Introduction to MATLAB
Lab 1

Mark A. Yoder and Bruce A. Ferguson

Objectives

The purpose of this lab is to play with MATLAB, specifically, using its command-line interface. While playing you should learn many of the commands we will be using throughout the quarter. Don't be afraid to try things. You can't break it. Specifically, you should achieve the following:

- Learn to use the MATLAB command-line interface
- Learn basic syntax and command structure
- Learn to create signals as arrays associated with a time axis array

Pre-Lab

1. The first pre-lab exercise is to get MATLAB and the signal processing toolbox installed on your laptop. (You may use a version of MATLAB already installed from a previous course). MATLAB is available from the tibia server: start_menu>run>\\tibia\public\Apps\Matlab. The signal processing toolbox is available with the network version, or is available in the student edition from the bookstore. Be sure to have MATLAB installed and running before you come into lab. Each student will need a copy and their computer in lab. There will not be time during lab to do the installation. **Note:** *If you are using the network version of MATLAB, you must be connected to the network to use the software*.
2. Obtain a laboratory notebook for ECE 300. The use of a used notebook is allowed as long as sufficient space is available for this course's lab entries. Create a table of contents for ECE 300, and a prelab page for lab 1. Make an entry in your lab notebook indicating the date and time that you and your partner each successfully installed MATLAB. **Note:** *This lab, and all labs, requires recording in a lab notebook. Guidelines are available on the course lab webpage.*
3. Read the handout "Simulating Waveforms in MATLAB" before lab.

Notes on Using MATLAB

MATLAB can be used in two different ways – through the command line interface and using MATLAB scripts. It is important that you know the difference between the two and when to use each. You will practice each in your first two labs in ECE 300. The first lab focuses on the command line interface.

The command line interface is the easiest way to use MATLAB, and is the default mode on startup. When you first start MATLAB, you will find that a "command window" opens as part of the MATLAB window. Using this command window, you can enter commands at the prompt (>>) for MATLAB to evaluate. This is convenient when doing simple calculations and getting help. A major disadvantage of the command line interface is that the work you do cannot be saved for future use.

Procedure: There are three exercises for today's lab

1. Play Time – record notes and observations in your laboratory notebook. (Tip – It is always good to compare your observations with your expectations, or theory and measurement, especially when they disagree. There is no need to write things down for obvious steps.)

(a) Explore the MATLAB help capability. Try the following (type at the command prompt – note that the text following the % is a comment; such text may be omitted without altering the command function):

```
help              % this is the basic help command
help plot
helpwin plot      % opens a help window on the plot command
help colon
lookfor filter    % keyword search
```

(b) Use MATLAB as a calculator. Try the following:

```
pi*pi-10
sin(pi/4)
ans^2*3           % ans holds the last result
```

(c) Practice variable name assignment in MATLAB. Try the following:

```
x = sin(pi/5);
cos(pi/5)         % assigned to what?
y = sqrt(1-x*x)
ans
```

NOTE: The semicolon at the end of a statement will suppress the echo to the screen.

(d) Experiment with vectors in MATLAB. Think of the vector as a set of numbers in a one-dimensional array. Try the following:

```
kset = -3:11;
kset
length(kset)
cos(pi*kset/4) % compute cosine
```

Using your lab notebook and a pen, explain using mathematical expressions how the last example computes the different values of cosine. The `cos` command introduces the concept of implicit type assignment. MATLAB senses that the cosine argument contains a vector, so it implicitly creates a vector to store the results of the `cos` command. The length of the `cos` vector is automatically set equal to the length of the `kset` vector.

(e) Create a time vector in MATLAB. Try the following:

```
helpwin linspace
t = linspace(0,0.1,100);
t
x = cos(2*pi*20*t);    % compute cosine
x
```

(f) Make sure that you understand the colon notation. In particular, explain what the following MATLAB code will produce:

```
help colon
jkl = 2:4:17
jkl = 99:-1:88
ttt = 2:1/9:4
tpi = pi*[2:-1/9:0]
```

(g) Extracting and/or inserting numbers in a vector is very easy to do. Consider the following definition:

```
x = [ones(1,3),[5:2:13],zeros(1,4)]
x(6:9)
```

Explain the result echoed from `x(6:9)`. Now write a single statement that will replace `x(6:9)` in the existing vector `x` with zeros.

