GEORGIA INSTITUTE OF TECHNOLOGY

SCHOOL of ELECTRICAL and COMPUTER ENGINEERING

ECE 2026 Fall 2023

Lab #00: Introduction to MATLAB

Date: Week of 08/21/2023

Important Note: This is the first lab of the semester and is mainly to guide you through the MATLAB tool, in preparation for the subsequent assignments. If you have previous exposure to MATLAB, follow this lab to refresh your MATLAB skills; if you are new to MATLAB, attend online tutorial sessions to acquire necessary MATLAB skills and use this lab as an exercise. This initial lab is different from the rest of the labs as it does not require you to submit a completion report. It does include the verification step (see below), which will provide examples how you'll for future labs include results in your report for submission.

In subsequent assignments, each lab will guide you through three stages: Pre-Lab, Exercise and Verification of Completion. The Pre-Lab stage requires you to read the assignment and familiarize yourself with what the lab is about and what additional tools, equipment or apps you may need in the main exercises. In the Exercise stage, you need to follow the instructions to complete a number of small tasks. Specific steps are to be taken to record the result when you complete a task. The results are then collated electronically into a Completion Report in the Verification of Completion stage for submission and grading. The completion report page at the end of each lab assignment enlists what needs to be answered, recorded and/or attached for submission. These include screen shots or files that show the output of the assigned exercise. All exercises will be done on your own computer.

Introduction

The objective of this lab is to help you regain your familiarity with MATLAB, a programming tool most suitable for signal processing studies. All labs of ECE2026 will involve use of MATLAB in some form.

1. Pre-Lab

Please read through the information below prior to attending your lab.

1.1 Overview

MATLAB will be used extensively in all the labs. The primary goal of this lab is to familiarize a student with using MATLAB. Please read Appendix B: *Programming in* MATLAB for an overview. Here are three specific goals for this lab:

1. Learn basic MATLAB commands and syntax, including the **`help`** system.
2. Write and edit your own script files in MATLAB, and run them as commands.
3. Learn a little about advanced programming techniques for MATLAB, i.e., vectorization.

1.2 Movies: MATLAB Tutorials

On the **`canvas`** course page, there are a number of movies on basic topics in MATLAB, e.g., colon operator, indexing, functions, etc. You will find these movies useful.

1.3 Getting Started

After logging in, you can start MATLAB by double-clicking on a MATLAB icon, typing **matlab** in a terminal window, or by selecting MATLAB from a menu such as the **START** menu under Windows. The following steps will introduce you to MATLAB.

a. View the MATLAB introduction by typing **intro** at the MATLAB prompt. This short introduction will demonstrate some of the basics of using MATLAB.

b. Run the MATLAB help desk by typing **helpdesk**. The help desk provides a hypertext interface to the MATLAB documentation. Two links of interest are **Getting Help** and **Getting Started with MATLAB**.

c. Explore the MATLAB help capability available at the command line. Try the following:

```
help
help edit
help plot
help colon          %<--- a VERY IMPORTANT notation
help ops
help zeros
help ones
lookfor filter      %<--- keyword search
```

NOTE: it is possible to force MATLAB to display only one screen-full of information at once by issuing the command **more on**).

d. **control-c** will stop the execution of any MATLAB command. For example, using **ctl-C** while **lookfor** is running will force it to stop and print out all the results it has found so far.

e. Run the MATLAB demos: type **demo** and explore a variety of basic MATLAB commands and plots.

f. Use MATLAB as a calculator. Try the following:

```
pi*sqrt(2)
0.5*pi*pi - 1.5
sin(2*pi/3+0.1)
ans ^ 0.8118        %<--- "ans" holds the last result
```

g. Do variable name assignment in MATLAB. Try the following:

```
x = sin( 2*pi/5 );
cos( pi/4 );  %<--- assigned to what?
y = sqrt( 2.2 - x*x )
ans                 %<--- what value is contained in ans?
```

h. Complex numbers are natural in MATLAB. The basic operations are supported. Try the following:

```
z = -4 + 3i, w = 3 - 4j
real(z), imag(z)
abs([z,w])          %<-- Vector (or Matrix) constructor
conj(z+w)
angle(z)
exp( -j*pi )
exp(j*[ pi/2, 0, -pi/2 ])
```

2. Exercise

2.1 MATLAB Array Indexing

a. Make sure that you understand the **colon** notation. In particular, explain in words what the following MATLAB code will produce

```
jkl = -4 : 1
jkl = -2 : 4 : 31
jkl = 87 : -2 : 63
tt = 2 : (1/80) : 3.1
tpi = pi * [ 0:1/4:2 ]
rr = [-1:(1/3):1].*[1:2:14]  %% what is .*?
```

Use **`help arith`** to learn how the operation "**`.*`**" works for vectors; compare array multiplication (**`dot-star`**) to matrix multiplication.

