Lab 1
August 22, 2016
Matlab Overview

INSTRUCTIONS:
All lab submissions include a written report and source code in the form of an m-file. The report contains all plots, images, and figures specified within the lab. All figures should be labeled appropriately. Answers to questions given in the lab document should be answered in the written report. ***The written report must be in PDF format***. Submissions are done electronically through **my.ECE**.

1 Overview

In this course, we will be using MATLAB to process digital information such as sounds and images. MATLAB is a scripting language that is adept at matrix manipulation. Therefore, it is important to understand basic linear algebra concepts such as vectors, matrices, and matrix multiplication. The purpose of this lab is to have students familiarize themselves with MATLAB and introduce useful lingo and code.

Begin by opening MATLAB on your computer. The screen that appears should be similar to Figure 1.
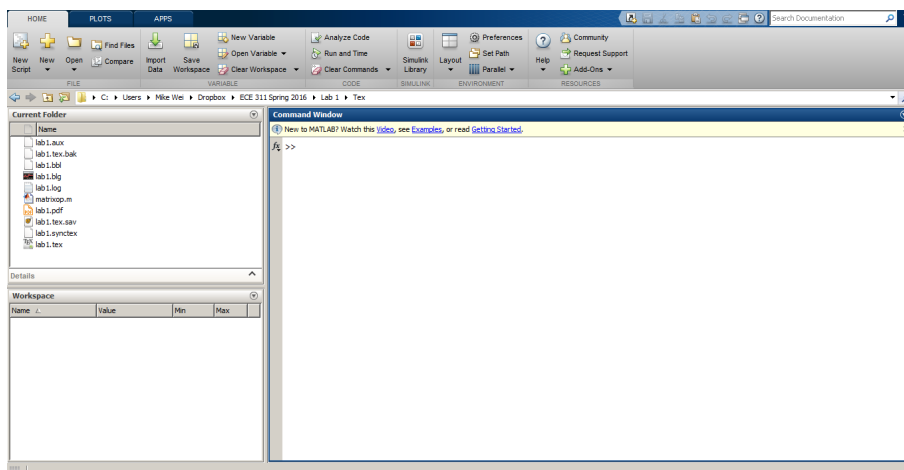

Figure 1: Hi MATLAB!

You should familiarize yourself with the layout of MATLAB. Notice the sections titled **Current Folder**, **Workspace**, and **Command Window**. **Current Folder** shows you the files within the current folder. **Workspace** shows the variables currently allocated for the current session. Other useful information such as dimension

and data type are given as well. Each time you restart MATLAB, your workspace variables will be cleared. MATLAB **commands** (functions) can be entered in the command window.

The command window is essentially the MATLAB console. Functions, scripts, and commands can be called through the command window. The command window can be cleared usinsg **clc**. Variables in your workspace can be cleared using **clear** [*variable name*]. To clear all variables, use **clear all**. Figures that contain plots or images can be closed manually or using the command **close all**.

MATLAB is known for its great documentation. To learn how to use any command, simply type **help** [*command name*] into the console to get a brief explanation. Replacing **help** with **doc** brings up a separate window containing a detailed explanation complete with examples and pictures. This is extremely useful!

