CLAMV Gnuplot quick tutorial

By Ibrahim Matewere.

Gnuplot is a free, command-driven, interactive, function and data plotting program.
The program can be downloaded from the Internet, a Google search would suffice.
But if you do not want to install it, you can use the installed version on the CLAMV Lab computers.
You can use the ssh command on Linux or putty on windows, to access the CLAMV lab computers and run the Gnuplot installed on them.
For instance on Linux terminal type: (replace the first part with your username)

ssh -Y imatewere@tlab043.clamv.jacobs-university.de

The above command fires up the ssh program and connects me to the CLAMV Lab computer 43.
Then on the terminal, type: gnuplot

tcsh% gnuplot

The above command gets you into the gnuplot shell:

gnuplot>

You can type commands in the gnuplot shell (shown above) after the prompt, (>).
In general, you can plot any of the math functions that you can work with in C programming Language.

The supported functions include:

Function	Returns
abs(x)	absolute value of x, |x|
acos(x)	arc-cosine of x
asin(x)	arc-sine of x
atan(x)	arc-tangent of x
cos(x)	cosine of x, x is in radians.
cosh(x)	hyperbolic cosine of x, x is in radians
erf(x)	error function of x
exp(x)	exponential function of x, base e
inverf(x)	inverse error function of x
invnorm(x)	inverse normal distribution of x
log(x)	log of x, base e
log10(x)	log of x, base 10
norm(x)	normal Gaussian distribution function
rand(x)	pseudo-random number generator
sgn(x)	1 if x > 0, -1 if x < 0, 0 if x=0
sin(x)	sine of x, x is in radians
sinh(x)	hyperbolic sine of x, x is in radians
sqrt(x)	the square root of x
tan(x)	tangent of x, x is in radians
tanh(x)	hyperbolic tangent of x, x is in radians

Bessel, gamma, ibeta, igamma, and lgamma functions are also supported. Many functions can take complex arguments.
Binary and unary operators are also supported.

The Basics:
The splot and the plot commands are the two most basic commands in gnuplot. The plot command is used to plot 2-dimensional plot while the splot is used to do 3-dimensional plots.
Here is the syntax of the plot command:

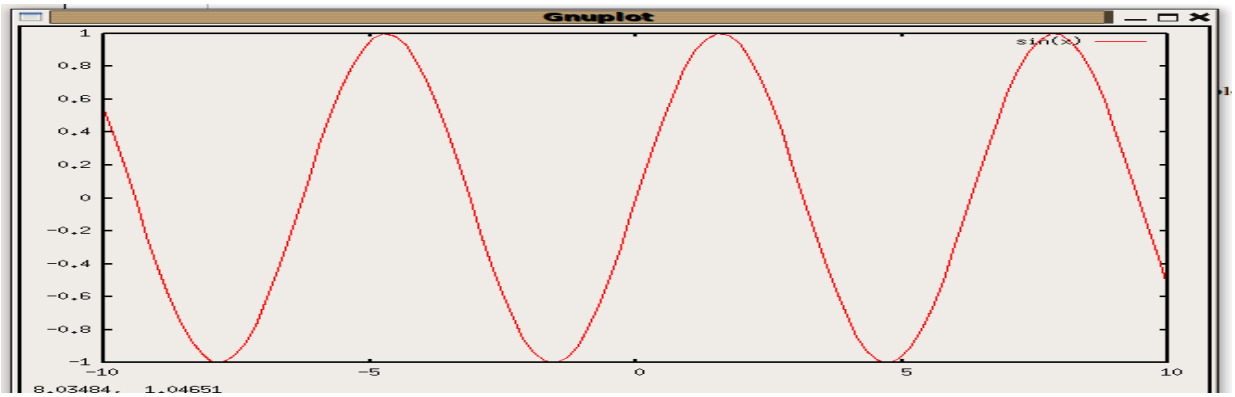
```
plot {[ranges]}
            {[function] | {"[datafile]" {datafile-modifiers}}}
            {axes [axes] } { [title-spec] } {with [style] }
            {, {definitions,} [function] ...}
```

