University of Illinois at Urbana-Champaign
Department of Electrical and Computer Engineering

ECE 311: Digital Signal Processing Lab

LAB 1: **Introduction to MATLAB**
Fall 2015

---

1 Overview

The goal of this lab is to get familiar with MATLAB and to learn the basics of how to create, represent, manipulate, and analyze discrete time signals in MATLAB.

2 Getting started

- Get familiar with MATLAB by using tutorials and demos found in MATLAB. You can click `Start` ≫ `MATLAB` ≫ `Demos` to open a comprehensive help screen.
- At the prompt, type `help` or `doc` followed by the command to get information about the command. Try the following:

```
>>help plot
>>help stem
>>help cos
>>help sin
>>help exp
>>doc for
>>doc ones
>>doc zeros
```

- What if you do not know a command name? Type `helpdesk` to start the help window. You can then type keywords in the search bar.
- Variables in MATLAB can hold numbers, vectors or matrices:

```
>>x = ones(1,1)
>>x = ones(1,5)
>>x = ones(5,1)
>>X = ones(5,5)
```

- Putting a semicolon after a variable assignment prevents unnecessary 'echoing':

```
>>x = ones(10,10)
>>x = ones(10,10)-0.5;
```

Exercise 1
1) Find MATLAB commands that generate random numbers.
2) Generate a 3×3 random matrix.

- The result of a computation can be assigned to another variable. If the result is not assigned to a variable, MATLAB stores it in the variable `ans`

```
>>x = 0.25*0.25;
>>x
x=
      0.0625
>>0.25*0.25
ans =
      0.0625
```

- It is very easy to manipulate complex numbers in MATLAB. You can assign a complex number of the form `a+b*1i` directly to a variable and perform arithmetic operations (it is recommended to use `1i` instead of `i` to accelerate your code):

```
>>x = 2+3*1i
>>y = 3+4*1i
>>x+y
ans =
      5.000 + 7.000i
```

Exercise 2
1) Find the MATLAB commands that calculate the conjugate, absolute value, real part and imaginary part of a complex number.
2) Calculate the conjugate, absolute value, real part and imaginary part of $3 + 4i$

- Loops in MATLAB: A simple `for` loop can be written as,

```
for n = 1:10
    x(n) = n - 1;
end
x,
```

The loop generates integers from 0 to 9. You can also use `while` loop instead of `for`. In general MATLAB is not very efficient with loops. We will see later that one can avoid loops by using the `colon` operator. See more information about flow control in Matlab in `help` under :
`MATLAB/Getting Started/Programming/Flow Control` .

3 MATLAB functions for representing signals

- The `colon` operator: The colon operator ':' can be used to define a vector. Let us say we want to create a vector `x` which holds the integers from 0 to 100. One way is to use a loop. Another way to do this is shown below,

```
>> x = [0:100];
```

In this case the brackets are optional. The same result is given by

```
>> x = 0:100;
```

- In general, signals are represented by row or column vectors, depending on the context. All vectors in MATLAB are indexed starting with 1. One may have to create an additional vector to properly keep track of signal index.

- Discrete-time complex exponentials form an important class of functions in the analysis of discrete-time signals and systems. A discrete-time complex exponential has the form α^n, where α is a complex scalar. The discrete-time sine and cosine signals can be built from complex exponential signals by setting $\alpha = e^{\pm i\omega}$,

$$\cos(\omega n) = \frac{1}{2}\left(e^{i\omega n} + e^{-i\omega n}\right)$$
$$\sin(\omega n) = \frac{1}{2i}\left(e^{i\omega n} - e^{-i\omega n}\right)$$

MATLAB has functions `cos`, `sin` and `exp` to manipulate these signals. Note that e^x is expressed as **exp(x)** in MATLAB.

Example 1
Generate the following signal

$$x[n] = \sin(\frac{2\pi n}{10}),\ n = 0, 1, 2, \cdots, 20$$

1) Define n

```
>> n = 0:20;
```

2) Generate $x[n]$

```
>> x = sin(2*pi*n/10);
```

3) Plot this signal using `stem` command

```
>> stem(n,x);
```

Exercise 3
1) Generate the following signal

$$x[n] = \exp(i\frac{3\pi n}{10}), n = 0, 1, 2, \cdots, 20.$$

3) Display the real and imaginary parts of this signal.