2   Scripts

It is useful to type commands or variable names directly into the command window when you want to execute a single command or view the contents of a variable. For lengthier code, the commands can be written in a MATLAB script known as an m-file. A new script can be created by going to **New→Script**.
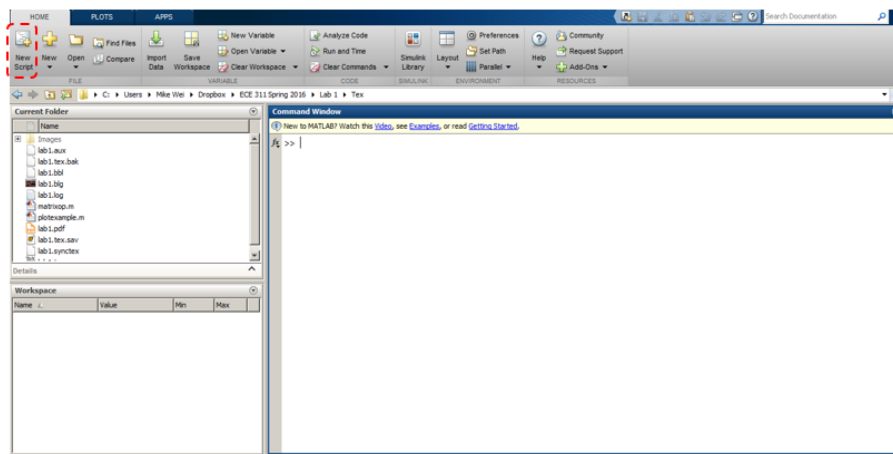

Figure 2: Clicking on the button in the dotted-red circle creates a new script.

Most scripts that we write should begin with the commands: **clc**, **clear all**, **close all**. This clears the command window, variables, and closes all figures. Doing so avoids clutter and accidental re-use of variables from previous scripts. To run a script, press *F5* or click the run button with your mouse cursor. Comments can be inserted into your script using the % sign. Two percents (%%) can be used to separate your script into sections where each section can be run separately by clicking *Run Section* or *Ctrl+Enter*.

```matlab
1   % This is a comment
2   clc, clear all, close all
3
4   %% Section 1
5   x = linspace(0,1,100);
6   y = x;
7   plot(x,y);
8
9   %% Section 2
10  x = linspace(0,1,100);
11  y = x.^2;
12  plot(x,y);
13
14  %% Section 3
15  x = linspace(0,1,100);
16  y = 3*x.^2+2;
17  plot(x,y);
```

Figure 3: This script contains a comment at the top and 3 sections. Each section can be run separately.

Scripts can be executed just like any other MATLAB command by typing the name of the script into the command window and pressing *Enter*. However, MATLAB is only aware of the scripts within its list of available paths in addition to the current folder. If your script is located outside MATLAB's scope, you must manually set the path or add the path to MATLAB's path list.
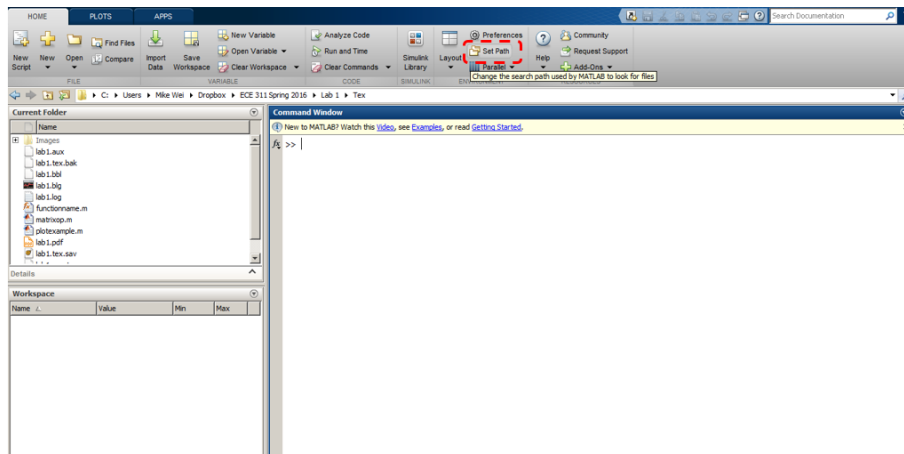

Figure 4: Additional paths can be added to MATLAB's path list. A script that is not within MATLAB's scope cannot be called.

Report Item: Create a folder called *ECE 311* in your home directory. Add the path of this folder to your list of MATLAB paths. Include a screen shot showing that this has been done in your report.