When unsure about a command, use **`help`**.

b. Extracting or inserting numbers in a vector is very easy to do. Consider the following definition of **`xx`**:

```
xx = [ zeros(1,5), linspace(0,1,6), ones(1,5) ]
xx(5:8)
size(xx)
length(xx)
xx(3:2:length(xx))
```

Explain the results echoed from each of the last four lines of the above code.

c. Observe the result of the following two assignments:

```
yy = xx; yy(4:2:8) = exp(0.5)*(1:2:5)
```

Now write a statement that will take the vector **`xx`** defined in part (b) and replace the even indexed elements (i.e., **`xx(2)`**, **`xx(4)`**, etc) with the constant π^e. *Use a vector replacement, not a loop.* Record this statement on the Verification Sheet and attach a screen shot.

Verification Sheet Fill-in

2.2 MATLAB Script Files

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

```
xk = cos( pi*(-1:2:25)/3 + pi); %<---comment: compute cosines
```

Explain how the different values of cosine are stored in the vector **`xk`**. What is **`xk(1)`**? Is **`xk(0)`** defined?

NOTES: the semicolon at the end of a statement will suppress the echo to the screen. The text following the **`%`** is a comment; it may be omitted.

b. (A taste of vectorization) Loops can be written in MATLAB, but they are NOT the most efficient way to get things done. It's better to **avoid loops** and use the colon notation instead. The following code has a loop that computes values of the cosine function. (The index of **`yy()`** must start at 1.) Rewrite this computation without using the loop (follow the style in the previous part).

```
yy = [ ];           %<--- initialize the yy vector to be empty
for k=-40:40
yy(k+41) = exp( -(k/10)*(k/20) );
end
plot(-40:40, yy)
```

Explain why it is necessary to write **`yy(k+41)`**. What happens if you use **`yy(k)`** instead? Also, explain the labels on the x-axis. This functional form is called a *Gaussian* form. Record your explanations on the Verification Sheet and attach a screen shot of the plot.

Verification Sheet Fill-in

c. Plotting is easy in MATLAB for both real and complex numbers. (See Appendix A of the textbook). The basic plot command will plot a vector **y** versus a vector **x**. Try the following:

```
x = [-2.8 -1.2 0 1.2 2.8 ];
y = x.*x - 1.8*x;
plot( x, y )
z = x + y*sqrt(-1); % a complex number consists of two parts
plot( z )           % complex values: plot real vs. imaginary
```

Use **help arith** to learn how the operation **xx.*xx** works when **xx** is a vector; compare array multiplication (**dot-star**) to matrix multiplication.

d. Learn more about arithmetic in vector and matrix forms. Try the following:

```
x = -3:2:3;
y = x.*x + (x*x')*x;
plot( x, y )
```

The apostrophe means transpose of a vector, i.e., turning a row vector into a column vector and vice versa. It of course also works for a matrix. (Note that there are more than one "apostrophe"-looking symbol; make sure which one works in MATLAB.) Take a screen shot of the plot and attach it to the completion report.

Verification Sheet Fill-in

e. Use the built-in MATLAB editor to create a script file called **mylab0.m** containing the following lines:

```
tt = -0.2 : 0.002 : 0.6;
xx = 4.* cos( 2*pi*5*tt + 1.);
plot( tt, xx, 'b-', tt, xx, 'r--' ), grid on %<--- plot a
                                                    %sinusoid
axis([tt(1) tt(end) -4 4]);   %what does this do?
title('TEST PLOT of a SINUSOID')
xlabel('TIME (sec)')
```

Mark the value of **xx** on your plot when it intersects with the time axis at 0 sec. The computed result, i.e., the array **xx**, is customarily considered the output of the program code. The concept of input-output will stay with us throughout the semester as we get into the "processing" aspect of the course objective.

From the plot which should look smooth, find the values of **tt** at which the wave attains a peak value. Which of these locations (values of **tt**) is the **closest to the origin** and what is the peak value? Give your answers on the Verification Sheet.

Verification Sheet Fill-in

f. Run your script from MATLAB. To run the file **mylab0** that you created previously, try

```
Mylab0        %<---will run the commands in the file
type mylab0           %<---will type out the contents of mylab0.m
                      %    to the screen
```

Lab #00

ECE-2026 Fall-2023

LAB COMPLETION REPORT

The report must include this page and the requested screen shots or files, which are attached after this page or included in a zip archive.

Name: Mythri Muralikannan Date of Lab: 21st Aug 2023

Part 2.1 Vector replacement using the colon operator:

```
xx(2:2:end) = pi^exp(1)   %assigns value of pi^e to every even index
```

Part 2.2(b) Write your vectorized statement below.