Instructor Verification (see last page)

(h) Loops can be used in MATLAB, but they are NOT the most efficient way to get things done. When efficiency is important, such as when dealing with very large vectors, it is better to avoid loops and use the vector notation instead. However, in this course, where we are beginners, the for loop is sufficient. Here is a loop that computes values of the sine function. (The index of `x()` must start at 1.)

```
x = []; % initialize the x vector to a null
for i=0:7
   x(i+1)=sin(i*pi/4)
end
x
```

Rewrite this computation without using the loop.

Instructor Verification (see last page)

(i) Complex numbers are natural in MATLAB. All of the basic operations are supported. Try the following:

```
z = 3+j*4
conj(z)
abs(z)
angle(z)
real(z)
imag(z)
exp(j*pi)
```

```
exp(j*[pi/4 - pi/4])
```

(j) Plotting is easy in MATLAB, for both real and complex numbers. The basic plot command will plot a vector `y` versus a vector `x`. Try the following:

```
x = [-3 -1 0 1 3];
y = x.*x-3*x;
plot(x,y)
z=x + x*j
plot(z)                 % complex values can be plotted
subplot(2,1,1), plot(x,y)
subplot(2,1,2), plot(z)
```

Use `helpwin arith` to learn how the operation `x.*x` works. Compare to matrix multiplication `x*x`. This "dot" arithmetic is an extremely important concept in MATLAB.

(k) Run the MATLAB demos; enter `demo` and explore some of the different demos of basic MATLAB commands and plots.

2. Manipulating Sinusoids with MATLAB. Create commands that will generate two five-kilohertz sinusoids with arbitrary amplitude and phase:

$$x_1(t) = A_1 \cos(2\pi 5000t + \phi_1)$$
$$x_2(t) = A_2 \cos(2\pi 5000t + \phi_2)$$

In MATLAB, this is implemented by first creating the time vector t, and then creating a vector holding the values of the sinusoids.

(a) Create a time vector t which will cover 3 periods of the waveforms x_1 and x_2. Make sure the plot starts at a negative time so that it will include one cycle before time t=0 *and* make sure that you have at least twenty samples per period of the wave. (Using 10-20 samples per period is a good guideline for how to specify time resolution in MATLAB simulations.)

(b) Select the value of the amplitudes and phases using a top secret recipe: use your age for A_1, and the largest digit of your SSN for A_2; for the phases, use the last two digits of your SSN for ϕ_1 (in degrees), and take $\phi_2 = -75^\circ$. Note: When doing computations, make sure to convert degrees to radians!

(c) Make a plot of both signals on the same graph over the specified range of *t*. Include appropriate title and axis labels. Print out your plot and paste it in your lab notebook.

(c) Verify that the phases of the two signals $x_1(t)$ and $x_2(t)$ are correct by examining the plot at $t = 0$, and also verify that each signal has the correct amplitude. Record your techniques and results in your lab notebook.

(d) Use `subplot(3,1,1)` and `subplot(3,1,2)` to make a three-panel subplot that puts both of these plots on the same page. Include a copy of your plot in your notebook.

(e) Create a third sinusoid as the sum $x_3(t) = x_1(t) + x_2(t)$. In MATLAB this amounts to summing the vectors that hold the samples of each sinusoid. Make a plot of $x_3(t)$ over the same range of time as used in the last plot. Include this as the third panel in the plot by using `subplot(3,1,3)`.

(f) Measure the magnitude and phase of $x_3(t)$ directly from the plot. Make sure that you **show on the plot how the magnitude and phase were measured**. Compare your measured magnitude and phase with the magnitude and phase predicted using phasor addition.

Instructor Verification (see last page)

Report

Record the results of all of your work in your lab notebook. Tape any printout (graphs, for example) into the notebook as specified in the Lab Manual for the course. Be sure that all members of your lab group sign the lab notebook, and hand in one notebook per group at the end of lab (both members must keep up-to-date lab notebooks).

Lab 1 Introduction to MATLAB
Instructor Verification Sheet

Paste this page in your laboratory notebook along with your other lab data and observations.

Name _____ Date of Lab: _____

Part 1(g) Vector indexing using colon operator:

Verified: _____ Date/Time: _____

Part 1(h) Show how you avoided the loop.

Verified: _____ Date/Time: _____

Part 2(f) Show plot and explain your measurements.

Verified: _____ Date/Time: _____