3 Vectors and Matrices

In digital signal processing, sounds and images are stored in matrices. The **zeros** command is used to allocate a matrix. A vector is considered a special case of a matrix where one of the dimensions is 1. The allocation of a 3×5 matrix Z is shown in Figure 5. Matrices can also be declared element-by-element. This is shown in Figure 6.

```
>> Z = zeros(3,5)

Z =

     0     0     0     0     0
     0     0     0     0     0
     0     0     0     0     0
```

Figure 5: Allocation of a matrix using **zeros**.

```
1  Z = [1,2; 3,4];
2  columnVector = [1;2;3;4];
3  rowVector = [1,2,3,4];
```

Figure 6: Matrices can also be allocated manually. The semi-colon can be interpreted as "next column".

To access a specific element in Z, one can write $Z(m, n)$, which returns the element in the m-th row and n-th column of Z. Note that MATLAB indexing begins at 1, not 0 (unlike C). A colon can be used to access a specific row or column in a matrix. For example, the m-th row can be accessed with $Z(m, :)$. Similarly, the n-th column can be accessed with $Z(:, n)$. To access a sub-matrix, one can write $Z(m : n, i : j)$ to access a matrix consisting of rows m through n and columns i through j of Z. An example is provided in Figure 7.

```
>> Z = rand(3,5)

Z =

    0.7060    0.0462    0.6948    0.0344    0.7655
    0.0318    0.0971    0.3171    0.4387    0.7952
    0.2769    0.8235    0.9502    0.3816    0.1869

>> Z(2,3)

ans =

    0.3171

>> Z(1:2,3:4)

ans =

    0.6948    0.0344
    0.3171    0.4387

>> Z(2,:)

ans =

    0.0318    0.0971    0.3171    0.4387    0.7952
```

Figure 7: Different matrix access patterns are shown.

Vectors can be allocated in several ways. Aside from **zeros**, a vector can also be allocated using colon notation. Some examples are shown in Figure 8. Access to vector elements is identical to that of a matrix (vectors are a special case of a matrix). For a N element vector, we consider a row vector to be $1 \times N$ matrix and a column vector to be a $N \times 1$ matrix. Often times, MATLAB will throw an error when it expects a row vector instead of a column vector and vice versa. To fix this error often requires taking the transpose of one or more matrices in involved. This can be done by using **transpose**.

```
>> V = 0:0.1:1

V =

      0    0.1000    0.2000    0.3000    0.4000    0.5000    0.6000    0.7000    0.8000    0.9000    1.0000

>> linspace(0,1,11)

ans =

      0    0.1000    0.2000    0.3000    0.4000    0.5000    0.6000    0.7000    0.8000    0.9000    1.0000
```

Figure 8: Some ways to allocate vectors in MATLAB.

> **Report Item:** Define a vector of length $N = 12$ equispaced points over the range $a = 0$ and $b = 1$. What is the delta or gap between two consecutive points in terms of a, b, and N?

4 Plots and Labels

A line plot can be created using **plot**. Plots are placed in the current active *figure*. A new figure can be created using **figure**. After creating a plot, the commands

xlabel, **ylabel**, and **title** can be used to label the plot. Be sure to label all figures appropriately!

```
1  x = linspace(0,1,100);
2  y = x.^2;
3  figure;
4  plot(x,y);
5  xlabel('x','fontsize',14);
6  ylabel('y','fontsize',14);
7  title('Title','fontsize',14);
```

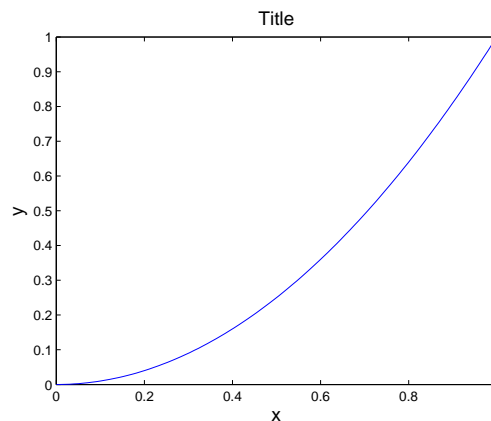Figure 9: Code used to generate a basic line plot.

