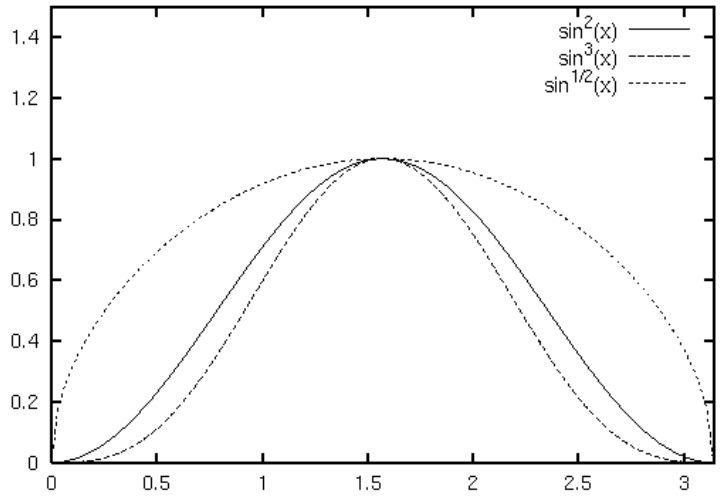

Figure 58: sample4.1

When texts in the legend contain super- / sub-scripts, text lines become very close each other. In the example above, the baseline skip was increased to 30% by `set key spacing 1.3`

9.2. I want to use Greek letters in a text

To use Greek letters in a text, you may try using enhanced postscript terminal just like the case of super- / sub-scripts.

```
gnuplot> set terminal postscript enhanced
```

The Greek letters can be displayed by {/Symbol a}. This gives "alpha" which corresponds to "a". The relation of the Symbol and alphabet is as follows.

ALPHABET	SYMBOL	ALPHABET	SYMBOL	alphabet	symbol	alphabet	symbol
A	Alpha	N	Nu	a	alpha	n	nu
B	Beta	O	Omicron	b	beta	o	omicron
C	Chi	P	Pi	c	chi	p	pi
D	Delta	Q	Theta	d	delta	q	theta
E	Epsilon	R	Rho	e	epsilon	r	rho
F	Phi	S	Sigma	f	phi	s	sigma
G	Gamma	T	Tau	g	gamma	t	tau
H	Eta	U	Upsilon	h	eta	u	upsilon
I	iota	W	Omega	i	iota	w	omega
K	Kappa	X	Xi	k	kappa	x	xi
L	Lambda	Y	Psi	l	lambda	y	psi
M	Mu	Z	Zeta	m	mu	z	zeta

You can also specify various postscript characters by octal codes, for example, /243 is a pound (L) mark, /247 is a section mark. See ps_guide.ps which comes with gnuplot source distribution in detail.
The next example is to draw the linear function y=Alpha x+ Gamma and two Greek letters with those values in the figure.

```
gnuplot> set terminal postscript eps enhanced
gnuplot> set xrange [ 0 : 5 ]
gnuplot> set label "{/Symbol a}=0.5, {/Symbol g}=0.2" at 2,0
gnuplot> plot 0.5*x-0.2 ti "y={/Symbol a}x-{/Symbol g}"
```

9.3. How do I adjust an interval between X,Y axes and their labels?

The interval can be controlled by the offset options of `set x|ylabel` command.

```
gnuplot> set xlabel "x" 0.0,1.0
```

This moves the X-axis label one letter upward. When you give a positive Y offset, the X label gets into the graph. While the offset is negative, distance between the X label and the graph becomes larger. Gnuplot tries to draw a graph as large as possible, so the graph hight becomes larger when the X-axis label moves upward. On the contrary the graph becomes small when the X-axis goes downward. Y offset = 0 Y offset = +5 Y offset = -5
To adjust the Y-axis label, use `set ylabel "Y-AXIS" +n,+m` where "+n" and "+m" are the Y-axis offset options. The following examples are to set the X offset "+5" and "-5". This affects the width of the graph. X offset = 0 X offset = +5 X offset = -5

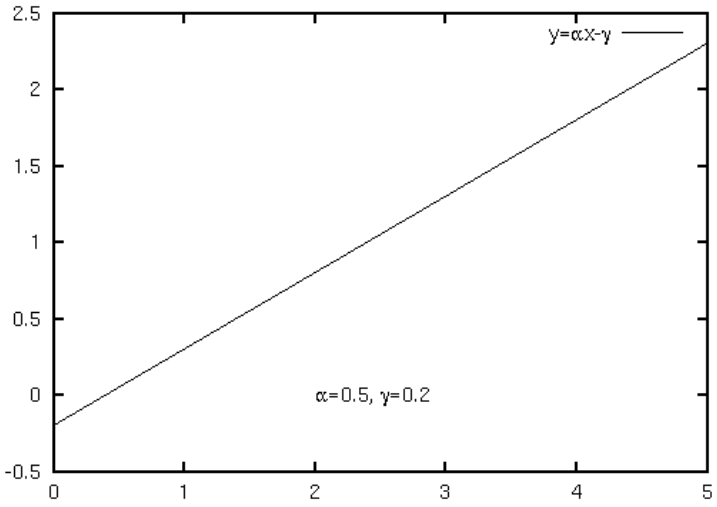

Figure 59: sample4.2

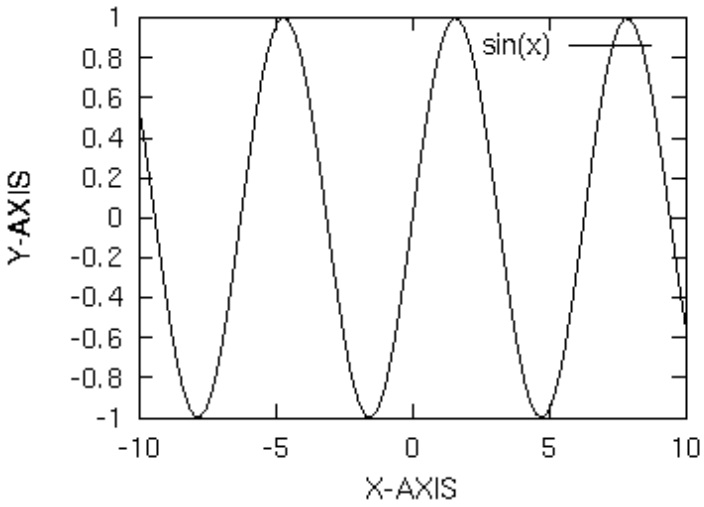

Figure 60: sample4.3

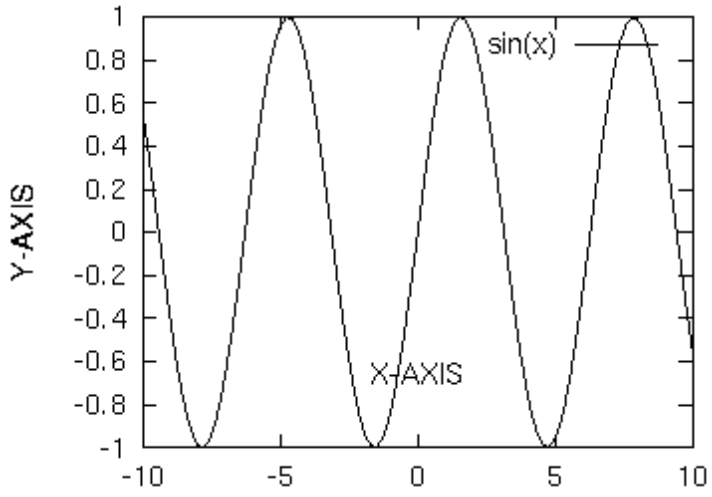

Figure 61: sample4.3a

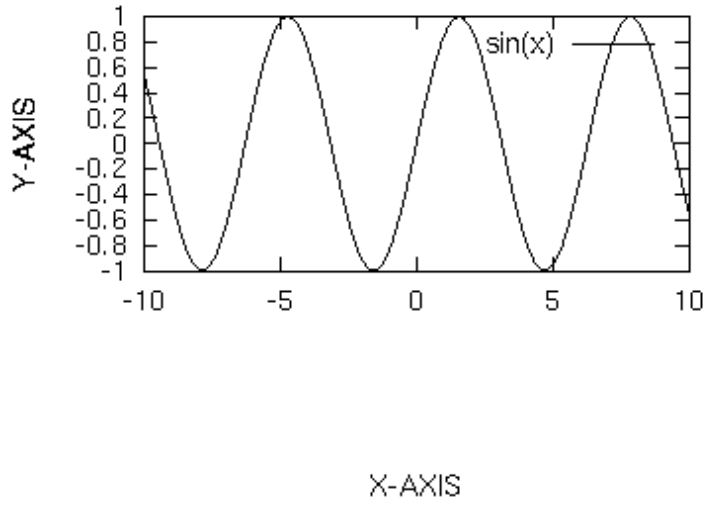

Figure 62: sample4.3b

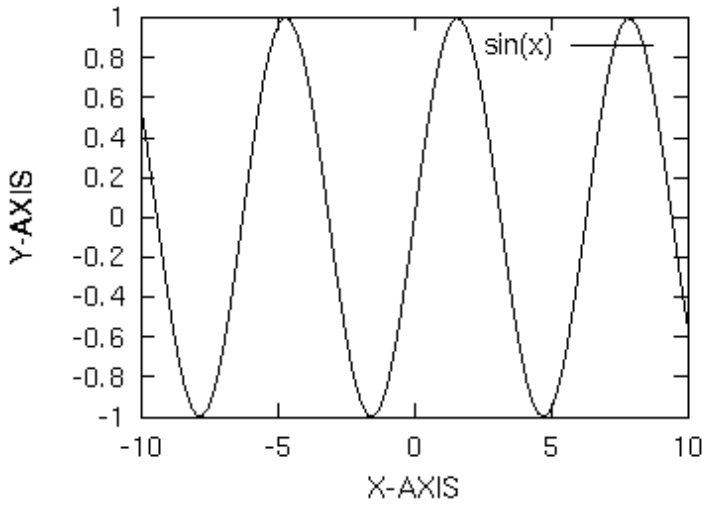

Figure 63: sample4.3

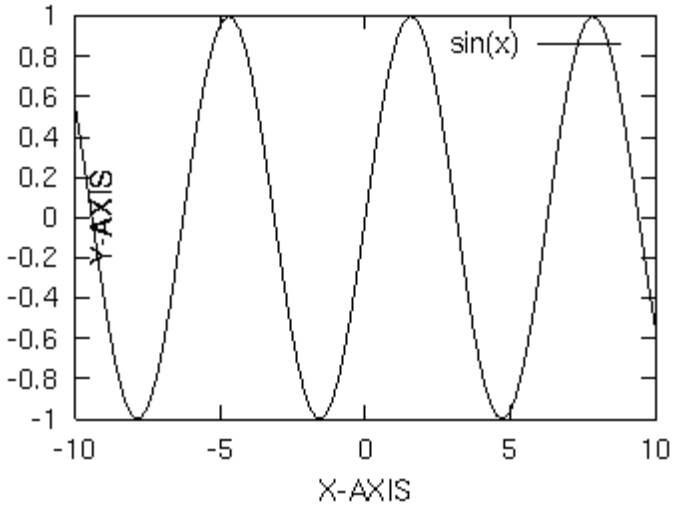

Figure 64: sample4.3c

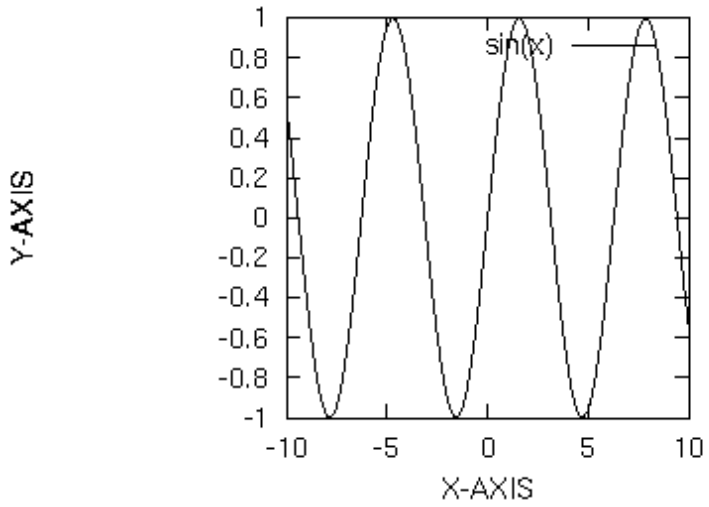

Figure 65: sample4.3d

10. About 2d Plot

10.1. I want to make a fixed-size plot

The area in which gnuplot draws a graph depends on axis and tic labels. To fix the size of the graph, you need to adjust margins by `set margin`. An option of this command is the number of letters. There are four kinds of margin, top (tmargin), bottom (bmargin), left (lmargin), and right (rmargin). As the defaults all margins are calculated automatically. Current settings can be displayed by:

```
gnuplot> show margin

        lmargin is computed automatically
        bmargin is computed automatically
        rmargin is computed automatically
        tmargin is computed automatically
```

The next shows an example in the case that all margins are given manually.

```
gnuplot> set lmargin 10
gnuplot> set bmargin 3
gnuplot> set rmargin 2
gnuplot> set tmargin 1
```

When the margins are defined explicitly like above, the size of graph does not change even formats of the X and Y tics are changed. To fix the position of axis labels, specify the offset options of `set x|ylabel` command.

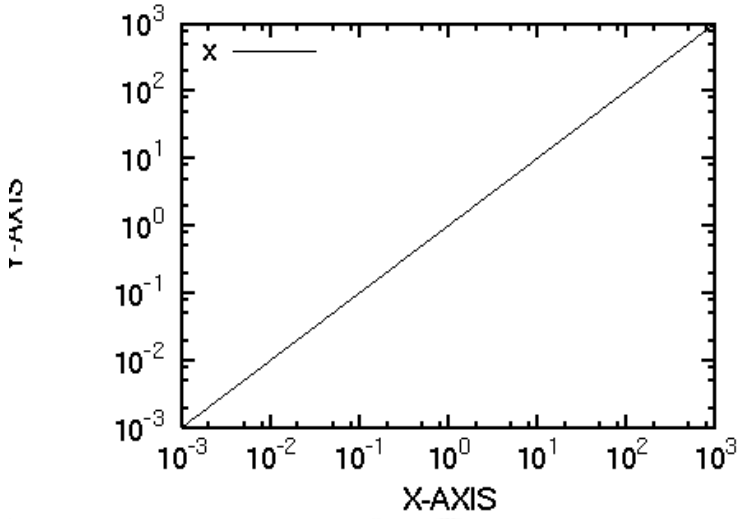

Figure 66: sample5.1a

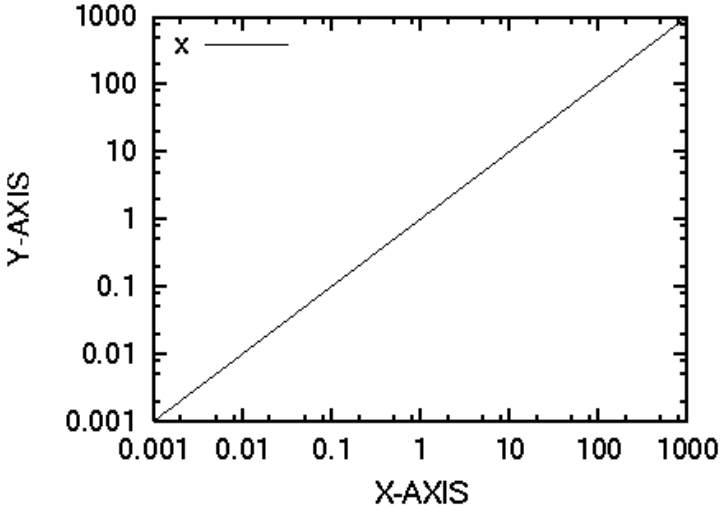

Figure 67: sample5.1b

10.2. I want to use both sides of Y-axes

It sometimes happens that one wants to put several graphs in one plot. With gnuplot you can use top and bottom axes, and left and right axes separately.
As the defaults Y2 axis is the same as the Y (left) axis. Let's make those two axes different, and plot sin(x) and its square at the same time. An option of `axis` in the `plot` command defines which axis is used for scaling. The syntax is `axis` + x1y1, x1y2, x2y1, x2y2. For example `axis x1y2` means that this function or data will be scaled by the bottom X-axis and the right Y-axis.

```
gnuplot> set xrange [0:2*pi]
gnuplot> set yrange [-1:1]
gnuplot> set y2range [0:1]
gnuplot> plot sin(x)    axis x1y1, \
              sin(x)**2 axis x1y2
```

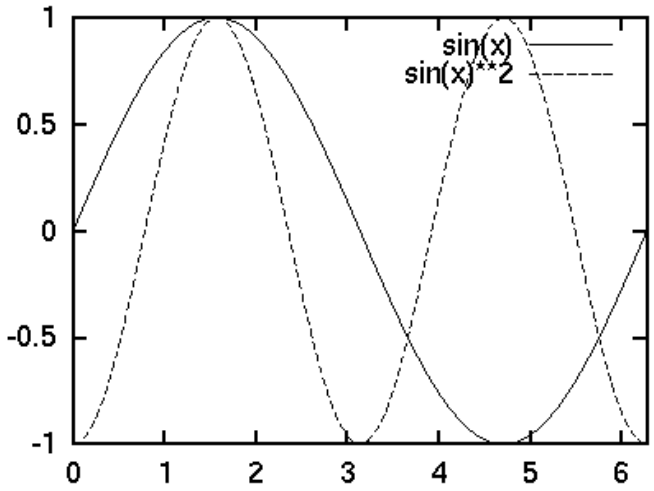

Figure 68: sample5.2a

In this case the tics on the right axis is the same as those on the left axis. To make those different one can use a `nomirror` option to the `set ytics` command, and `set y2tics` makes different axis on the right border.

```
gnuplot> set y2tics 0, 0.2
gnuplot> set ytics nomirror
```

When you use the both axes, you need to specify which axis is used for each data. An arrow is often used for this purpose. This can be done by `set arrow`.

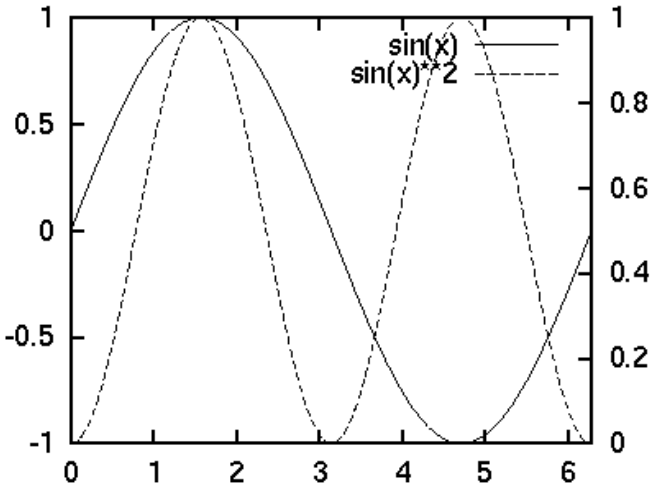

Figure 69: sample5.2b

10.3. I want to erase axes

Gnuplot draws top, bottom, left, and right frames. To erase those frames, use `set border` n . An integer is assigned for each frame line, the bottom is 1, left 2, top 4, and right 8. The option n is the sum of those integers. For example, only X1 axis is shown if n=1, X1 and Y1 axes for n=3, all four border lines for n=31. The `set border` command only affects the border lines, so that tics remain even if n=0. In order to erase axes, you need `set nox|ytics` or `set x|ytics nomirror`. The following example shows how to erase the top and right border.

```
gnuplot> set border 3
gnuplot> set xtics nomirror
gnuplot> set ytics nomirror
```

10.4. I want to draw a square or fixed aspect ratio figure

This subject had been known for the old version gnuplot as a very difficult matter, but it is supported nowadays. To make a square plot, give an option `square` to the `set size` command.

```
gnuplot> set size square
```

Similarly, to fix the aspect ratio:

```
gnuplot> set size ratio 2
```

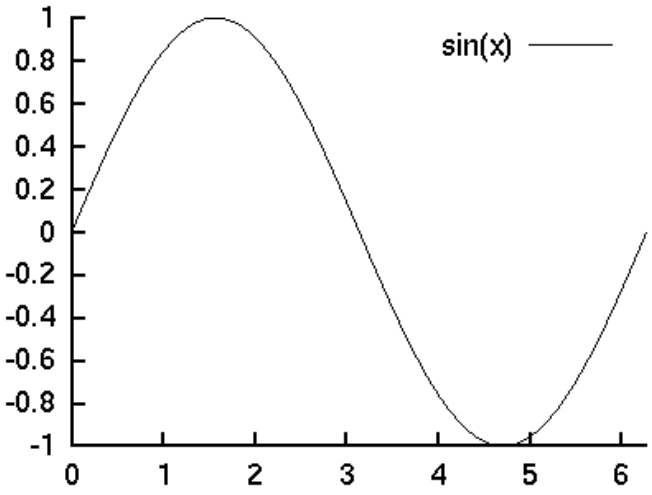

Figure 70: sample5.3

In this case the Y axis length is two times longer than the X axis. This ratio is independent of the values of the X and Y axes. In order to set the scales so that the unit has the same length on both the X and Y axes, give negative value for the ratio. If the ratio is -1, the unit length for the X axis is the same as that for the Y axis. If -2, the Y axis becomes two times longer.

10.5. I want to draw a zero axis

Use the `set x|yzeroaxis` command. If no option is provided, the zero-axis line is drawn by line-type 0 (dotted line). The options `ls` line_style, `lt` line_type, `lw` line_width control the style of the zero axis. In the case of `lt -1` , the line becomes the same as the border lines.

```
gnuplot> set xzeroaxis lt -1
gnuplot> set yzeroaxis
```

10.6. I want to get rid of small bars which appear at the top of error bars

Small bars are placed on the top (bottom) of the error bars when data with the errors are plotted. This lines are sometimes bothersome when the number of data is large. To get rid of them:

```
gnuplot> set bar 0
```

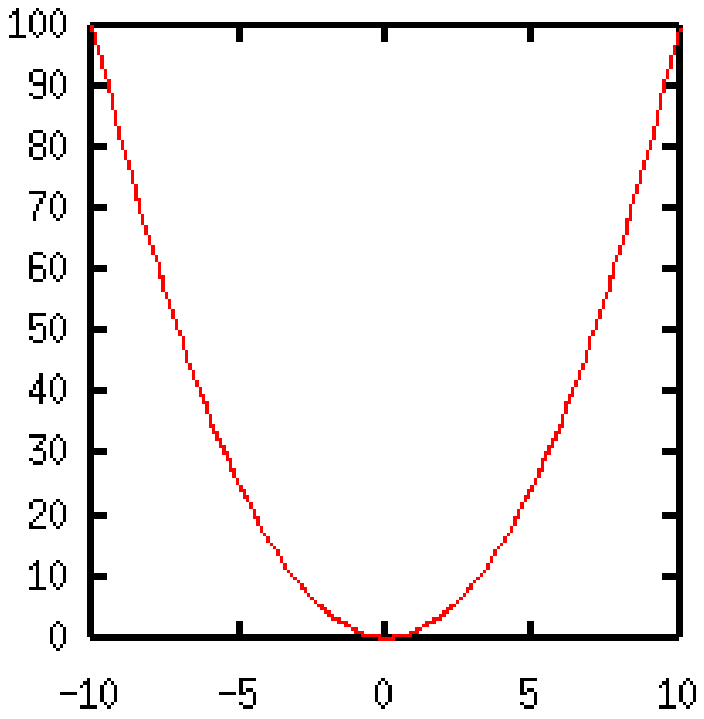

Figure 71: square

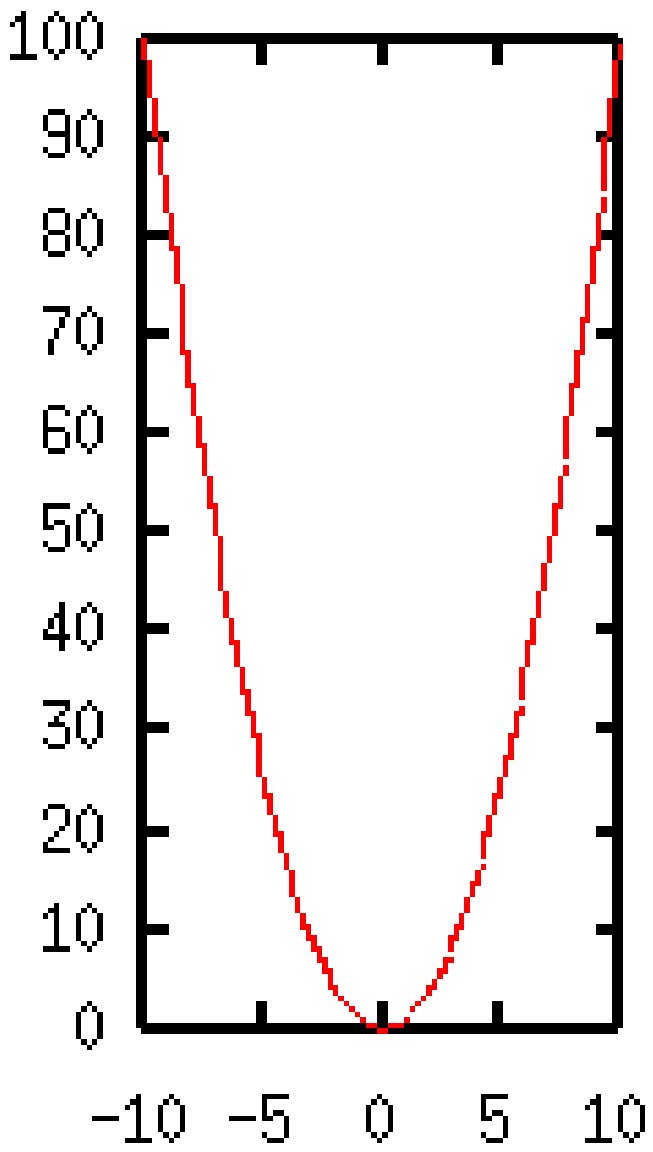

Figure 72: ratio 2

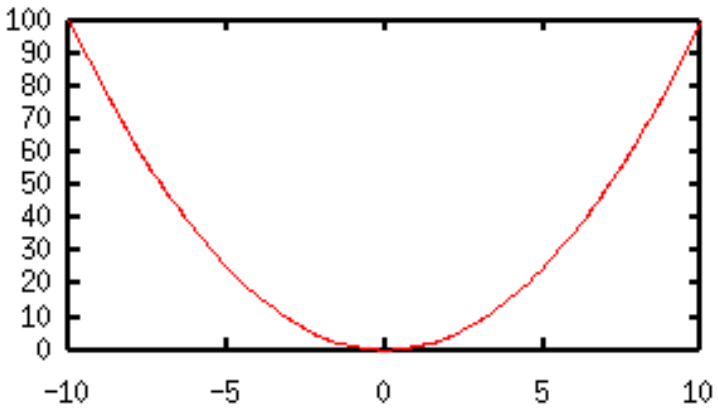

Figure 73: ratio 0.5

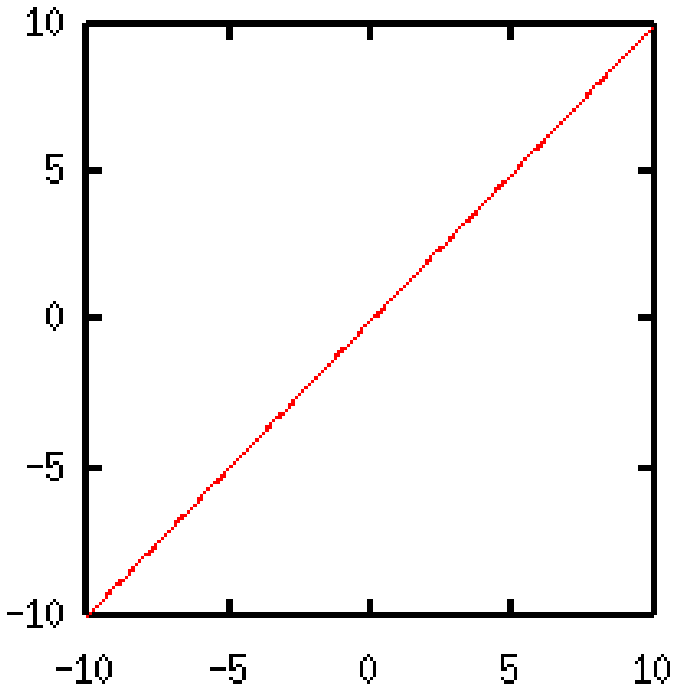

Figure 74: ratio -1

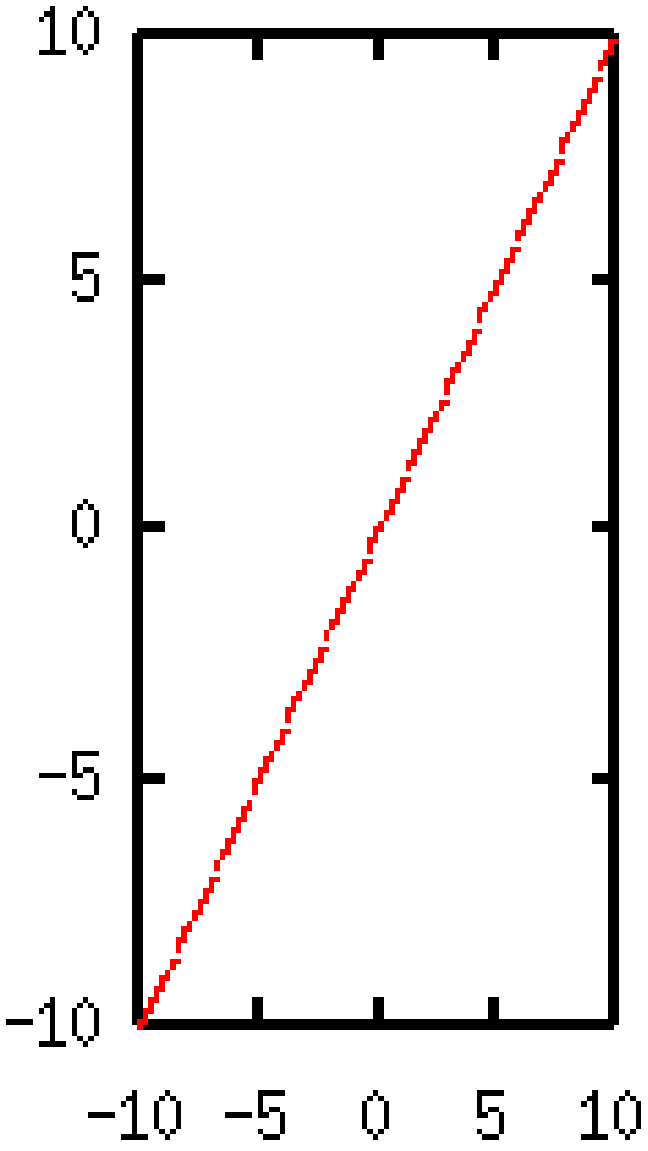

Figure 75: ratio -2

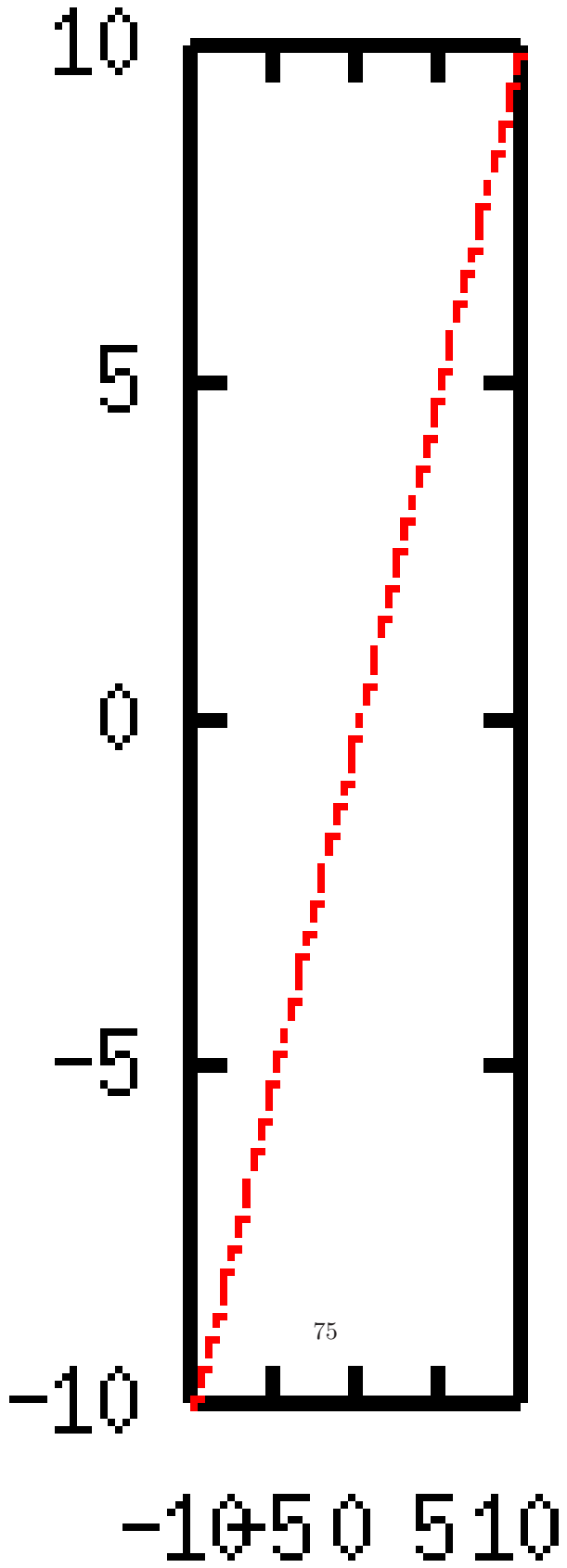

Figure 76: ratio -4

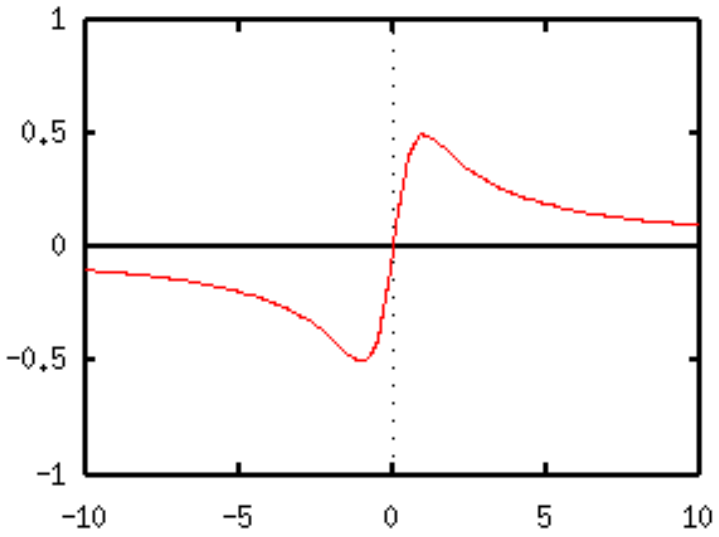

Figure 77: sample5.5

The option is the length (default 1) of the bar. If 0, the bar disappears.
Note that even if you change the size of points by `set pointsize`, the length of this bar does not change. You'd better to change the bar size at the same time.

```
gnuplot> set pointsize 3
gnuplot> set bar 3
```

10.7. I want to make letters larger

It depends on the devices with which you are plotting the figure. Commands of `set label` or `set title` accept a font option, so you can use the larger fonts if those are provided for the device. But it is usually difficult to control the size of fonts.
If you are using the postscript terminal, scaling of the font is easy. Instead of enlarging the letters, make the whole figure smaller. Then the font size becomes large relative to the figure size. It is possible to resize the PostScript figure, so that it is no problem even if your figure is too small.

```
gnuplot> set size 0.3,0.3
```

With the command above the whole size is reduced to 30%. An Encapsulated PostScript terminal is used, and the figure is enlarged again at printing. This case is rather extreme, but the reduction in the size of 0.5 – 0.7 yields a sufficient result.
If you want to control the font precisely in your PostScript figure, use the `font` options of label, title, and specify the font-shape and its size. An option for the `set terminal` command also has a default font size option. In the following example, we used 16pt

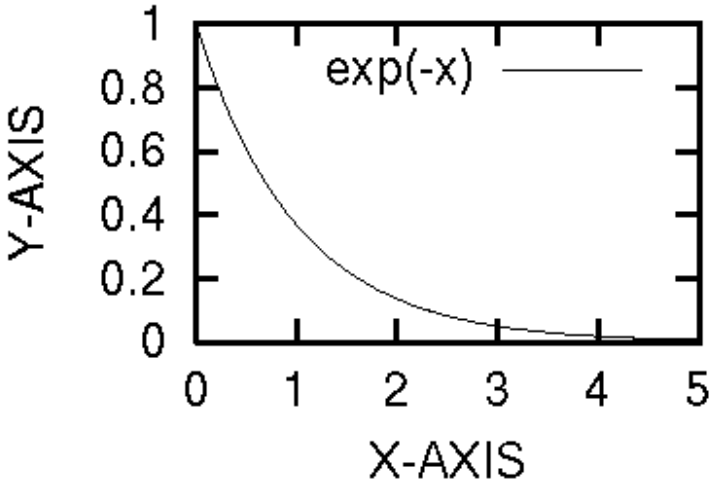

Figure 78: sample5.7a

Helvetica for the basic font, and the title and axis names are shown by the different fonts.

```
gnuplot> set terminal postscript enhanced "Helvetica" 16
gnuplot> set title "Damping Function" font "Times-Roman,40"
gnuplot> set xlabel "X-AXIS" font "Helvetica,20"
gnuplot> set ylabel "Y-AXIS" font "Times-Italic,32"
gnuplot> plot exp(-x)
```

Well, it can be done as shown above, but I don't think it is convenient. Although gnuplot generates a nice figure, you can decorate your figures with other tools like Tgif.