To get more information on any command such as the plot command type: help plot

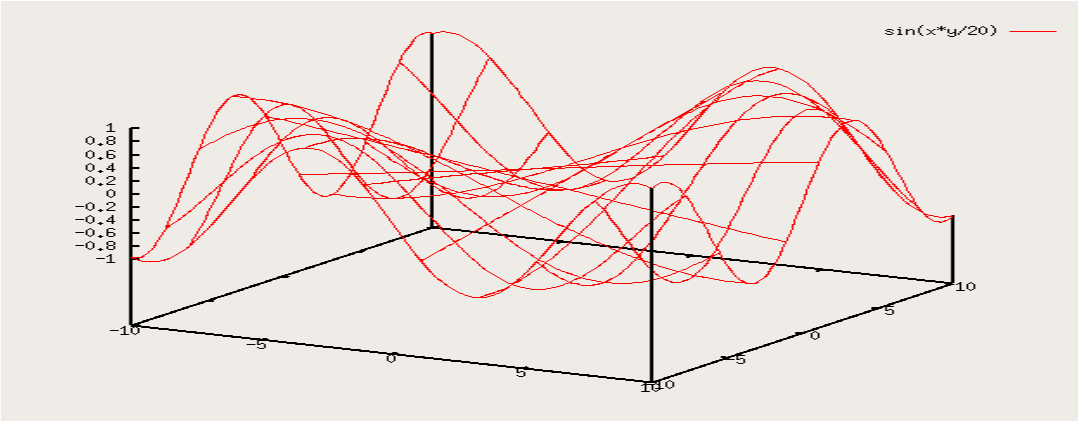
Plotting functions:
For instance to plot the sine function you can type:

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

and that produces this window:

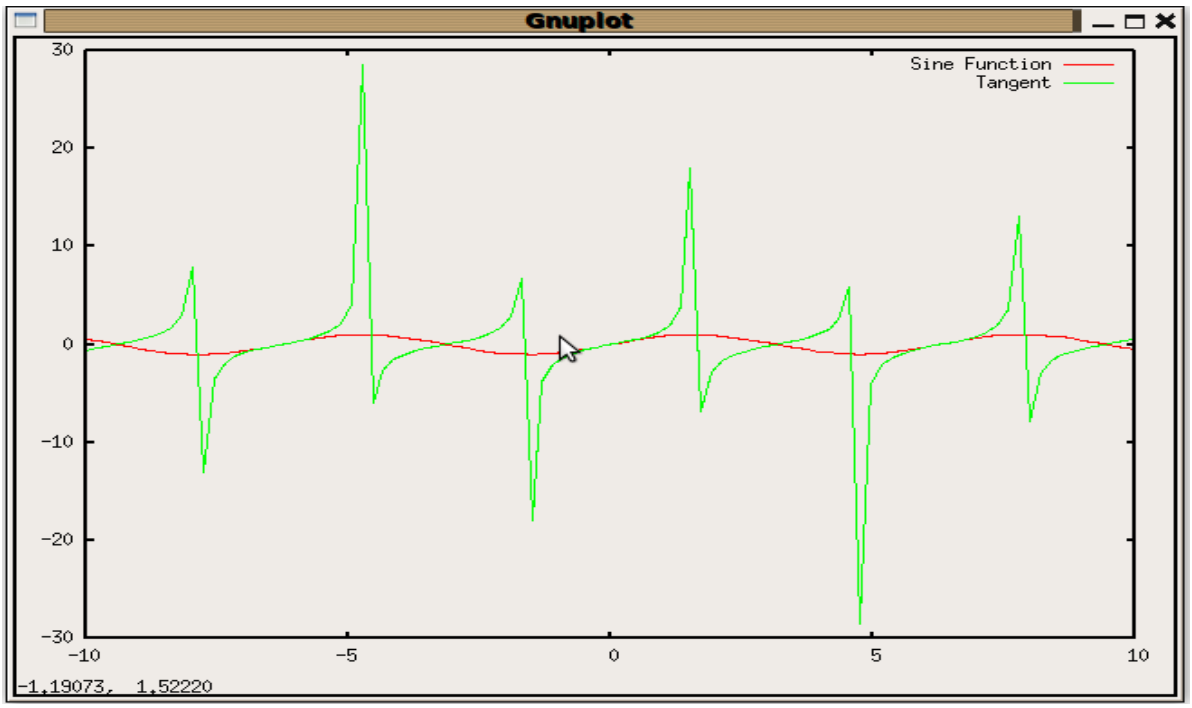


another example would be to plot a 3d plot:
gnuplot> splot sine(x*y/20)
the plot below is displayed:



now the command below plots two functions with legends:

```
gnuplot>  plot sin(x) title 'Sine Function', tan(x) title 'Tangent'
```