Figure 10: Line plot generated from code in Figure 9.

Report Item: Use **linspace** to define a vector x with $a = 0$, $b = 5$, and $N = 100$. Use this vector to create a line plot of $f(x) = x^2 \log(x) \sin(x)$ (*Hint: Use element-by-element operations*). Make sure to properly label both x and y axes and add an appropriate title.

Plots are generally used for 1-D continuous signals. For discrete signals, **stem** should be used. Both stem and plot require you to provide vectors containing values for the x and y axes. The length of these vectors must be identical.

Multiple plots can be superimposed onto the same plot using **hold**. When you want this behavior to be active, input the command **hold on**. When you want to de-activate this behavior, use **hold off**. Several plots can also be placed side-by-side on the same figure using **subplot**. The syntax for this command is **subplot** [*rows*][*cols*][*index*] where index refers to a specific row and column where the plot will appear.

```
1  x = linspace(−1,1,500);
2  f1 = sin(2*pi*1*x);
3  f2 = sin(2*pi*5*x);
4  figure;
5  subplot(211);
6  plot(x,f1);
7  subplot(212);
8  plot(x,f2);
```
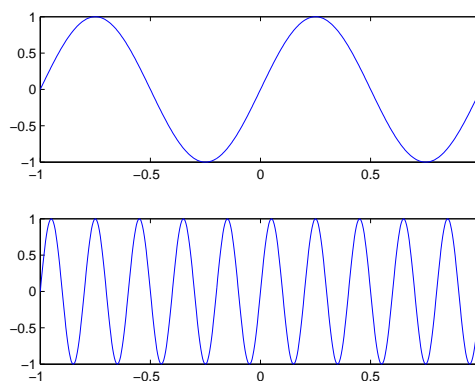
Figure 11: A simple subplot example.

Figure 12: The subplot generated from code in Figure 11.

Plots can be saved by going to the top-left of the figure window **File→Save as...**.

Report Item: Use colon notation to define $x = [0, 5]$ with $\Delta x = 0.2$. Implement the functions $f(x) = x^2 e^{\sqrt{x}}$ and $g(x) = 3\sqrt{x} + \sin(8\pi x)$ without using any loops. Use **subplot** to plot both functions in the same figure. Repeat using **stem**.

5   Loops

Loops are generally discouraged in MATLAB unless absolutely necessarly. However, no programming language would be complete without loops. Therefore, we briefly mention them here. MATLAB provides many of the familiar looping constructs. So if you're already familiar with loops in other language, the idea is the same. MATLAB allows *for* and *while* loops. Examples are shown below:

```matlab
% For loop.
for i = 1:10
    disp(['i = ', num2str(i)]);
end

% While loop.
counter = 1;
while counter < 10
    counter = counter + 1;
end

% Another for loop
for i = linspace(−1,1,5)
    disp(['i = ', num2str(i)]);
end
```

Figure 13: Some examples of loops in MATLAB.

If a variable used outside a loop has the same name as the loop variable, that variable will be overwritten by the loop variable. Be careful! In general, we want to avoid using loops in places where matrix operations can be used instead.

6   Matrix Operations

Given a matrix A and B, multiplication is $A * B$. If the number of columns in A is not the same as the number of rows in B, MATLAB will return an error. This is a common source of error. The dimensions of a matrix can be determined using **size**. If the dimensions are mismatched, one often needs to transpose one or more matrices. This can be done using the **transpose** command.