10.8. I want to connect all points with some smooth curves

Gnuplot has a provision for data smoothing with the cubic-splines or the Bezier curves. To display the smoothed curve, use the `smooth` option in the `plot` command. There is a difference between those smoothing methods. The spline function is an interpolation between the data points, while the Bezier curve is an approximation of the data trend. The following example is a comparison of the spline function and the Bezier curves. The same data are plotted in the three ways, the original data which are shown by the symbols, the curve smoothly interpolated with the spline function, and the Bezier curve.

```
gnuplot> plot "test.dat" using 1:2 notitle with points, \
>             "test.dat" using 1:2 smooth csplines \
```

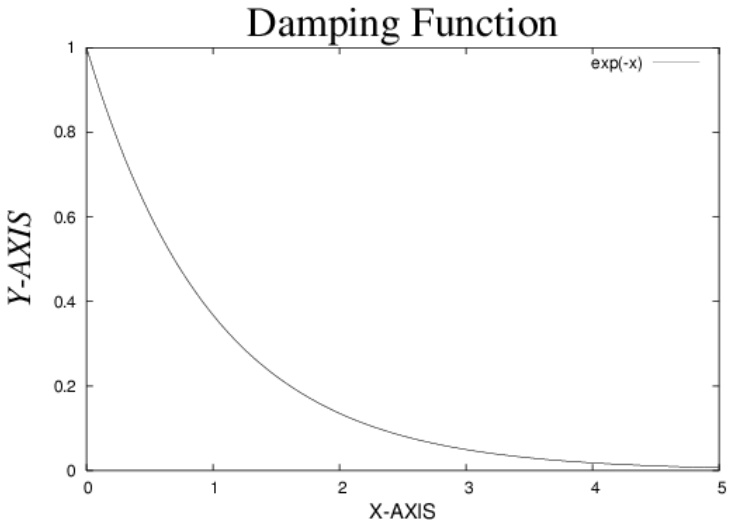

Figure 79: sample5.7b

```
>                                title "spline" with lines,\
>             "test.dat" using 1:2 smooth bezier \
>                                title "bezier" with lines
```

The spline option `csplines` connects all data points smoothly. On the other hand the Bezier curve is not an interpolation but it smoothes the data. The spline function can also be used for the data smoothing by the option `acsplines`, with which one can draw an approximation curve of the data. The example above the X and Y data are only needed, but to make an approximation curve one needs weights (uncertainties) of all data points. The next example shows how to smooth experimental data with the Bezier curve and the spline function.

```
gnuplot> plot "test.dat" using 1:2:3 notitle with yerrorbars, \
>             "test.dat" using 1:2:3 smooth acsplines \
>                                    title "acsplines" with lines,\
>             "test.dat" using 1:2   smooth bezier \
>                                    title "bezier"    with lines
```

The Bezier curve chases the variation of the data, but the spline function expresses a rough trend of them. Sometimes one needs to draw a curve by "eye guide" in the plot of experimental data. Gnuplot can do it very easily.

The weights of the data are needed to make the approximation curve with the spline. If the weights are the same for all the data points, you can give an equal weight 1.0 by `using 1:2:(1.0)`.

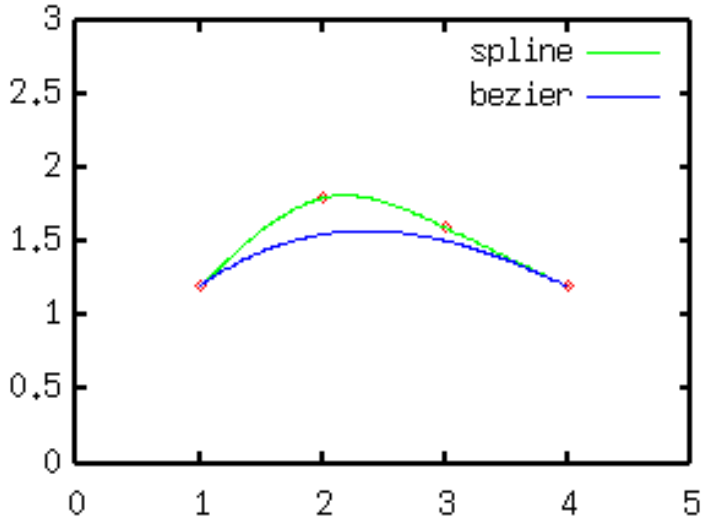

Figure 80: sample5.8a

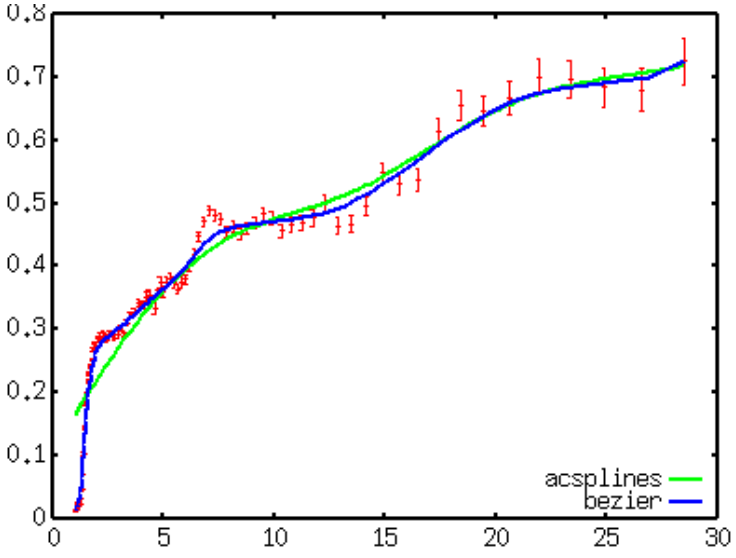

Figure 81: sample5.8b

10.9. I want to erase points those are on border

Data points or lines near the border line can be clipped. The command `set clip` controls the method of this data clip. There are three types of the data `clip`, `points`, `one`, and `two`. In order to explain the difference of those types, the following example is used.

```
#  X     Y
   1.0   1.0
   2.0   1.5
   3.0   2.0
   4.0   1.5
   5.0   1.0
```

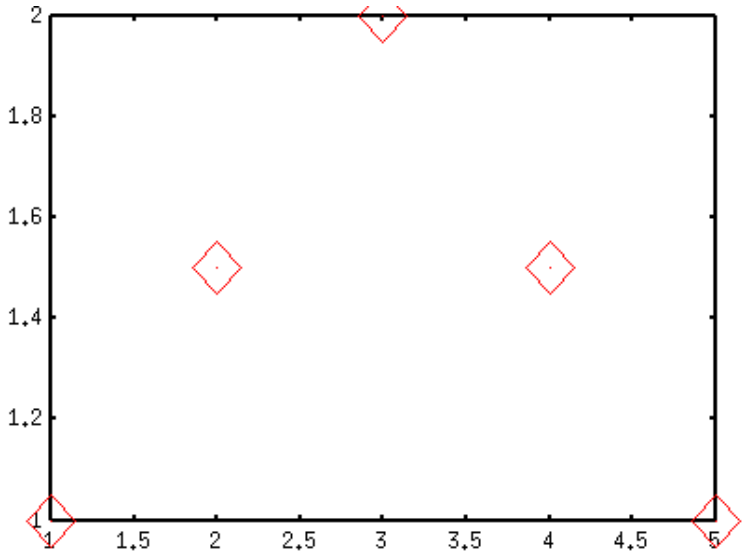

Figure 82: sample5.9a

When the clip is defined the points on the border (X=1, 3, and 4) disappear.

```
gnuplot> set clip points
gnuplot> plot "test.dat" notitle with points
```

By the way gnuplot clips data automatically if those are very close to the border lines. For example, even if the Y range in the above figure is enlarged to 2.1, the data point at X=3 is still clipped. I don't know the criteria of this — which point is clipped and which is shown.
The next clip type is `set clip one`, which defines a behavior of lines near the border. When there are two points, one is inside the graph and the other is outside, and if

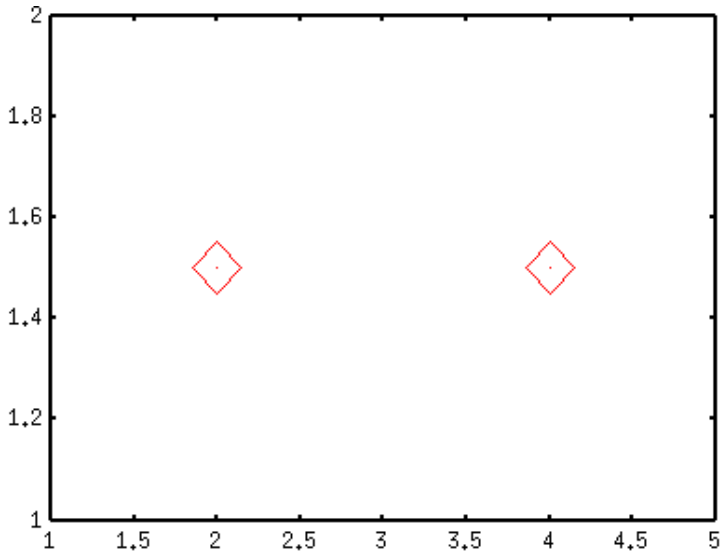

Figure 83: sample5.9b

those two points are connected by a line which crosses the border line, there are two choices to control this. The first one is to draw a line from the inside point to the border line and truncate the line there. This is the default. Alternatively such lines can be erased by `set noclip`.
When a function is displayed, gnuplot calculates the X,Y values at certain points which are defined by `sampling rate` (100, default), and those points are connected by small lines. So that gnuplot draws truncated-lines instead of a curve. If some point is outside the graph, the line crosses the border line. The command `set clip one` specifies that the line is erased (noclip), or that the line is partly drawn from the end point to the border line (clip).
When the function y=sin(x) is drawn in a figure and the Y range is [0:1], the curve crosses the X axis several times. The function is displayed by `with linespoints`, then the points shown by a symbol are those gnuplot actually calculated.

```
gnuplot> set clip one
gnuplot> plot sin(x) with linespoints
gnuplot> set noclip one
gnuplot> replot
```

The last type of the clip is `set clip two`. This also controls when a truncated-line crosses the border line, but this is the case for that the both points are outside the graph. The default is `clip`. See the following example.

```
gnuplot> set yrange [-0.5:0.5]
gnuplot> set samples 10
```

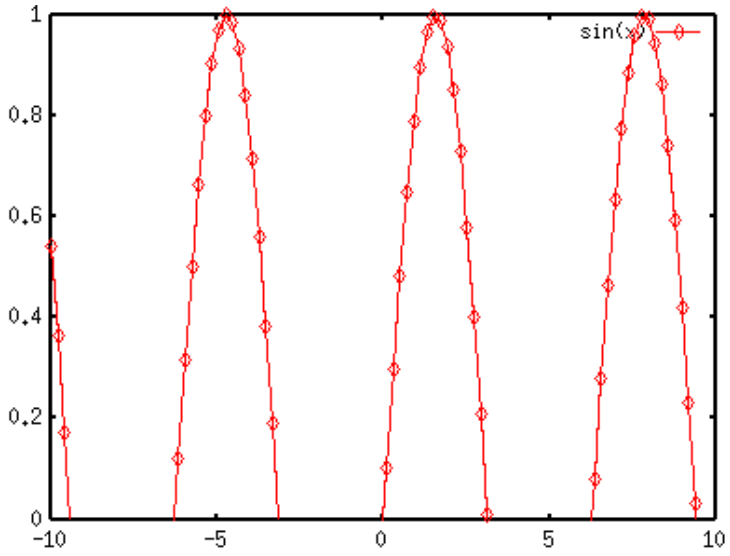

Figure 84: clip one

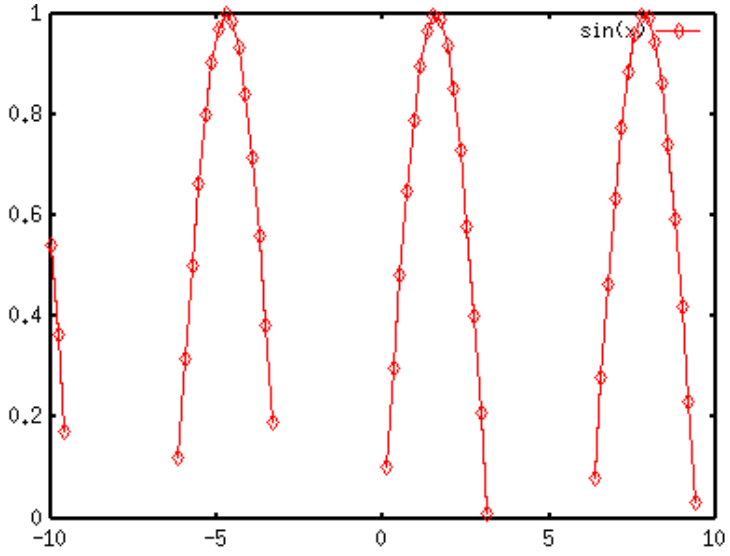

Figure 85: noclip one

```
gnuplot> set clip two
gnuplot> plot sin(x) with linespoints
gnuplot> set noclip two
gnuplot> replot
```

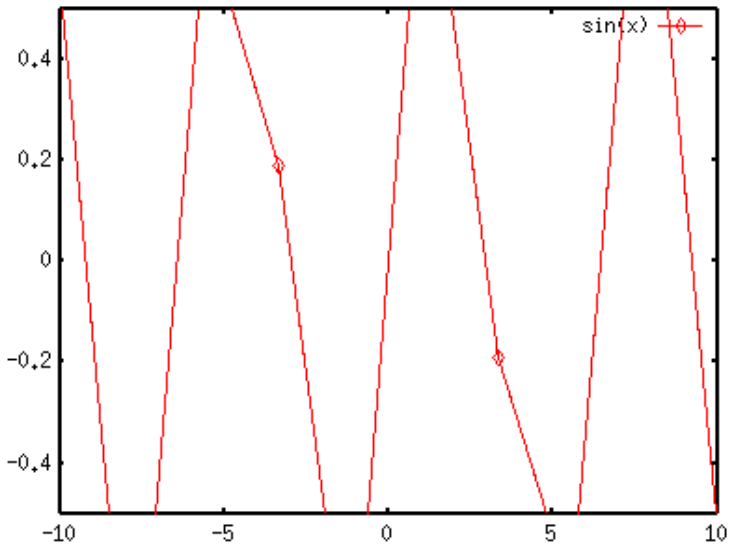

Figure 86: clip two

10.10. I want to draw several figures on one drawing

There are two methods to make a drawing which contains several figures. The first one is to use `multiplot`. The other way is to use the EPS format, then those files in this format are gathered by TeX or some drawing tools. Here we explain the `multiplot` command.
With the command `set multiplot` you enter the multi-plot mode, and the gnuplot command prompt becomes `multiplot>`. In this mode a new figure overlaps the old one. The next example shows the overlapping plot of three functions, y=x, y=x*x, and y=x*x*x.

```
gnuplot> set multiplot
multiplot> plot x
multiplot> plot x*x
multiplot> plot x*x*x
multiplot> set nomultiplot
```

Gnuplot determines the range, tics, and size of figure automatically for each plotting, then the plot gets into a mess. Let's try to fix the X and Y ranges [-10:10], and erase the legends by `no key` command since they overlap.

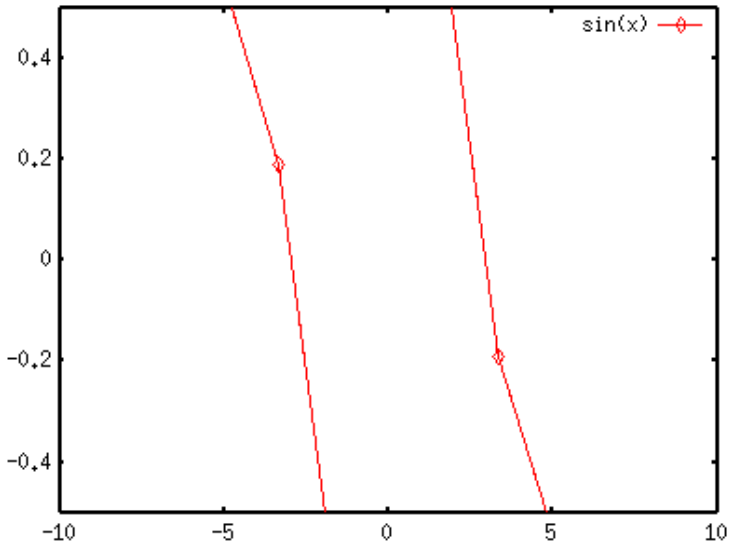

Figure 87: noclip two

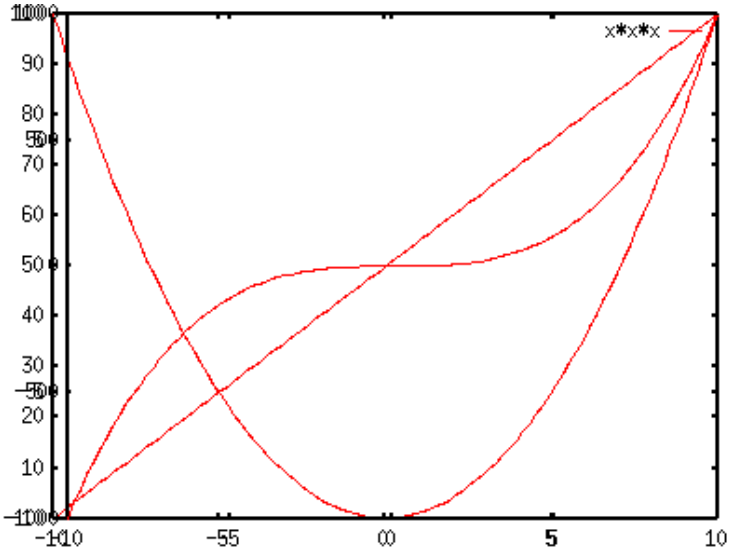

Figure 88: sample5.10a

```
gnuplot> set xrange [-10:10]
gnuplot> set yrange [-10:10]
gnuplot> set nokey
gnuplot> set multiplot
multiplot> plot x
multiplot> plot x*x
multiplot> plot x*x*x
multiplot> set nomultiplot
```

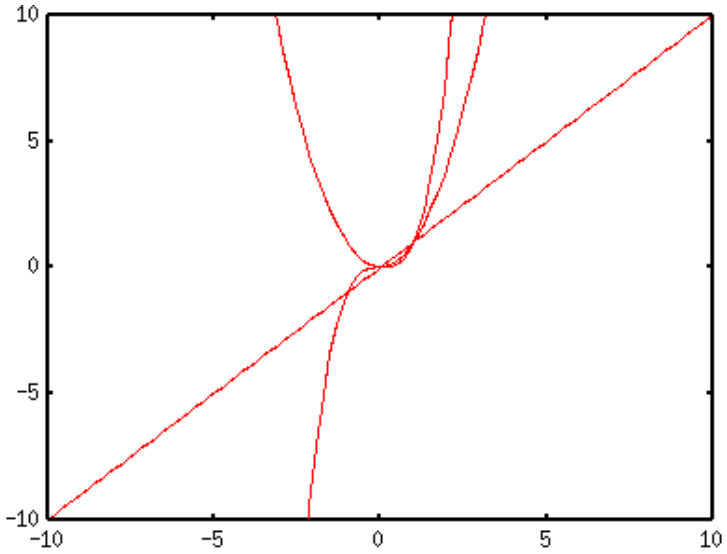

Figure 89: sample5.10b

Well, it's much better. But such thing can be done by usual plotting procedures. With the `multiplot` command you can draw several figures in one page by moving the origin of each figure. To do that, use `set size` and `set origin` commands.
As an example of multiplot, four Lissajous' figures are plotted in one drawing. The Lissajous's figure can be drawn by a parametric representaion of two functions, x=sin(n*t) and y=sin(m*t). Here we draw the case of n=3,5 and m=2,4.

```
gnuplot> set parametric

        dummy variable is t for curves, u/v for surfaces
gnuplot> set noxtic
gnuplot> set noytic
gnuplot> set nokey
gnuplot> set size square 0.3,0.3
gnuplot> set rmargin 0
```

```
gnuplot> set lmargin 0
gnuplot> set tmargin 0
gnuplot> set bmargin 0
gnuplot> set multiplot
multiplot> set origin 0.1,0.1
multiplot> plot sin(3*t),cos(2*t)
multiplot> set origin 0.1,0.5
multiplot> plot sin(3*t),cos(4*t)
multiplot> set origin 0.5,0.1
multiplot> plot sin(5*t),cos(2*t)
multiplot> set origin 0.5,0.5
multiplot> plot sin(5*t),cos(4*t)
multiplot> set nomultiplot
```

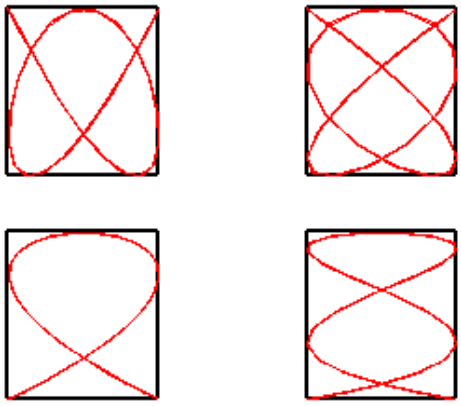

Figure 90: sample5.10c

In order to make those figures the same size, the tics are erased by `noxtic` and `noytic`, and the four margins are set to be zero. Usually it is difficult to adjust the position of each figure appropriately. There is no easy way to determine the values of `origin`, and you have to do it by "trial-and-error". When your figure has tics and labels, the size of figure also depends on them. The adjustment of the positions becomes very severe for this case.

10.11. I want to draw a grid at minor tics

The command `set grid` makes grids at the major tics position. In order to make them at the minor tics:

```
gnuplot> set mxtics 5
gnuplot> set mytics 5
gnuplot> set grid xtics ytics mxtics mytics
```

This gives five grids at the minor tics between the major ones. You can specify the grid position by `xtics ... mytics` options of the `set grid` command. If `set grid noxtics noytics mxtics mytics`, the grids appear at the minor tics position only (but I guess no one likes such a graph though...)

10.12. I want to draw two axes in the different scale

Sometimes one wants to make two axes in one figure — for example, the X-axis is a temperature which is represented by both Celsius (C) and absolute (K). There are several ways to do it, for example, display two figures by multiplot, or use the opposite axis. Here is an example of use of `multiplot`.
Let's make two X-axes. At first, make the figure height smaller, set the bottom margin zero, then enter the multiplot mode.

```
gnuplot> set size 1.0,0.7
gnuplot> set bmargin 0
gnuplot> set yrange [0:5]
gnuplot> set multiplot
```

Raise the figure slightly, and plot the first graph. In a following example we draw a function y=exp(x)-1. The unit of x is "minutes".

```
multiplot> set origin 0,0.3
multiplot> set xrange [0:2]
multiplot> set xtics 1
multiplot> set xlabel "Time [min]"
multiplot> plot exp(x)-1 notitle
```

Now, lower the figure, and draw the X-axis only. To avoid overlapping, adjust tics by `set noytics`, `set xtics nomirror` . Finally, plot a function which does not fit to the current drawing (I mean, it is outside the graph). In this example it was y= -1

```
multiplot> set origin 0,0.15
multiplot> set xrange [0:120]
multiplot> set xtics nomirror 30
multiplot> set noytics
multiplot> set xlabel "Time [sec]"
multiplot> set border 1
multiplot> plot -1 notitle
```

```
multiplot> set nomultiplot
gnuplot>
```

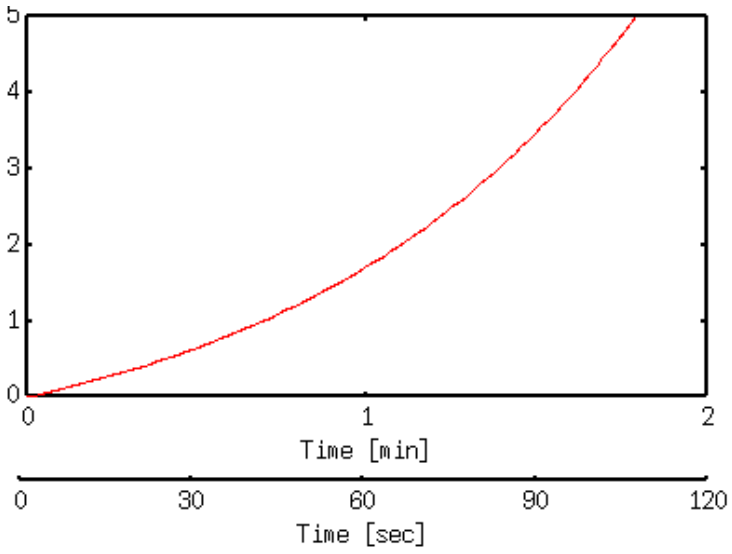

Figure 91: sample5.12

Well... probable there are more elegant ways to do it than this example, but practically it is no problem. It is not easy to adjust precisely the distance between two X-axes. To make it easier, add a new axis with other drawing programs other than gnuplot such as Tgif.

10.13. I want to make grid at an arbitrary position

The command `set grid` makes grid lines at the major tics positions, and those are equidistant usually. To make the grid at an arbitrary position, adjust the major tics position by the command `set tics`.
In the following example, three tics are displayed at 0.5, 1.2, and 2.5 by the `set xtics` command.

```
gnuplot> set xtics ("0.5" 0.5, "1.2" 1.2, "2.3" 2.3)
gnuplot> set grid
```

In order to make the arbitrary grid keeping the tic position equidistant, use the opposite axis. Choose the grid positions by `set x2tics`, then make the grid according to the X2 axis by `set grid` command option.

```
gnuplot> set xtics 0,1
gnuplot> set x2tics ("0.5" 0.5, "1.2" 1.2, "2.3" 2.3)
gnuplot> set grid noxtics x2tics
```

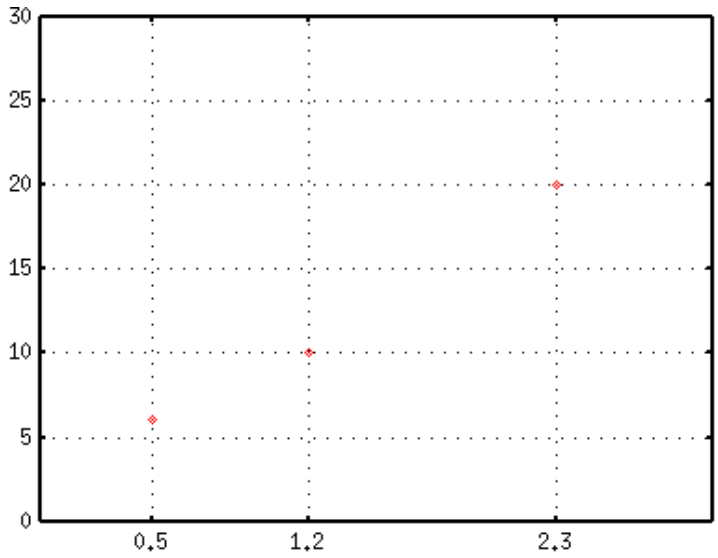

Figure 92: sample5.13a

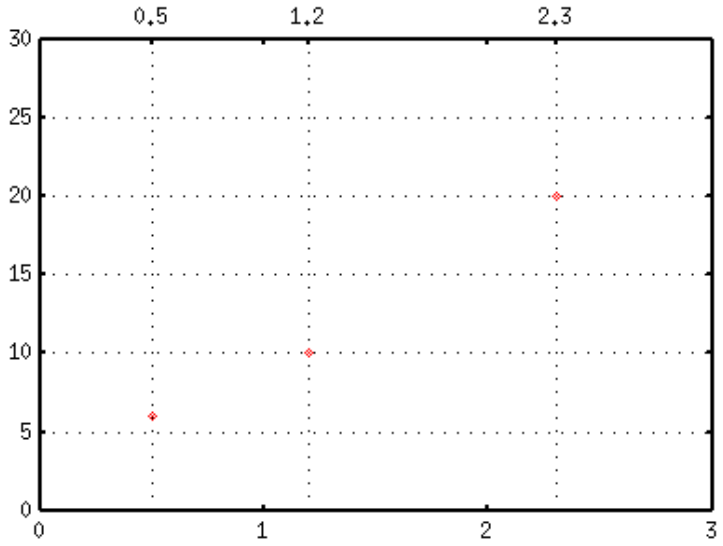

Figure 93: sample5.13b

If you want to remove numbers at the top of this figure, `set x2tics "" 0.5`. Those numbers disappear but X2 major tics still remain.