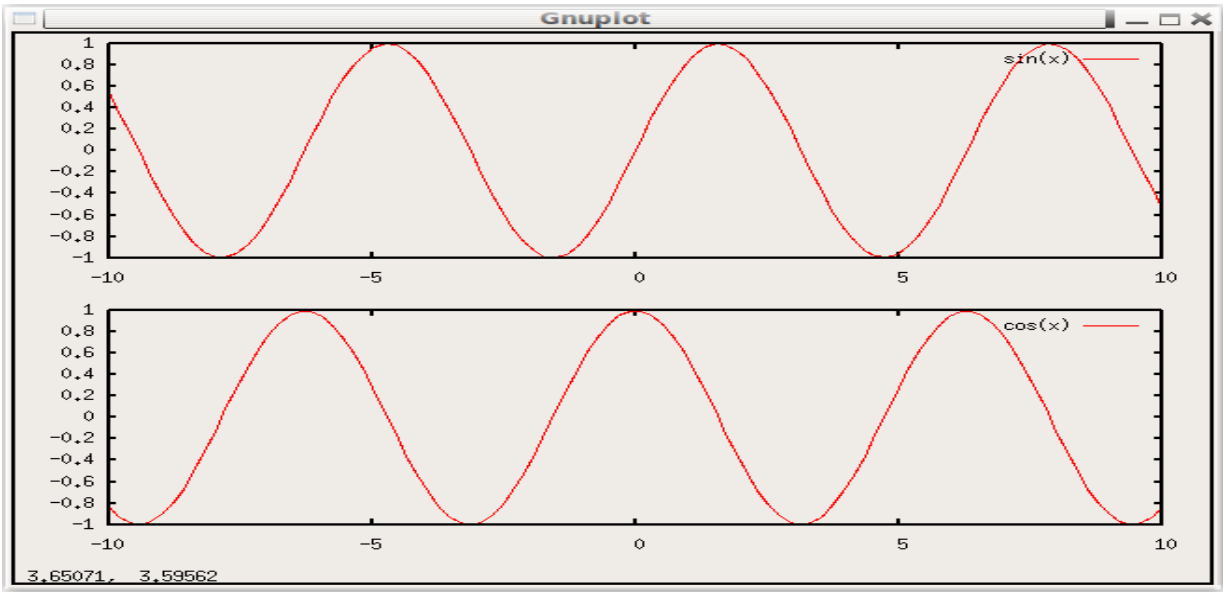


MULTI-PLOT

Gnuplot can plot more than one figure in a frame (like subplot in matlab) i.e., try:

```
set multiplot;                          # get into multiplot mode
set size 1,0.5;
set origin 0.0,0.5;   plot sin(x);
set origin 0.0,0.0;   plot cos(x)
unset multiplot                         # exit multiplot mode
```

This produces the plot below:



Plotting Data

However, most of the times, we work with data in files that is produced in experiments, not with continuous functions as those used in the examples before. The nice thing is that Gnuplot is very good when it comes to plotting data from experiments.

