Introduction to MATLAB

Objectives

The purpose of this lab is to play with MATLAB. 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. The best source of information on MATLAB is the `help` command.

Pre-Lab

This week's lab has no pre-lab, however the laboratory instructor must verify the appropriate steps in the lab procedure by initialing on the **Instructor Verification** sheet (attached). When you have completed a step that requires verification, simply demonstrate the step to the instructor.

It is only necessary to turn in the last problem (with explanations) as this week's lab memo. Staple the Instructor Verification and Lab Summary sheet to your memo to show that the appropriate steps were demonstrated to the instructor.

Play Time

(a) Explore the MATLAB help capability. Try the following:

```
help
help plot
helpwin plot
help colon
lookfor filter % keyword search
```

The text following the **%** is a comment; it may be omitted.

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

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

(c) 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 (separate page)

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

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

(e) Do 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.

(f) Experiment with vectors in MATLAB. Think of the vector as a set of numbers. Try the following:

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

Explain how the last example computes the different values of cosine.

(g) Loops can be done in MATLAB, but they are NOT the most efficient way to get things done. It's better to **avoid loops** and use the vector notation instead, but here is a loop that computes values of the sine function. (The index of **`x()`** must start at 1.) Rewrite this computation without using the loop (as in the previous part).

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

Instructor Verification (separate page)

(h) Complex numbers are natural in MATLAB. 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 ])
```

(i) 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 + y * j
plot(z) % complex values can be
plotted
subplot(2,1,1), plot(x,y)
subplot(2,1,2), plot(z)
subplot(1,1,1)
```

Use ***helpwin arith*** *to learn how the operation* **x.*x** *works; compare to matrix multiply.*

(j) Use the MATLAB editor ("File -> New -> M-File") to create a file called **foo.m** containing the following lines:

```
tt = -2 : 0.05 : 3;
xx = sin( 2*pi*0.789*tt );
% plot a sinusoid
plot( tt, xx ), grid
title('TEST PLOT of SINUSOID')
xlabel('TIME  (sec)')
```

Run your function from MATLAB by typing its name (**foo**) at the MATLAB prompt (note that foo must reside in MATLAB's search path; see **helpwin path** for details). You can verify the contents of foo by typing **type foo**.

(k) Edit **foo.m** to use the **hold on** function followed by another plot command to add a plot of **0.5*cos(2*pi*0.789*tt)** to the plot created above.

Instructor Verification (separate page)

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

Manipulating Sinusoids with MATLAB

Hand in a short memo of this problem.
Generate two 5 kilohertz sinusoids with arbitrary amplitude and phase.

$$x_1(t) = A_1 \cos(2p\ 5000\ t + ø_1)$$
$$x_2(t) = A_2 \cos(2p\ 5000\ t + ø_2)$$

1. Select the value of the amplitudes and phases at random. 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 $ø_1$ (in degrees), and take $ø_2 = -75°$. NOTE: when doing computations, make sure to convert degrees to radians!

2. Make a plot of both signals over a range of t that will exhibit approximately 3 cycles. Include appropriate title and axis labels. Make sure the plot starts at a negative time so that it will include one cycle before time **t=0 AND make sure that your have at least 20 samples per period of the wave**.

3. Verify that the phase of the two signals $x_1(t)$ and $x_2(t)$ is correct at $t=0$, and also verify that each one has the correct maximum amplitude.

4. 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. See `help subplot`.

5. 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)`.

6. Measure the magnitude and phase of $x_3(t)$ directly from the plot. Turn in this **plot with sufficient annotation to show how the magnitude and phase were measured**.

Instructor Verification

Staple this page to the end of your Lab Memo.

Name: _____ Date: _____

Part c `x(6:9)`	
Part g `no loops`	
Part k `foo`	

Lab Summary

In general, your write-up (memo) should indicate that you have acquired a better understanding of the topics treated by the laboratory assignment. Here is a simple worksheet for you to fill out in order to assess your understanding of this week's objective: a working knowledge of the basics of MATLAB. If you do not know the answers to these questions go back to the Lab and try to figure them out in MATLAB (remember the commands **`help`** (**`helpwin`**) and **`lookfor`**!).

1) You saw how it easy it is for MATLAB to generate and manipulate vectors (i.e., 1-dimensional arrays of numbers). For example, consider the following:

```
y = 0:10;
y= zeros(1,25);
y= 1:0.25:5;
```

(a) How would you modify one of the above lines of MATLAB code to create a vector that steps from 0 to 20 in steps of 1/5?

(b) How would you modify one of the lines of code to create a vector of two hundred 100s?

2. You also learned that MATLAB has no problem handling complex numbers (see part (i), for example). Consider the following line of code: `y=3+5j`

   (a) How do you get MATLAB to return the magnitude of the complex number?

   (b) How do you get MATLAB to return the *phase* of the complex number **`y`**? What are the units of the answer?

3. In part (l), you learned that multiple lines of MATLAB code can be stored in a file with a **`.m`** extension. MATLAB then executes the code in the order that it appears in the file. Consider the following file, named `example.m`

```
f = 200;
tt = 0:1/(20*f):4/f;
z = exp(j*2*pi*f*tt)
subplot(2,1,1)
plot(real(z))
title('Real part of exp(j*2*pi*200*tt)')
subplot(2,1,2)
plot(imag(z))
title('Imaginary part of exp(j*2*pi*200*tt)')
```

   (a) How do you execute the file from the MATLAB prompt?

   (b) Suppose the file were named `example.dog`. Would it run? How could you change it to make it work in MATLAB ?

   (c) Assuming the m-file runs, what do you expect the plots to look like? If you're not sure type in the code and run it.