10.14. A small figure in a figure

Making a small figure inside a figure: Sometimes you may see a magnified plot in a margin to make a crowded part clear. To put such a magnified window, use multiplot. We make about 1/4 of the full-size figure inside the main plot. Firstly draw a main graph in the multiplot mode.

```
gnuplot> set xrange [ 0 : 20 ]
gnuplot> set yrange [ 0 : 6 ]
gnuplot> set xtics 5
gnuplot> set ytics 1
gnuplot> set multiplot
multiplot> set origin 0.0,0.0
multiplot> set size 1.0,1.0
multiplot> plot "file.dat" u 1:2:3 notitle with yerrorbars,\
>               "file.cal" u 1:2   notitle with lines
```

Move the origin to the vacant place (0.45,0.1), and draw a small graph there. The X and Y ranges should be determined to enlarge the place where you want to magnify. The small figure is the same as the main one except for the ranges, so that you can use the `replot` command.

```
multiplot> set origin 0.45,0.1
multiplot> set size 0.5,0.5
multiplot> set xrange [ 1 : 5 ]
multiplot> set yrange [ 2.4 : 3.0 ]
multiplot> set ytics 0.5
multiplot> replot
multiplot> set nomultiplot
gnuplot>
```

Now you can see clearly the part where there exists many data points. We chose the origin and size so as not to overlap those two figures. It is impossible to erase lines and points of the main figure which intrude the small one with gnuplot.
X and Y axis names for the small graph should be empty, even the main figure has those stuff. When you want to make a reduced size EPS, the `set size` command should be outside the multiplot mode. Otherwise gnuplot defines BoundingBox in your Postscript file so as to include a whole screen. You can see an example in our gallery.

10.15. A simple bar-chart

Gnuplot (if older than ver.3.6) draws open rectangulars when `with boxes` is used to make a bar-chart, as you can see an example. Gnuplot ver.3.8 and newer can draw a

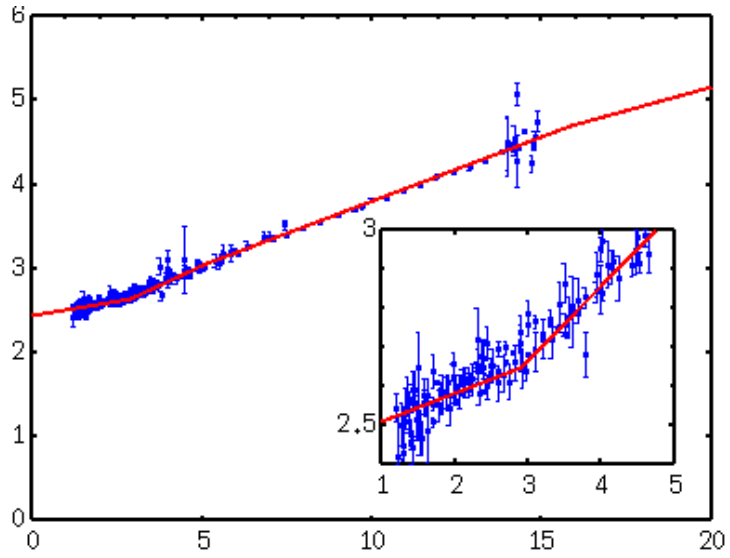

Figure 94: sample5.14

filled box.
If you are making a Postscript figure, the easiest way is to draw a figure `with impluses` with very thick lines. In the next example we defined the line type 1 which is 50 times thicker than the default width.

```
gnuplot> set term postscript eps enhanced color
gnuplot> set linestyle 1 lt 1 lw 50
gnuplot> # for gnuplot ver.4
gnuplot> # set style line 1 lt 1 lw 50
gnuplot> plot "test.dat" using 1:2 with imp ls 1
```

Gnuplot Ver.4 has an option to fill the boxes, which is `with boxes fs [pattern | solid] (fillstyle)`. Also you can use `set style fill` to define the color-fill pattern separately.
In the case of `with boxes fs pattern`, the pattern number is used to specify the filling pattern. In the case of `solid` option, density of the filled box is given by a value ranging from 0 to 1. In the example below, (3) stands for the bar-width.

```
gnuplot> plot "test.dat" usi 1:2:(3)      w boxes fs pattern 1,\
gnuplot>      "test.dat" usi ($1+5):2:(3) w boxes fs solid 0.7
```

10.16. Display a graph with normal probability axis

In a field of powder or aerosol engineering, they often plot the size distribution of particles. It often shows a "log normal distribution", which becomes a straight line on

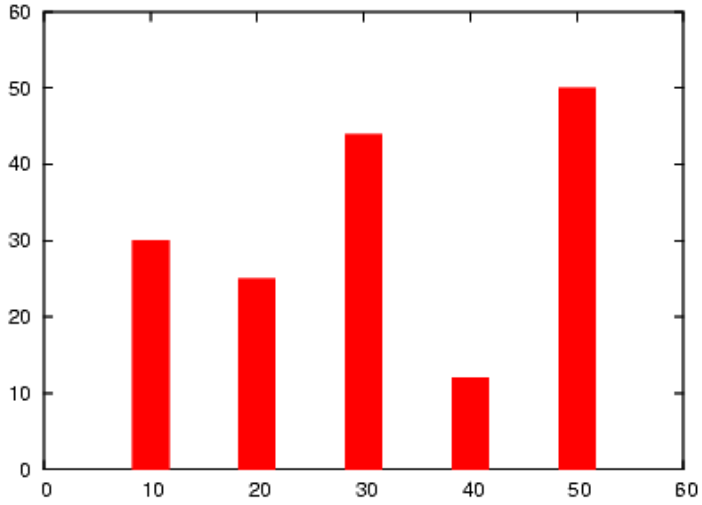

Figure 95: sample5.15

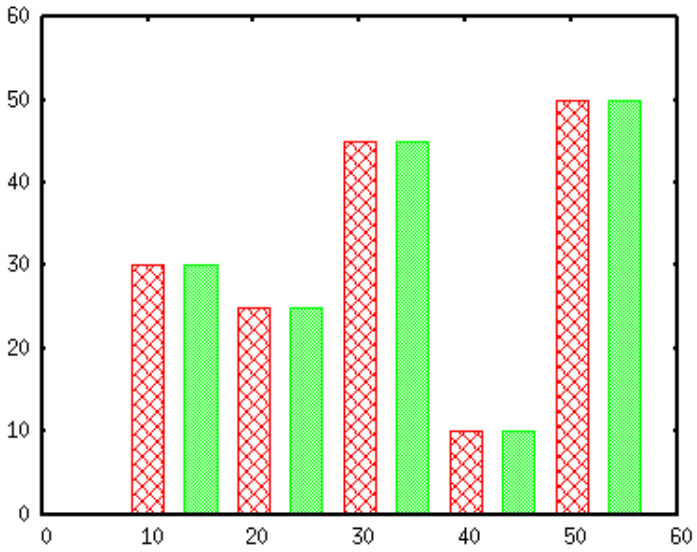

Figure 96: sample5.15b

the "normal probability"(fraction) vs log(size) plot, and so called "normal probability axis" is used. With two built-in functions - the normal distribution norm() and the inverse normal distribution invnorm() - of Gnuplot one can draw the normal probability axis. The following script shows how to make "normal probability axis" for y. Other modified scales can be made in the same way.

```
gnuplot> set ytics ("0.1" invnorm(0.001),"1" invnorm(0.01),"5" invnorm(0.05),\
>          "10" invnorm(0.1),"20" invnorm(0.2),"30" invnorm(0.3),\
>          "40" invnorm(0.4),"50" invnorm(0.5),"60" invnorm(0.6),\
>          "70" invnorm(0.7),"80" invnorm(0.8),"90" invnorm(0.9),\
>          "95" invnorm(0.95),"99" invnorm(0.99),"99.9" invnorm(0.999))

gnuplot> yrange [invnorm(0.0001):invnorm(0.9999)]
gnuplot> set ylabel "Cumulative mass fraction (%)"
gnuplot> set xlabel "Diameter (mm)"
gnuplot> set logscale x
gnuplot> set grid
gnuplot> plot "sample.dat" using 1:(invnorm($2)) notitle w lp
```

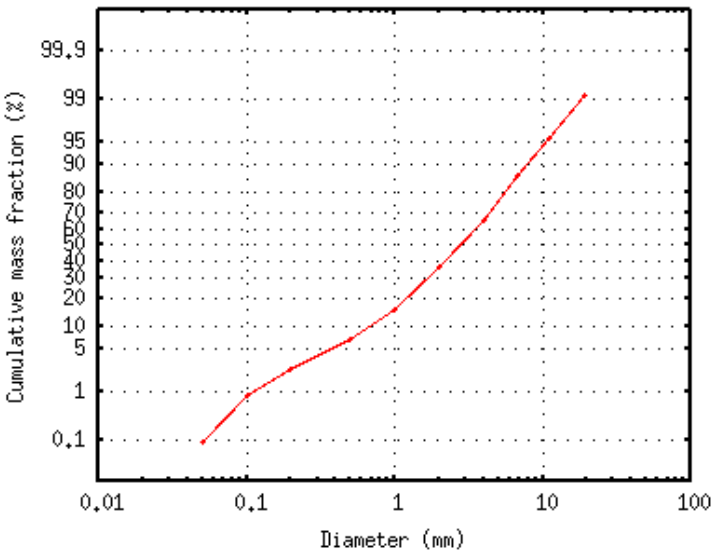

Figure 97: sample5.17a

In this example the X-axis is in the log-scale since the log-normal distribution is assumed for the data.

10.17. How can I print values at each datapoint?

Sometimes it is helpful if values of the points are shown numerically near those points. However gnuplot cannot treat the values in a data file as a text, we have to read those numbers with some program, and show them by `set label`. Suppose we have the following data, those data are shown by points, and we print those values near the points.

```
  0.40  80
  3.00  70
  3.00  60
  6.00  50
  9.00  40
 12.00  30
 13.00  20
  5.00  20
  8.00  20
```

Since we cannot make it with gnuplot alone, here we generate the `label` commands with Perl. Gnuplot needs the next lines.

```
gnuplot> set label "(0.4,80)" at 0.4,80
gnuplot> set label "(3.0,70)" at 3.0,70
```

Those lines can be generated as:

```
 % perl -ane 'print "set label \"($F[0],$F[1])\" at $F[0],$F[1]\n"' file.dat > label.plt
```

Now we read those gnuplot commands at plotting, the numerical values are shown at each point.

```
gnuplot> load "label.plt"
gnuplot> plot 'file.dat' u 1:2 smooth csp with lines ,\
>             'file.dat' u 1:2 w points
```

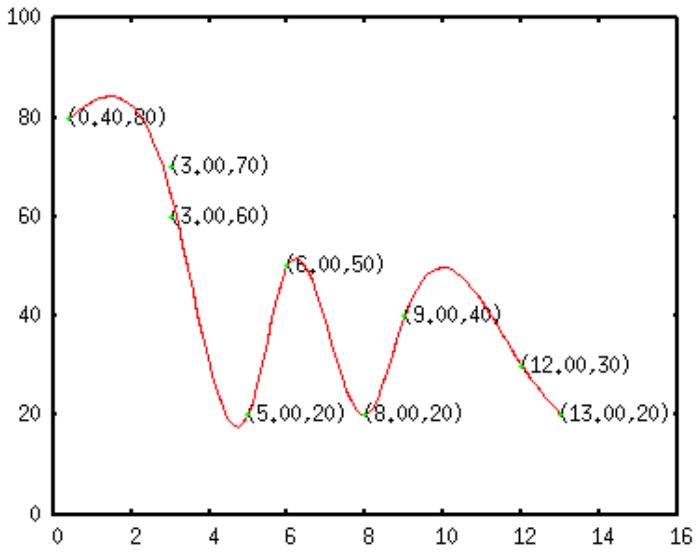

Figure 98: sample5.18

11. About 3d Plot

11.1. Why the origin of Z-axis is not on the XY-plane?

In the `splot`, the zero-point of Z-Axis is placed above the XY plane. To move this origin onto the XY plane, use `set ticslevel`. When ticslevel=0, the zero of Z-Axis moves to on the XY plane.

```
gnuplot> set ticslevel 0
gnuplot> splot (x**2)*(y**2)
```

11.2. I want to make a surface mesh finer

In a 3-dimensional plot, mesh size (XY coordinate) is determined by `isosample`. The smaller this number, the more rough mesh you get. The default is 10.
If the mesh is too rough, it sometimes makes trouble when a hidden line option `set hidden3d` is used. The following example shows a 3-dimensional plot of the function z=sin(x)*cos(y) with the isosample of 10. The top drawing is without the hidden line removal, and the lower is with it. To avoid this problem, make isosample larger. The next we set it to 40. To make X and Y meshes different, try `set isosample 20,40`.

```
gnuplot> set isosample 40
```

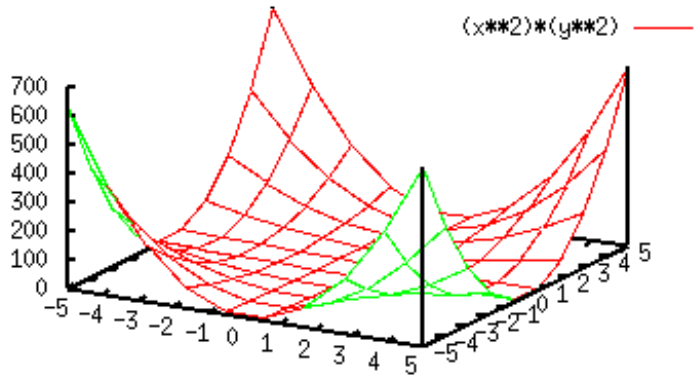

Figure 99: sample6.1

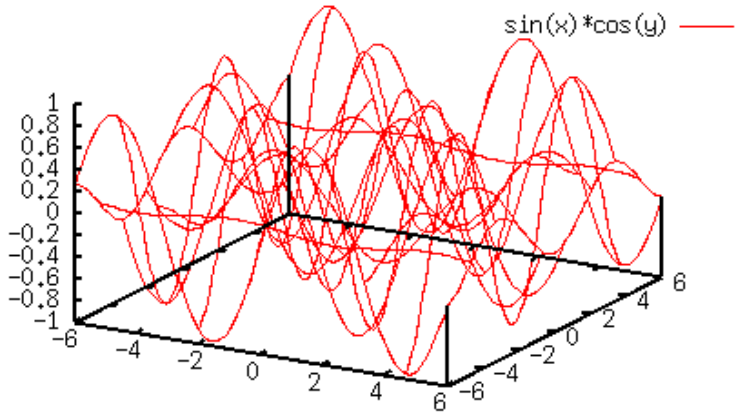

Figure 100: sample6.2a

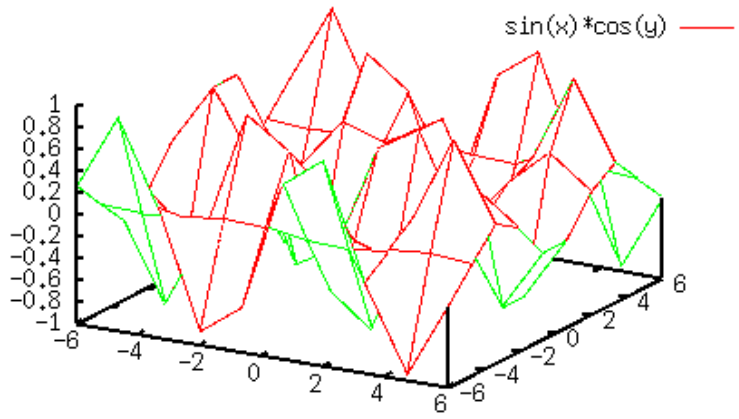

Figure 101: sample6.2b

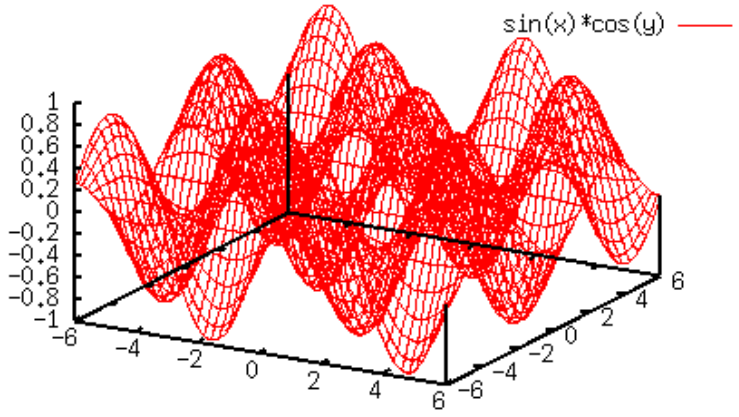

Figure 102: sample6.2c

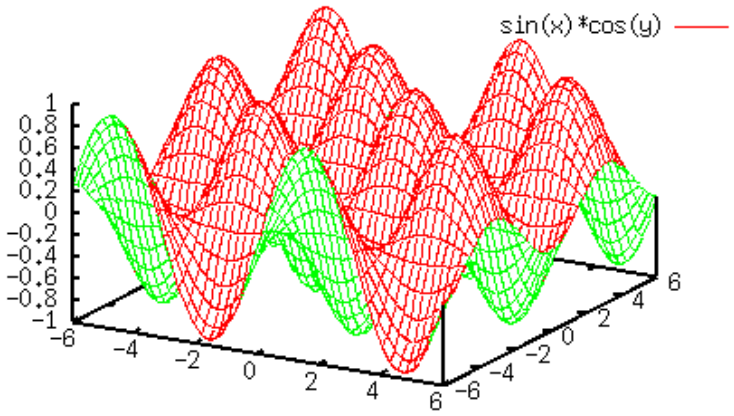

Figure 103: sample6.2d

11.3. How do I change a view point?

The view point in the 3-dimensional plot is controlled by `set view` command. The default setting is,

```
gnuplot> show view

        view is 60 rot_x, 30 rot_z, 1 scale, 1 scale_z
```

Initially (before rotation), your screen is parallel to the X-Y plane and the Z-axis is perpendicular to that. Firstly, the X-axis is rotated to 60 degrees (rot_x), then the

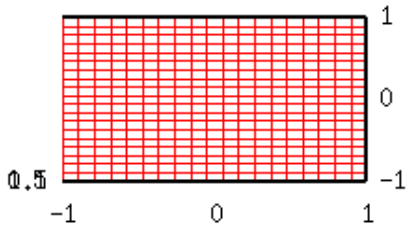

Figure 104: sample6.3a

Z-axis slants. Next, the new Z-axis is rotated to 30 degrees (rot_z). This is the view

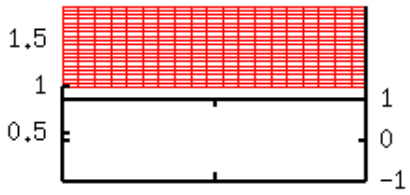

Figure 105: sample6.3b

point which splot sets it as default. The `set view` command rotates the graph.

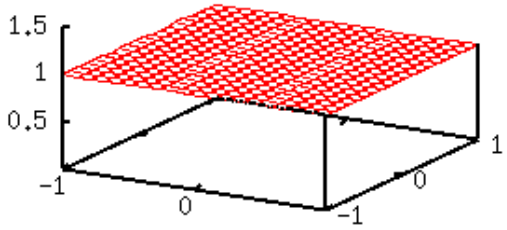

Figure 106: sample6.3c

11.4. How do I change contours?

Contour lines can be controlled by the `set cntrparam` command.

```
gnuplot> set contour
gnuplot> set cntrparam levels 10
gnuplot> set cntrparam levels incremental -1, 0.2, 1
gnuplot> set cntrparam levels discrete -0.2, -0.5, 0.2, 0.5
```

The above example shows three ways to control the contours. The keyword `levels` defines the level to which contour curves are drawn. The above, `levels 10` indicates

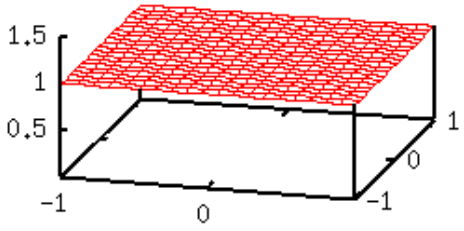

Figure 107: set view 60,15

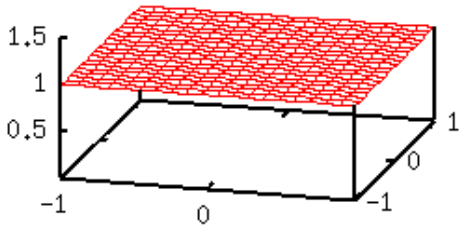

Figure 108: set view 60,45

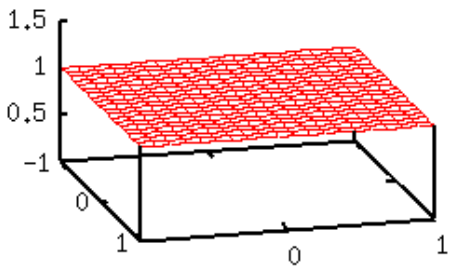

Figure 109: set view 60,75

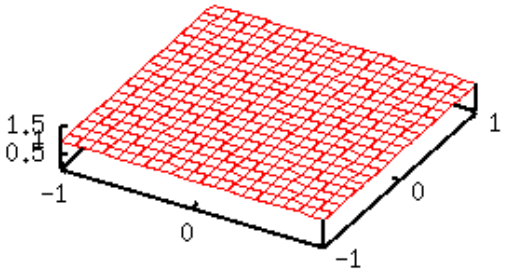

Figure 110: set view 15,30

that ten contour lines are drawn. The next, `incremental` defines the initial, increment, and final values. You can also define the height at which the contour lines are to be shown by the command `levels discrete`
A legend for contour lines is placed in the legend of the graph. To erase the contour's legend, use the `set noclabel` command. In this case all line kinds for the contour lines become the same, which is just next line kind to the surface plot.
The line kinds of the contours are the next to that used to the surface. I guess there is no way to control those line kinds arbitrarily. You have to use some external tools such as Tgif to edit the style of the contour lines.

11.5. I want to draw only contours on the 2-dimensional plot

When your view point is right above the XY plane, you get a two-dimensional plot of the contours lines. Firstly set the view point at 0,0 by `set view 0,0`, and indicate `set nosurface` to hide the surface.

```
gnuplot> set contour base
gnuplot> set nosurface
gnuplot> set view 0,0
```

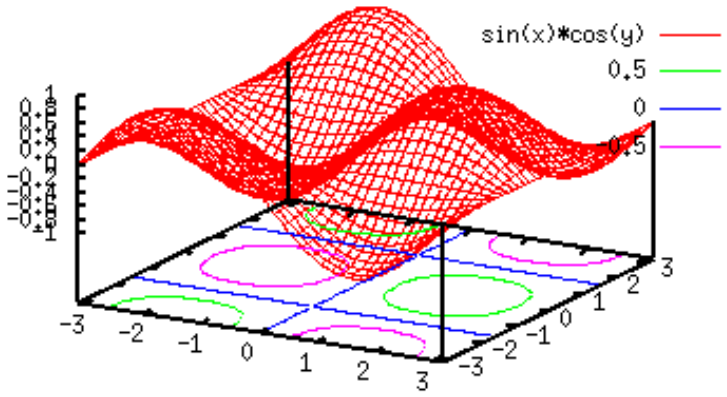

Figure 111: set view 60,30

The Y-axis goes to the right-side by the `set view 0,0` command. If you want to move it to the left-side, use `set view 180,180`. The Y label becomes horizontal in this method. If you want to make it vertical, use "table" as follows.
You can make a better two-dimensional plot of the contours by using the table terminal. The contour lines are once written on a data file with the `set term table` command, which produces two-dimensional data of various contour lines. See an example in our Gallery.

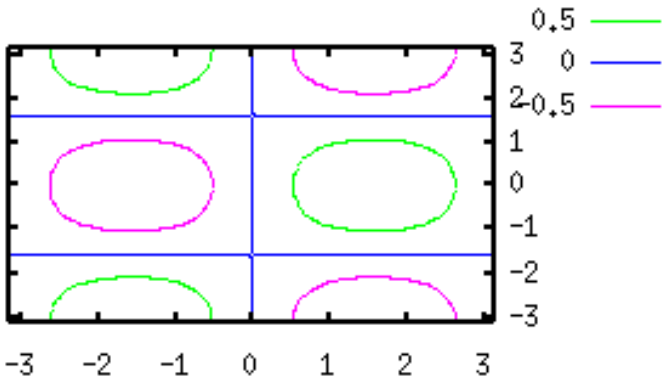

Figure 112: set view 0,0

```
gnuplot> set term table
gnuplot> set output "table.dat"
gnuplot> replot
gnuplot> set output
gnuplot> set term x11
gnuplot> plot "table.dat" using 1:2 with lines
```

Ver4 only: The two-dimensional plot with a color-map can be done easily with the pm3d terminal.

11.6. How do I draw a 3-dim. grid graph from 3-Dim scattered data

Suppose your data to be plotted are 3-dimensinonal scattered data (x,y,z) with arbitrary order. With the `splot` command, gnuplot generates points or lines in a 3-dim. space.

```
# X           Y          Z
  9.862e-01 -8.062e-02   0.001
  9.786e-01 -1.134e-01   0.002
  9.720e-01 -1.382e-01   0.003
 .....
 -1.849e-01 -2.165e-01  80.000
 -2.412e-01 -1.301e-01  90.000
 -2.611e-01 -4.825e-02 100.000

gnuplot> set xrange [-1:1]
gnuplot> set yrange [-1:1]
gnuplot> set ticslevel 0
```

```
gnuplot> splot "datafile.dat" u 1:2:3 with lines
```

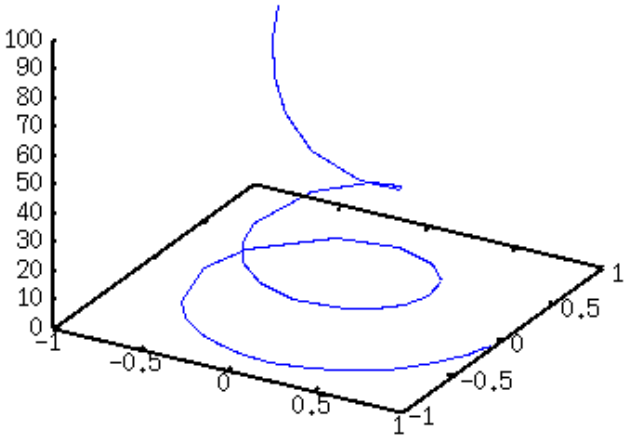

Figure 113: sample6.6a

In order to generate a 3-dim. grid graph from those data, use `set dgrid3d`. Height of each mesh is determined automatically by a weighted average of the data inside the grid. The number of mesh in the X- and Y-directions can be specified by `set dgrid3d x-mesh, y-mash`.

```
gnuplot> set dgrid3d 30,30
gnuplot> set hidden3d
gnuplot> splot "datafile.dat" u 1:2:3 with lines
```

11.7. How do I draw a colored 3D figure?

[ver.4] ONLY !
Gnuplot draws 3D figures with lines and / or points, in addition, ver.3.8 or 4.0 allows you to draw a color-mapped 3D figure by setting `pm3d`. The figure is shown on your screen as well as PostScript or some image formats like PNG/JPG.

```
gnuplot> set xrange [-2:2]
gnuplot> set yrange [-2:2]
gnuplot> set pm3d
gnuplot> splot exp(-x*x)*exp(-y*y)
```

You can choose color or gray-scale by the terminal option of `color/monochrome`. The gray-scale figure can be drawn by `set palette gray`, but the surface grid lines are still colored in this case. The following two examples are for the PostScript terminal.

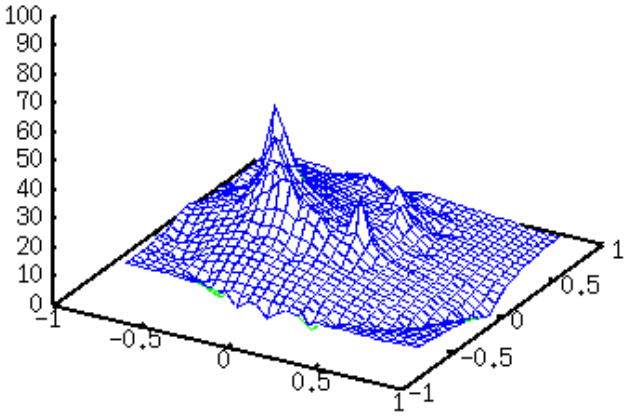

Figure 114: sample6.6b

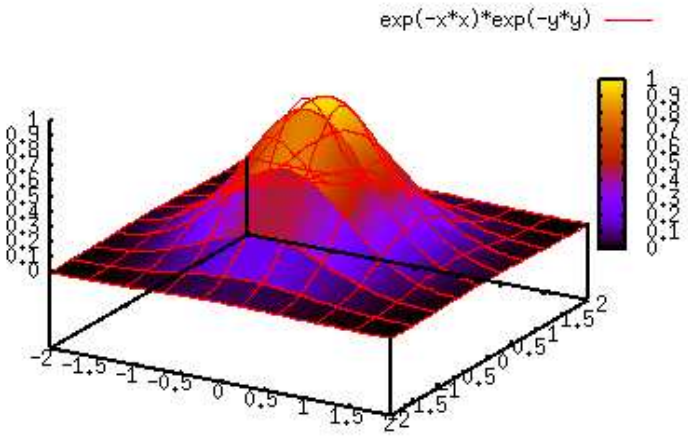

Figure 115: sample6.7a

```
gnuplot> set term postscript eps enhanced color
gnuplot> set output "color.eps"
gnuplot> replot
gnuplot> set term postscript eps enhanced monochrome
gnuplot> set output "mono.eps"
gnuplot> replot
```

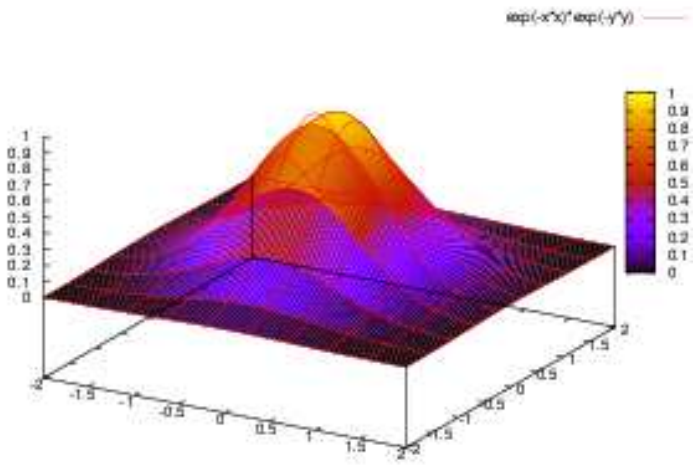

Figure 116: sample6.7b

11.8. I want to draw colors for contours

[ver.4] ONLY !
A contour map of gnuplot is shown by lines. The colored contour by pm3d is something like a colored density distribution, which means, each piece of colored surface is mapped on the bottom/top plain. To show this on the bottom, add the `at b` option to the `set pm3d` command.

```
gnuplot> set pm3d at b
gnuplot> set ticslevel 0.8
gnuplot> set isosample 40,40
gnuplot> splot x*x*exp(-x*x)*y*y*exp(-y*y)
```

With pm3d, it is very easy to see the bottom surface in the 2D plot.

```
gnuplot> set pm3d map
gnuplot> splot x*x*exp(-x*x)*y*y*exp(-y*y)
```

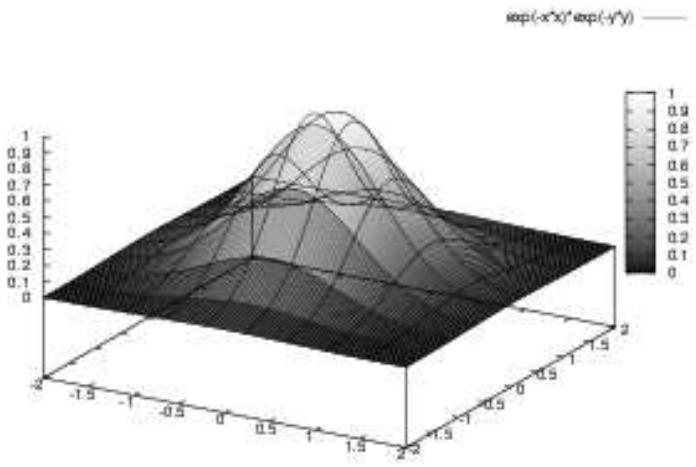

Figure 117: sample6.7c

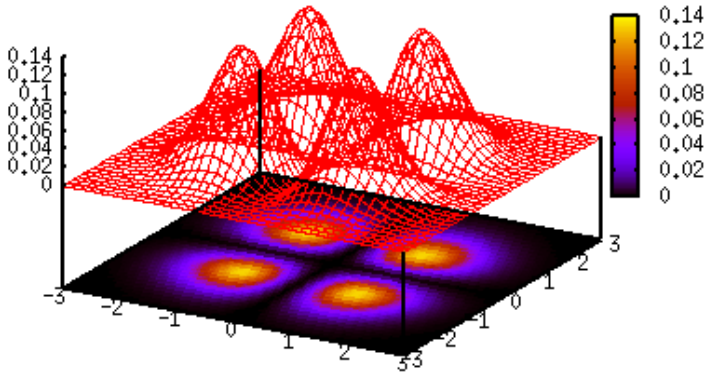

Figure 118: sample6.8a

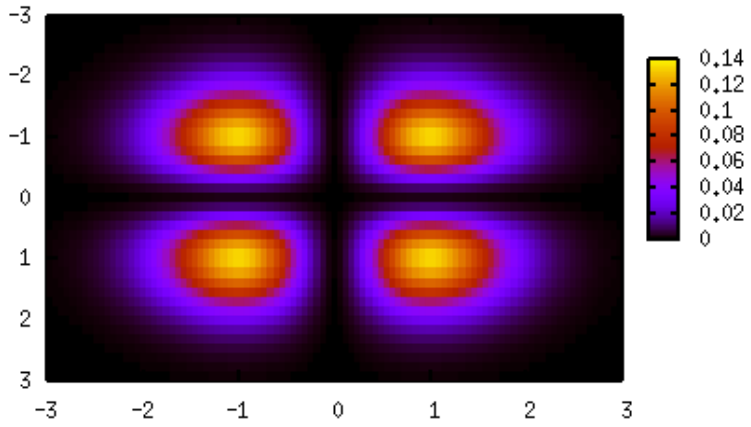

Figure 119: sample6.8b

11.9. Pseudo 3D Bar graph

Since gnuplot cannot draw a 3D-bar graph, we need a little trick to make it. Suppose we have the following 3D data.

```
# X     Y     Z
  0.0   0.0   2.0
  0.0   1.0   3.5
  1.0   0.0   1.0
  1.0   1.0   3.0
```

