Introduction to MATLAB
Lab 1

Mark A. Yoder and Bruce A. Ferguson

Prelab: None

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

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 two exercises for today's lab

1. Play Time – getting used to MATLAB

(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
```

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. Try the following and think about how this is different from the linspace command.

```
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. Assume we don't know how the vector x was created, for example it may be data read from an instrument, and we want to replace the 6th through 9th elements 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 + j*x.*x
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.

2. Manipulating Sinusoids with MATLAB. Create commands that will generate three 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 page (use subplot to make two graphs) over the specified range of *t*. **Include appropriate title and axis labels** (use the help command to look up the built-in functions *title, xlabel, ylabel.*) *Print out your plot, you will need to turn it in with this lab.*

(d) *Assume that the two signals that you just plotted were downloaded into MATLAB directly from the oscilloscope, and you want to verify your measurements.* Verify that each signal has the correct amplitude. Determine the phase difference between the signals using $x_1(t)$ as the zero phase signal. **Show your work on the printed plots.** Compare the estimated phase difference with the known phases (on the printout containing your plot).

Instructor Verification (see last page)

(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 all three signals ($x_1(t)$, $x_2(t)$, and $x_3(t)$) on the same page (use subplot to make three graphs). over the same range of time. **Include appropriate title and axis labels.** *Print out your plot, you will need to turn it in with this lab.*

(f) *Again, assume you just downloaded these signals from the oscilloscope.* Measure the magnitude of $x_3(t)$ and its phase relative to $x_1(t)$ directly from the plot. Make sure that you **show on the printout of the plot how the magnitude and phase were measured**. Use phasor analysis to determine the correct magnitude and phase of $x_3(t)$, based on the magnitude and phase of $x_1(t)$ and $x_2(t)$. Compare your estimate of the phase difference with the known phase difference.

Instructor Verification (see last page)

Lab 1 Introduction to MATLAB
Instructor Verification Sheet

*Turn in this page as well as plot for parts **2(d)** and **2(f)***

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(d) Show plot and explain your measurements.

Verified: _____ Date/Time: _____

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

Verified: _____ Date/Time: _____