Computational physics

Assignment 1: Some introductory Linux commands and plotting

In this assignment, you will introduce yourself to the `gnuplot` plotting package. We will run through the basics this week. You will frequently be expected to make plots in future weeks. If you are already comfortable with some other plotting package and have access to it, you may use that other package after this week's assignment. I am choosing gnuplot to teach you in this class because it is relatively powerful, relatively easy to use and available across a range of platforms (i.e. Mac, Windows, Linux). It should be a good option for you any time you need to make a plot and don't have any particular requirements about how.

If you are ever stuck on how to do something in gnuplot, you may check this tutorial, but I'll give you a few of the basics first.

To start, open up gnuplot by typing:

`gnuplot`

from the command line.

Now, let's try a few of the main features that don't require a data set to be input:

`plot sin(x)` should plot a sine wave.

`plot cos(x)` should plot a cosine function.

So now you have seen that you can make simple plots of functions in `gnuplot`.

The next stage is to plot data files.

Now, download the file `plotdata_intro` from my course web site.

Make sure you are in your home directory:

`cd`

The `cd` command changes directories to whatever follows it. The tilde represents the home directory. Tilde followed by a username on the system represents the home directory of a particular user. Typing `cd` without following it with a directory name will also send you to your home directory.

Now, make a directory for this week's exercises, using the make directory command, `mkdir`:

`mkdir assignment_plotting`

and copy the `plotdata_intro` file into that directory. You will first need to see whether the file has been downloaded into your home directory or your Downloads directory.

Search your own home directory for files, using the list command: `ls`

If the file is there, then copy it to the `assignment_plotting` directory using the copy command, `cp`. If you wish to save a few bytes of disk space, you may move the file, instead of copying it, by using the move command, `mv` instead of the copy command.

`cp plotdata_intro assignment_plotting`

Note, that if you try to or move a file, the first thing after `cp` is where you are copying from, and the second is where you are copying to. You should

avoid using spaces in filenames in Linux for this reason.[1] Linux is smart enough to recognize whether the second word is a directory which already exists[2]. If you copy to a directory name, what will happen is you will get a copy of the file with the same filename in the other directory. If the second word is *not* a directory name, then the file will be copied to a new name in the same directory. Everything that goes for copying also goes for moving. Moving is essentially copying, followed by deleting the original file.

The copy command is extremely important. As you start programming, you will run into cases where you have a program that works, but you want to make modifications to it. If you make a copy, then can can leave the original in good working order, while messing around with a new version.

If the file is not in your home directory, then look for it in the Downloads directory:

`ls Downloads`

If you find it there, then copy it over to the `assignment_plotting` directory. If you still don't find it, let me know.

Now the columns in the file are fake data I made up to look something like what would happen if we had time in the first column and height in the second column, and the uncertainty on the height in the fourth column, for an experiment in which we dropped a weight from an initial height of 490 meters in a vacuum, and made a measurement once every second.

So, first, let's plot the data. First, start `gnuplot` back up.

Then, type:

```
plot ‘‘plotdata_intro’’ with yerrorbars
```

`gnuplot` will figure out which column is which.

Then plot both the data and the function that we expect will fit the data:

```
plot ‘‘plotdata_intro’’ with yerrorbars, 490-4.9*x**2
```

Note that the double asterisk is how an exponent is typed into `gnuplot`.

Now, look at the plot, and see that the model goes through most of the data points. Eventually, we will discuss proper curve fitting, in which instead of guessing at everything, you guess at the functional form of the model (which in this case would be $y = y_0 + v_0 t - \frac{1}{2} g t^2$), and let the computer find the best fitting values (which in this case would be the values of y_0, v_0 and g).

Finally, make files that you can print out or include in a report or could print out. To do this, you need to understand how gnuplot outputs the results of plots. It does this with a "terminal".[3] The default terminal is an `xterm` window, which appears on your screen. To dump the output to a file, you need to change that. Type:

---

[1] It is possible to have filenames with spaces, but you then need to express the names using backslashes, which is more trouble than it's worth in most cases. Just use underscores instead of spaces if you want readable multi-word filenames.

[2] Linux is *not* smart enough to recognize that you forgot to create the directory before copying.

[3] This terminology goes back to an era where one computer would have many people using it at the same time, and the box at which they were typing was a terminal. The terminal is the "end of the line".

`set term postscript`
to make a "postscript" file as the output.

Then type:
`set output ‘‘outfile.ps’’`
`replot`

You should then have a file in your directory called `outfile.ps` with your plot. It will be in the postscript format. You can choose any file name you want for your file, but of course it's often very useful to choose file names that you will remember a year from now. If you make several plots with different parameters, but the same basic physics, then it becomes useful to put the parameter values in the file name. E.g. if you were doing the falling object with different values of y_0, you might call your files `falling_obj_y0_10.0` and `falling_obj_y0_20.0` if the values were 10.0 and 20.0, respectively. The computer doesn't care, but it can make your life easier later on if you start a project and put it away for a while and then try to come back to it.[4]

Now, you are ready to print the plot out. You can just type:
`lpr outfile.ps`
and the computer will send the file to your default printer. Or, if you want to make it into a pdf file so that you can email it to your mom, who probably doesn't have a postscript file reader, you can type:
`ps2pdf outfile.ps`
and you will have a new file called `outfile.pdf`
created.

There are other options within gnuplot for types of files you can create. If you want to make a JPG or GIF or PNG graphics file, just google it and see if it's possible and what the relevant command is.

[4] Also, be thankful you were born when you were. I'm one of the younger faculty in the department, and I still remember an era when file names could only be 8 characters long!

What you must do: the basic assignment

I will not check to see that you have copied the files over properly unless you ask me to do this during the lab session, but you should be making a new directory for every assignment to help keep everything in a format where you can find things later on, so you really should do this.

This week's assignment is fairly straightforward. Just follow the instructions on the previous three pages, and produce a printout of the plot you have been asked to make.

The challenge problem

Using either books or documentation available on google, figure out how to make a new version of your plot. The new version should (1) be in color and (2) have noticeably thicker lines than the original version (3) have the $x-$ axis labelled "Time" and the $y-$axis labelled "Height".

A few things to remember with computing: (1) the information on google is often superior to what is in books (2) for a lot of purposes, like figuring out how to make plots, either you'll know it works or know it doesn't work almost right away, so it may be easier just to go to a semi-reliable source and give it a try than to go to a book and try to find what you need to know. You should *not* apply the same rationale to physics, especially paper and pencil physics. When you are solving a homework problem, if you use the wrong formula, you will get an answer most of the time, but the answer will be wrong.