The first line means $Z = 2$ when $0 < X < 1$, $0 < Y < 1$, and it corresponds to the area (1) in the next figure. The data file does not contain maximal values. The ranges of X and Y are implicitly assumed as $1 < Y < 2$, $1 < Y < 2$, and the Z value is 3.0. We will make a 3D histogram like this. Now we expand the data with some programs/tools. See the figure above at $X = 0$, the Y values are step function changing $Y = 2.0$, 2.0, 3.5, and 3.5. This step function is expressed as one line in a 3D space. Same can be consider for $X = 1$ and $X = 2$.

```
# X     Y     Z
  0.0   0.0   2.0
  0.0   1.0   2.0
  0.0   1.0   3.5
  0.0   2.0   3.5
```

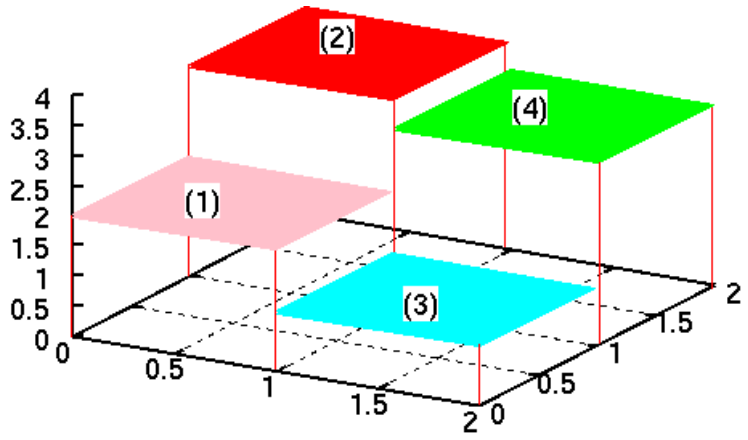

Figure 120: sample6.9

```
1.0   0.0  2.0
1.0   1.0  2.0
1.0   1.0  3.5
1.0   2.0  3.5

1.0   0.0  1.0
1.0   1.0  1.0
1.0   1.0  3.0
1.0   2.0  3.0

2.0   0.0  1.0
2.0   1.0  1.0
2.0   1.0  3.0
2.0   2.0  3.0
```

Once you prepare your data in such a way, we just depict them with lines. A program to process the data can be .... your homework.

```
gnuplot> set hidden3d
gnuplot> splot "test.dat" with lines
```

[ver.4] ONLY !
When your gnuplot is ver.4, a funcy figure can be made with pm3d.

```
gnuplot> set pm3d
gnuplot> splot "test.dat" with pm3d
```

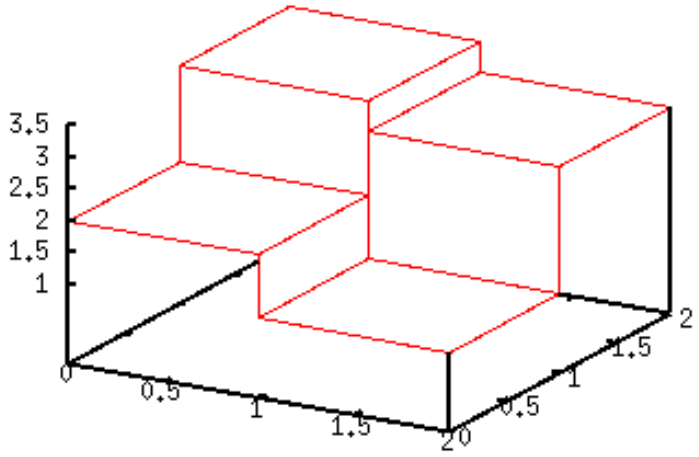

Figure 121: sample6.9a

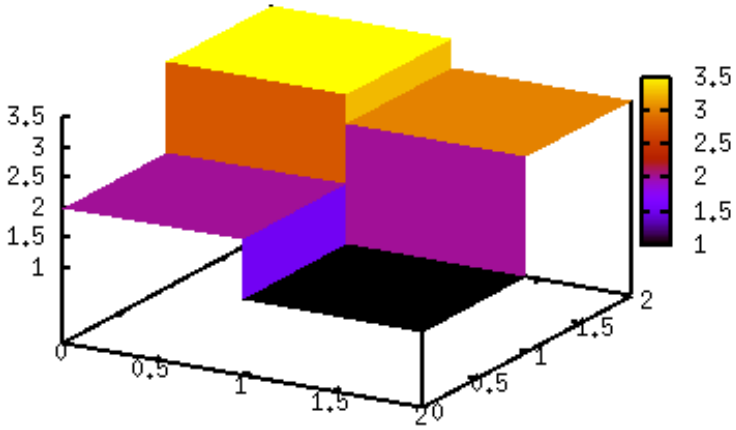

Figure 122: sample6.9b

11.10. How can I change the colors in a 3D figure?

[ver.4] ONLY !
As a default, pm3d uses a color map which varies from black to yellow via blue and red. To change this color gradient, use the `set palette` command. You need to add some options to control the color map. Suppose we have a function (or data) whose Z values vary from -3 to 1. f(x,y) = (2*sin(x)-1)*exp(-y) is such a function. Now we want to use red for the maximum value (Z=1), blue for the minimum value (Z=-3), and do not want to use any colors at Z=0.

```
gnuplot> set ticslevel 0
gnuplot> set pm3d
gnuplot> set palette defined (-3 "blue", 0 "white", 1 "red")
gnuplot> splot (2*sin(x)-1)*exp(-y) with pm3d
```

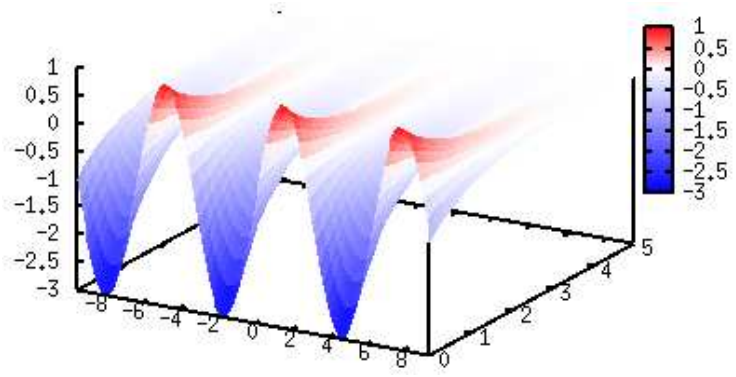

Figure 123: sample6.10a

The option `defined` is used here. The followed numbers/values are some Z values and colors there. Pm3d interpolates colors in between two defined Z points. In the example above, the color changes from blue to white in the Z=-3 to 0 range.
In this example, the color for Z=0 is the same as the background color which is white. Such a color-mapping might be useful when you want to dim some area where your data are less important. For example, you have a 2-dimensional density distribution, you may want to exclude some area in which the density is zero. This can be understood with the example above if we project the 3D data onto 2D plane. The negative area is shown by blue, positive is red, and the other area looks empty.

```
gnuplot> set pm3d map
gnuplot> replot
```

Another method to change colors is to use the option of `rgbformulae`, followed by three intergers. Those numbers are used for three colors – Red(R)/Green(G)/Blue(B), and each number is an index of some built-in functions. For example `rgbformulae 0,0,3`

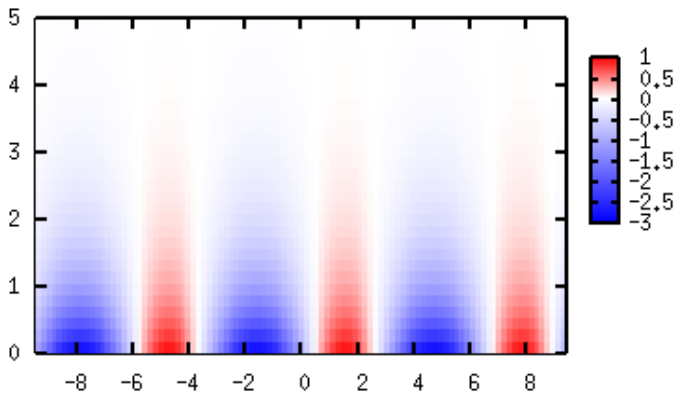

Figure 124: sample6.10b

means, gnuplot does not use Red and Green, but the height of Z values is expressed by a linear color variation from black to blue.
There are 37 functions to change the color, which have an index from zero to 36. The function index 3, which was used in the example above, is a simple linear function. Default color setting of gnuplot is 7,5,15 where the function number 7 is sqrt(x), 5 is $x\hat{5}$, and 15 is sin(360x). The defined functions with those indices you can see by `show palette rgbformurae`.
In general, to obtain a suitable color palette with the `palette rgbformulae` command is hard. However, the `help palette rgbformulae` command may give you some hints. For example, a combination of 33,13,10 is a "rainbow", which is like this.

```
gnuplot> set palette rgbformulae 33,13,10
gnuplot> splot -x*x with pm3d
```

Other colorings gnuplot help recommends are as follows: Now we got another "rainbow"

```
gnuplot> set pm3d map
gnuplot> set size square
gnuplot> set palette rgbformulae 22,13,-31
gnuplot> splot -(x*x+y*y)
```

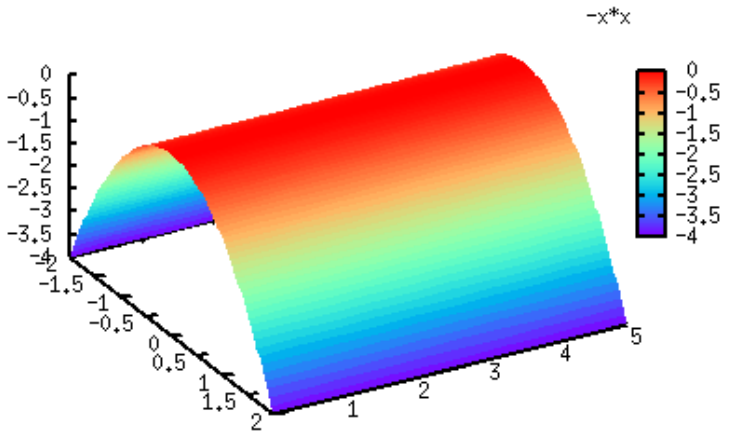

Figure 125: sample6.10c

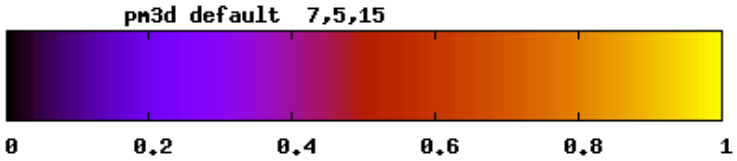

Figure 126: cb1

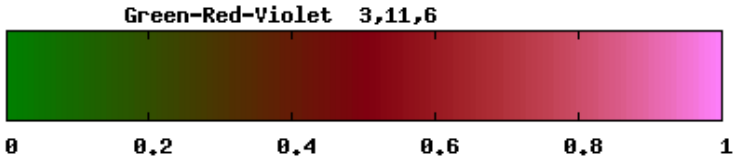

Figure 127: cb2

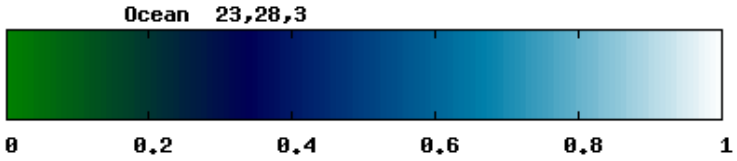

Figure 128: cb3

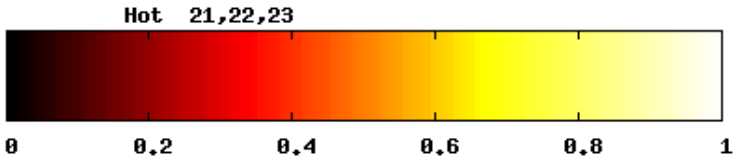

Figure 129: cb4

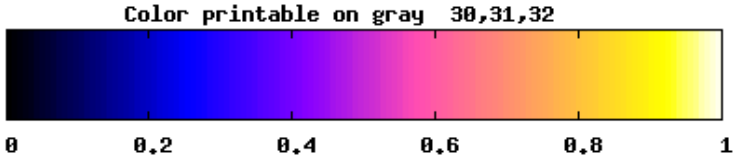

Figure 130: cb5

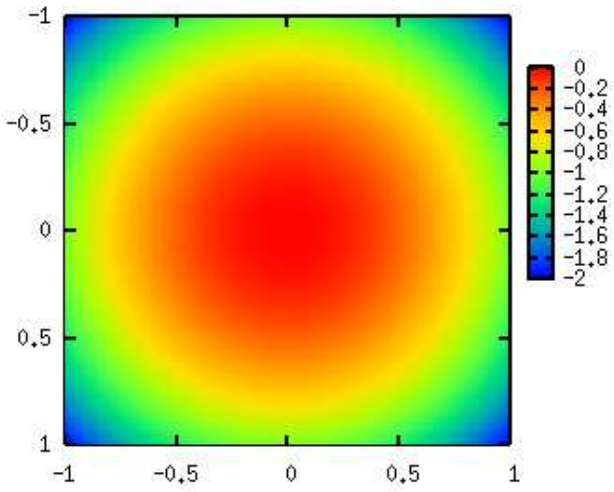

Figure 131: sample6.10d

12. About Polar Coordinate

12.1. Plotting data in the 2-dimensional polar coordinates

In the polar coordinate the data to be plotted are similar to the rectangular coordinate – (X,Y), but the data represent X="angle" and Y="radius". The data format is the same as the usual two-dim. data. The default unit of angle is radian and the range is 0 to 2*pi. If you want to use a degree unit, `set angles degrees`.
Let's think about the following two-dimensional data, which are shown in the rectangular coordinate, X range is 0 to 180 degrees, Y range is -1 to 1. We plot this in the

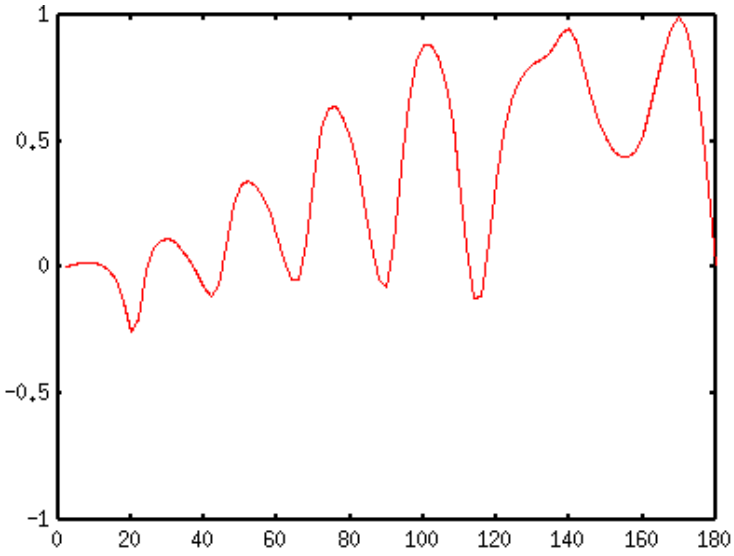

Figure 132: sampleP1.1

polar coordinate.

```
gnuplot> set polar

        dummy variable is t for curves
gnuplot> set angles degrees
gnuplot> plot "datafile.dat" with lines
```

Now, adjust the ranges of X and Y, and make the graph square. This is a likely appearance for the polar coordinate plot.

```
gnuplot> set size square
gnuplot> set xrange [-1:1]
gnuplot> set yrange [-1:1]
gnuplot> replot
```

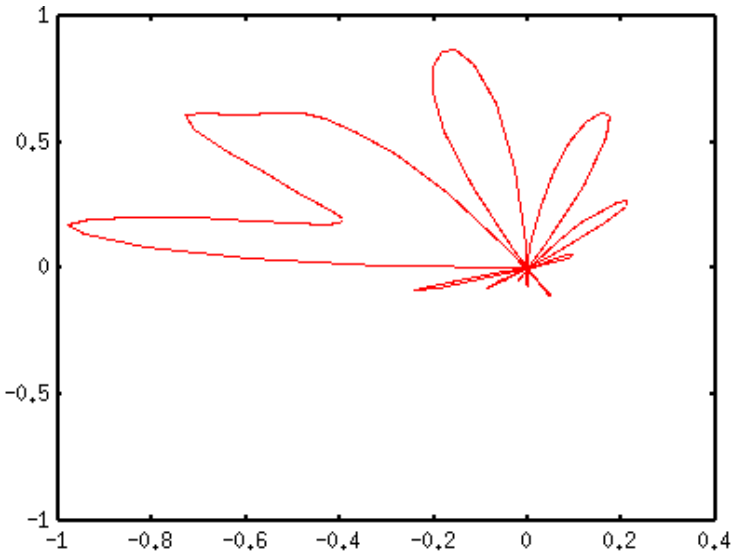

Figure 133: sampleP1.2

12.2. Drawing lines from data-points to the origin

n the above example, each (angle, radius) point is connected by a piecewise line. To draw lines from the origin to each data point, use `with impules` option.

```
gnuplot> plot "datafile.dat" with impulses
```

12.3. Drawing grids

There are two kinds of grid in the polar coordinate. The first one is to draw them at the major tics, which is controlled by the `set grid` command. This is the same as the usual rectangular coordinate plot. The other one is circles and radial lines which are drawn by the `set grid polar` angle command. The angle defines the radial lines interval (default 30 degrees).

```
gnuplot> set grid polar
```

The circular grids depends on both the major tics of X and Y axes. In the above figure tics for X and Y are 0.5, so that the circles are drawn at the interval of 0.5. If the major tics are changed as follows, the circular grids are drawn at 0.3,0.5,0.6,0.9, and 1.0.

```
gnuplot> set xtics 0.5
gnuplot> set ytics 0.3
```

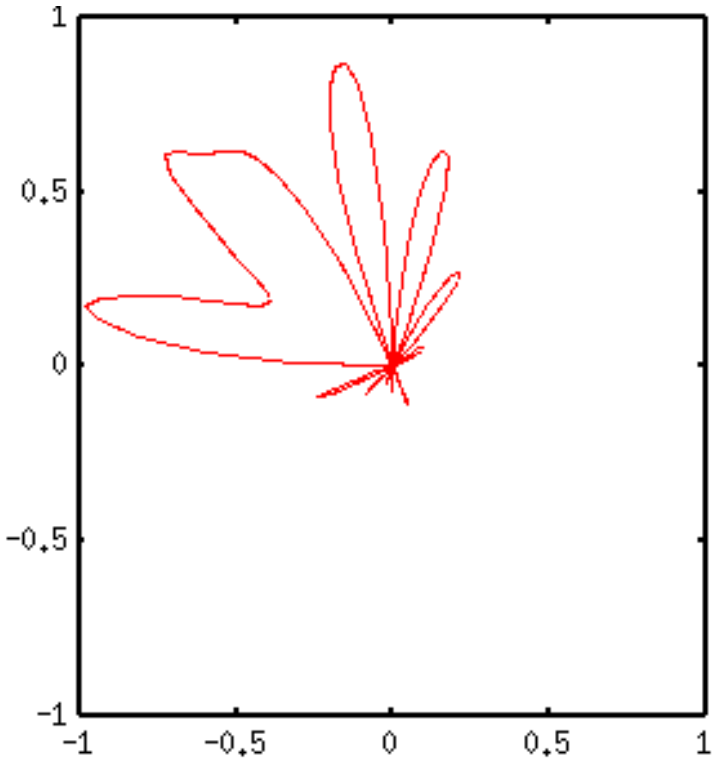

Figure 134: sampleP1.3

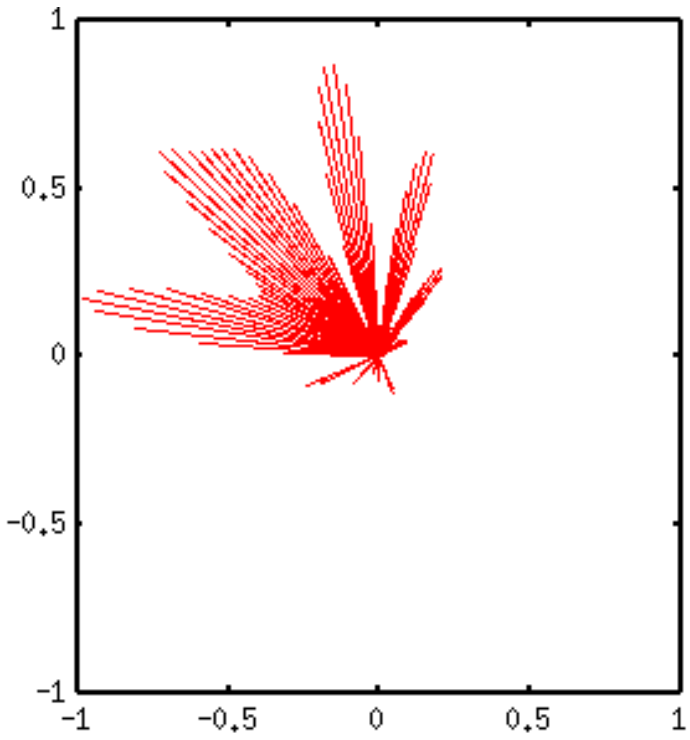

Figure 135: sampleP2.1

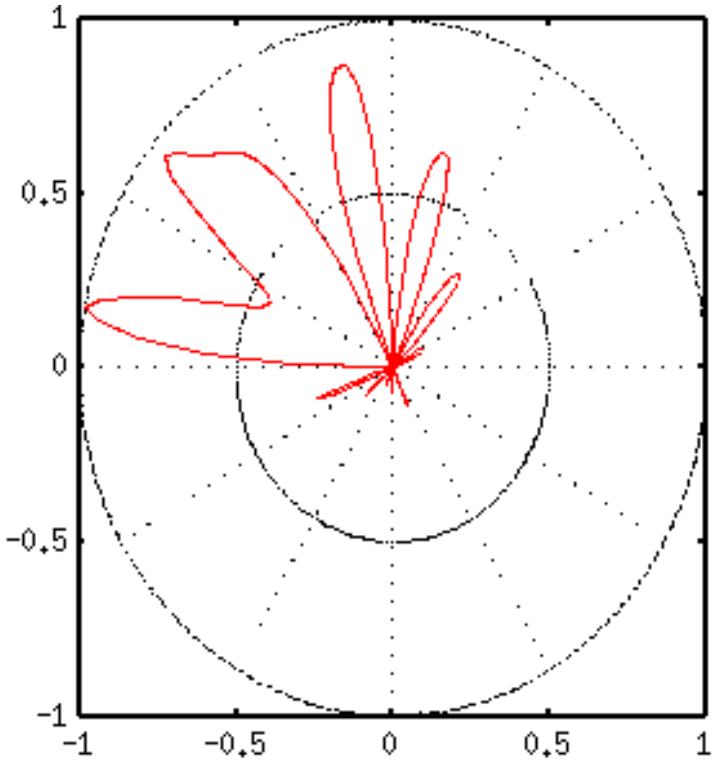

Figure 136: sampleP3.1

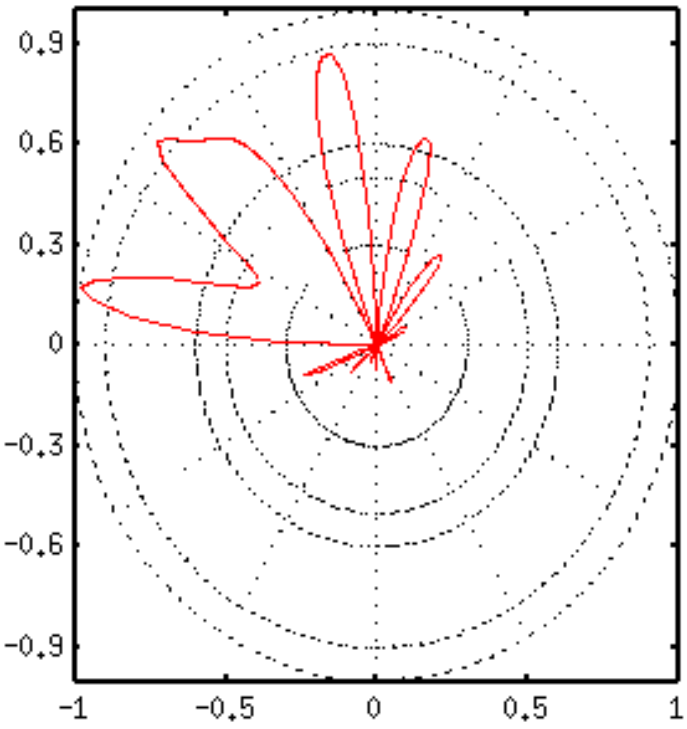

Figure 137: sampleP3.2

13. About Parametric Functions

13.1. Use of parameters

In the usual 2-dimensional plot of gnuplot, the Y coordinate is expressed by y=f(x), however you can also use a parametric expression which uses the parameter t

$$x = f(t)$$
$$y = g(t)$$

With this expression, more complicated functions can be plotted with gnuplot. Note that the 3-dim. plot with two parameters u,v is given in the spherical harmonics section.
First of all, you need to use the command `set parametric` to tell gnuplot that the function is defined by a parameter. Then, the `plot` command followed by a function f(t) which is the X-coordinate and a function g(t) for Y-coordinate, is give like, `plot f(t),g(t)`

13.2. to draw a vertical line

The most simple but it is impossible to express by the y=f(x) form is a vertical line which is x=const. This function can be written as:

$$x = const \quad y = t$$

with the parameter t, when t is varied. The range of t is controlled by the command `set trange` .

```
gnuplot> set parametric

        dummy variable is t for curves, u/v for surfaces
gnuplot> const=3
gnuplot> set trange [1:4]
gnuplot> set xrange [0:5]
gnuplot> set yrange [0:5]
gnuplot> plot const,t
```

In this case the vertical line is draw at x=3. Since we used `set trange [1:4]` , the range of this truncated line is from 1 to 4. If `trange` not set, the vertical line is drawn from the bottom to top border lines.

13.3. to draw a circle, polygons

The parametric expression of a circle is

$$x = sin(t) \quad y = cos(t)$$

and the circle can be drawn if one changes the t parameter from 0 to 2pi. The graph is "squared" here, and the t range is given by an option of `plot` command.

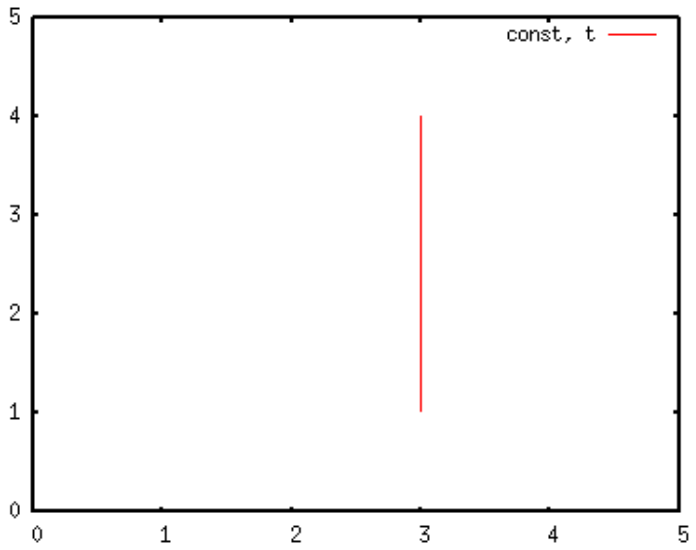

Figure 138: sampleQ2.1

```
gnuplot> set parametric

        dummy variable is t for curves, u/v for surfaces
gnuplot> set size square
gnuplot> set xrange [-1:1]
gnuplot> set yrange [-1:1]
gnuplot> plot [0:2*pi] sin(t),cos(t)
```

The parameter t is not changing continuously, and actually this is controlled by the value which is set by the `set samples` command. The default value is 100. In the case of `set samples 8` , gnuplot generates eight t values from zero to 2*pi, and the graph becomes a regular heptagon. If you need a regular N-gon, just type `set samples N+1`. The 2-dim. parametric representation is convenient to draw a function which is in a polar coordinate. The 2-dim. polar coordinate has two variables which are radius r and angle theta. The gnuplot parameter t is for the theta, and the radius r is expressed by a function of angle, namely r(t). A (x,y) coordinate is given by

$$x = r(t) * cos(t) \quad y = r(t) * sin(t)$$

The circle is a special case of which r(t)=const. When the radius of circle is proportional to t, you get a spiral.

```
gnuplot> set xrange [-10*pi:10*pi]
gnuplot> set yrange [-10*pi:10*pi]
gnuplot> plot [0:10*pi] t*sin(t),t*cos(t)
```

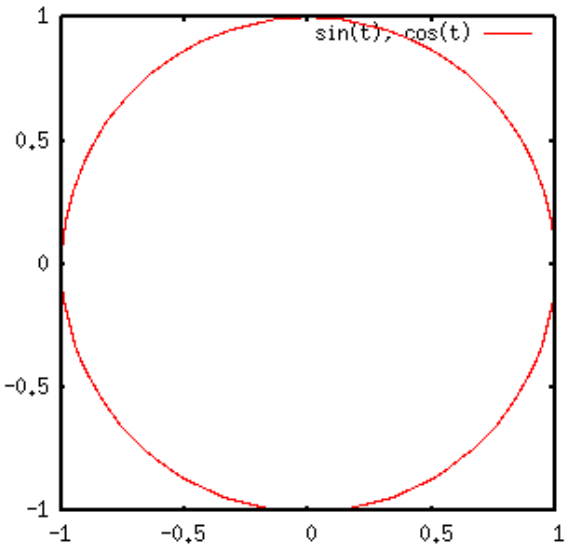

Figure 139: sampleQ3.1

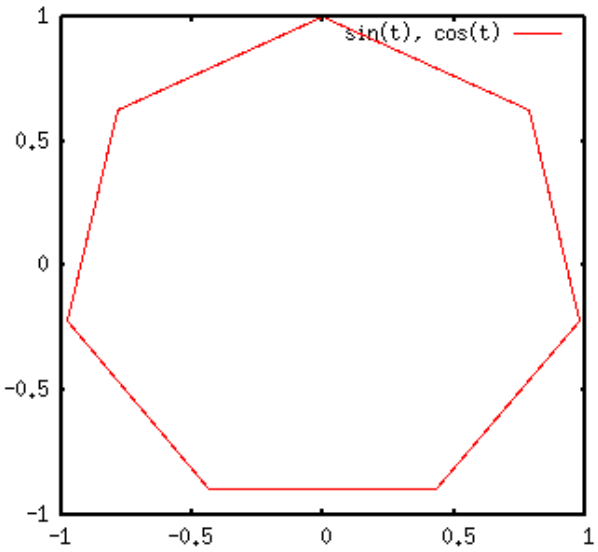

Figure 140: sampleQ3.2

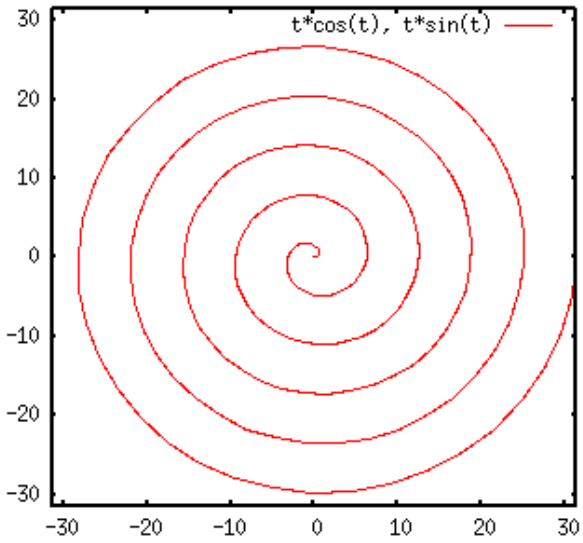

Figure 141: sampleQ3.3

The following example shows r(t)=const*(1+cos(t)), which is called Cardioid.

```
gnuplot> set parametric

        dummy variable is t for curves, u/v for surfaces
gnuplot> r(t) = 1+cos(t)
gnuplot> plot [0:2*pi] r(t)*cos(t),r(t)*sin(t)
```

13.4. Exchange X and Y-axes

Functions are normaly expressed by y=f(x), but the parametric expression allows us to make a graph of x=f(y). The y values are the same as t, and the x values are calculated with a function of f(t).

```
gnuplot> set parametric

        dummy variable is t for curves, u/v for surfaces
gnuplot> c=2*pi
gnuplot> set size square
gnuplot> set trange [-c:c]
gnuplot> set xrange [-c:c]
gnuplot> set yrange [-c:c]
gnuplot> plot c*sin(t),t with lines, t,c*cos(t) with impulses
```

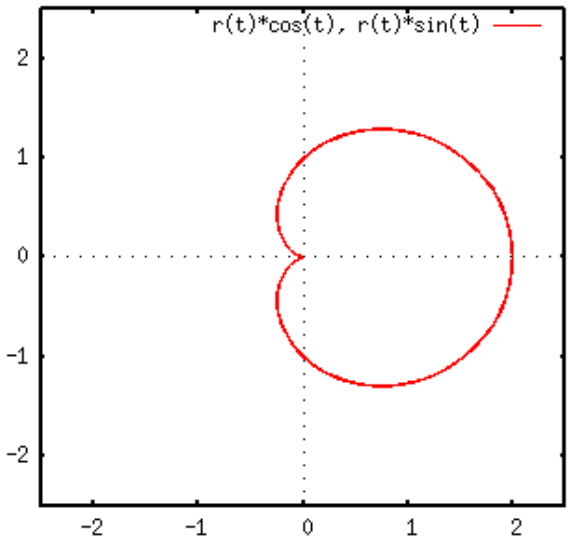

Figure 142: sampleQ3.4

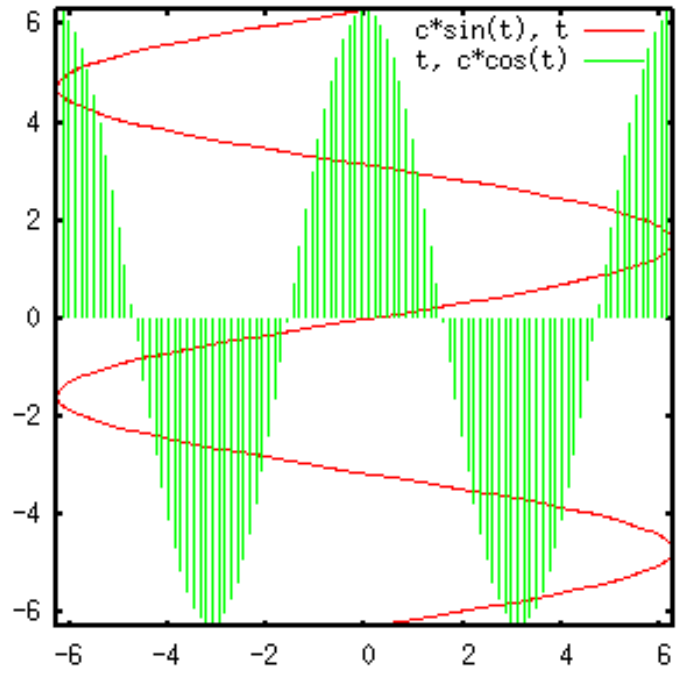

Figure 143: sampleQ4.1

Two functions are shown, one (green stripe) is y=2pi*cos(x), and the other (red solid line) is x=f(t)=2pi*sin(y).
The option `with impulse` draws a vertical line from the Y=0 axis. If you use `with impulses` for the red curve which is x=2pi*sin(y), you still get a vertical stripe, not a horizontal one.

