20.309 Matlab Tutorial
By Marcio von Muhlen
Fall 2006

Adapted from previous tutorials by Maxim Shusteff, Nate Tedford and other BE students

Where to use Matlab: In the 20.309 lab (16-252), any Athena station or your own computer.

Where to get Matlab (optional): http://mit.edu/matlab/www/

How to use Matlab (please print and bring to lab):
http://openwetware.org/images/5/5f/MatlabIntro.pdf (Intro to Matlab file 2 in the website)
We will follow these tutorial notes, with additional material below.

Additional Material:

In experimental work you will often encounter the need to fit data to arbitrary (non-polynomial) functions. Matlab provides a variety of methods for doing this, one of which is the function `LSQCURVEFIT`. Take a few moments to glance through the help file to see what we're getting in to. The various paragraphs indicate that this function behaves differently depending on the number of input and output variables specified in calling it.

In order to generate a fit you need to tell Matlab the form of the equation you believe the data should follow. Bear in mind that Matlab has no way of knowing or telling you if you're incorrect, so it's up to you to account for this. The way you do this is by creating a Matlab function file. For example, for an equation of the form: $f(x) = \frac{C_1 * X}{1 + C_2 * X}$

The function file might look like:

```
function yhat = myfun(beta, x)
c1 = beta(1);
c2 = beta(2);
yhat = (c1*x ./ (1 + c2*x));
```

which, since it's a function, must be named `myfun.m`. The coefficients to be fitted are here contained in the input variable `beta`. Make sure to have this function in your working directory.

Let's try to fit the following data to this function using `LSQCURVEFIT`:
(you can find it in digital form at http://openwetware.org/images/8/8f/Fitdata.txt, or just copy and paste)

X	f(X)
1.0000	1.6523
1.2743	1.8215
1.6238	2.0050
2.0691	2.1106
2.6367	2.3456
3.3598	2.4778
4.2813	2.5952
5.4556	2.8175
6.9519	2.8945
8.8587	3.0125
11.2884	3.0001
14.3845	2.9454
18.3298	3.1056
23.3572	3.1850
29.7635	3.2454
37.9269	3.1170
48.3293	3.2313
61.5848	3.2092
78.4760	3.3027
100.0000	3.2677

Create a matrix `fitData` which contains this information. You could use:

```
>>fitData = dlmread('fitdata.txt', '\t', 1, 0);
```

Assuming fitdata.txt exists in your working directory. The 1 in the third input tells Matlab the data is starting on the 2nd row of the text file (Aside: Matlab indexes occasionally start with 0 as in JAVA, but usually start with 1 – you can tell different coders wrote different parts of this software).

Assign the X column to the variable X and the f(X) column to the variable Y, then plot the data with a semilog plot.

```
>> X = fitData(:,1);
>> Y = fitData(:,2);
>> semilogx(X,Y, 'bd')        %'bd' specifies blue diamond, unconnected data points
```

We need to provide Matlab with initial guesses for our coefficient values. At this point it's a good idea to plot our function to see if the data makes sense in regards to it. We can start by guessing 1 for both coefficients; in a real experiment your prior knowledge of what you are measuring would determine these guesses.

```
>>hold on
>>semilogx(X, (1 ./(1 + 1*X))
```

Now let's have Matlab figure out the best fit (least squares sense) of the data to this function, then plot the data and fit together.

```
>> Cfs = LSQCURVEFIT(@myfun, [1 1], X, Y)
>> hold off
>> semilogx(X,Y, 'bd', X, (Cfs(1)*X ./(1 + Cfs(2)*X)), 'k-') %what does 'k-' do?
```

Here we plotted two sets of data in the same command. See the help file for `semilogx` to understand how this works. Notice that although our guess wasn't very good, Matlab had no trouble finding good coefficients. This won't always be the case with more complicated data and functions.

Additional Resources:
MIT's MATLAB answer page: http://web.mit.edu/answers/matlab/

Another MATLAB tutorial, with additional material on looping and solving ODE's in MATLAB: http://openwetware.org/images/7/76/MatlabTutorial2.pdf