In addition, explain why it is necessary to write **`yy(k+41)`** in the loop. What happens if you use **`yy(k)`** instead? How would you change the plot command so that the extent of the x-axis would be from –10 to +30?

vectorized statement:

```
yy = [zeros(1,81)];
yy(1:81) = exp (-((-40:40)/10).*((-40:40)/20));
plot(-40:40, yy)
xlabel("x axis scaled x by 10")
ylabel("values of yy")
```

It is necessary to write yy(k+41) as array indexing begins at 1. If we use yy(k) instead it would result in an error due to an incorrect input of array index 0.

```
plot(-10:30, yy)
```

Part 2.2(d) Write out the result of **`y`**.

```
y =
```

Part 2.2(e) Give answers to the following:

Peak value = 2.16121	**`tt`** = -0.032

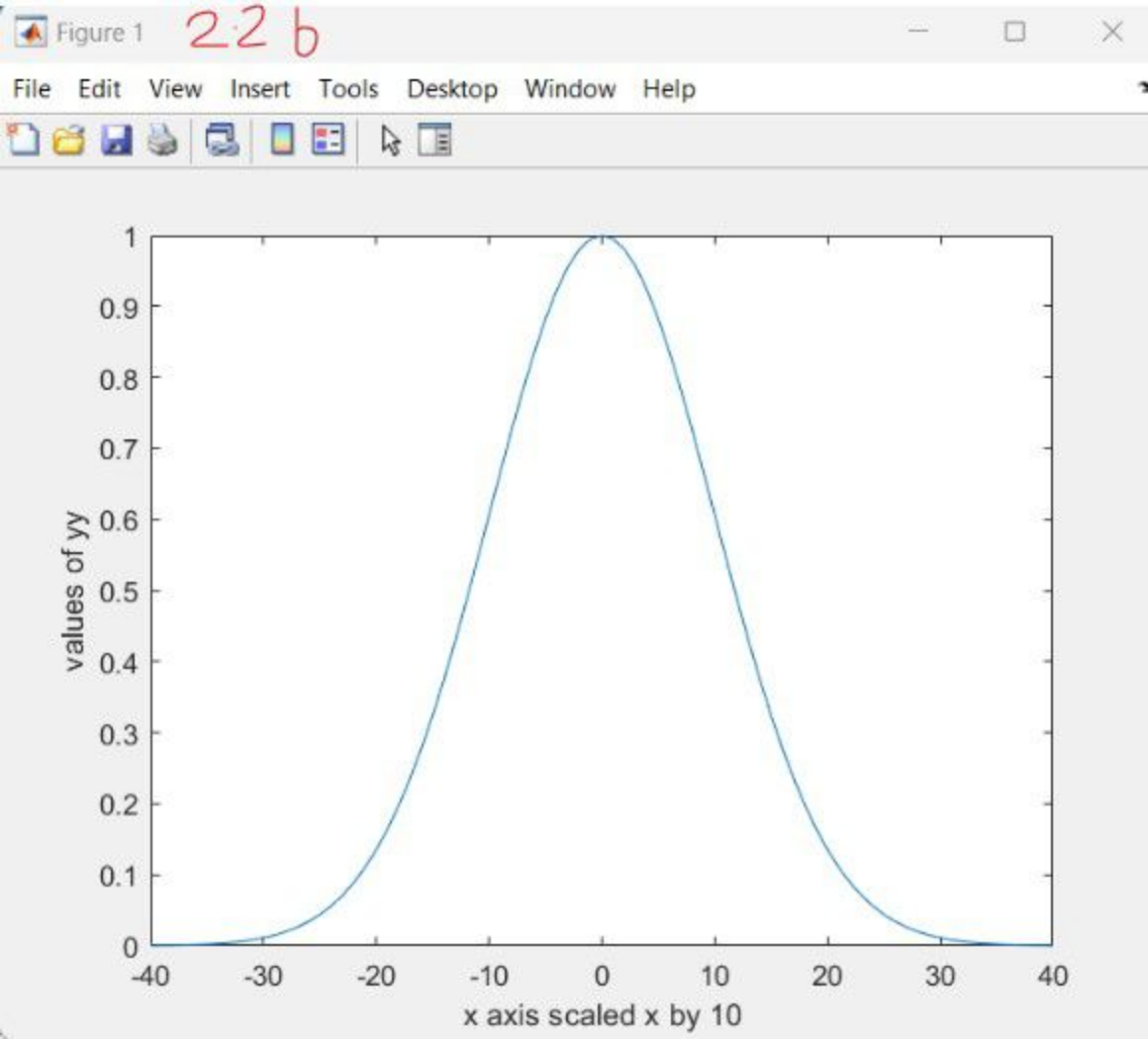
Figure 1
2.2 b
File Edit View Insert Tools Desktop Window Help
values of yy
x axis scaled x by 10
1
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
-40
-30
-20
-10
0
10
20
30
40

Figure 1 2.2d

File Edit View Insert Tools Desktop Window Help