14. Plotting Numerical Data and Data Files

Probably most of the gnuplot lovers in a scientific field use this program to draw a graph of some calculated results or experimental data. They see their results on their monitor, make some corrections, comparison of the calculated result with the experimental data, and so on. If it seems fine, the figure is saved in a postscript format and send it to a printer, otherwise an EPS file is included in a TeX document...

14.1. What is the format which gnuplot can recognize?

1. 2-dimensional data
2. 3-dimensional data
3. Matrix

14.1.1. 2-dimensional data

In a data file, the data columns are separated by a white-space or tab. If a line begins with "#", this line is ignored. Generally gnuplot can read any data format if one specifies the format. See gnuplot help "using".

```
#  X     Y
   1.0   1.2
   2.0   1.8
   3.0   1.6
```

For example, if you have two-dimensional data, one line contains a pair of X and Y values, like the example above. If X or Y values have uncertainties, you need an extra column to give the errors. The order of columns is not important because you can specify which column is used for X or Y data. If the first column is the X data, and the second is the Y data, use `using` option as:

```
gnuplot> plot "test.dat" using 1:2
```

If `using` is omitted, the first column is used for the X data, and the second is for the Y data automatically.
There are two ways to make an error-bar for the Y value. The first one is that the Y value has uncertainties of plus/minus Z. The second one is that the Y value has a range [Z1,Z2]. In this case the lengths of the error bars below and above Y value are different. The former needs three columns, and the latter needs four columns.

```
#  X     Y     Z
   1.0   1.2   0.2
   2.0   1.8   0.3
   3.0   1.6   0.2
```

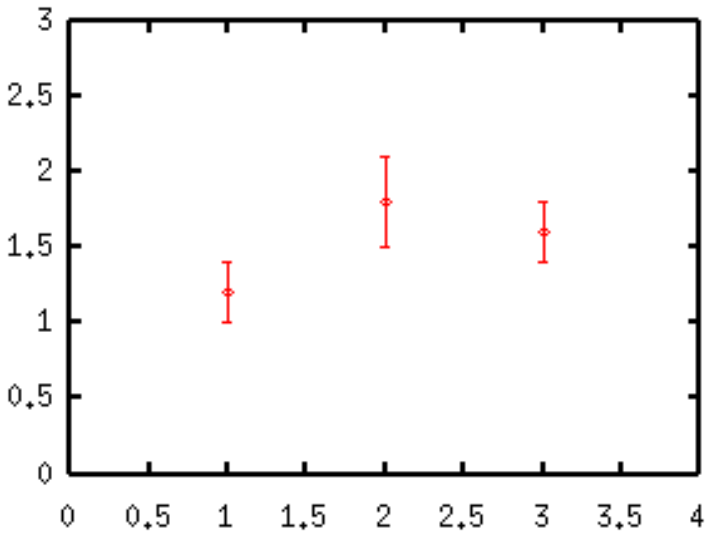

Figure 144: sample7.1a

```
#  X     Y     Z1    Z2
   1.0   1.2   0.8   1.5
   2.0   1.8   0.3   2.3
   3.0   1.6   1.0   2.1
```

To plot those data,

```
gnuplot> plot "test.dat" using 1:2:3 with yerrorbars
gnuplot> plot "test.dat" using 1:2:3:4 with yerrorbars
```

one needs the `using` option.
The number of data column required for data plotting depend on a kind of figure, which is summarized below.

Data Format	Column	using	with
(X,Y) data	X Y	1:2	lines, points, steps, linespoints, boxes, etc.
Y has an error of dY	X Y dY	1:2:3	yerrorbars
X has an error of dX	X Y dX	1:2:3	xerrorbars
Y has an error of dY, and X has an error of dX	X Y dX dY	1:2:3:4	xyerrorbars
Y has a range of [Y1,Y2]	X Y Y1 Y2	1:2:3:4	yerrorbars
X has a range of [X1,X2]	X Y X1 X2	1:2:3:4	xerrorbars
Y has a range of [Y1,Y2], and X has a range of [X1,X2]	X Y X1 X2 Y1 Y2	1:2:3:4:5:6	xyerrorbars

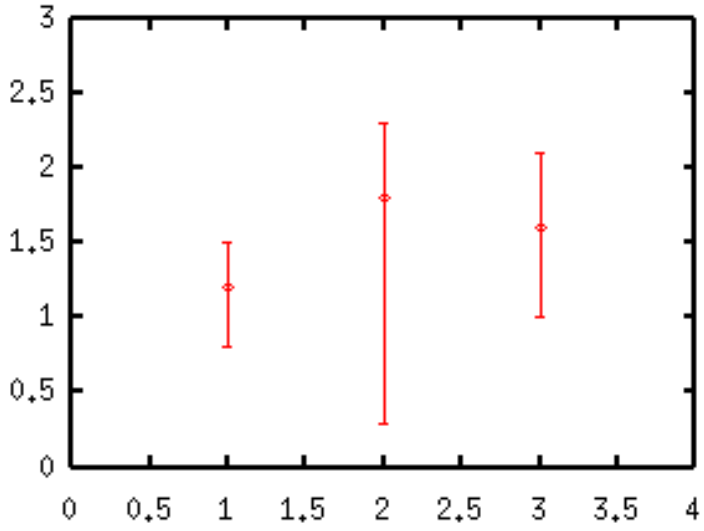

Figure 145: sample7.1b

14.1.2. 3-dimensional data

Three dimensional data are specified by (X,Y,Z). In the default, it represents a point in a 3-dimensional space. When one uses `with lines` option to draw lines, gnuplot shows 3-dim. lines or surfaces depending on a data format. The following example contains 4 blocks data, which are separated by one blank line. The number of (X,Y) pairs are different for the each data block. In such a case, gnuplot does not draw a surface but 3-dim. lines.

```
# X   Y    Z
  0   0    0
  0   1    1
  0   2    4
  0   3    9
  0   4   16
  0   5   25

  1   0    1
  1   1    2
  1   2    5
  1   3   10
  1   4   17

  2   0    4
  2   1    5
```

```
  2   2    8
  2   3   13

  3   0    9
  3   1   10
  3   2   13
```

```
gnuplot> splot "test3d.dat" using 1:2:3 with lines
```

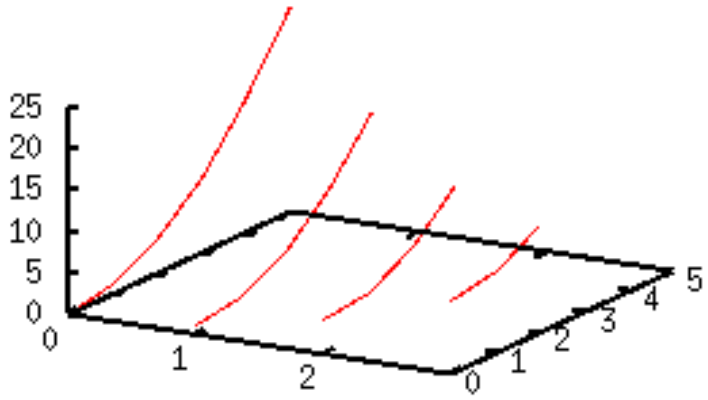

Figure 146: sample7.1c

Now, if the number of (X,Y) pairs is the same:

```
# X   Y    Z
  0   0    0
  0   1    1
  0   2    4
  0   3    9
  0   4   16
  0   5   25

  1   0    1
  1   1    2
  1   2    5
  1   3   10
  1   4   17
  1   5   26

  2   0    4
  2   1    5
```

```
2    2     8
2    3    13
2    4    20
2    5    29

3    0     9
3    1    10
3    2    13
3    3    18
3    4    25
3    5    34
```

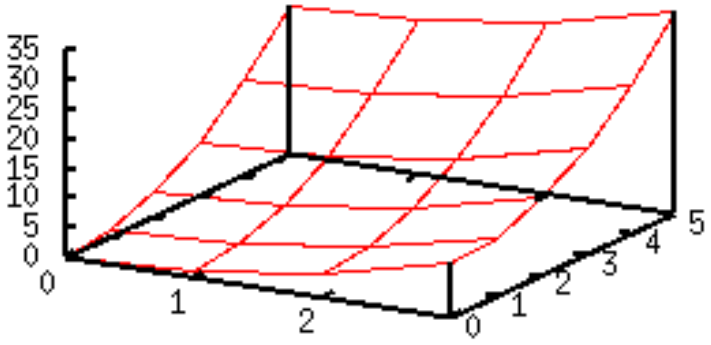

Figure 147: sample7.1d

The surface grid appears. Although the Y values (second column) in the each block of this example are the same, gnuplot also draws the surface grid even if the Y values are different (but the number of data points are the same.) Such data are treated as the grid data. When your data are the grid data, gnuplot can draw a contour map, or hidden line processing can be done. The next is a simple test — the Y values in the last block are doubled (changed from Y=5 to Y=10). If you want to remove the surface-grid but your blocks contain the same number of data point, separate the each block by two blank lines.

```
# X   Y     Z
  0   0     0
  0   1     1
  0   2     4
  0   3     9
  0   4    16
```

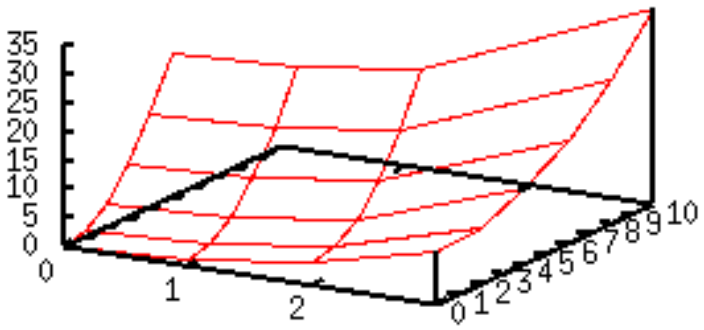

Figure 148: sample7.1e

```
0    5    25


1    0     1
1    1     2
1    2     5
1    3    10
1    4    17
1    5    26


2    0     4
2    1     5
2    2     8
2    3    13
2    4    20
2    5    29


3    0     9
3    1    10
3    2    13
3    3    18
3    4    25
3    5    34
```

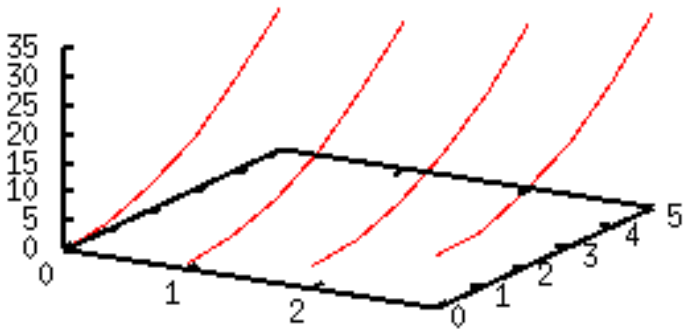

Figure 149: sample7.1f

14.1.3. Matrix

The matrix data format is an alternative way to give 3-dim. data. When X and Y values (XY grid) are fixed, the matrix format is more convenient. In this format, raw is the direction of X, and column is Y.

```
 0    1    4    9
 1    2    5   10
 4    5    8   13
 9   10   13   18
16   17   20   25
25   26   29   34
```

To plot this data, use `matrix`. When the data are represented by the matrix format, the X and Y coordinates are the index of row and column. In this case the X range is [0:3] and the Y range is [0:5]. If you want to change those number, use the `set x|ytics` command. The next shows how to change the X range from [0:2] to [100:300].

```
gnuplot> set xtics ("100" 0, "200" 1, "300" 2)
gnuplot> splot "test3d.dat" matrix with lines
```

14.2. How do I plot several data sets in a single file?

In order to plot several data those are stored in one file, use `using` and `index`. Here, "data" means a set of XY pairs. Gnuplot draws one line or prints the same symbol at each data-point.
When those data have the same X-values, prepare the data in a table format, and use the `using` option to specifiy Y-data. The next example file contains three Y-values at the same X-value.

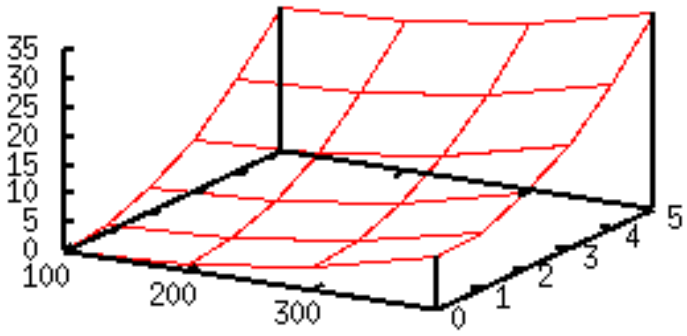

Figure 150: sample7.1g

```
#   X          Y1          Y2          Y3
   -1.0000     0.0000      0.0000      1.0000
   -0.9000     0.5700      1.1769      0.7150
   -0.8000     1.0800      1.4400      0.4600
   -0.7000     1.5300      1.4997      0.2350
   -0.6000     1.9200      1.4400      0.0400
   -0.5000     2.2500      1.2990     -0.1250
   -0.4000     2.5200      1.0998     -0.2600
   -0.3000     2.7300      0.8585     -0.3650
   -0.2000     2.8800      0.5879     -0.4400
   -0.1000     2.9700      0.2985     -0.4850
    0.0000     3.0000     -0.0000     -0.5000
    0.1000     2.9700     -0.2985     -0.4850
    0.2000     2.8800     -0.5879     -0.4400
    0.3000     2.7300     -0.8585     -0.3650
    0.4000     2.5200     -1.0998     -0.2600
    0.5000     2.2500     -1.2990     -0.1250
    0.6000     1.9200     -1.4400      0.0400
    0.7000     1.5300     -1.4997      0.2350
    0.8000     1.0800     -1.4400      0.4600
    0.9000     0.5700     -1.1769      0.7150
    1.0000     0.0000     -0.0000      1.0000

gnuplot> plot "test.dat" using 1:2 with lines,\
              "test.dat" using 1:3 with lines,\
              "test.dat" using 1:4 with lines
```

When the X-values are different, the whole data are separated into several blocks, and put two blank lines between the each block. (Note: here we use the term "block"

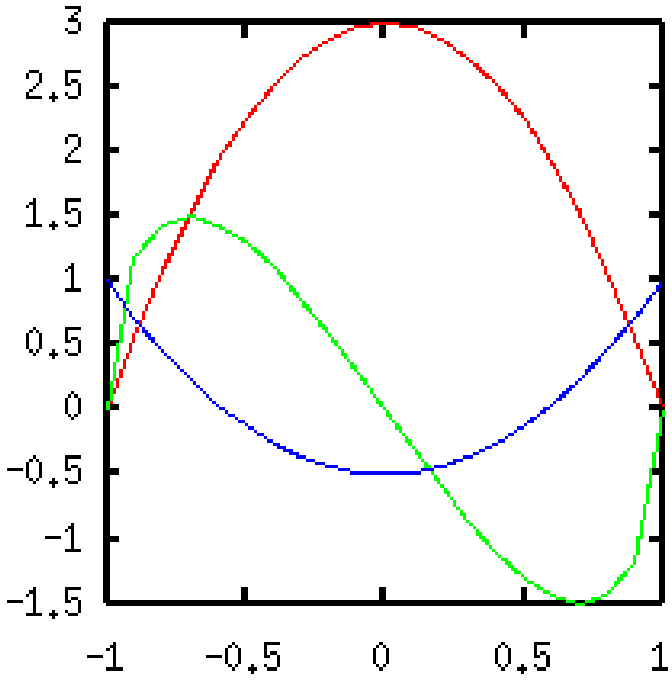

Figure 151: sample7.2a

for a set of data points in a general meaning, and this is different from the gnuplot's definition of "datablock". The datablock in gnuplot is a set of data points separated by a single blank line.)

```
#  X     Y     Yerror
   1.0   1.2   0.1
   2.0   1.8   0.1
   3.0   1.6   0.1


   1.1   0.8   0.2
   2.1   0.3   0.2
   3.1   1.0   0.2


   1.2   1.5   0.3
   2.2   2.3   0.3
   3.2   3.1   0.3
```

```
gnuplot> plot "test.dat" using 1:2 with lines
```

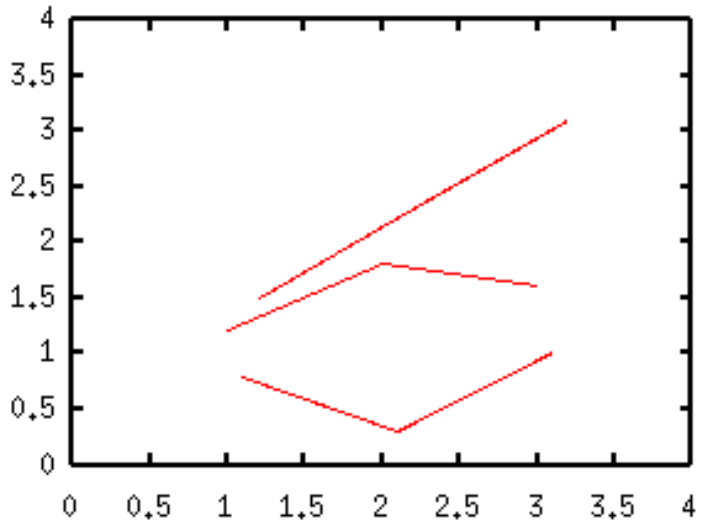

Figure 152: sample7.2b

```
gnuplot> plot "test.dat" using 1:2:3 with yerrorbars
```

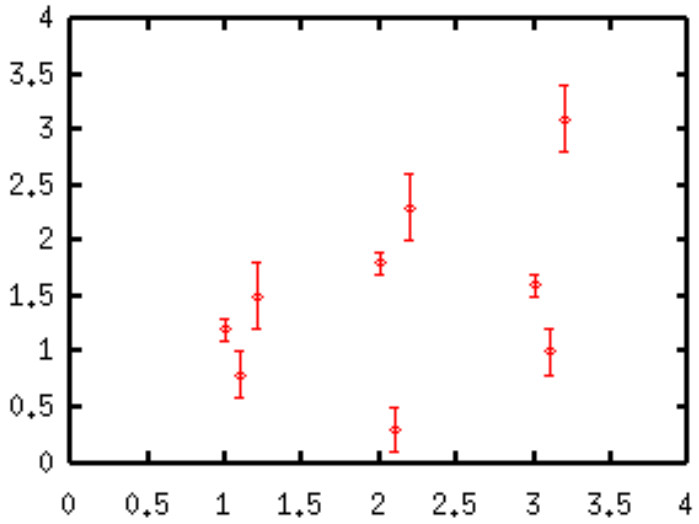

Figure 153: sample7.2c

As you can see above, all the data in the different data blocks are drawn by the same line/symbol type. To make them different, specify a data block by `index`. The first data block has an index of zero, the second is one, and so on.

```
gnuplot> plot "test.dat" index 0 using 1:2:3 with yerrorbars,\
              "test.dat" index 1 using 1:2:3 with yerrorbars,\
              "test.dat" index 2 using 1:2:3 with yerrorbars
```

You can specify the range of index. For example, to use the same symbol for the first and second blocks, and change the symbol for the third block:

```
gnuplot> plot "test.dat" index 0:1 using 1:2:3 with yerrorbars,\
              "test.dat" index 2   using 1:2:3 with yerrorbars
```

14.3. I want to modify values in my data file when plotting

You can make some simple calculations and change the data of the column which is specified by the `using` option. With the following data we show four cases simultaneously — Y-values themselves, Y-values are doubled, squared, and logarithm is taken.

```
#  X     Y
   1.0   1.2
   2.0   1.8
   3.0   1.6
```

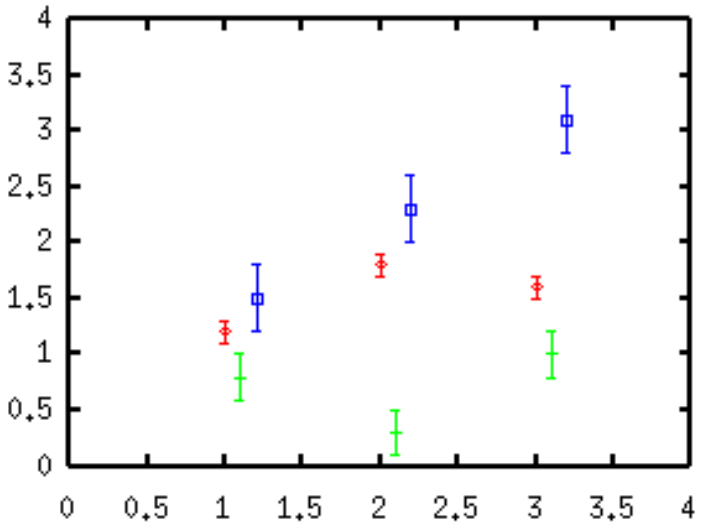

Figure 154: sample7.2d

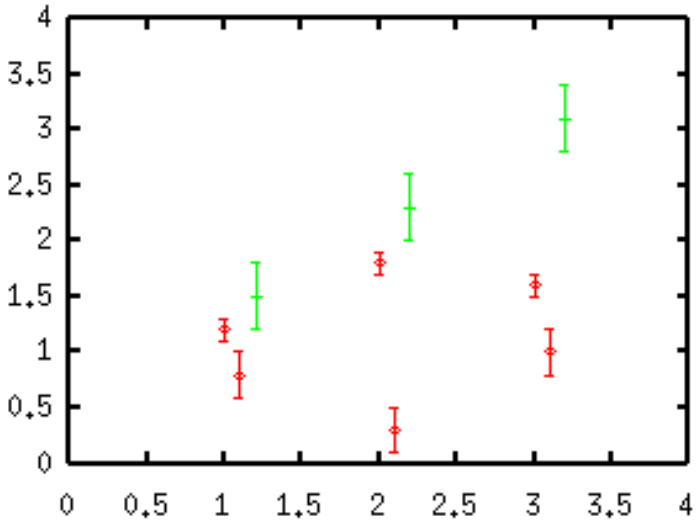

Figure 155: sample7.2e

```
gnuplot> plot "test.dat" using 1:2          with points,\
              "test.dat" using 1:($2*2)     with points,\
              "test.dat" using 1:(sqrt($2)) with points,\
              "test.dat" using 1:(log($2))  with points
```

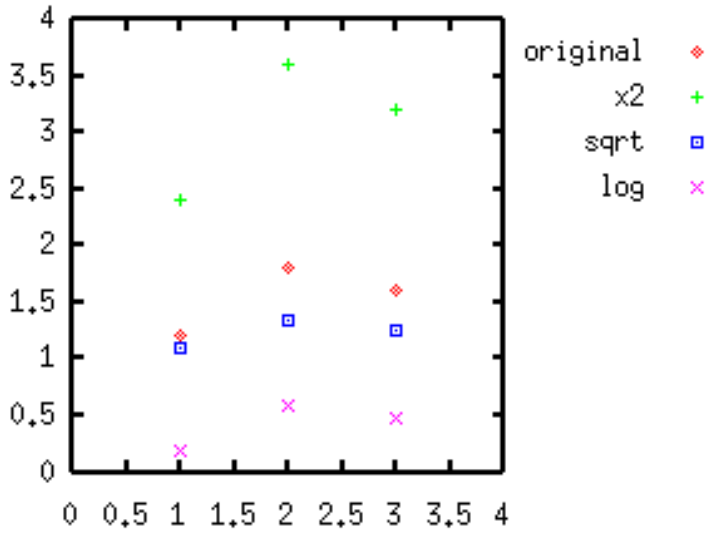

Figure 156: sample7.3

Specify the n-th column by `$n` , and make some operation. The expression should be put into parenthesis. An expression like `using 1:sqrt($2)` does not work. You can also make some calculations which need several columns. For example, `using 1:2:($2*$3)` makes a product of the second and third columns.
Sometimes this technique is useful when you have a data file which contains Y-value uncertainties, and the uncertainties are represented by a relative error (%). To draw error bars your data uncertainties should be absolute. To convert from relative uncertainties into absolute ones, `using 1:2:($2*$3/100.0)`, where the second column is the Y-data, and the third column is the percent errors.

14.4. I want to put some plotting commands in a data file

Gnuplot uses two files, one is the data and another is the control command. If your data are not so complicated, you can put the plot command into one file together with the data themselves. Basically it is the same as a usual command file, but the data file to be plotted is not an external one but it is just "-", and the numerical data follow. Gnuplot stops reading the data when a line begins with the letter "e".

```
set xrange [0:5]
```

```
set yrange [0:3]
plot "-" using 1:2:3 w yerrorbars
#  X     Y     Z
   1.0   1.2   0.2
   2.0   1.8   0.3
   3.0   1.6   0.2
   4.0   1.2   0.2
end
pause -1
```

Prepare a control file which includes the data just like above, and give the name as a command line option when gnuplot is invoked.

```
 % gnuplot "test.plt"
```

By the way, you cannot use the `replot` command in this method. Gnuplot tries to read the data from the standard input again, because Gnuplot does not remember the data.

14.5. I want to eliminate some data points

There are two ways to skip some data in a data file – the first one is to edit the data file, alternatively you can use the `every` option. To skip some points in your data by editing the data file, put some letter like "?" just before the number. This letter is defined by the `set missing` command, but any kinds of letters which cannot be converted into numbers work too.
The next example is to draw graphs of (X,Y1) and (X,Y2).

```
#file1
#  X     Y1    Y2
   1.0   1.2   1.5
  ?2.0   1.8   0.8
   3.0   1.6   1.1
#file2
#  X     Y1    Y2
   1.0   1.2   1.5
   2.0   1.8  ?0.8
   3.0   1.6   1.1
```

In the left drawing, the second X value has "?" mark, so that the second data points (2.0,1.8) and (2.0,0.8) are skipped. In the right drawing, the point (2.0,1.8) still is alive, but (2.0,0.8) is erased.

```
gnuplot> plot "test1.dat" using 1:2 with linespoints,\
              "test1.dat" using 1:3 with linespoints

gnuplot> plot "test2.dat" using 1:2 with linespoints,\
              "test2.dat" using 1:3 with linespoints
```

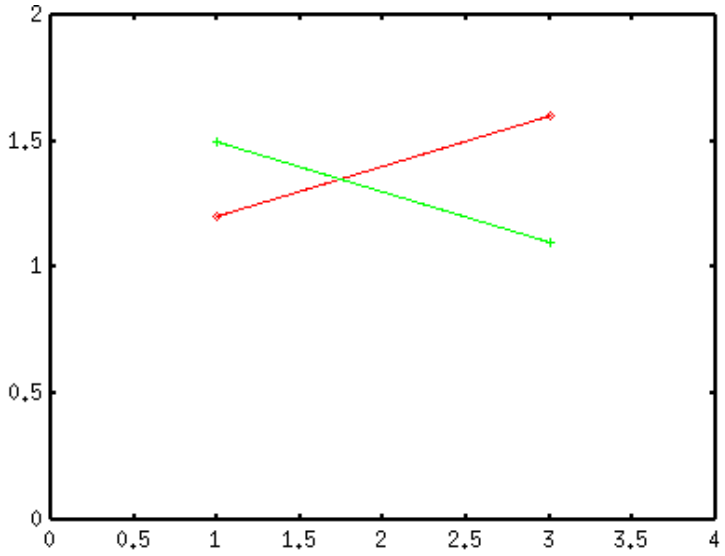

Figure 157: sample7.5a

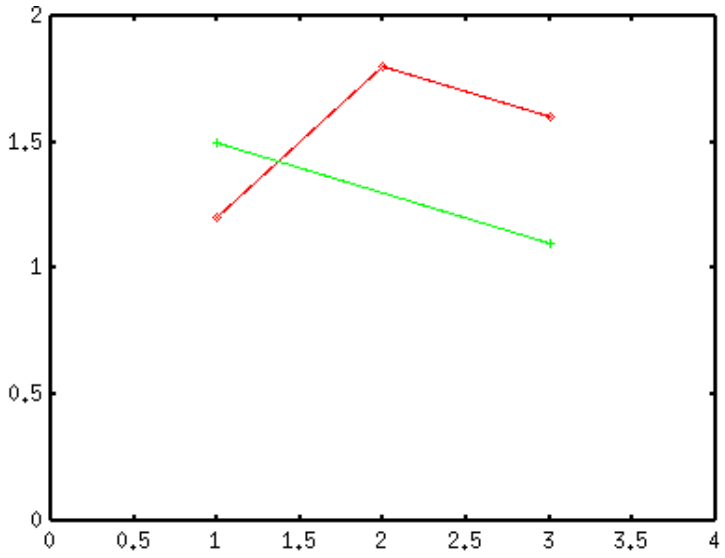

Figure 158: sample7.5b

This function is useful if you want to alter the range of plot in a table formatted data. Let's plot the following data in log-scale, the second (Y1) and the third (Y2) columns are plotted simultaneously.

```
#  X     Y1    Y2
   1.0   0.0   0.1
   2.0   0.0   0.1
   3.0   0.0   0.1
   4.0   0.1   0.2
   5.0   0.6   0.4
   6.0   1.0   0.9
   7.0   1.2   1.7
   8.0   1.3   2.4
```

The Y1-values are zero in the X range of [1:3], so you cannot take log here. Gnuplot produces a vertical line at the lowest point which has a positive value (here, that is X=4). To remove this vertical line, put "?" at Y1=0.0, like "?0.0". Then the Y1

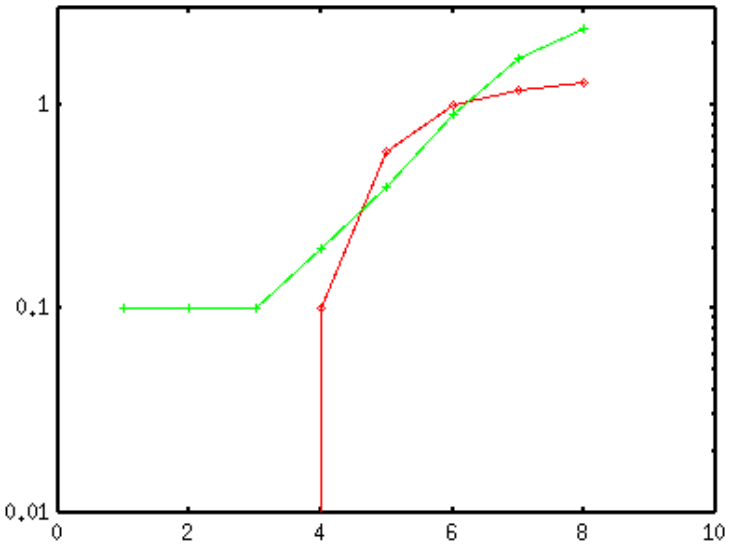

Figure 159: sample7.5c

data (the red line) are treated as that they begin at X=4. The same thing you can do with the plot command option `every ::3` . See the next topic.