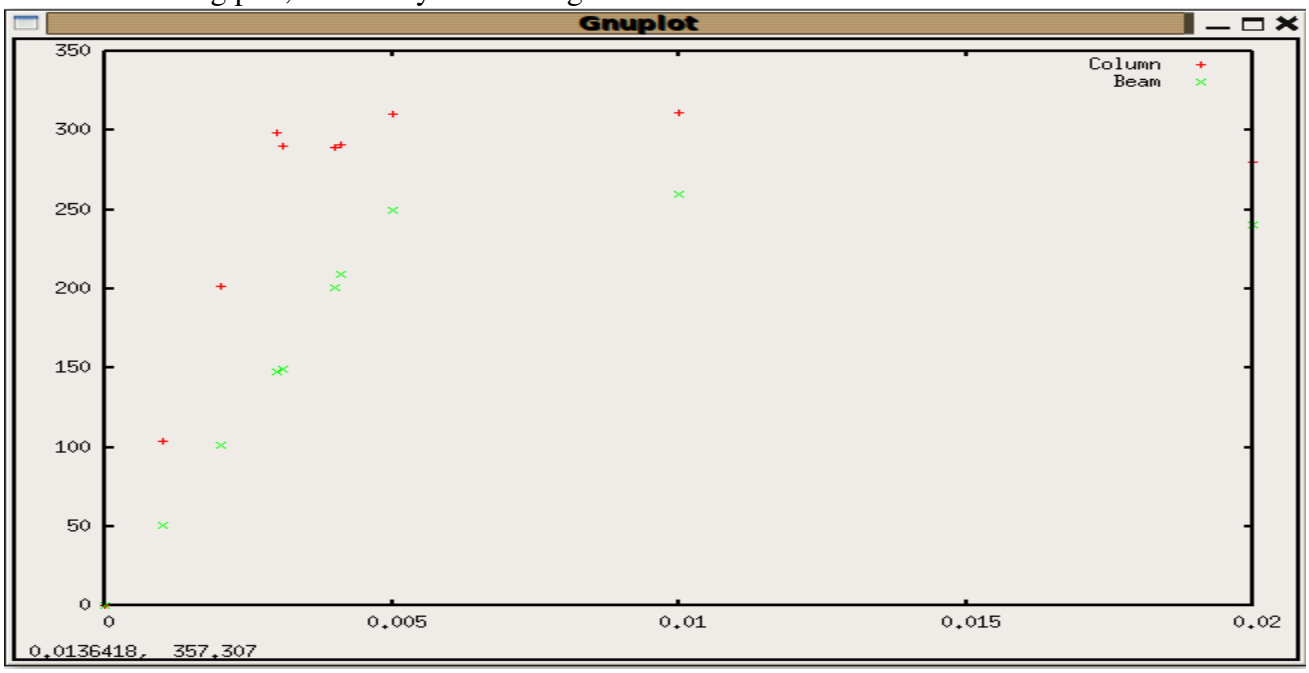
Discrete data contained in a file can be displayed by specifying the name of the data file (enclosed in quotes) on the **plot** or **splot** command line. Data files should have the data arranged in columns of numbers. Columns should be separated by white space (tabs or spaces) only, (no commas). Lines beginning with a # character are treated as comments and are ignored by Gnuplot. A blank line in the data file results in a break in the line connecting data points.

For example your data file, force.dat , might look like:

```
# This file is called   force.dat
# Force-Deflection data for a beam and a bar
 # Deflection    Col-Force       Beam-Force
     0.000             0             0
     0.001           104            51
     0.002           202           101
     0.003           298           148
     0.0031          290           149
     0.004           289           201
     0.0041          291           209
     0.005           310           250
     0.010           311           260
     0.020           280           240
```

gnuplot> plot "force.dat" using 1:2 title 'Column',"force.dat" using 1:3 title 'Beam'

This the following plot, it not very clear though:



To find out more about working with datafile, type either:

help plot datafile

or : help splot datafile

The latter is for making 3-dimensional plots from datafile.

plotting data files with other comment characters

If your data file has a comment character other than # you can tell Gnuplot about it. For example, if your data file has "%" comment characters (for Matlab compatability), typing

```
gnuplot>  set datafile commentschars "#%"
```

indicates that either a "#" or a "%" character starts a comment.

Getting a hard-copy of your plot for printing

To create a post-script of your plot, follow the steps below:
First copy the lines below and save them in a file called: save.plt

```
# File name: save.plt - saves a Gnuplot plot as a PostScript file
# to save the current plot as a postscript file issue the commands:
#  gnuplot>   load 'saveplot'
#  gnuplot>   !mv my-plot.ps another-file.ps
set size 1.0, 0.6
set terminal postscript portrait enhanced mono dashed lw 1 "Helvetica" 14
set output "my-plot.ps"
replot
set terminal x11
set size 1,1
```

after that type:

gnuplot> cd "/home/imatewere" #you replace it with you home directory
gnuplot> load 'save.plt'
gnuplot> !mv my-plot.ps another-name.ps #note the ! Is used to execute a system command

Issue this command to see if the postscript files are created:
gnuplot> !ls # this is the normal linux ls command

On Linux, you can view postscript files easily, but on windows you have to use GhostView.

