ECE 481: Introduction to MATLAB

Dr. Anthony Maciejewski

Overview: For the computer projects assigned in ECE 481, students are asked to program in MATLAB. MATLAB is an extremely versatile, high-level computing language with user-friendly graphical user interface and well-documented support. The purpose of this document is to give you a quick introduction into some features which will be useful for the specific projects assigned in ECE 481. As you work more with MATLAB, you will develop your own preferred approaches to different problems. Searching for help online or within MATLAB itself will likely lead to a dozen new approaches. This document is not intended to introduce all the features of MATLAB but merely provide some initial tools so you can begin working with the program.

Suggested Resources: MATLAB is a ubiquitous programming language and as such, there are hundreds of online resources available. If you are looking for a specific command in MATLAB, a quick Google search can likely return just what you needed. As the creators of MATLAB, MathWorks is the single best source of help on all MATLAB topics. There are numerous tuturials, targetted discussion forums, and programming examples available at: *www.mathworks.com*. If you get stuck on the specifics of a command, you can also ask MATLAB for help directly using the *help* command in the Command Window as will be demonstrated later in this document.

Getting Started in MATLAB: MATLAB is available on all computers within the engineering building. Student versions are also available through MathWorks for less than $100. MATLAB offers several Toolboxes, including Robust Control Toolbox and Signal Processing Toolbox, with topic specific functions; however, for ECE 481, no special toolboxes are required. Instead, much of the work will be accomplished by writing MATLAB scripts.

To get started, simply open MATLAB on your computer. You will be greeted by the MATLAB Command Window which should resemble the screenshot shown in Fig. 1. Note, there may be some slight differences if you are using a campus computer as these screenshots were taken from the R2013a student version but all the same features are available.

To get started, I recommend working in one of two domains within MATLAB: the Command Window or script files. The Command Window is useful for line by line commands and provides immediate answers. It is useful for quick calculations or outputting data for debugging. Script files are more like C++

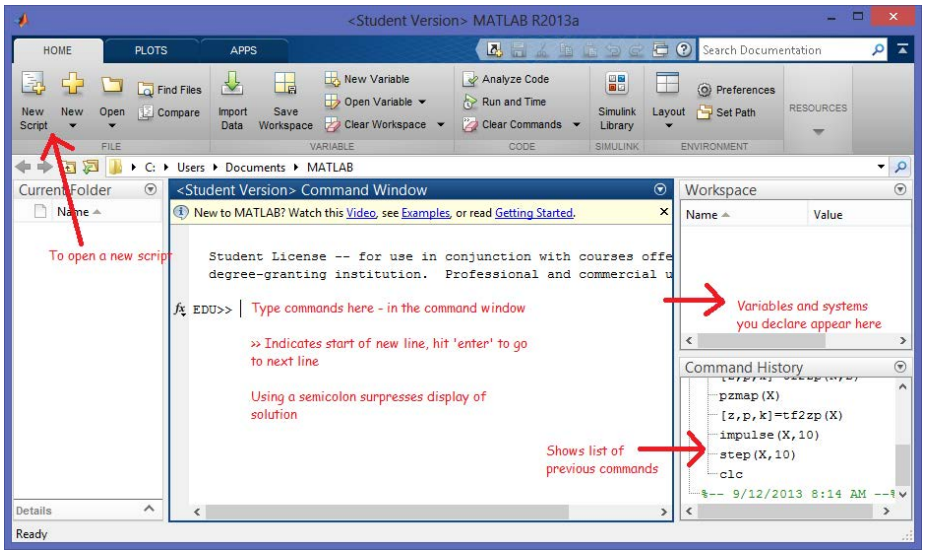

Figure 1: MATLAB Command Window

code and allow for longer, repeatable simulations, and complex operations. You can use scripts to call other scripts, format and plot data, etc. The two computer projects for ECE 481 will require you to write at least one MATLAB script, preferably several.

Writing Scipts: You can use scripts to write either a script file or a function. To illustrate how both scripts and functions can be used, consider writing a script to plot a cosine wave at a user-specified frequency. Although there are more concise ways to achieve the task, bare with the example which highlights several different features which may come in handy during ECE 481.

First, open a new script by clicking on the *New Script* icon at the top of the MATLAB opening screen (see Fig. 1). Within the script, type or copy and paste the following snippet of code:

```
% Demonstrate how to use a script and function call to plot cosine
% Note: this script is saved as plot_cosine.m

freq=1;
amplitude=2;
time=0:0.01:2*pi;

cosine=get_cosine(freq,amplitude,time);

plot(time,cosine)
    title('Plot of Cosine Function')
    xlabel('Time (s)'), ylabel('Magnitude')
    axis([0,2*pi,-amplitude-0.5,amplitude+0.5])
```

and save the script as *plot_cosine.m* in your working folder. The script you typed begins with two comments (denoted with a % symbol). Three variables are then declared - the first two being constants and the time variable being a vector containing values in [0,2π) in 0.01 intervals. Next, a new variable is declared which depends on a function yet to be created. Even before writing the function, we can tell *get_cosine* takes the three variables already declared and returns a new variable, *cosine*. The remaining lines of the script simply plot the generated cosine as a function of time, labels the resulting figure, and sets the axes so the plot is nicely scaled.

Before we can test the script, we need to make the *get_cosine* function so open another script file and type the snippet:

```
function cosine=get_cosine(freq,amplitude,time)
    w=2*pi*freq;
    cosine=amplitude*cos(w*time);
end
```

Save the function as *get_cosine.m* in your working folder. The file name must match the name of the function. Similar to C++, note the first line of the function call (sans *function*) matches the function call from your script file. Although often recommended, this is NOT a requirement. It is required to match the format (variable=function(three variables)) but not the names. The first line of the function could be replaced with:

```
function bob=get_cosine(tony,john,brad)
```

and by making subsequent substitions still work fine. Personally, changing names helps in certain occassions but often causes more confusion when debugging!

With all the pieces in place (a whopping two!), it is time to run the script. Going back to the *plot_cosine.m* file, click the green play button titled *Run* on the right side of the top toolbar. A simple cosine plot should appear. Alternatively, you could run the script from the Command Window by typing *plot_cosine* into the command prompt then pressing enter. To test just your function call from Command Window, you can type *get_cosine(1,5,0:0.01:2*pi)*.

After running your script, notice the four variables created appear in your Workspace (right side of Fig. 1). The size and value of each variable is also shown. It is always good to do a logic check and ensure the actual size of the variables you created match what you expected. You will be multiplying vectors and matrices quite often for the computer projects and sometimes a simple transpose didn't get typed but can be detected by checking the Workspace dimensions.

Useful Tips: If you are getting compilation errors or banging your head against the wall, here are some tips and suggestions which may help:

- To access the MATLAB command documentation, you can always type 'help' followed by the command in question in the Command Window.

For example, typing " help tf" into the command window will return MathWorks documentation on how to use transfer functions in MATLAB.

- Make sure the files you are calling are in the same working folder and the working folder is your current folder shown on the left side of Fig. 1.
- Break the big project down into small, testable segments. Use functions to accomplish repeated tasks and keep the code more organized. Comments are also useful for keeping track of what you want a segment of code to accomplish.

Conclusion: Coding is very much an art form. The goal of this short introduction was to give you some of the technical requirements to coding in MATLAB so you can get started successfully but there are many different ways to accomplish the same task, some more elegant than others but most are simply different. Learn a style that works for you, check online if you get stuck on a specific command or ask for help, but mostly be patient and have fun!