14.6. How do I plot a part of data in a file?

To specify a range of the data to be plotted, use the `every` option in the plot command. To skip every two lines, say `plot "test.dat" every 2`

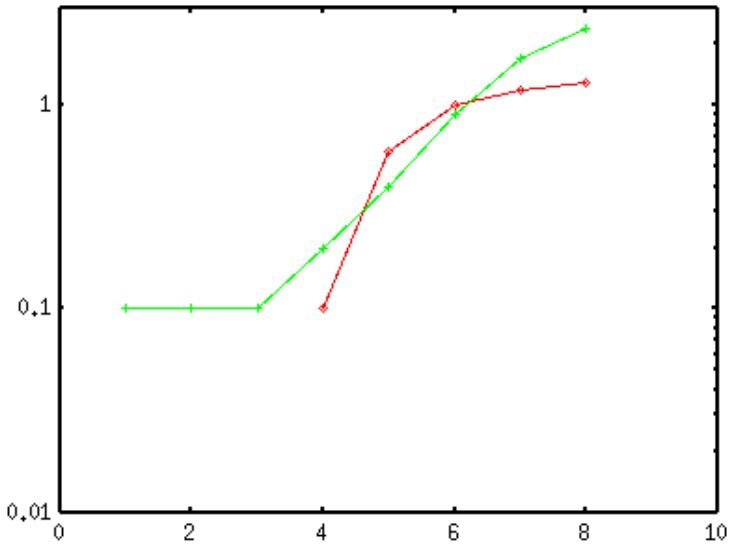

Figure 160: sample7.5d

When the data file contains several data blocks those are separated by a single blank line, you can skip the data block by the `every` option. To skip every two block, try `plot "test.dat" every :2`

```
 every I:J:K:L:M:N
I  Line increment
J  Data block increment
K  The first line
L  The first data block
M  The last line
N  The last data block
every 2  plot every 2 line
every ::3  skip the first 3 lines
every ::3::5  plot from the 4-th to 6-th lines
every ::0::0  plot the first line only
every 2::::6  plot the 1,3,5,7-th lines
every :2  plot every 2 data block
every :::5::8  plot from 5-th to 8-th data blocks
```

Alternatively (if you are on the UNIX-like system), a part of your data file can be plotted by using the unix commands, "head" and "tail".

```
gnuplot> plot "< head -10 test.dat" using 1:2 with lines
gnuplot> plot "< tail -3 test.dat" using 1:2 with lines
gnuplot> plot "< head -5 test.dat" using 1:2 with lines,\
```

```
>        plot "< tail -5 test.dat" using 1:2 with points
```

The first "plot" command says to plot the first 10 lines in the data file "test.dat", and the second "plot" means to show the last 3 lines in the data file. The next lines are an example to draw a graph of the data file "test.dat" — the first 5 points are shown by lines, and the last 5 points are by symbols.

14.7. How do I use UNIX commands inside gnuplot

With gnuplot you can modify the data in your file at plotting , but the flexibility is limited. For example, the data to be plotted are separated into two files - X data are in 'a.dat' and Y data are in 'b.dat', or you want to modify a specific data point in the file. Of course the best way is to edit your datafile before you plot it, but sometimes UNIX commands are helpful.
Here we use the following commands.

- sort
- paste
- sed
- awk

14.7.1. sort

With the `sort` command, you can concatenate / merge several data files.

```
#file1.dat
   1.0   0.1
   2.0   0.2
   5.0   0.5
   8.0   0.8
#file2.dat
   5.0   1.0
  10.0   1.2
  12.0   1.3

 % sort -n file1.dat file2.dat
   1.0   0.1
   2.0   0.2
   4.0   0.6
   5.0   0.5
   5.0   1.0
   8.0   0.8
  10.0   1.2
  12.0   1.3
```

Two datasets are shown by symbols, then all data points are connected by one line.

```
gnuplot> plot "< cat -n file1.dat file2.dat" using 1:2  w l,\
>             "file1.dat" u 1:2 w p,\
>             "file2.dat" u 1:2 w p
```

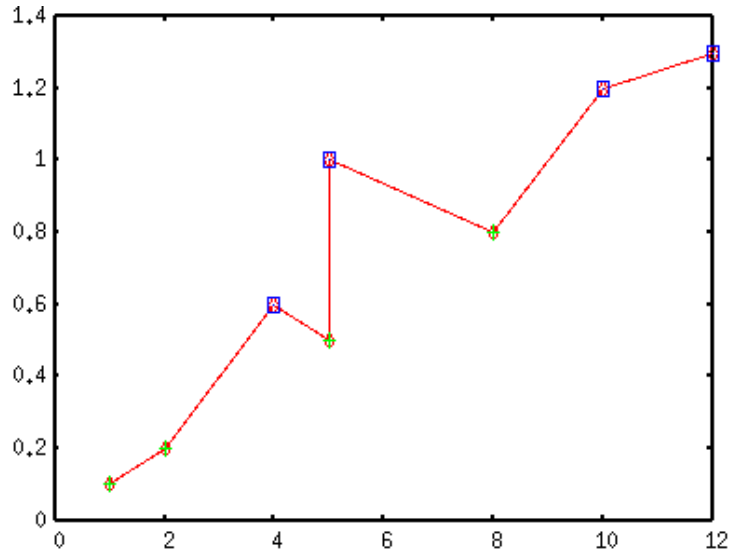

Figure 161: sample7.7a

14.7.2. paste

`Paste` command puts more than two files side by side.

```
#file1.dat
   1.0   0.1
   2.0   0.2
   3.0   0.5
   4.0   0.8
#file2.dat
   1.0   0.6
   2.0   1.0
   3.0   1.2
   4.0   1.3
\end {verbatim}
\begin{verbatim}
 % paste file1.dat file2.dat
   1.0   0.1       1.0   0.6
```

```
2.0   0.2       2.0   1.0
3.0   0.5       3.0   1.2
4.0   0.8       4.0   1.3
```

To draw figure of (X,Y) pairs when X data are in 'file1.dat' and Y data are in 'file2.dat':

```
gnuplot> plot "< paste file1.dat file2.dat" using 2:4  w lp
```

You can compare two files with this command. The `paste` command writes Y data of 'file1.dat' in the second column, and those of 'file2.dat' are in the fourth column, therefore difference between two data sets are expressed by `$2-$4`, and the ratios are by `$2/$4`.

```
gnuplot> plot "< paste file1.dat file2.dat" using 1:($2/$4) w points
```

14.7.3. sed

The stream editor `sed` is very powerful tool to edit your text data. It is impossible to explain everything here, so I show some useful examples. The above data file 'file1.dat' is used as the example.
Shift X data of 1.0 to 1.2.

```
gnuplot> plot '<sed "s/^   1.0/   1.2/g" file1.dat'
```

Show all data points, but draw a line which does not pass through (2.0,0.2).

```
gnuplot> plot 'file1.dat'                   u 1:2 with points,\
>             '<sed "/^   2.0/d" file1.dat' u 1:2 with lines
```

Remove the third line.

```
gnuplot> plot '<sed "3d" file1.dat' u 1:2 with points
```

14.7.4. awk

`Awk` is also very powerful language with which you can manipulate each data column in a file.
Cumulative values of Y data are plotted with a bar graph. The last value becomes the sum of Y data.

```
gnuplot> plot "<awk '{x=x+$2; print $1,x}' file1.dat" with boxes
```

Change Y into zero when Y is less than 0.2.

```
gnuplot> plot  "<awk '{print $1,($2<0.1) ? 0.0 : $2}' file1.dat" with lines
```

Multiply Y-values by 5 when X is in the range of [1:3].

```
gnuplot> plot  "<awk '{print $1,($1<=3 && $1>=1) ? $2*5 : $2}' file1.dat" with lines
```

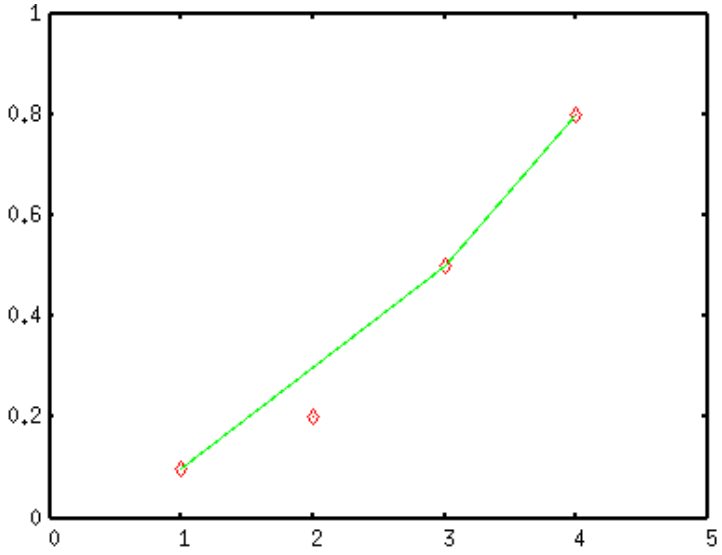

Figure 162: sample7.7b

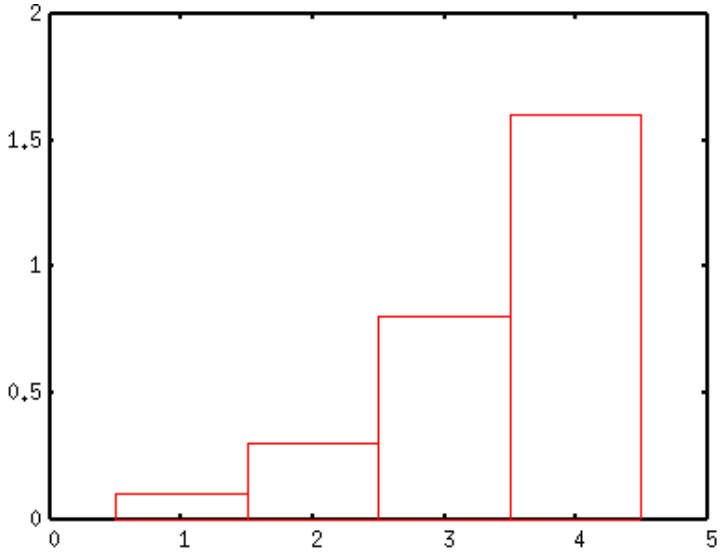

Figure 163: sample7.7c

14.8. plotting time-sequence data

To plot a data file which contains date or time, you need to tell gnuplot that the data are date/time type by the command `set xdata time` or `set ydata time`, then specify the data file format by `set timefmt`. Let's begin with the following example:

```
2004-02-09  310
2004-02-10  185
2004-02-11  239
2004-02-12  132
2004-02-13   85
2004-02-14   57
2004-02-15    8
```

A corresponding `timefmt` to read this data is "%Y-%m-%d". In this case zero's are padded when month and/or date are less than 10, but gnuplot can read data without this padding (2004-2-9, for example).

```
gnuplot> set xdata time
gnuplot> set timefmt "%Y-%m-%d"
gnuplot> plot "sample.dat" using 1:2 with boxes
```

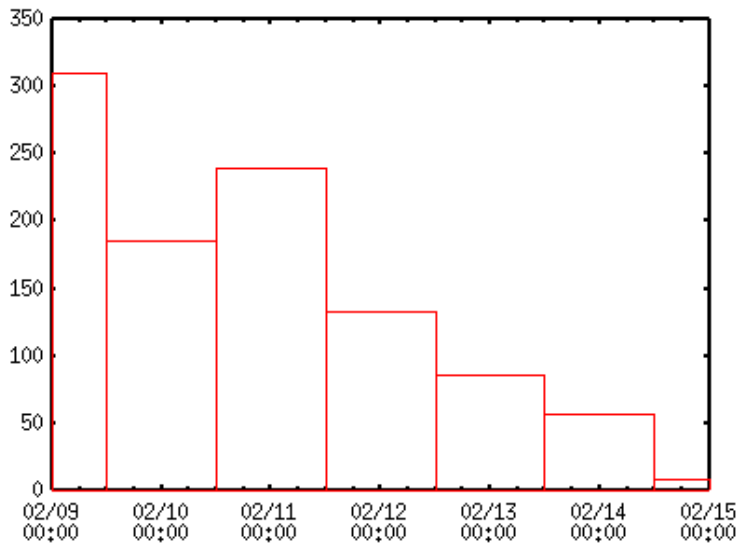

Figure 164: sampleS1.1

Reading date/time data with gnuplot is rather flexible. Here are some examples of the format given for `timefmt`.

```
 data     timefmt     comment
```

```
2004/4/6  %Y/%m/%d  2004/04/06 works well
December/96  %B/%y  warning if mis-spelled
2004/Jan  %Y/%b  3-letters abbreviation
1970/240  %Y/%j  "%j" is a day of the year (1-365)
02:45:03  %H:%M:%S  "%H", 24-hour
1076909172  %s  seconts since 1/1/1970 00:00
```

14.9. Changing output text

When gnuplot reads in date/time type data as X-axis data, it generates appropriate X-tics automatically. In the example above, which are time sequence data for one week, gnuplot writes "month/day" on the X-axis, plus time of "00:00" which we may not need. To control the output format, use `set format` . This format specification not the same as usual numerical output, but it is a format for the strftime function of UNIX library.

```
gnuplot> set xdata time
gnuplot> set timefmt "%Y-%m-%d"
gnuplot> set format x "%b/%a"
gnuplot> plot "sample.dat" using 1:2 with boxes
```

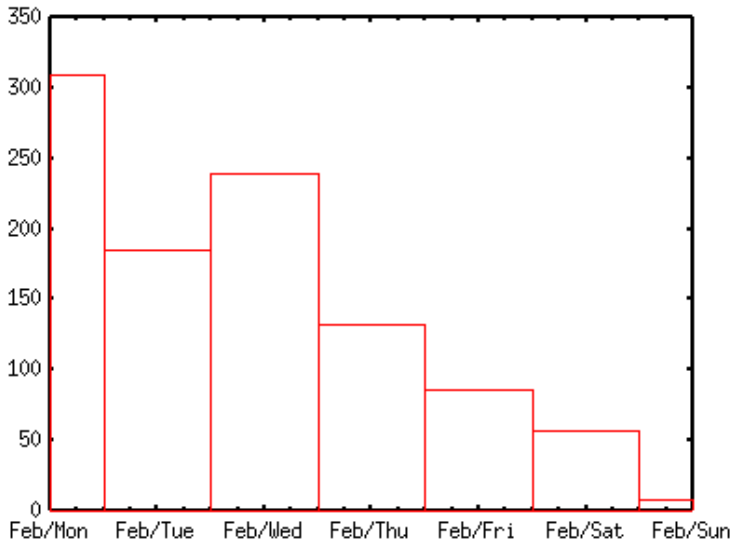

Figure 165: sampleS2.1

In addition to the format-characters shown above, you can use "%a" which does not exist in `timefmt` . With the "%a" format gnuplot gives you a weekday name. See manual of strftime function in detail. Anyway, formats such as "Y, y, m, d, H, M, S" are probably enough for almost all cases.

15. After Plottng

15.1. I want to put a figure in my TeX document

Gnuplot has a latex terminal which is capable of generating a PicTeX graph. This terminal, however, tends to makes a lot of data points to show a graph, and sometimes you cannot process it with the TeX system because a computer memory is not enough. An easy way to include your graph into the TeX document, make the graph in an EPS format and use the LaTeX graphics (graphicx) package.
Firstly, make a figure with gnuplot, and use the enhanced EPS terminal to make an EPS file.

```
gnuplot> set term postscript eps enhanced
gnuplot> set output "test.eps"
gnuplot> set key left top
gnuplot> set size 0.5,0.5
gnuplot> set xrange [0:4]
gnuplot> set yrange [0:4]
gnuplot> set xlabel "Energy [MeV]"
gnuplot> set ylabel "Cross Section [b]"
gnuplot> set linestyle 1 lt 1
gnuplot> set linestyle 2 lt 1 pt 7
gnuplot> # for gnuplot ver.4
gnuplot> # set style line 1 lt 1
gnuplot> # set style line 2 lt 1 pt 7
gnuplot> plot 0.2536*x+1.1717      title "LESQ fit" \
>                                  with lines      linestyle 1,\
>             "test.dat" usi 1:2:3 title "data" \
>                                  with yerrorbars linestyle 2
```

If you don't change the figure size, letters in the generated EPS become too small in comparison with the graph. To make them larger, use the `set term` option to specify the larger font, or use the `set size` command to make the whole figure smaller, just like the example above. The font size is the same even you change the size of figure, so that the letters become larger relative to the graph.
When symbols in your figure have error bars, the PostScript driver draws the bars with the specified line type. If those are not the number 1 (solid line), gnuplot the error bars with draws dotted or dashed lines. To avoid this, use `linestyle` to set the linetype `lt=1` and change the symbol number `pt`.
To paste a graph in your TeX(LaTeX2e) document, use graphics package.
`includegraphics EPS file` command inserts the figure at that location. The EPS file name is specified at EPS file. If you want to chance the figure size, use
`resizeboxX sizeY size` command, which enlarges or shrinks the figure size. If you want to keep the aspect ratio, put "!" into the X or Y size. In the next example, a figure is inserted into the figure environment, a caption is given, and the figure width is set to 120mm.

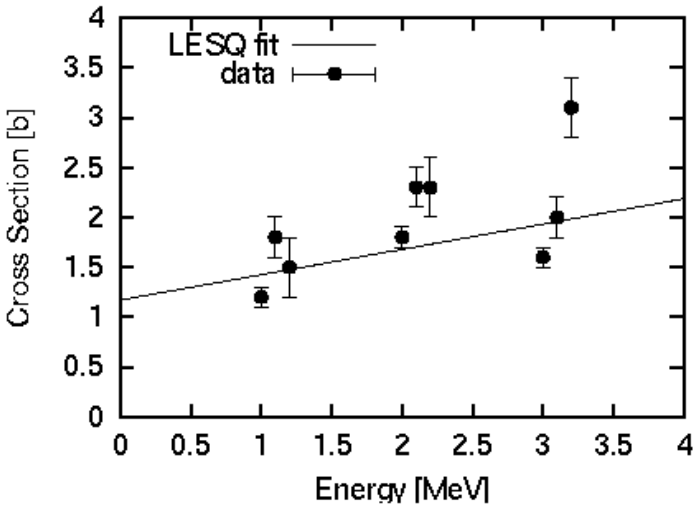

Figure 166: sample8.1a

```
\documentclass{article}
\usepackage{graphics}
\begin{document}
  \begin{figure}
    \begin{center}
      \resizebox{120mm}{!}{\includegraphics{test.eps}}
      \caption{This is a sample figure.}
      \label{test}
    \end{center}
  \end{figure}
\end{document}
```

15.2. I want to merge several figures into one figure

There are several ways to merge some figures, for example, use multiplot, combine EPS figures into one drawing. Maybe the simplest way to do is to use a LaTeX tabular environment, and arrange the EPS figures.
When you have 4 figures, and those are aligned into 2x2 matrix. The Y label appears only the left figures, while the lower figures have X label. In addition, a caption is given for 4 drawings, namely those 4 plots are regarded as one figure.
Firstly, those 4 figures are prepared by 4 different EPS files. At that time, all figures should be in the same size. As an example, here shows the graphs of y=sin(x), cos(x), sin(2x), and cos(2x).

```
gnuplot> set term postscript eps enhanced
```

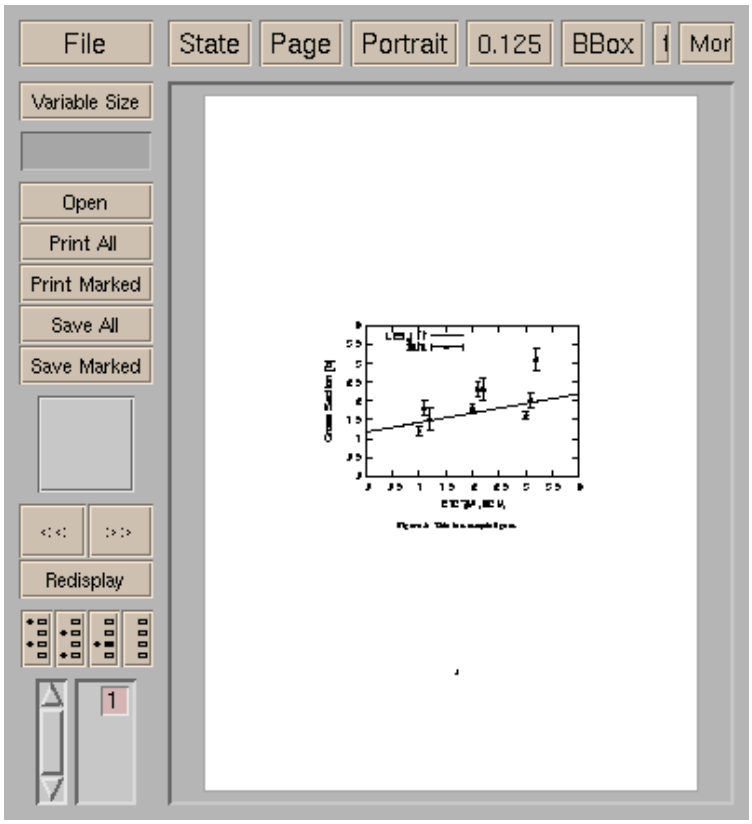

Figure 167: sample8.1b

```
gnuplot> set lmargin 5
gnuplot> set bmargin 3
gnuplot> set rmargin 2
gnuplot> set tmargin 2
gnuplot> set size 0.5,0.5
gnuplot> set xrange [0:2*pi]
gnuplot> set yrange [-1:1]

gnuplot> set output "test1.eps"
gnuplot> set xlabel ""
gnuplot> set ylabel "sin x"
gnuplot> plot sin(x) notitle

gnuplot> set output "test2.eps"
gnuplot> set xlabel ""
gnuplot> set ylabel ""
gnuplot> plot sin(2*x) notitle

gnuplot> set output "test3.eps"
gnuplot> set xlabel "X [nodim.]"
gnuplot> set ylabel "cos x"
gnuplot> plot cos(x) notitle

gnuplot> set output "test4.eps"
gnuplot> set xlabel "X [nodim.]"
gnuplot> set ylabel ""
gnuplot> plot cos(2*x) notitle
```

Those EPS figures are inserted into the tabular environment, and aligned.

```
\documentclass{article}
\usepackage{graphics}
\begin{document}
\begin{figure}
  \begin{center}
    \begin{tabular}{cc}
      \resizebox{60mm}{!}{\includegraphics{test1.eps}} &
      \resizebox{60mm}{!}{\includegraphics{test2.eps}} \\
      \resizebox{60mm}{!}{\includegraphics{test3.eps}} &
      \resizebox{60mm}{!}{\includegraphics{test4.eps}} \\
    \end{tabular}
    \caption{This is sample figures.}
    \label{test4}
  \end{center}
\end{figure}
\end{document}
```

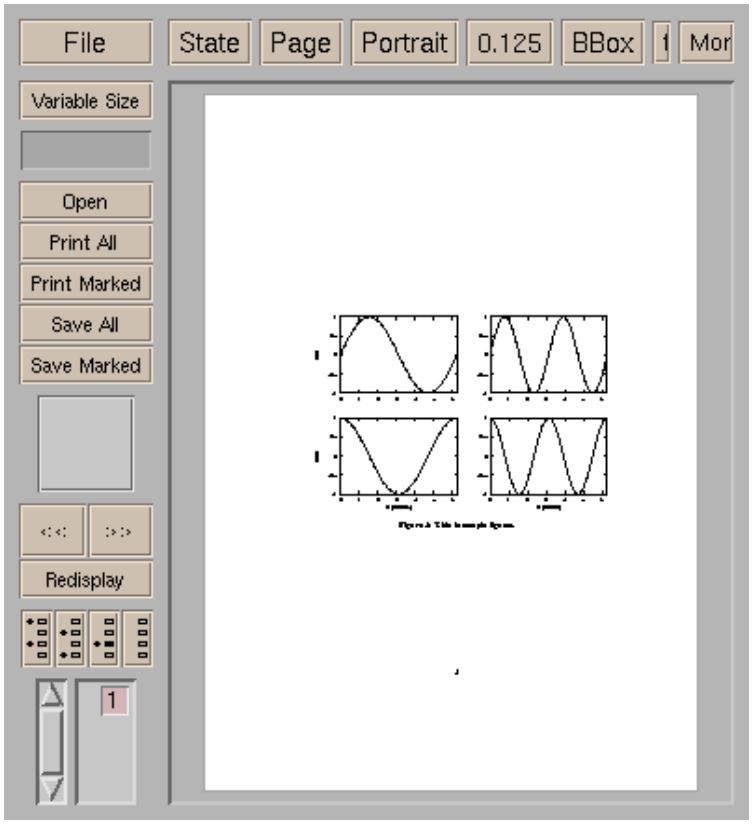

Figure 168: sample8.2b

If you want to make one PS file which contains several figures, use the tabular environment (not table float), and cut off with the dvips command line option `-pp` . With the `-E` option, you can make another EPS file which contains several figures. See the manual of divps.
Of course the same thing you can do with a drawing tools like Tgif or commercial softwares.

15.3. How do I convert a figure into image formats like PNG?

An image format that gnuplot can generate directly is ppm and png. Gnuplot ver.3.7 can make a GIF too. But the GIF file generated by gnuplot is not compressed because of a LZW patent problem, then the file size is usually very large. In addition GIF is no more supported by gnuplot3.8. Main reasons of use of GIF are to put the graph into your Web pages. If so, you can use PNG (Portable Network Graphics) too, as many browsers are capable of displaying the PNG files, and usually the size of PNG file is smaller than GIF.

There are several ways of image format conversion. Here are some examples.

- Display a graph on a X terminal, dump the screen window by "xwd", convert it into various formats with "xv", "ImageMagick", "Gimp", etc.
- Make a graph in the ppm format, then convert it with "ImageMagick", "xv", "pbmplus", "netpbm", "Gimp".
- Make it in a PostScript format, then convert it with "GhostScript" (gs supports several image formats).
- Make a PostScript file, display it with "GhostScript", dump the window by "xwd", convert it with "xv", "ImageMagick", "Gimp", etc.

Netpbm (ppmtoXXX commands) is useful for automatic conversion. The screen dump method with xwd is probably the easiest way.
Firstly display something with gnuplot, then dump that window with xwd.

```
gnuplot> set yrange [-6:6]
gnuplot> set hidden3d
gnuplot> set isosample 40
gnuplot> splot x*sin(y)
gnuplot> shell
 % xwd > screen.xwd
 % exit
exit

gnuplot>
```

Open this "screen.xwd" with xv, Imagemagick (display command), or Gimp, and save it in a image format like PNG. Otherwise, with netpbm

```
 % xwdtopnm < screen.xwd | ppmtopng > screen.png
```

This generates PNG too. The next PNG figure was made with this technique. With GhostScript you have to check what image formats your gs supports first. This can be done with the `--help` option of gs.

```
 % gs -help
 GNU Ghostscript 5.10 (1998-12-17)
 Copyright (C) 1997 Aladdin Enterprises, Menlo Park, CA.  All rights reserved.
 Usage: gs [switches] [file1.ps file2.ps ...]
 Most frequently used switches: (you can use # in place of =)
  -dNOPAUSE           no pause after page   | -q       'quiet', fewer messages
  -g<width>x<height>  page size in pixels   | -r<res>  pixels/inch resolution
  -sDEVICE=<devname>  select device         | -dBATCH  exit after last file
  -sOutputFile=<file> select output file: - for stdout, |command for pipe,
                                          embed %d or %ld for page #
```

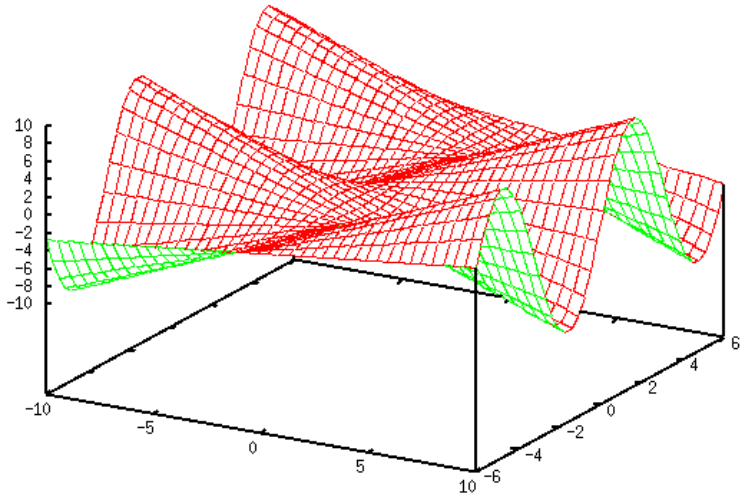

Figure 169: sample8.3b

```
Input formats: PostScript PostScriptLevel1 PostScriptLevel2 PDF
Available devices:
   x11 x11alpha x11cmyk x11gray2 x11mono ml600 npdl epag escpage lbp310
   lbp320 lips2p lips3 lips4 lips4c lips4v lips4vc fmpr mjc180 mjc360 mjc720
   mj500c pr150 jj100 bj10v bj10vh md5000 dmprt deskjet djet500 laserjet
.....
```

Here is an explanation how to convert a PS file into jpeg with gs. First, you make an EPS file with gnuplot, then convert it with gs.

```
% gs -dNOPAUSE -sDEVICE=jpeg -sOutputFile=test.jpg -q
  -dBATCH -g500x350 test.eps
```

The command above should not be folded but in one-line. The `-sOutputFile` specifies the output file name. When `-sOutputFile=-` , the output goes to STDOUT. The `-g` option means the pixel size of image, which depends on the figure size in the EPS file. The JPEG format, as shown below, is mainly used for a photo image, and it is not suitable for images which have strong contrast at borders. To show a graph in a image format, GIF and PNG are better than JPEG EPS by gnuplot, displayed with gv convert into jpeg with gs

15.4. How do I change colors in a PostScript figure?

You cannot specify colors of lines and symbols directly from gnuplot. One of the easiest way to do is, make an obj file of Tgif by setting `set term tgif`, and edit your figure with Tgif, then save it in a color PS format.
Here is a tricky way to change the colors. Edit the postscript file by hand. The PS file

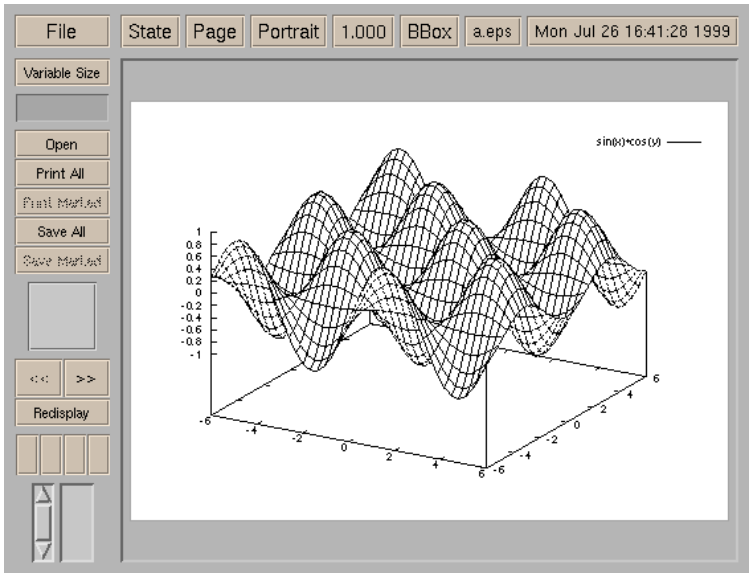

Figure 170: sample8.3c

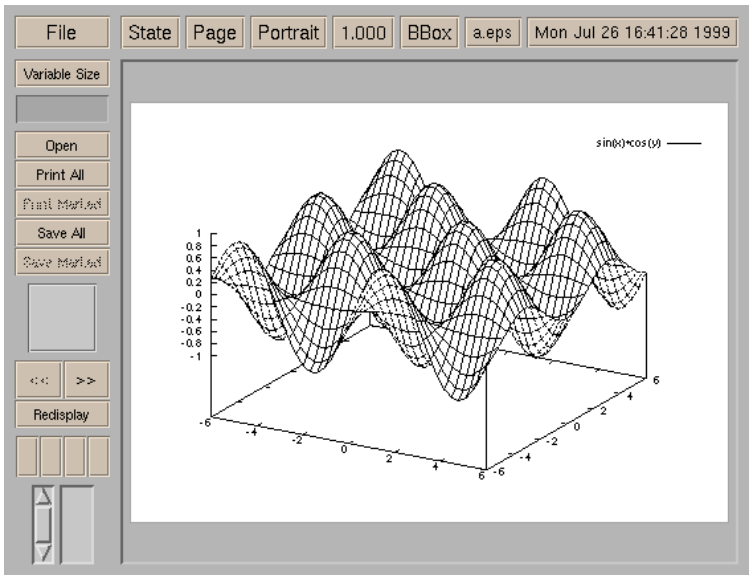

Figure 171: sample8.3c

generated by gnuplot is rather easy to read. With the following gnuplot commands you can get a color index. Of course the `test` command yields the color index, too.

```
gnuplot> set term postscript enhanced color
gnuplot> set output "colorindex.ps"
gnuplot> set size 0.5,0.5
gnuplot> set noborder
gnuplot> set nokey
gnuplot> set linestyle  1 linetype  1 linewidth 8
gnuplot> set linestyle  2 linetype  2 linewidth 8
gnuplot> set linestyle  3 linetype  3 linewidth 8
gnuplot> set linestyle  4 linetype  4 linewidth 8
gnuplot> set linestyle  5 linetype  5 linewidth 8
gnuplot> set linestyle  6 linetype  6 linewidth 8
gnuplot> set linestyle  7 linetype  7 linewidth 8
gnuplot> set linestyle  8 linetype  8 linewidth 8
gnuplot> set linestyle  9 linetype  9 linewidth 8
gnuplot> set linestyle 10 linetype 10 linewidth 8
gnuplot> # for gnuplot ver.4
gnuplot> # set style line 1 line type 1 linewidth 8
gnuplot> # ... etc
gnuplot> set noxtics
gnuplot> set ytics nomirror 1
gnuplot> set yrange [ -1.5 : 10.5 ]
gnuplot> plot  1 w l ls  1,  2 w l ls  2,  3 w l ls  3, 4 w l ls  4,\
gnuplot>       5 w l ls  5,  6 w l ls  6,  7 w l ls  7, 8 w l ls  8,\
gnuplot>       9 w l ls  9, 10 w l ls 10, -1 w line -1, 0 with line 0
gnuplot> pause -1
```

