Visualizing data with Gnuplot

Written by Dave Pawlowski, September 26, 2011

Introduction

There are many different programs available that can help with data visualization. Most students are familar with Microsoft Excel, but if you ever publish a paper, write a book, give a talk, or present a poster, chances are you'll want to use software that creates more professional looking graphics without all the work required to make Excel do so. Gnuplot is an example of one such piece of software, and it has the added benefit of being free, as well as widely available on many different systems.

One of the nice things about Gnuplot is that it is command driven, which is something you should be getting used to, and it is interactive. This means that when you start a gnuplot session, you can directly interact with the variables that you create, as you'll see below.

Gnuplot is installed on chuck, however, it is not available on the EMU systems yipe or emunix. In order to use Gnuplot effectively, you should have an X-windows system or be forwarding an X-windows system to your local computer. If this connection is prohibitively slow, then I reccomend installing Gnuplot. You can get a pre-compiled executeable file for Windows and OSX (if you are using a Mac, see me).

In class, you should be using gnuplot via chuck. At home, you are free to use which ever system you want.

For more information see the Gnuplot manual.

Getting started

On a Linux machine, you start Gnuplot by typing

```
% gnuplot
```

at the prompt. This will launch the gnuplot program which you will interact with via the gnuplot command prompt. You can get help for any function in gnuplot by typing help 'command name'.

You can store and print variables in gnuplot very simply:

```
gnuplot> a = 40
```

```
gnuplot> print a
```

Math operations work just like you would expect:

```
gnuplot> b = 5 * 10/2.
gnuplot> print b
```

When you plot something in Gnuplot, by default gnuplot will create the plot using its own windows system (on OSX, this is called Aqua). This is no good if you are not sitting in front of the machine that gnuplot is running on, as X11 doesn't know how to forward an Aqua window. You need to tell Gnuplot to use the X11 X-windows system instead. Do this by typing:

```
gnuplot> set terminal x11
```

You should get a message :"Terminal type set to 'X11' "

Alternatively, you can create a file called '.gnuplot' in your home directory and add this line to it. That way, the terminal will be set properly everytime you start Gnuplot.

Functions

Gnuplot supports most common mathematical functions, which 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.
```

Plotting

The most common commands are the plot and splot commands. plot is used to plot 2-D functions and data, while splot is for 3-D surfaces and data.

To plot a function, simply type plot at the prompt. Try a few examples:

gnuplot> **plot sin(x)**

gnuplot> **splot sin(x*y/20.)**

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

These examples all plot continuous data, but discrete data can also be plotted. Use emacs to open up a file called data.dat.

```
#This is a temporary file called data.dat
#It contains force-deflection data for a beam and a bar
#Gnuplot automatically ignores lines that start with #
# 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
```

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.

You can plot this data using:

```
gnuplot> plot “data.dat” using 1:2 title ’Column’, “data.dat” using 1:3 title ’Beam’
```

where 1:2 and 1:3 refer to the columns in the data.dat file.

In addition, you can include data from multiple files. Create a data2.dat file that is similar to data.dat. Then, you can plot both files using:

```
gnuplot> plot “data.dat” using 1:2 title ’Column Material 1’, \
> plot “data2.dat” using 1:2 title ’Column Material 2’
```

Note that the \character is the line continuation character. Be sure to not type any spaces after \.

Customization

Pretty much everything can be customized on the plot, such as the ranges of the axes, the labels of the x and y axes, the style of data point, the style of the lines connecting the data points, and the title of the entire plot. There are two ways to set customization features, when you actually plot the data, using the plot command, and before you plot the data, using the set command. For the most part, customization that applies to the entire plot, such as the axes, plot title, axis labels, etc are set using set. Customization for each dataset on a plot is set when you call plot.

Plot command
Changing the data columns, titles, and line/point styles are specified when the plot command is issued. Plots can be created using one of several styles: lines, points, linespoints, impulses, dots, steps, fsteps, hiseps, errorbars, xerrorbars, yerrorbars,xyerrorbars, boxes, boxerrorbars,boxxyerrorbars, financebars,candlesticks, or vector. You can specify which style you wish to use as follows:

```
gnuplot> plot “data.dat” using 1:2 title ’Column’ with lines, \
> “data.dat” u 1:3 t ’Beam’ w linespoints
```

Note that the words: using, title, and with can be abbreviated as: u, t, and w. Try the different styles to see what each of them look like.

Set command
Customization of the axis ranges, axis labels, and plot title, as well as many other features are specified using the set command. Here are a few examples:

```
Create a title:                 > set title "Force-Deflection Data"
```

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

If you plot some data and decide that you want to change something using the set command, simply type

gnuplot> **replot**

and Gnuplot will redo the last plot with your updated changes. You can customize other features using set such as: arrow, border, clip, contour, grid, mapping, surface, time, view, and many more. Refer to the gnuplot help, or the Gnuplot manual for more information.

Comment Characters

If for some reason you have a data file that uses characters other than # to denote comments, for example %, you can let Gnuplot know using:

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

This will cause Gnuplot to treat both # and % as denoting a comment.

Plotting to a postscript

Having an X-window pop up and display your plot is useful when you want to quickly visulize your data. However, it isn't much use if you want to include that plot in a paper or presentation. For this reason, it is possible to create a PostScript (ps) file using Gnuplot. PostScript is the standard for high quality graphics, and if needed, can easily be converted to any other filetype using common unix utilities.

Below are typical commands used to create such a file:

gnuplot> **set size 1.0, 0.6**

gnuplot> **set terminal postscript portrait enhanced color dashed "Helvetica" 14**

gnuplot> **set output "my-plot.ps"**

Use help set terminal for help with the 2nd command. You can then enter the postscript sub-menu to see all the options and what they do.

Any plot command that follows the above set commands will be sent to the postscript file (my-plot.ps) instead of an X-window.

Scripts

As you can imagine, using Gnuplot in the manner that we have been doing can be problematic if you make a mistake and need to re-enter several lines of code again, or, you want to make the same plot for several different sets of data. For this reason, it is possible to create scripts for Gnuplot to read. All this means is that you can put one or more Gnuplot commands in a file and use Gnuplot to execute it.

Below is an example script caled force.p (again, the suffix is meaningless) that will make use of the data.dat file that you have already created.

force.p

```
# 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 at 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    "data.dat" using 1:2 title 'Column' with linespoints , \
           "data.dat" using 1:3 title 'Beam' with points
```

Then, you can run the script by typing:

gnuplot> **load 'force.p'**

or alternatively, from the shell prompt:

```
% gnuplot force.p
```

Multiple plots

It is possible to plot multiple graphs on one plot:

```
gnuplot> set multiplot
gnuplot> set size 1.0,0.5
gnuplot> set origin 0.0, 0.5
gnuplot> plot sin(x)
gnuplot> set origin 0.0,0.0
gnuplot> plot cos(x)
gnuplot> unset multiplot
```

The last line will get you out of multiplot mode.

Curve fitting

Lastly, it is possible to do some curve fitting using gnuplot. Below is a script that does so, and plots the output in a file called my-plot.ps. Go through it and see if you can make sense of each line. You have to tell gnuplot the functional form that you are trying to use to fit the data with. Then, gnuplot will figure out the coefficients for you. Once you have them, you can plot the data and the fit.

fit.p

```
#Script to fit a function to data
f1(x) = a1*tanh(x/b1)                          # define the function to be fit
a1 = 300; b1 = 0.005;                          # initial guess for a1 and b1
fit f1(x) 'data.dat' using 1:2 via a1, b1

f2(x) = a2 * tanh(x/b2)                        # define the function to be fit
a2 = 300; b2 = 0.005;                          # initial guess for a and b
fit f2(x) 'data.dat' using 1:3 via a2, b2

set size 1.0, 0.6
set terminal postscript portrait enhanced color
set output "my-plot.ps"
```

```
set key at 0.018,150 title "F(x) = A tanh (x/B)"        # title to key
set title "Force Deflection Data \n and curve fit"      # note newline
set pointsize 1.5                                       # larger point
set xlabel 'Deflection, {/Symbol D}_x (m)'              # Greek symbols
set ylabel 'Force, {/Times-Italic F}_A, (kN)'           # italics

plot    "data.dat" using 1:2 title 'Column data' with points, \
      "data.dat" using 1:3 title 'Beam data'   with points, \
      a1 * tanh( x / b1 ) title 'Column-fit: A=309, B=0.00227', \
 a2 * tanh( x / b2 ) title 'Beam-fit: A=260, B=0.00415'
```

Summary

This information should help you get started making professional looking plots. As always, there is more that you can do with Gnuplot. There is even a Matlab style programming environment that makes use of Gnuplot, called Octave, that is free. The best place to start for learning more difficult tricks is just search the web. Chances are that someone has already done what you would like to do.