CS 403: Lab 7: Analysis with Matlab

Getting Started

1. Boot into Linux.
2. Use the code and example problems from Wednesday.

Matlab

1. Today, we'll be running our model problem and analyzing its output using Matlab. It is up to you what experiments you'd like to run, but this is probably a good time to work on PS 3. First, compile `rad1d`–the compiler options and pcgm version you use won't matter much, but for maximum speed, use pcgmL with the -O2 option.
2. Configure one of the .cmnd files however you want. Then run the model problem.
3. Start Matlab by typing `matlab &`. A large white window subdivided into several other windows should appear. This is Matlab. Working with Matlab is controlled from a command line, much like UNIX or DOS, but Matlab is much more powerful. The command line is located in the "Command Window." You may close the other windows if you like, they're useful, but I won't be talking about them. The first thing we need to do is to get our data (initial condition e.g. C.txt, C2.txt, etc and the answer Cfinal.txt) into Matlab. It is easy to read simple text files (text files organized like a one or two dimensional table of numbers) into Matlab: at the command prompt (the ≫) type

   ≫ `load C3.txt`

   (or whatever initial condition file you used). The command above will cause Matlab to read in the data in C3.txt and store it in an array of numbers called C3. Load Cfinal.txt. You should now have two arrays stored in your "workspace"–these are variables that we can manipulate from the command line. To see what variables are in your workspace, type

   ≫ `whos`

   You should see something like

```
  Name   Size   Bytes  Class

    C3  1000x1   8000  double array
Cfinal  1000x1   8000  double array
   ans   1x268    536  char array

Grand total is 2268 elements using 16536 bytes
```

This tells us that C3 and Cfinal each have 1000 rows and 1 column. To see the first five rows in C3, type

$$\gg \text{C3(1:5)}$$

4. Seeing the numbers is not very useful, what we want is a plot; specifically, we want a plot of C3 as a function of x. Before we can plot this, we need to create an array with our grid points. As we specified, our grid contains m evenly spaced points between 0 and L. The simplest way to create such a grid in Matlab, is to use the linspace function. First, we need to create variables containing m and L. We can get m either from the .cmnd file we ran or from the length of one of the arrays:

$$\gg \text{m=length(Cfinal);}$$

L is set only in the .cmnd file. Look up the value in the file you used (it is on the second line), I used $L = 10$:

$$\gg \text{L=10;}$$

If you want, type whos to verify that m and L are in your workspace. To see the value of a variable, just type its name and press return. To create our grid, enter

$$\gg \text{x=linspace(0,L,m);}$$

(be sure to put the ";" on the end, otherwise all the values of x will be displayed). We can check that this grid is the same as that created implicitly in rad1d by computing dx by taking the difference between the first two elements in x:

$$\gg \text{x(2)-x(1)}$$

Does this agree with the output of rad1d?

5. Now for the plot. To plot C3 using a blue line, enter

$$\gg \texttt{plot(x,C3,'b')}$$

The letter in quotes determines the color that will be used, other options include red='r', cyan='c', black='k', yellow='y'. Try plotting Cfinal using a red line. Did it work? What happened to C3? By default, calling plot causes Matlab to erase the figure window before creating the new plot. In order to plot C3 and Cfinal together, we need to turn this behavior off. You can do this by typing

$$\gg \texttt{hold on}$$

You should now be able to plot several lines together. You can erase a plot by closing the window, or by typing clf.

6. You should now make several runs with rad1d and compare the results. Remember, that every time you run `rad1d` it will overwrite the old values in Cfinal.txt. You can archive your results by renaming the files using the UNIX statement `mv Cfinal.txt NEWFILENAME` where NEWFILENAME is a name that has meaning to you. Alternatively, you can load Cfinal into Matlab after each run and giving it a new name. The easiest way to do this is to use the functional form of load:

$$\gg \texttt{NEWNAME=load('Cfinal.txt');}$$

which reads in the data from Cfinal but saves it in the variable NEWNAME (or whatever you want) rather than Cfinal. If you would like to save one of your plots (perhaps for PS 3), type

$$\gg \texttt{print -djpeg FNAME.jpg}$$

This will save the figure window in JPEG format in the file FNAME.jpg. Good luck!