Ten line types are defined in the PostScript terminal, those are numbered from 1 to 9. Above, functions y=1 to y=10 are displayed with the various line types. In order to see the color clearly very thick lines are used there.
Do the same thing but with a B/W PostScript. Remove the `color` option, and save the output into "monoindex.ps". You can see the difference between "colorindex.ps" and "monoindex.ps" with the diff command on UNIX. The difference is only one line, `/Color true def` and `/Color false def` . Therefore you can choose color or B/W without gnuplot by editing this part `/Color true |false`.
ets read the PS file generated above, you can easily find the place where colors are defined. Near line 40th, you see the following section (extra spaces are inserted here to align).

```
/LT0 {PL [                                        ] 1   0   0   DL} def
/LT1 {PL [4 dl 2   dl                             ] 0   1   0   DL} def
/LT2 {PL [2 dl 3   dl                             ] 0   0   1   DL} def
/LT3 {PL [1 dl 1.5 dl                             ] 1   0   1   DL} def
/LT4 {PL [5 dl 2   dl 1 dl 2 dl                   ] 0   1   1   DL} def
```

```
/LT5 {PL [4 dl 3   dl 1 dl 3 dl                    ] 1   1   0   DL} def
/LT6 {PL [2 dl 2   dl 2 dl 4 dl                    ] 0   0   0   DL} def
/LT7 {PL [2 dl 2   dl 2 dl 2 dl 2 dl 4 dl          ] 1   0.3 0   DL} def
/LT8 {PL [2 dl 2   dl 2 dl 2 dl 2 dl 2 dl 2 dl 4 dl] 0.5 0.5 0.5 DL} def
```

Those lines correspond to the line types 1 to 9. Three numbers in LT0, "1 0 0" defines RGB (Red, Green, and Blue), then LT0 becomes red. In the same manner, LT1 is green, LT2 is blue...and (1,0,1)=Magenta, (0,1,1)=Cyan, (1,1,0)=Yellow, (0,0,0)=Black, (1,0.3,0)=Orange, and (0.5,0.5,0.5)=Gray. Now you can change the line color by changing these RGB values.
To make a gradation from red to yellow, fix the R=1 and Blue=0, and change G from 0 to 1 gradually.

```
/LT0 {PL [                                         ] 1   0    0   DL} def
/LT1 {PL [4 dl 2   dl                              ] 1   0.1  0   DL} def
/LT2 {PL [2 dl 3   dl                              ] 1   0.2  0   DL} def
/LT3 {PL [1 dl 1.5 dl                              ] 1   0.3  0   DL} def
/LT4 {PL [5 dl 2   dl 1 dl 2 dl                    ] 1   0.4  0   DL} def
/LT5 {PL [4 dl 3   dl 1 dl 3 dl                    ] 1   0.5  0   DL} def
/LT6 {PL [2 dl 2   dl 2 dl 4 dl                    ] 1   0.6  0   DL} def
/LT7 {PL [2 dl 2   dl 2 dl 2 dl 2 dl 4 dl          ] 1   0.7  0   DL} def
/LT8 {PL [2 dl 2   dl 2 dl 2 dl 2 dl 2 dl 2 dl 4 dl] 1   0.8  0   DL} def
```

The results are as follows. The first image is generated with gnuplot, while the second one is made with editing the PS file as described above. With this method you can use any colors in you figure. An automatic conversion of those colors can be achieved with some programs like perl or sed. ColorPS generated by gnuplot PS file edited
Now, go back to the postscript data above, you can see "[4 dl 2 dl]" which defines a line pattern to draw dotted, dot-dashed, and dashed lines. For LT1, the pattern is defined as "[4 dl 2 dl]". This gives a solid line of 4 unit length and a space of 2 unit length, so that it becomes a dashed-line. "[5 dl 2 dl 1 dl 2 dl]" defines a solid line of 5 unit length, space of 2, line of 1, and space of 2, and so on. If you need an extra line pattern, edit here to make your own.
In the PS file you can find the following 2 lines, those are just above the definition of LT0.

```
/LTb { BL [] 0 0 0 DL } def
/LTa { AL [1 dl 2 dl] 0 setdash 0 0 0 setrgbcolor } def
```

The line LTb is used for the border of graph, and LTa is for the zero axes. Their color is black since RGB is "0 0 0". If you want to change those color, edit here.

15.5. I want to get rid of a right-side margin in a square figure

A square figure can be drawn by `set size square`. However, an EPS file generated with gnuplot is not square but still rectangular, and you get extra margin. The reason is that gnuplot determines BoundingBox so as to cover the whole screen. In order to get

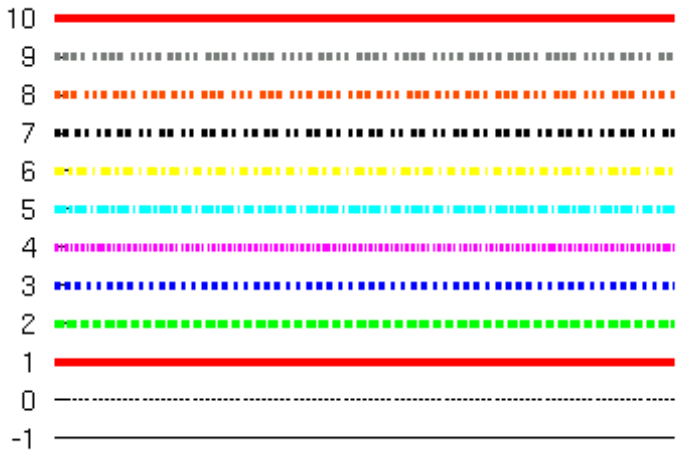

Figure 172: sample8.4a

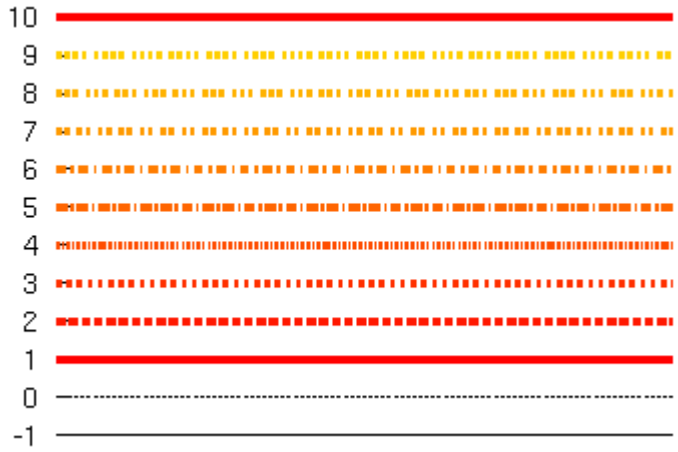

Figure 173: sample8.4b

rid of the right-side margin, you need to edit the EPS file by hand. The BoundingBox is defined at the top of the generated EPS file.

```
%%BoundingBox: 50 50 410 302
```

This "410" is too large in this case, then reduce it. An appropriate BoundingBox coordinate depends on the label and margin, so that you may find the best number after some try-and-errors. In the above case, about 320 works fine because the vertical size is 302.
If you are working on Unix or Linux, and your system has GhostScript, probably you have `ps2ps, eps2eps` commands. Those shell scripts convert your PostScript graphs generated by gnuplot into more proper PS file. You can adjust your BoundingBox with the following command.

```
% eps2eps input.eps output.eps
```

16. Miscellaneous Stuff

16.1. Ternary operator (A ? B : C)

The ternary operator works just the same as that in C-language. The ternary operator is expressed by A ? B : C. Firstly A is evaluated, if it is true (non-zero) B is executed; otherwise C is executed. The next example is a graph which is incontinuous at x=0, y=exp(-x) for x¿0, and y=exp(4*x) for x¡=0.

```
gnuplot> set xrange [-5:5]
gnuplot> plot x>0 ? exp(-x) : exp(4*x)
```

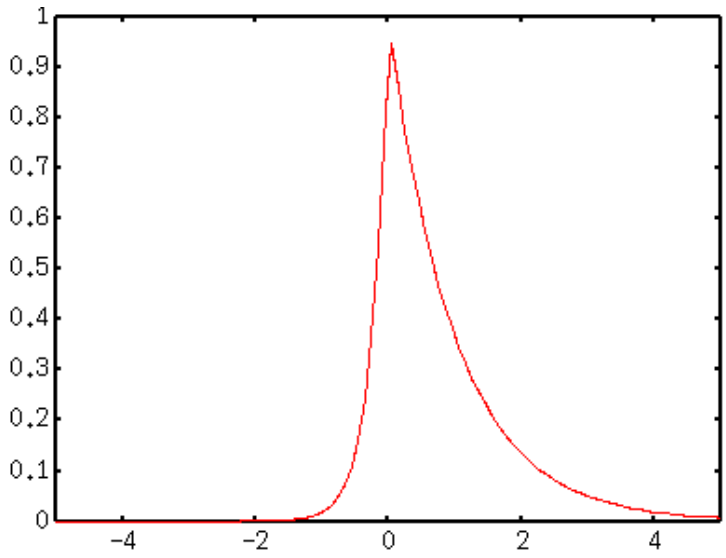

Figure 174: sample9.1

More complicated functions can be definced with this ternary operator. Here is an example of user-defined function which varies discontinuously along with the X values.

```
gnuplot> f(x)= (abs(x)<1) ? 100 : ( (abs(x)<2) ? 50 : ((abs(x)<3) ? 20: 10 ))
gnuplot> set xrange [-5:5]
gnuplot> set yrange [0:150]
gnuplot> set sample 1000
gnuplot> plot f(x)
```

This function gives 100 if the absolute value of X is less than 1, 50 for $1 <= |X| < 2$, 20 for $2 <= |X| < 3$, and 10 otherwise. The Y values are not continuos between those X regions, but gnuplot connects those discrete Y values by line. The `set sample` command makes those "connected parts" be as vertical as possible.

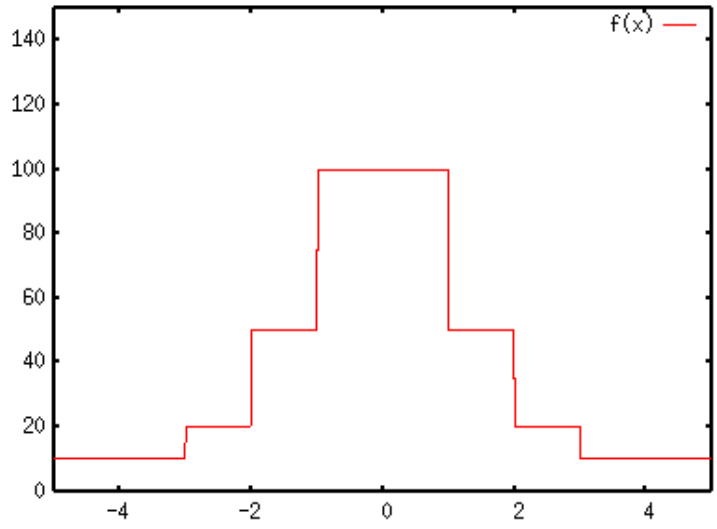

Figure 175: sample9.1b

16.2. Broken functions

Gnuplot ignores any mathematically undefined expressions like zero division (/0), as noted in Gnuplot FAQ. With this together with the ternary operator shown above, you can plot discontinuos functions.

```
gnuplot> f(x) = (abs(x)>pi/2) ? sin(x+pi/2) : 0
gnuplot> g(x) = (abs(x)>pi/2) ? sin(x) : 1/0
gnuplot> set xrange [-2*pi:2*pi]
gnuplot> set yrange [-1:1]
gnuplot> plot g(x) w l lw 2,f(x)
```

We comapre two sine functions, f(x) and g(x). f(x) is zero when $|x|$ is smaller than 90 deg, while g(x) has a special expression, one over zero. I have moved f(x) slightly to see the difference of those two.

16.3. Loop

Gnuplot has a `reread` and `if` commands. With those you can make a simple loop. The command `reread` re-reads the file again. When gnuplot finds this command in the control file, an endless-loop begins. To terminate the loop, use the `if` command, and you can determine the number of iteration.
Let's make a simple animation by rotating a 3D plot. To rotate the drawing, change the view point angle from 0 to 360 with 5 degrees step. Firstly, make a "loop.plt" file whose content is as follows. This is a figure of function `z=exp(-x*x)*erf(y)`.

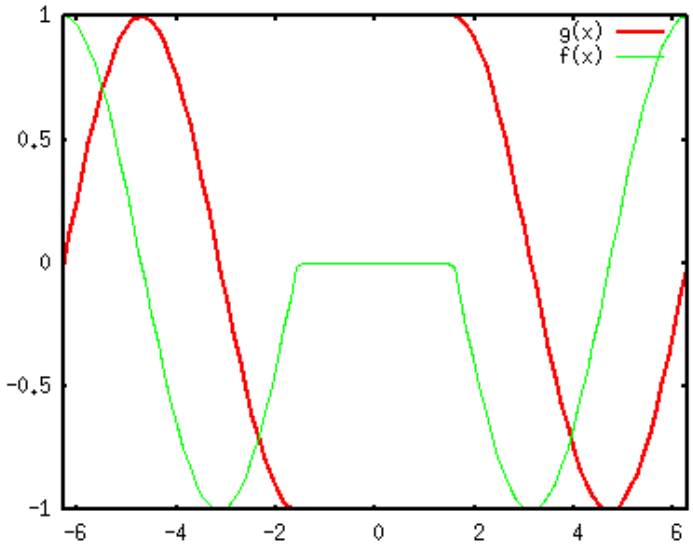

Figure 176: sample9.1c

```
theta = theta + 10
set view 60, theta
splot exp(-x*x)*erf(y)
if(theta<360) reread
```

An initial value of `theta` and the other setting are given outside the 'loop.plt' file. The endless loop begins when gnuplot reads the `load "loot.plt"` command. You can make an animated GIF with some tool, but usually you have to pay for it :-p

```
gnuplot> set nokey
gnuplot> set noxtics
gnuplot> set noytics
gnuplot> set noztics
gnuplot> set border 0
gnuplot> set isosamples 40, 40
gnuplot> set hidden3d
gnuplot> set xrange [ -5 : 5 ]
gnuplot> set yrange [ -5 : 5 ]
gnuplot> theta = 5
gnuplot> load "loop.plt"
```

17. Special Functions

17.1. Spherical Harmonics

To show an example of 3-dim functional plot, here we present the well-known Spherical Harmonics. The function in the polar coordinate is expressed by parameters, and the function for some l and m values are displayed.

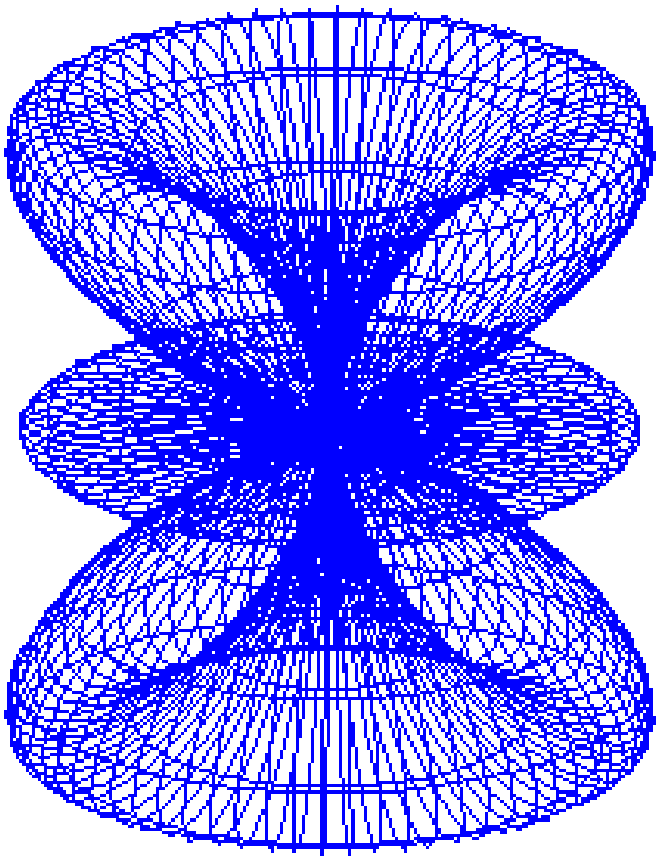

Figure 177: sphtitle

17.1.1. Parametric Expression

There are two ways to display a 3-dimensional function. The first one is to use variables x,y , and define z values as z=f(x,y). The other method is to define a 3-D function by

two parameters u,v as :

$$x = f(u, v)$$
$$y = g(u, v)$$
$$z = h(u, v)$$

When your function f(x,y) is not so complicated, the former method is easier.

```
gnuplot> f(x,y)=sin(x)*cos(y)
gnuplot> splot f(x,y)
```

However a function defined in the polar coordinate is hard to express in a form "z=", so that we need to use a parametric representation. For example, a sphere of unit radius is expressed as x$\hat{2}$+y$\hat{2}$+z$\hat{2}$=1, then one has to define two functions (z ¿ 0 and z ¡ 0), namely z=sqrt(1-x*x-y*y) and z = -sqrt(1-x*x-y*y).
The sphere can be expressed with two parameters u and v those define the condition that the radius is constant. Suppose u and v are the angles as follows, and alter those values from zero to 360 under the condition that the radius r does not change. With

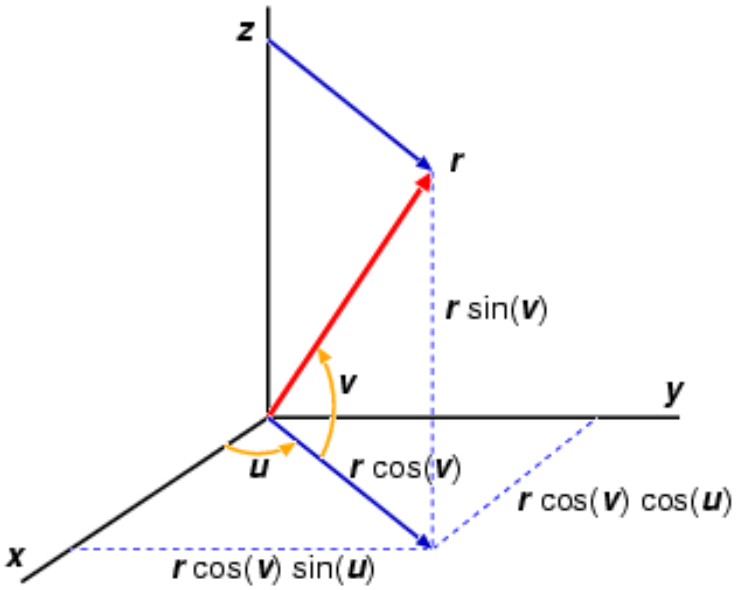

Figure 178: coordinate1

the angles u,v and radius r, an arbitrary coordinate x,y,z can be expressed as follows:

$$x = r * cos(u) * cos(v)$$
$$y = r * sin(u) * cos(v)$$
$$z = r * sin(v)$$

The sphere can be defined in the polar coordinate so as that the radius r=a=constant, then one can get the sphere when the parameter r is replaced by a constant value. When a=1, the functions above represent the sphere of unit radius.

$$x = cos(u) * cos(v)$$
$$y = sin(u) * cos(v)$$
$$z = sin(v)$$

Therefore to draw a 3-D sphere with gnuplot:

```
gnuplot> set parametric

        dummy variable is t for curves, u/v for surfaces
gnuplot> set angle degree
gnuplot> set urange [0:360]
gnuplot> set vrange [-90:90]
gnuplot> set isosample 72,36
gnuplot> set ticslevel 0
gnuplot> set size 0.7,1.0
gnuplot> a=1
gnuplot> splot a*cos(u)*cos(v),a*sin(u)*cos(v),a*sin(v)
```

The default angle unit is radian. The second line `set angle degree` changes the angle unit into degree. The third and fourth lines tell gnuplot to change the parameters u,v from zero to 360 deg. We changed the aspect ratio by the `set size` command in order to make the sphere be spherical on your screen (otherwise the drawing becomes like a pancake).
The roughness of wire-frame can be controlled by `set isosample m,n`. The larger the number is, the finer the mesh becomes. The number m is for the parameter u, while n is for v.
The functions used above can be more in a convenient form when one defines functions:

```
gnuplot> fx(u,v)=cos(u)*cos(v)
gnuplot> fy(u,v)=sin(u)*cos(v)
gnuplot> fz(v)=sin(v)
gnuplot> splot a*fx(u,v),a*fy(u,v),a*fz(v)
```

17.1.2. Spherical Harmonics

The spherical harmonics in the polar coordinate Y[lm](theta,phi) is given by: where P[lm] is the associated Legendre function. The value l is integer, l=0,1,2,3... For each l value, m takes the value of m=-l,-l+1,...l-1,l so that there are 2l+1 different m values. A square of this function is normalized to unity, ¡ Y[lm] — Y[l'm'] ¿ = delta(ll') delta(m,m'). The theta runs from 0 to pi, while phi runs from 0 to 2pi.
In the gnuplot parametric representation, the angle u is the same as phi, while the

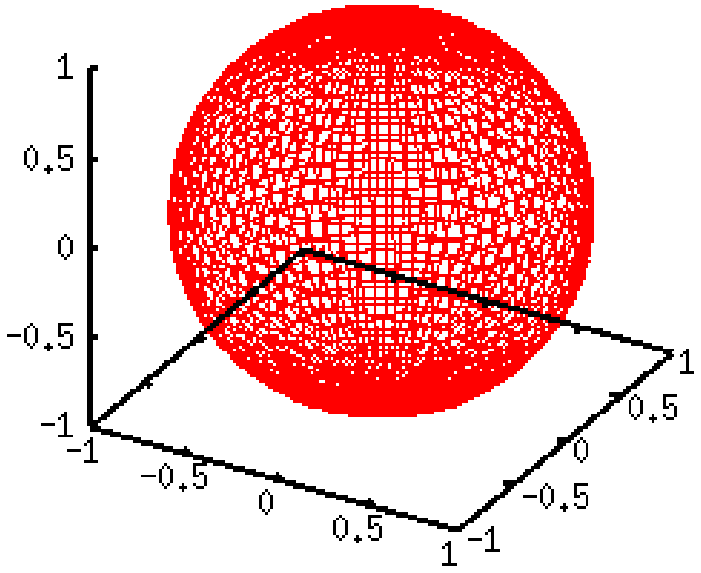

Figure 179: sphere

$$Y_l^m(\theta,\phi) = (-1)^{(m+|m|)/2}\sqrt{\frac{2l+1}{4\pi}\,\frac{(l-|m|)!}{(l+|m|)!}}P_l^{|m|}(\cos\theta)e^{im\phi}$$

Figure 180: eq

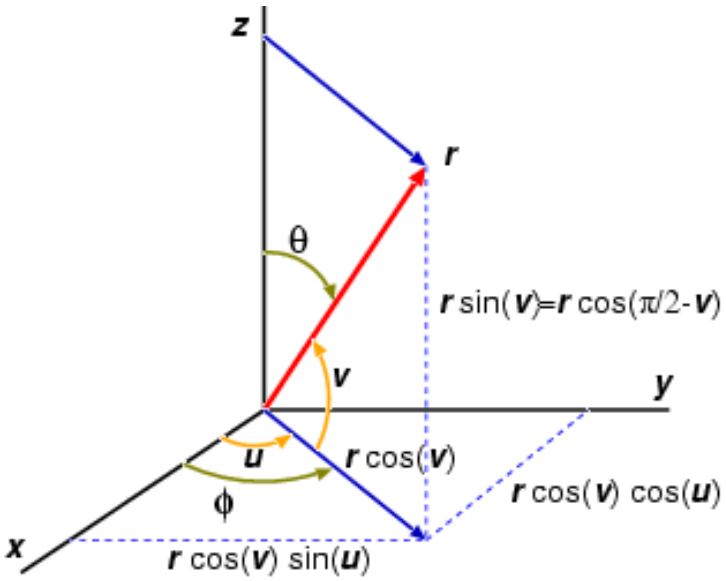

Figure 181: coordinate2

definition of theta is different from v. From the left figure, theta = pi/2 - v, we get:

$$cos(theta) = sin(v)$$
$$sin(theta) = cos(v)$$

The simplest spherical harmonics can be obtained by setting l=0 and m=0. This yields a constant value of 1/sqrt(4pi). A square of this, —Y[00]—$\hat{2}$=1/4pi, is of course constant, and this is a sphere with the radius of 1/4pi. It is easy to draw this function with gnuplot. The procedure is the same as the previous page.

```
gnuplot> set parametric

        dummy variable is t for curves, u/v for surfaces
gnuplot> set angle degree
gnuplot> set urange [0:360]
gnuplot> set vrange [-90:90]
gnuplot> set isosample 36,18
gnuplot> set ticslevel 0
gnuplot> set size 0.65,1.0
gnuplot> a=1.0/(4*pi)
gnuplot> fx(u,v)=cos(u)*cos(v)
gnuplot> fy(u,v)=sin(u)*cos(v)
gnuplot> fz(v)=sin(v)
gnuplot> splot a*fx(u,v),a*fy(u,v),a*fz(v)
```

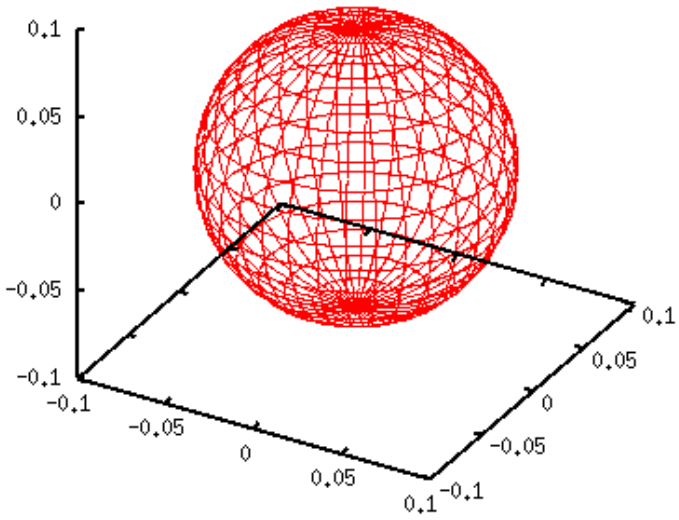

Figure 182: sph00

The left graph is a square of Y[00], which is a sphere whose center is the origin and the radius of 1/4pi. Since m=0, the `exp(-im phi)` term disappears and the function is independent of `phi`. The spherical harmonics becomes real when m is even. If m is an odd number the function has an imaginary term.
Next, we consider the case of l=1 and m=0. This function becomes Y(theta)=sqrt(3/4pi)cos(theta). As shown above, cos(theta) is given by sin(v). The function can be plotted when the x,y,z coordinates are multiplied by sin(v). In the following example we show —Y[10]—$\hat{2}$=3/4pi $\cos\hat{2}$(theta).

```
gnuplot> a=3.0/(4*pi)
gnuplot> g(v)=sin(v)*sin(v)
gnuplot> splot a*g(v)*fx(u,v),a*g(v)*fy(u,v),a*g(v)*fz(v)
```

The cross-section of spherical harmonics can be shown by setting the u range, [0:180], and use the `hidden3d` option.

```
gnuplot> set urange [0:180]
gnuplot> set hidden3d
gnuplot> splot a*g(v)*fx(u,v),a*g(v)*fy(u,v),a*g(v)*fz(v)
```

17.1.3. Various Angular Momenta and Magnetic Quantum Numbers

l=1,m=-1,+1 In the quantum mechanics, l and m values of the spherical harmonics Y[lm] are the angular momentum and magnetic momentum numbers. Here we show

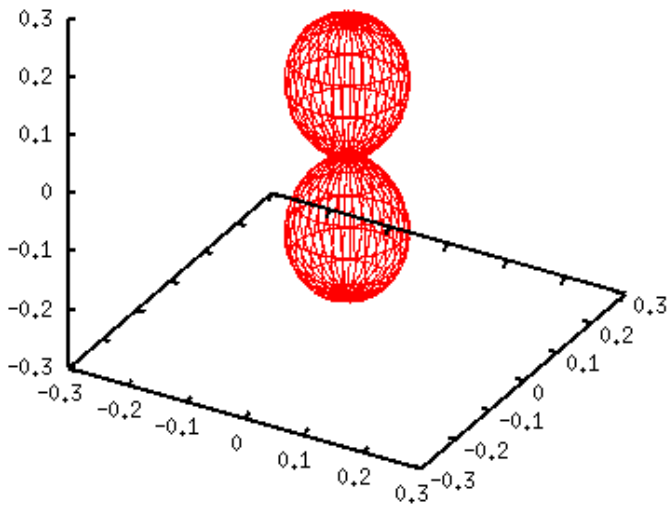

Figure 183: sph10

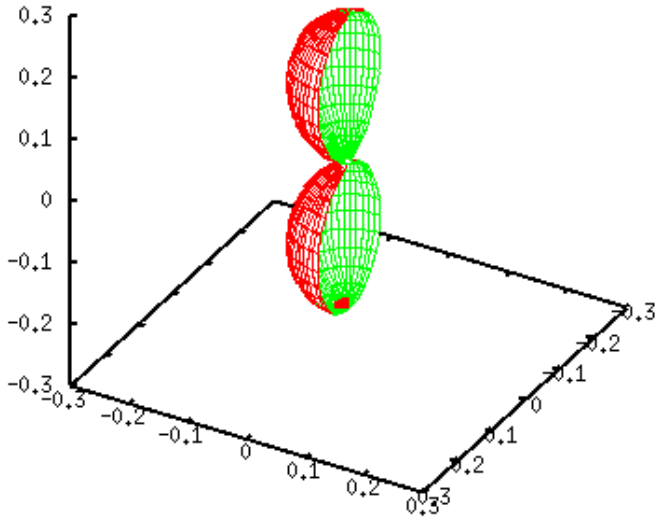

Figure 184: sph10a

several spherical harmonics which have relatively small l values. The method is the same as the previous section — express x,y,z by parameters u and v.
When m is odd, the spherical harmonics contains an imaginary part which comes from exp(-im phi) term. For m=-1 and 1, the function becomes as follows:

```
 Y[1,-1](t,p) =  sqrt(3/8pi) sin(t) exp(-i p)
   Y[1, 1](t,p) = -sqrt(3/8pi) sin(t) exp( i p)
```

—Y—$\hat{2}$ can be calculated easily by multiplying Y and its complex conjugate, and both above become the same function, Y(t)=3/8 pi $\sin\hat{2}$(t). The complex conjugate function of the spherical harmonics can be given by the relation, $(\text{Y[l,m]})\hat{*}$ = (-1)m̂ Y[l,-m].

```
gnuplot> set parametric

        dummy variable is t for curves, u/v for surfaces
gnuplot> set angle degree
gnuplot> set urange [0:360]
gnuplot> set vrange [-90:90]
gnuplot> set isosample 36,18
gnuplot> set ticslevel 0
gnuplot> set size 0.65,1.0
gnuplot> a=3.0/(8*pi)
gnuplot> fx(u,v)=cos(u)*cos(v)
gnuplot> fy(u,v)=sin(u)*cos(v)
gnuplot> fz(v)=sin(v)
gnuplot> g(v)=cos(v)*cos(v)
gnuplot> splot a*g(v)*fx(u,v),a*g(v)*fy(u,v),a*g(v)*fz(v)
```

The left torus-shape figure is —Y[11]—$\hat{2}$. Since this function contains sin(t)=cos(v), we defined g(v)=cos(v)*cos(v) .

l=2,m=-2,-1,0,+1,+2 When l=2, there are 5 different functions those correspond to m=-2 to 2.

```
  Y[2,-2](t,p) =  sqrt(15/32pi) sin(t)sin(t) exp(-2i p)
   Y[2,-1](t,p) =  sqrt(15/8pi) sin(t)cos(t) exp(-i p)
   Y[2, 0](t,p) =  sqrt(5/16pi) (3cos(t)cos(t)-1)
   Y[2, 1](t,p) = -sqrt(15/8pi) sin(t)cos(t) exp( i p)
   Y[2, 2](t,p) =  sqrt(15/32pi) sin(t)sin(t) exp( 2i p)
```

Here we show the case of m=0,1,2. A function of negative m value is the same as that for —m—.