Customization of Your plots : the set command

Customization of the axis ranges, axis labels, and plot title, as well as many other features, are specified using the set command. Specific examples of the set command follow. (The numerical values used in these examples are arbitrary.) To view your changes type: replot at the gnuplot> prompt at any time.

```
Create a title:                  gnuplot> set title "Force-Deflection Data"
Put a label on the x-axis:       gnuplot> set xlabel "Deflection (meters)"
Put a label on the y-axis:       gnuplot> set ylabel "Force (kN)"
Change the x-axis range:         gnuplot> set xrange [0.001:0.005]
Change the y-axis range:         gnuplot> set yrange [20:500]
Have Gnuplot determine ranges:   gnuplot> set autoscale
Move the key:                    gnuplot> set key 0.01,100
Delete the key:                  gnuplot> unset key
Put a label on the plot:    gnuplot> set label "yield point" at 0.003, 260
Remove all labels:               gnuplot> unset label
Plot using log-axes:             gnuplot> set logscale
Plot using log-axes on y-axis:   gnuplot> unset logscale; set logscale y
Change the tic-marks:           gnuplot> set xtics (0.002,0.004,0.006,0.008)
Return to the default tics:      gnuplot> unset xtics; set xtics auto
```

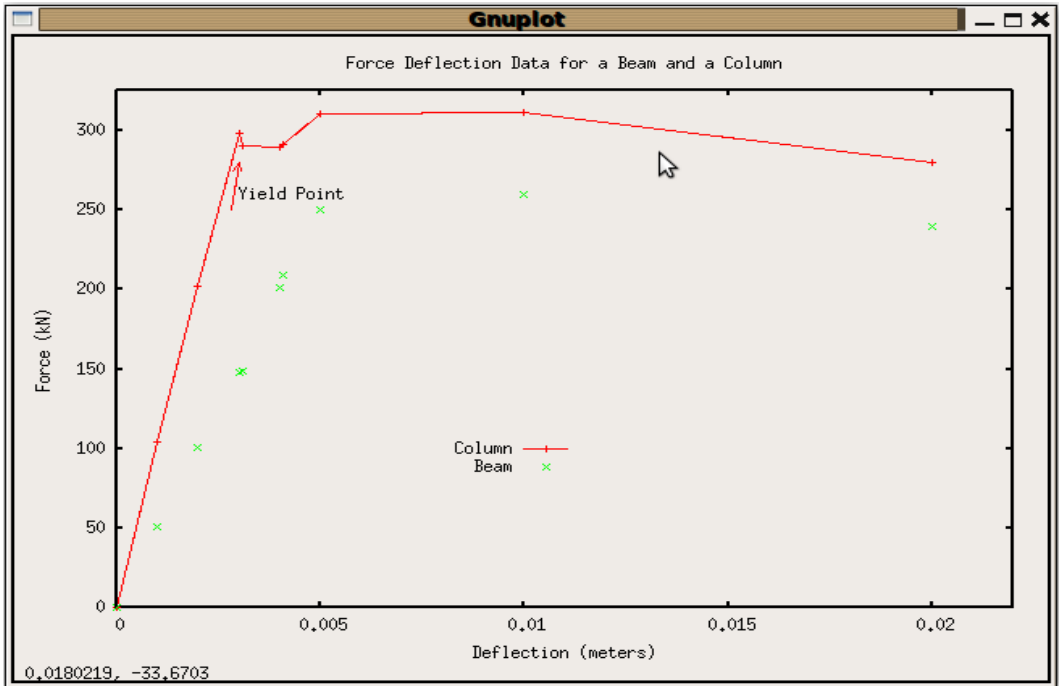
GNUPLOT SCRIPTS

Sometimes, several commands are typed to create a particular plot, and it is easy to make a typographical error when entering a command. To stream- line your plotting operations, several Gnuplot commands may be combined into a single script file. For example, the following file will create a customized display of the force-deflection data:

```
# Gnuplot script file for plotting data in file "force.dat"
# This file is called   force.p
set   autoscale                        # scale axes automatically
unset log                              # remove any log-scaling
unset label                            # remove any previous labels
set xtic auto                          # set xtics automatically
set ytic auto                          # set ytics automatically
set title "Force Deflection Data for a Beam and a Column"
set xlabel "Deflection (meters)"
set ylabel "Force (kN)"
set key 0.01,100
set label "Yield Point" at 0.003,260
set arrow from 0.0028,250 to 0.003,280
set xr [0.0:0.022]
set yr [0:325]
plot    "force.dat" using 1:2 title 'Column' with linespoints , \
      "force.dat" using 1:3 title 'Beam' with points
```

Then the total plot can be generated with the command:

```
gnuplot> load 'force.p'
```



Other resources:

A lot of useful information about gnuplot found on the following sites:

- http://www.duke.edu/~hpgavin/gnuplot.html
- http://t16web.lanl.gov/Kawano/gnuplot/intro/plotcalc-e.html
- http://gnuplot.sourceforge.net/demo/
- http://gnuplot.sourceforge.net/demo_4.0/
- http://www.gnuplot.info/