Additionally, we often require element-by-element multiplication instead of matrix multiplication. This can be done using $A.*B$. Similarly, element-by-element division is $A./B$. Matrix addition and subtraction are inherently element-by-element operations and can be expressed without a dot. For example, $A + B$ or $A - B$. Element-by-element power of a matrix is $A.\hat{}n$. Element-by-element operations can reduce the need to use for loops. For example, if we want to plot a function $f(x, y)$, we can do so by defining x and y and writing a double *for* loop to compute $f(x, y)$. Alternatively, we can use **meshgrid**. An example is given below:

```matlab
x = linspace(−2,2,200);
y = linspace(−2,2,200);
[xx,yy] = meshgrid(x,y);
f = yy + xx.^2 + yy.^3.*xx;
imagesc(x,y,mag2db(abs(f)));
axis xy;
```

Figure 14: Code that implements a 2-D function using **meshgrid**.

Report Item: Use meshgrid to define a 400×300 grid spanning $x = [-5, 5]$ and $y = [-4, 4]$, respectively. Compute $f(x, y) = \text{sinc}(\sqrt{x^2 + y^2})$ without using any for loops. Plot the result using **imagesc**. Make sure the axes are labeled. Repeat for $f(x, y) = \text{sinc}(x)\text{sinc}(y)$.

---

7   Functions

MATLAB functions are written by writing a script. MATLAB allows multiple output arguments in addition to multiple input arguments. The syntax for a function script is quite specific. The script and command must have the same name. Figure 15 shows an example of a simple MATLAB function. The function can be called the same way as a command. Since functions are written within scripts, the path of the script must be known to MATLAB in order to call it.
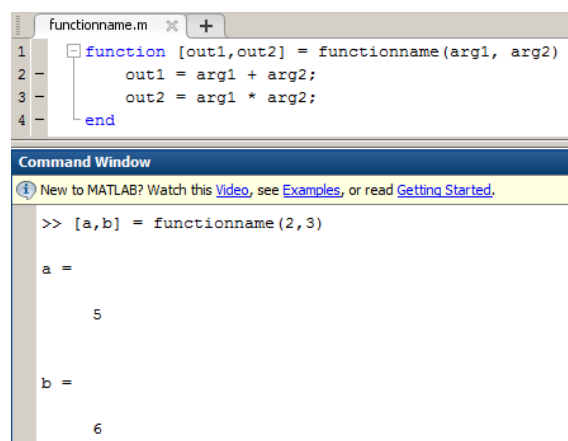

Figure 15: Basic MATLAB function.

Report Item: Write a function called **myDFT** that implements the DFT using two *for* loops. There is only one input and one output, $x(n)$ and $X(k)$, respectively. Verify your results using MATLAB's built-in **fft**.

$$X(k) = \sum_{n=0}^{N-1} x(n)e^{-i2\pi kn/N}, \quad k = 0, \dots, N-1$$

The output of myDFT is complex. Test your function on $x = [1, 3, -4, -3, 4]$. Plot the real and imaginary parts (using **real** and **imag**) of the output using **stem** on the same plot using the **hold** command.

---

8   Sounds and Images

In this course, we will need to manipulate sounds and images. Using **audioread**, audio files of various formats can be read into MATLAB. Sounds (or any vector, for that matter) can be played back using **soundsc**. Proper audio files include the sampling rates of the audio samples they contain.

> **Report Item:** Load *speechsig.mat*. A variable called x should appear in your workspace. The sampling frequency for this file is not given. Play the sound using **soundsc** and manually adjust the sampling frequency until it sounds reasonable. What frequency did you choose? What is the range of human hearing (approximately)?

Images can be read using **imread** and displayed using **imshow**. For grayscale images, we will often use **imagesc** instead. **imagesc** defaults to a jet colormap. We can change the colormap using **colormap**. For example, **colormap('gray')** will give us a grayscale image. Additionally, we will make use of 2-D FFT which can be viewed more naturally using images instead of line plots.

> **Report Item:** MATLAB contains several built-in images. Load *clown* using the **load** command. A matrix X defining the image itself should appear, along with its associated **colormap**—map. Use **imagesc** to display the image. Set the colormap to map using **colormap**. Include this image in your report. Using subplot, plot the 17th row and the 49th column of the clown image in separate plots. Include this plot in your report. Finally, take the transpose of the clown image and display it using **imagesc**. Include this image in your report.