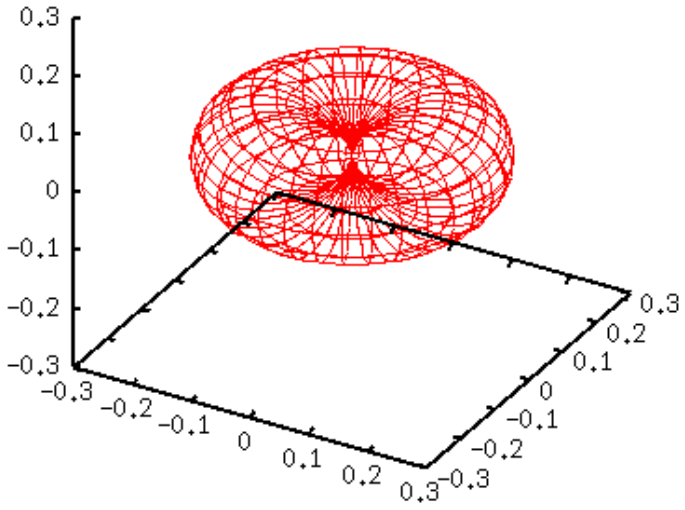

Figure 185: sph11

```
gnuplot> a= 5.0/(16*pi)
gnuplot> g(v)= (3*sin(v)*sin(v)-1)**2
gnuplot> splot a*g(v)*fx(u,v),a*g(v)*fy(u,v),a*g(v)*fz(v)


gnuplot> a=15.0/( 8*pi)
gnuplot> g(v)= (sin(v)*cos(v))**2
gnuplot> splot a*g(v)*fx(u,v),a*g(v)*fy(u,v),a*g(v)*fz(v)


gnuplot> a=15.0/(32*pi)
gnuplot> g(v)= (cos(v)*cos(v))**2
gnuplot> splot a*g(v)*fx(u,v),a*g(v)*fy(u,v),a*g(v)*fz(v)
```

17.1.4. Deformed Nucleus (Legendre Expansion)

[ver.4] ONLY !
The spherical harmonics Y[lm](theta,phi) is reduced to a simple Legendre function p[l](cos(theta)) scaled by a constant when m=0, which is independent of phi. The

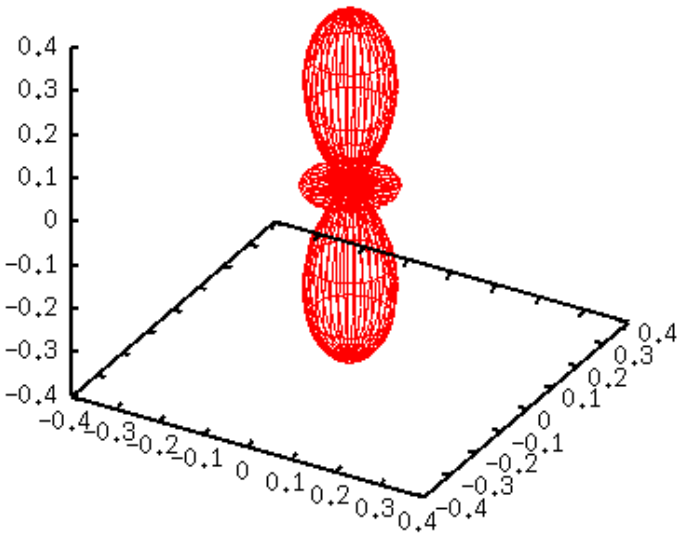

Figure 186: sph20

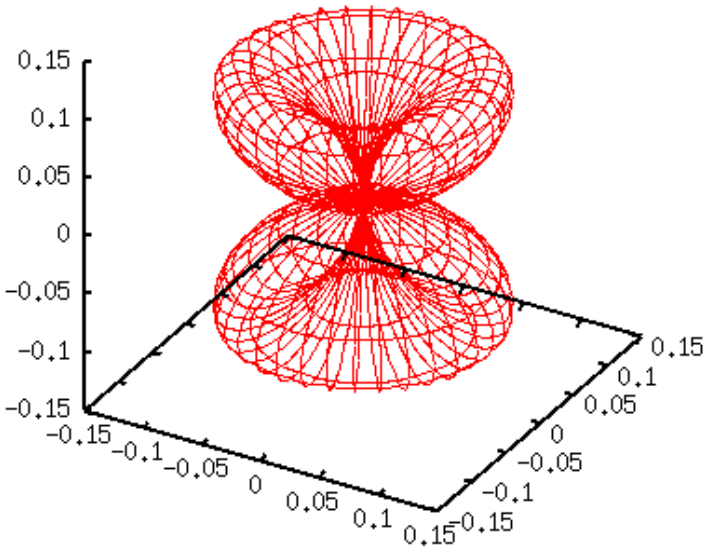

Figure 187: sph21

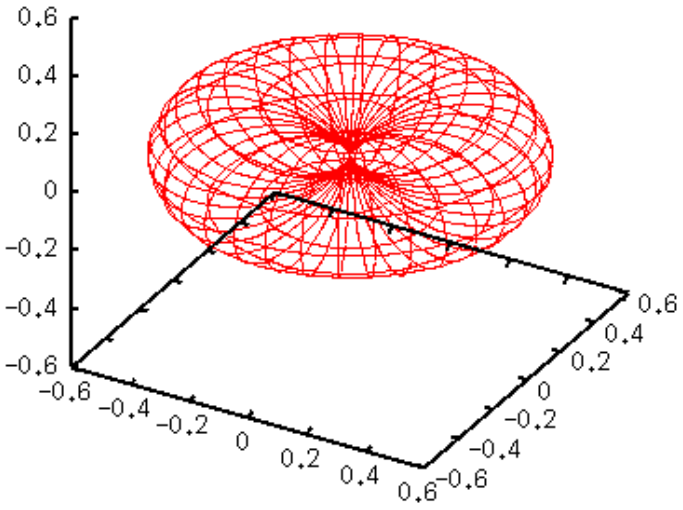

Figure 188: sph22

following equations and figure are Y[l](theta) for several l-values.

$$
\begin{aligned}
Y_0(\theta) &= \sqrt{\frac{1}{4\pi}} \\
Y_1(\theta) &= \sqrt{\frac{3}{4\pi}}\cos\theta \\
Y_2(\theta) &= \sqrt{\frac{5}{4\pi}}\frac{1}{2}\left(3\cos^2\theta - 1\right) \\
Y_3(\theta) &= \sqrt{\frac{7}{4\pi}}\frac{1}{2}\left(5\cos^3\theta - 3\cos\theta\right) \\
Y_4(\theta) &= \sqrt{\frac{9}{4\pi}}\frac{1}{8}\left(35\cos^4\theta - 3ß\cos^2\theta + 3\right)
\end{aligned}
$$

Using this Y[l](theta) with l=lambda=even terms, shape of a deformed nucleus can be expanded as follows:

$$
\begin{aligned}
R &= R_0\left\{1 + \sigma_\lambda\beta_\lambda Y_\lambda(\theta)\right\} \\
&= R_0\left\{1 + \beta_2 Y_2(\theta) + \beta_4 Y_4(\theta) + ..\right\}
\end{aligned}
$$

where the beta a parameter of deformation. If beta=0, the nucleus is spherical. The 3-dim. shape given by this equation is shown with gnuplot. As it is already shown in the previous section, we express the (x,y,z) coordinate with the angles u,v and radius r.

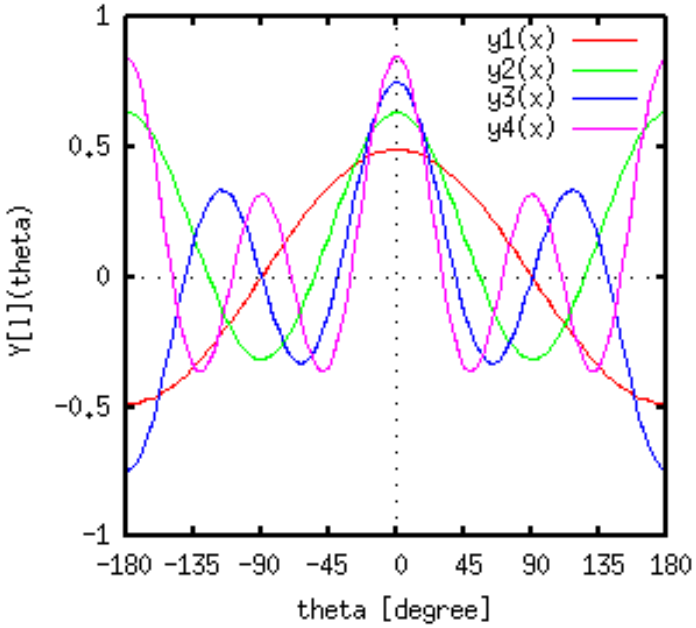

Figure 189: yfunc

```
   x = R(theta)*cos(u)*cos(v)
   y = R(theta)*sin(u)*cos(v)
   z = R(theta)*sin(v)
```

where the theta is the angle measured from the Z-axis, so that the relation between theta and v is theta = pi/2-v. To draw the surface, the parameters u,v are varied from 0 to 360 deg. In the case of beta_2 = 0.3, beta_4 = 0.1, and R_0 = 1:

```
gnuplot> set parametric

        dummy variable is t for curves, u/v for surfaces
gnuplot> set angle degree
gnuplot> set urange [0:360]
gnuplot> set vrange [0:360]
gnuplot> set isosample 16,16
gnuplot> set ticslevel 0
gnuplot> set view 75,25
gnuplot> set size 0.7,1.0
gnuplot> set xrange [-2:2]
gnuplot> set yrange [-2:2]
gnuplot> set zrange [-2:2]
gnuplot> set urange [0:360]
gnuplot> set vrange [0:360]
gnuplot> y0(t)=1.0
gnuplot> y2(t)=sqrt(5.0/(4*pi))*( 3.0*cos(t)**2 -  1.0             )/2.0
gnuplot> y4(t)=sqrt(9.0/(4*pi))*(35.0*cos(t)**4 - 30*cos(t)**2 +3.0)/8.0
gnuplot> b2=0.3
gnuplot> b4=0.1
gnuplot> r(t) = 1 + b2*y2(0.5*pi-t) + b4*y4(0.5*pi-t)
gnuplot> fx(u,v)=cos(u)*cos(v)
gnuplot> fy(u,v)=sin(u)*cos(v)
gnuplot> fz(v)=sin(v)
gnuplot> set pm3d
gnuplot> splot r(v)*fx(u,v),r(v)*fy(u,v),r(v)*fz(v) with lines
```

The deformation parameters, beta_2 and beta_4 can be positive or negative. Here are some examples for some combinations of beta_2 and beta_4. The beta_2 parameters are taken to be -0.4 or 0.4, and for each beta_2, we changed the beta_4 value from -0.2 to 0.2. When beta_2 is positive the shape of nucleus is prolate, while it becomes oblate if beta_2 is negative.

```
gnuplot> set border 0
gnuplot> unset xtics
gnuplot> unset ytics
gnuplot> unset ztics
```

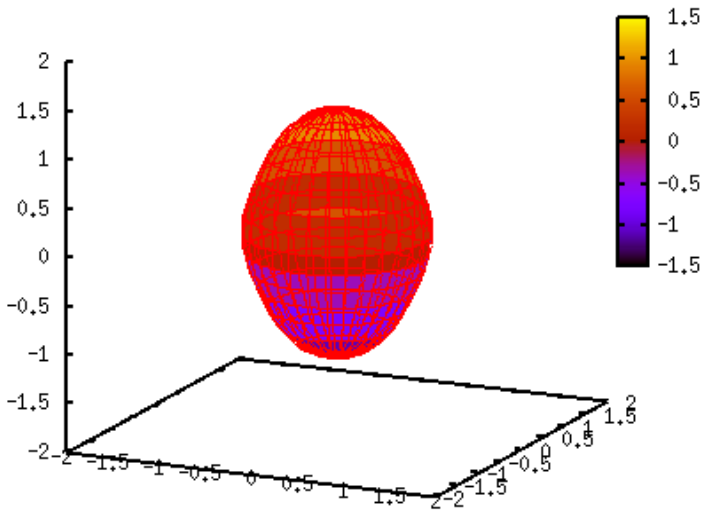

Figure 190: legendre

```
gnuplot> unset colorbox
gnuplot> b2 = -0.4 ; b4 = -0.2 ; replot ; pause -1
gnuplot> b2 = -0.4 ; b4 =  0.0 ; replot ; pause -1
gnuplot> b2 = -0.4 ; b4 =  0.2 ; replot ; pause -1
gnuplot> b2 =  0.4 ; b4 = -0.2 ; replot ; pause -1
gnuplot> b2 =  0.4 ; b4 =  0.0 ; replot ; pause -1
gnuplot> b2 =  0.4 ; b4 =  0.2 ; replot ; pause -1
```

17.2. Fractal

We generate a simple fractal image with gnuplot. To calculate fractal graphics, functions must be defined in a recursive form. At first we explain how to define the recursive functions with gnuplot, then the Mandelbrot set and self-squared fractal images are shown.

17.2.1. Recursive Definition

You can define a function recursively, so that you can include your defined function in itself. For example, a factorial of integer number N!=FAC(N) is written as FAC(N)=N*FAC(N-1), you can define this in gnuplot as:

```
gnuplot> fac(n) = n * fac(n-1)
```

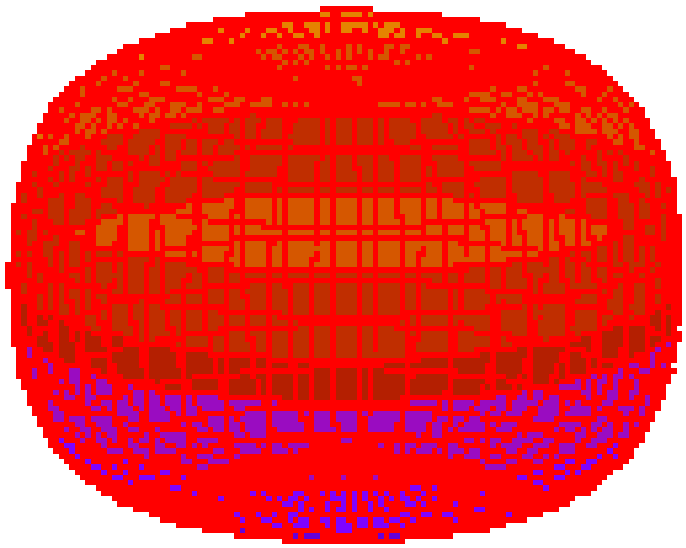

Figure 191: legendre1

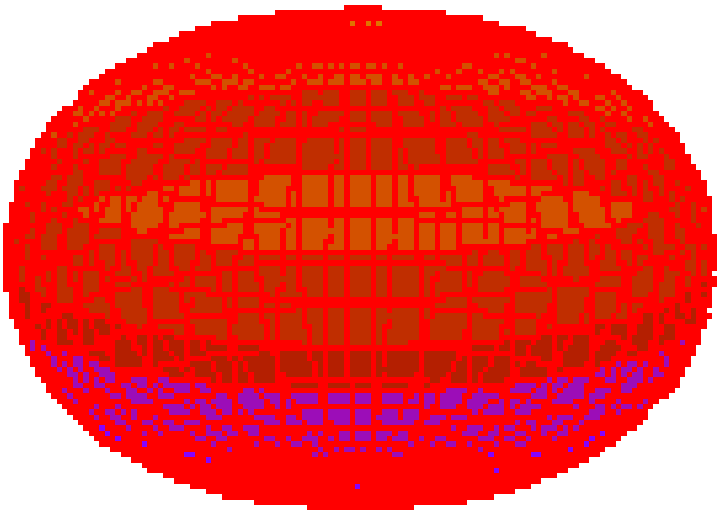

Figure 192: legendre2

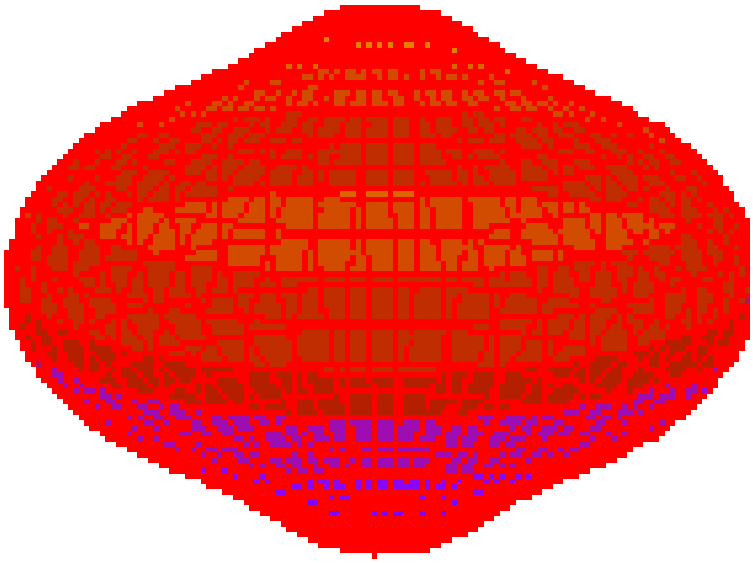

Figure 193: legendre3

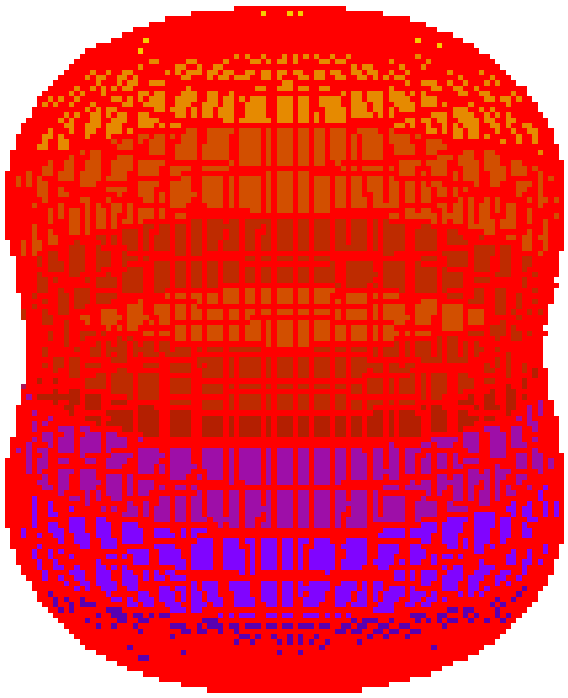

Figure 194: legendre4

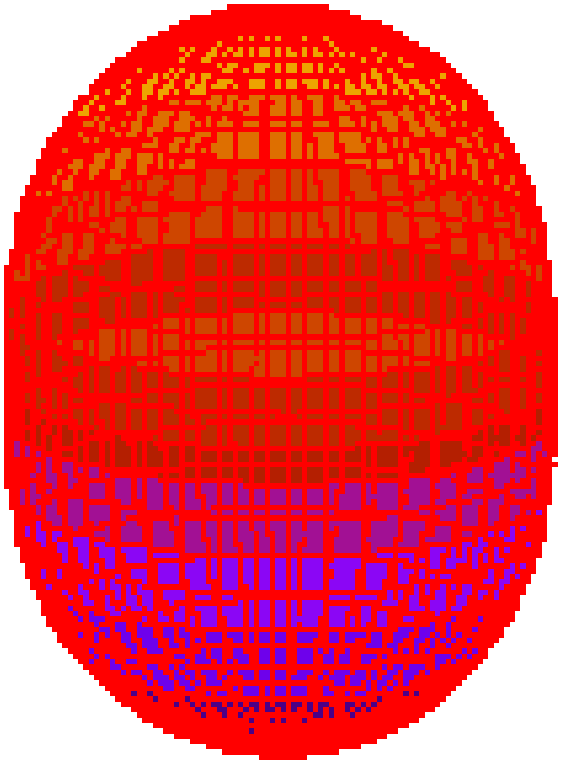

Figure 195: legendre5

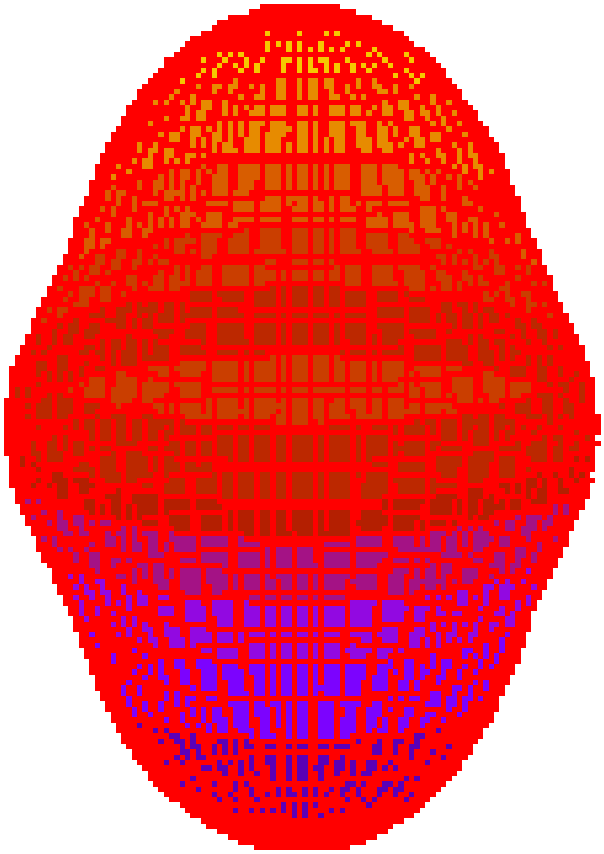

Figure 196: legendre6

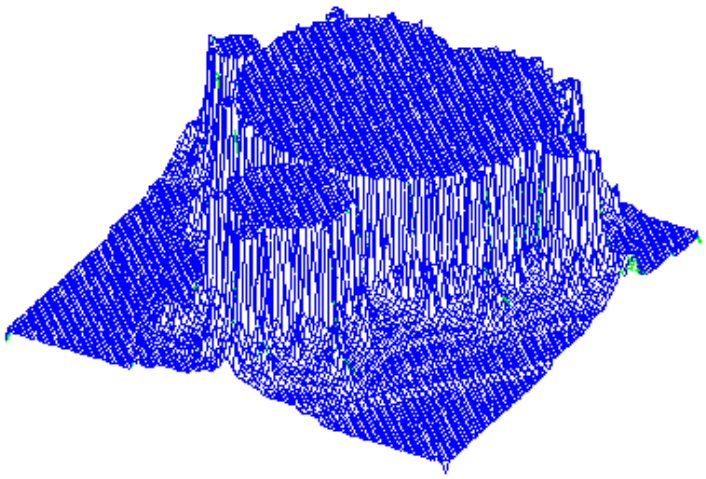

Figure 197: mandeltitle

This one-line function is infinite-loop in which the variable N decreases by 1. In order to terminate the loop, we use a ternary operator. The next example tells gnuplot to stop the loop when N=0.

```
gnuplot> fac(n) = (n==0) ? 1 : n * fac(n-1)
```

This means, when N is equal to zero, just after the '?' is evaluated, otherwise fac(n-1) is executed, and again, the same function is called but its argument is n-1. The function-call is terminated when its argument is zero.
To calculate N!, N should be integer. When the argument is a real number, we use a gnuplot function int() to make it integer. Perhaps you also need to include the restriction that N must be positive, although we don't consider this.

```
gnuplot> fac(x) = (int(x)==0) ? 1.0 : int(x) * fac(int(x)-1.0)
```

Now you can calculate the function fac(x), as follows:

```
gnuplot> fac(x) = (int(x)==0) ? 1.0 : int(x) * fac(int(x)-1.0)
gnuplot> print fact(1)
1.0
gnuplot> print fact(5)
120.0
gnuplot> print fact(5.5)
120.0
gnuplot> print fact(20)
2.43290200817664e+18
```

It is known that N! can be approximated by the Stirling formula.

$$N! \approx \sqrt{2\pi N} N^N e^{-N}$$

Let's compare the Stirling formula with our function fac(x).

```
gnuplot> stirling(x) = sqrt(2*pi*x) * x**x * exp(-x)
gnuplot> set xrange [1:15]
gnuplot> set yrange [1:1e+10]
gnuplot> set log y
gnuplot> set sample 15
gnuplot> plot stirling(x) notitle with lines,\
>             fact(x)     notitle with points
```

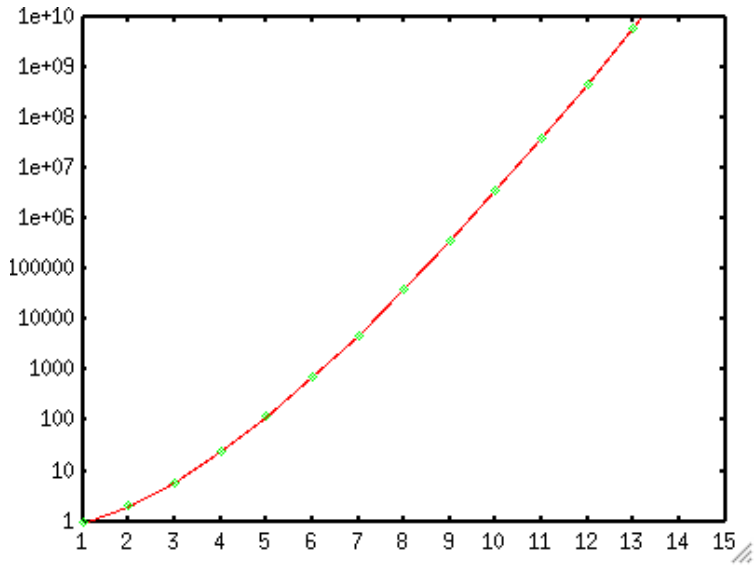

Figure 198: fact

17.2.2. Mandelbrot Set

Suppose we have a complex value A, and we calculate the following recurrence relation. If —z(n)—$\hat{2}$ does not diverge when n becomes very large numbers, the complex value A belongs to the Mandelbrot set.

```
z(0)   = 0.0
 z(n+1) = z(n)*z(n) + A
```

We regard the XY plane as the complex plane, and the point A is defined on it. Calculate the recurrence relation above, and if the —z—$\hat{2}$ value begins to diverge at a certain number of iteration n, we give this number n as the Z value at (X,Y) point. You can draw a 3-dim plot of the Mandelbrot set with those (X,Y,Z) data. Firstly we define this recurrence relation as a recursive function. This function returns the number of iteration n when —z—$\hat{2}$ becomes larger than a certain real number (2, in this case). The upper limit of iteration is 1000, which we need to avoid an infinite-loop.
In gnuplot a complex variable z whose real part is a and imaginary is b is written by z=a,b. Functions real(z) and imag(z) return the real and imaginary part, respectively. The absolute value of a complex z is abs(z). On the other hand there is no easy way to make a complex value from a real value, we define a new function, complex(x,y)=x*1,0+y*0,1

```
complex(x,y) = x*{1,0}+y*{0,1}
mandel(x,y,z,n) = (abs(z)>2.0 || n>=1000) ? \
                  n : mandel(x,y,z*z+complex(x,y),n+1)
```

The coordinates, x,y, are the location of the value A on the complex plane. The function mandel calculates z$\hat{2}$+A, and when its absolute value exceeds 2.0, the function returns the number of iteration, n. When you call this function you have to give initial values of z and n which are 0,0 and 0, in addition to the values of x and y.

```
gnuplot> set xrange [-1.5:0.5]
gnuplot> set yrange [-1:1]
gnuplot> set logscale z
gnuplot> set isosample 50
gnuplot> set hidden3d
gnuplot> set contour
gnuplot> splot mandel(x,y,{0,0},0) notitle
```

You may often see a nice computer graphics of the Mandelbrot set. To make such an image on your computer, each pixel of the image is alloted to a certain point on the complex plane, and calculate the recurrence relation for the pixel. When the calculation diverged, put a color-point at the pixel, and the color is changed by the number of iteration. Here we drew a 2-dim. Mandelbrot image with a simple X Window program. The plateau in the graph above corresponds to the black part in the right image.
As you can see in the left image, a structure of the magnified Mandelbrot set is very complicated. We showed those fractal images by using very fine contour lines, however, a line-art like gnuplot makes is not so good for playing with computer graphics. A new version of gnuplot, version 3.8, can draw nicer color-3D graph.

17.2.3. Julia Set (Self-Squared Fractal)

Same as the Mandelbrot set. We calculate the recurrence relation with a complex variable A, and look for a set of numbers with which the value of —z(n)—$\hat{2}$ does not diverge for a large number of n. The differences from the Mandelbrot set are that the

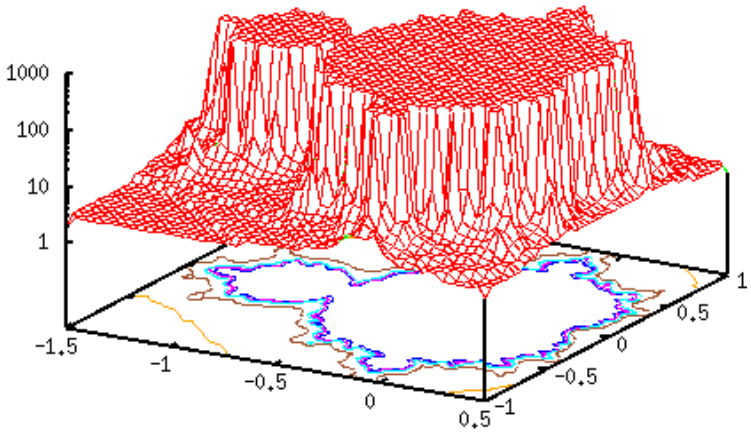

Figure 199: mandel1

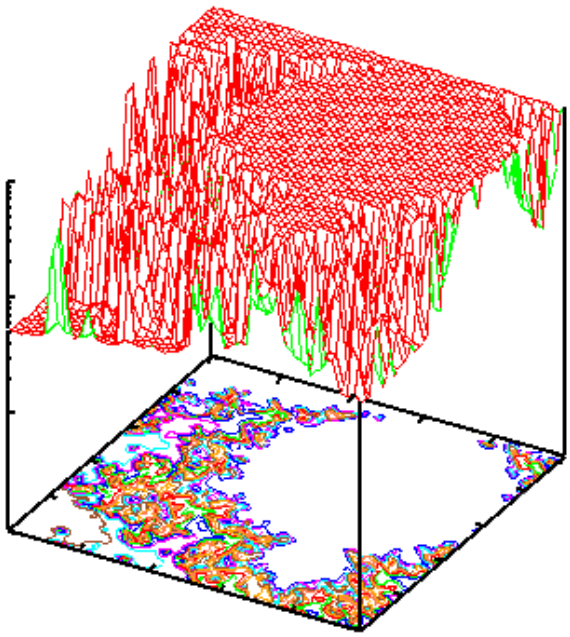

Figure 200: mandel2

initial value of the recurrence relation is the (x,y) coordinate on the complex plane, and the number of A is arbitrary.

```
z(0)   = X + iY
z(n+1) = z(n)*z(n) + A
```

The function calculate the function above is just the same as our mandel function defined in the previous section. The complex value A is given outside the function, and the (X,Y) coordinate is used as the initial value. Here we employ A=-0.37-0.612 i.

```
gnuplot> set xrange [-0.5:0.5]
gnuplot> set yrange [-0.5:0.5]
gnuplot> set logscale z
gnuplot> set isosample 50
gnuplot> set hidden3d
gnuplot> set contour
gnuplot> a= -0.37
gnuplot> b= -0.612
gnuplot> splot mandel(a,b,complex(x,y),0) notitle
```

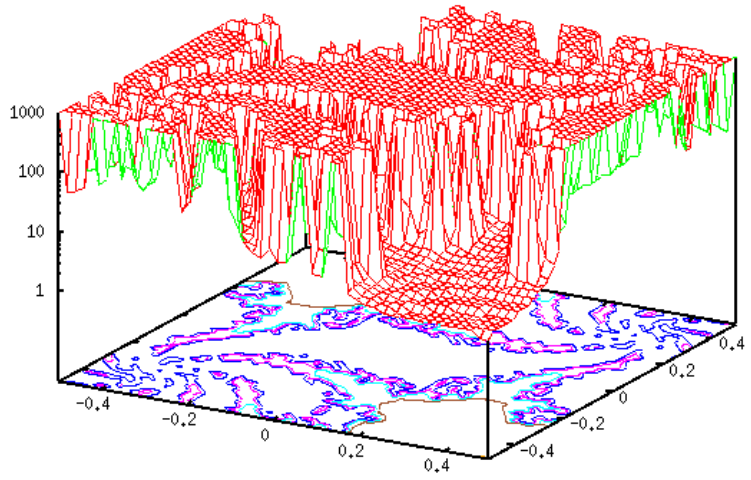

Figure 201: selfsq1

You can make a great variety of images by changing the complex constant A. In fact, a very small change in the constant result in a completely different picture. For example, the value of A was -0.37 -0.612 i in the figure above, but if you change the imaginary part into 0.6, you get:

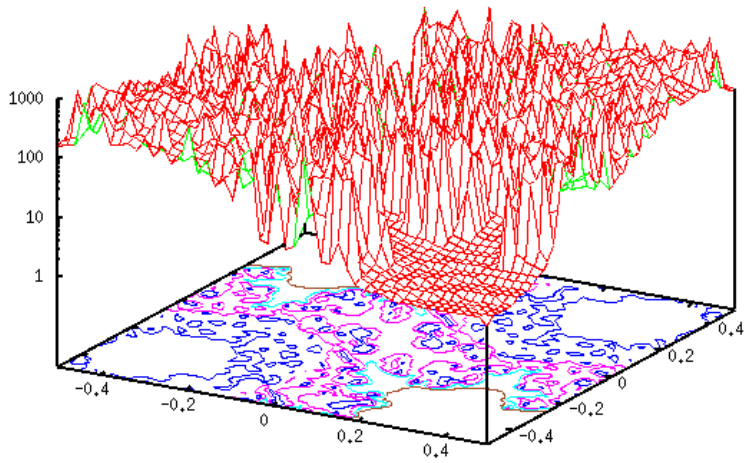

Figure 202: selfsq2