Gnuplot tutorial/assignment #1, Physics 209

First of all, get logged into one of the computers in Henn 205.

The main point of this assignment is to become comfortable with gnuplot. There is nothing to turn in specifically for this assignment, but towards the end of this tutorial, you should create a plot that becomes a part of experiment 2.

1. Enter your `physics` login id and password
2. right-click on the desktop, and select "open terminal"
3. type `gnuplot` into the terminal window

Ok, so now you have a gnuplot command line. Next, you should get a copy of the gnuplot notes, available on the PHYS 209 web page at: http://www.physics.ubc.ca/~phys209/ by clicking on: Intro to Gnuplot.

You can start a web browser from the linux command line with: `firefox &`, or from within gnuplot with: `!firefox &` The `&` tells the shell (the command interpreter) that it should run the command in the background, and give you back your command prompt. The ! tells gnuplot that it should pass your command on to the shell, rather than trying to interpret it itself.

If you try to view pdf documents from within firefox and it asks you what to do with them, you can choose 'Browse' and enter: `/usr/bin/xpdf` as the program you want to use to view pdf documents.

- Follow through the gnuplot notes and generate the plots as they are done there.
- You can save yourself lots of re-typing by saving a gnuplot load file. Try making one of the plots suggested in the notes - make sure it has a title and labelled axes. Now type:
  `save "my_plot.plt"`

  Gnuplot will save a file that contains all the commands necessary to create that plot. If you quit out of gnuplot (with `exit`) and restart it, you can type: `load "my_plot.plt"` and your plot should reappear. If your plot uses data from a data file, the data **is not** stored in the plot file!

  The way I use gnuplot is to get the basics of my plot set up with the command line, then I'll save a plot file, and then open the plot file in a text editor. I make final adjustments in the text editor, and after each change I reload the plot file into gnuplot.

  Its often very useful to have more than one unix window available to run commands in. You can open a new window the same way as the first one, by right-clicking and choosing "open terminal." You can run gnuplot in one window, and run other commands in the other window without quitting out of gnuplot.

  There are several text editors available on `physics`. Some commonly used ones are: `emacs, vi, gedit` and `pico`. Emacs takes a little getting used to, but is very widely used, and very powerful. There is a tutorial built into emacs. If you start emacs by typing: `emacs &` at the unix command prompt, then ctrl-h, then t, you can follow it along (you should know that the

cursor keys and page-up, page-down keys work the way you would expect, so some of the first few pages of the tutorial are more complicated than they need to be).

`pico` and `gedit` are very easy to use and if you don't want to learn emacs right now, `gedit` is a good choice.

When you load a file into gnuplot, it just reads all the commands as if you had typed them, so you can change the order of the commands and add new ones as you see fit.

- Printing from gnuplot is a little complicated, as you can see from the gnuplot notes. Another approach to the "printit" file suggested there is to use the plot file we created, and simply add the contents of the printit file to the end. I would suggest removing the line: `!lpr gnuplot.eps`, because otherwise every time you load your plot file, you'll send a plot to the printer and deplete your print quota quickly! Instead, you can preview the postscript file you generated with: `!gv gnuplot.eps &` from the gnuplot command line (or `gv gnuplot.eps &` from the unix command line) to make sure your plot looks good, and then choose to print it from within gv. You can print the postscript file directly from the command line with `lpr gnuplot.eps`.

- You can do calculations on your data before plotting. For example, using the strange.dat file mentioned in the notes, you could do something like: `plot 'strange.dat' us 1:($2/$3)` which will take the entry from column 2 and divide it by the entry of column 3 before plotting. These expressions can become quite complicated, and can include functions like sin, cos, ln etc. You need to place the () around an entry that does any calculations, to let gnuplot know its got to do more work than just plot the numbers.

- Make a plot of your current vs voltage measurements from part one of Experiment 2. On top of your data, plot a line for the dependence of current on voltage that you expect from Ohm's law and the resistance of your resistor. Ensure your graph has labels on the x and y axes, and that it has a title.

- Finally, one way to see how well your data fits the theory is to look at the residuals - the difference between the theory and the measurements. To your earlier graph, add new points to show the residuals. Each residual point should have the same error bars that the raw data had. This final graph should be permanently attached into your lab book and turned in with Experiment 2.