## Lab #1 -- Getting Started with Matlab

## Overview, Syntax, and Simple Function Evaluation and Plotting

### Overview

The name Matlab stands for **Mat**rix **Lab**oratory. This is important since we will see that the numerical capability in Matlab is built upon its roots as a tool for performing linear algebra analysis. In particular, every variable is considered as a 2-D array, and this interpretation and the associated capability that this gives makes for a very powerful computational and visualization tool. This flexibility, however, also makes Matlab a little difficult to comprehend for the novice.

Our goal today will be to introduce enough basic concepts so that you can write a short Matlab program to evaluate and plot simple one-dimensional functions. We will do a lot more over the course of the semester, but this is a good starting point, since the capability to evaluate and plot some function, f(x) vs. x, is essential in most fields of study.

By the end of this lab you should be quite comfortable with several basic operations in Matlab and be able to evaluate and plot 1-D functions of the form f(x) vs. x. If you are already familiar with Matlab, think of this lab as a quick refresher. However, if are only a novice or a very first-time user, then there is a lot of really good stuff demonstrated here – so make sure you understand what is happening at each step of the exercises to follow. **And have some fun…**

**Getting Started** -- Try the following tasks:

| | | |
|---|---|---|
| **1** | Startup Matlab | **double click the Matlab icon** or **click Start/Matlab** (note that, once the program fully opens, you can close all the 'extra' windows, if desired, leaving only the **Command window** open -- since this is where we will focus today...) |
| 2 | Show some syntax | in the **Command window**, type **x = 5** |
| | | now type **y = 6;** |
| | | type **whos** % shows workspace variables and their size |
| 3 | Create 1-D (vector) and 2-D (matrix) arrays | **xx = [-2 -1 0 1 2]** % enter individual data |
| | | **yy = -2:0.5:2** % use colon operator |
| | | **zz = linspace(-2,2,9)** % use **linspace** command |
| | | **A = [1 2 3; 4 5 6]** % a 2×3 array |
| | | **B = [1 0 1; 4 3 -1; 0 0 1; 2 3 -4]** % a 4×3 array |
| | | **C = B′** % ′ is the **transpose operator** (gives a 3×4 array) |
| | | **whos** % shows existing variables in the **workspace** |
| 4 | Clear workspace | **clear all** |
| | | **whos** % note that everything in the **workspace** is now gone |

| | | |
|---|---|---|
| 5 | Evaluate a function and create a simple plot (lots to discuss here) | **t = 0:1:10;**<br>**f = exp(-0.8*t).*cos(5*t);** % note the **dot arithmetic** |
| | | **plot(t,f, 'r-'), grid on, title('f(t) = e^{-0.8t}cos(5t)')** |
| | | **xlabel('time (seconds)'), ylabel('f(t)')** |
| 6 | Get general help | **help** or go to **?** on top menu bar |
| | | **help graph2d** or search and select from menu |
| | | **help plot**, **help title**, etc. or search + select from menu |
| 7 | Explore other possibilities -- don't be afraid to try things... | |

**Creating a Matlab Script File** -- Try the following tasks:

1. Under **Home**, select **New Script**
2. Get into the Matlab editor and write a few lines of Matlab code to evaluate and plot a function. You might try the example given above as a start. Type in the commands under item #5 and save the file (call it **test1**, which will be saved in the default directory as **test1.m**). Don't exit the editor yet -- leave it open so that you can correct any errors you may have made (and also to clean up the rather ugly plot that is generated by the above code). Also you might add a **clear all, close all** at the beginning of your script file -- why???
3. Return to the Matlab command window and simply type the name of the m-file, **test1**, to test it out (without the extension -- first name only). Or, within the editor itself, simply hit the **green Run button**.
4. Continue the above process of going between the Matlab workspace and the Editor to build and debug your Matlab program. This could be your first Matlab program -- now that wasn't so bad, was it???

**Using a Pre-existing Matlab Script File** -- Do the following:

1. Download the **pendulum_1.m** script m-file from the course website to the current Matlab folder. This file evaluates and plots several characteristics associated with the motion of a simple pendulum. The development of the physical and mathematical models for this relatively simple, but real system, is done in detail in the **pendulum_dynamics.pdf** file -- you should study this is some detail to see the complete development process. Also, you should consult the **MLabLesson1_getting_started.pdf** file for further discussions concerning of this illustrative example. These additional pdf files can be explored in more detail after this lab session…
2. For now, simply open the **pendulum_1.m** file in the Matlab editor and execute it (as above) to see what it does. At this point, the lab instructor will outline the basic structure of the code and he/she will point out several features within this script file, including the use of the **figure**, **plot**, **subplot**, **axis**, **legend**, etc. commands, how to include Greek symbols within the

title and x and y axis labels, and explain the various plot styles illustrated here. You should really try to understand all the logic and command syntax in this file -- it makes a great example that should prove to be quite useful in the future!!!

**Now it is YOUR turn to write some Matlab Scripts of your own** -- In the time remaining in this lab, you should get started with evaluating and plotting the following two functions using Matlab. Make sure the plots look good and that they are as informative as possible. Follow any special instructions that are given. Please complete these tasks outside of class if you run out of time during the lab period.

a. The temperature dependence of the mass diffusion coefficient D is given by an Arrhenius type equation,

$$D = D_o e^{-E_a/RT}$$

where $D_o$ is the pre-exponential constant, $E_a$ is the activation energy for diffusion, R = 8.314 J/mol-K is the gas constant, and T is the temperature in Kelvin. For diffusion of carbon into stainless steel, $D_o = 6.18$ cm$^2$/s, and $E_a = 187$ KJ/mol.

Your goal here is to make two plots of D vs. T for $200 \le T \le 800$ C. Put both curves in a 2×1 array on the same plot using Matlab's ***subplot*** command. In the first plot use a linear scale for both axes and in the second plot use a linear scale for T and a logarithmic scale for D. Which plot is more useful? Explain…

b. Evaluate and plot the following expression within Matlab for $0 \le x \le 100$ using Matlab's vector processing capability. Your goal should be a properly labeled plot that shows the key characteristics of y(x) over the given interval. Note that you may want to consider different options for axis scaling so that the important functional behavior is readily observable. From your plot, what are the minimum and maximum values of y(x) and the corresponding x values (use the zoom feature to get the results to two significant figures)? Does the plot appear reasonable considering the equation given? In your opinion, what is the best way to plot this function? Explain…

$$y(x) = \sqrt{\frac{100\left(1-0.01x^2\right)^2 + 0.02x^2}{\left(1-x^2\right)^2 + 0.1x^2}}$$