- MATLAB has several commands to help you label the plots appropriately, as well as to print them out. `title` places its argument (a string, e.g. **title('your title')**) over the current plot as the title. `xlabel` and `ylabel` allow you to label the axes. Every plot or graph you generate must have a title and the axes must be labeled clearly.

- You can control the font and text size using MATLAB's *'PropertyName', PropertyValue* notation (e.g. **xlabel('Frequency (Hz)','FontName','Arial','FontSize',16))**

Example 1 (continued)
4) Label the plot

```
>> title('A sine signal: sin(2*pi*m/10)');
>> xlabel('n (Samples)');
>> ylabel('Amplitude');
```

Exercise 3 (continued)
3) Label the plots.

- You can also extend the range of the sequences. For instance, begin with the signal

```
>>n = [-3:3];
>>x = [1:7];
```

If you want to pad the signal to the range $-5 \leq n \leq 5$, you can extend the index vector `n` and add additional elements to `x`,

```
>>n = [-5:5];
>>x = [0 0 x 0 0];
```

- What if want to extend this range by a large value? Use `zeros`:

```
>>n = [-100:100];
>>x = [zeros(1,95) x zeros(1,95)];
```

- We can define $x_1[n]$ to be the discrete-time unit impulse function and $x_2[n]$ to be the time-advanced version of $x_1[n]$, i.e., $x_1[n] = \delta[n]$ and $x_2[n] = \delta[n+2]$. We can do this by,

```
>>nx1 = [0:10];
>>x1 = [1 zeros(1,10)];
>>nx2 = [-5:5];
>>x2 = [zeros(1,3) 1 zeros(1,7)];
```

- Let us now plot the signals,

```
>>stem(nx1,x1)
>>stem(nx2,x2)
```

Note that if you want to plot one signal on top of the other, you need to use the **hold** command (you can also choose the colors of your curve as follows)

```
>>stem(nx1,x1,'b')
>> hold on
>>stem(nx2,x2,'r')
```

- MATLAB also allows you to add, subtract, multiply, divide, scale and exponentiate signals. let us define the signals `x1` and `x2`,

```
>>x1 = sin((pi/4)*[0:15]);
>>x2 = cos((pi/7)*[0:15]);
```

Try the following

```
>>y1 = x1 + x2
>>y2 = x1 - x2
>>y3 = x1 .* x2
>>y4 = x1 ./ x2
>>y5 = 2*x1
>>y6 = x1 .^2
>>y7 = x1 * x2
>>y8 = x1 * x2'
```

IMPORTANT: For multiplying, dividing, and exponentiating on a term by term basis ('element-wise'), you must precede the operator with a period '`.*`' instead of '`*`'. Also note '`x2'`' converts the row vector `x2` into a column vector, and takes the complex conjugate - this is the hermitian transpose (conjugate transpose) of the argument. If you want to transpose `x2` without conjugating it use `x2.'`.

4 MATLAB scripts and functions

- MATLAB allows us to create `m-files`. There are two types of `m-files`: `scripts` and `functions`
- A command script is a text file of MATLAB commands whose filename ends in a **.m** in the current working directory or elsewhere in your MATLAB path. A script has no input or output arguments. If you type the name of the file (without `.m`) at the command prompt the commands contained in the script file will be executed.

Example 1 (continued)
5) Create a script file based on the above tasks and run it from the terminal.

Exercise 3 (continued)
4) Create a script file based on the codes you have and run it from the terminal.

- An `m-file` implementing a function is a text file with a title ending `.m` whose first word is a `function`. The rest of the line specifies the input and output arguments.
- The following `m-file` is a function called foo. It accepts input `x` and returns `y` and `z` which are equal to `2*x` and `5/9*(x-32)` respectively

```
function [y,z] = foo(x)

%[y,z] = foo(x) accepts a numerical argument x and
% returns two arguments y and z, where y is 2*x and z is (5/9)*(x-32)
```

```
y = 2*x;
z = (5/9)*(x-32);
```

Try the following

```
>> help foo
>>[y,z] = foo(-40)
>>[y,z] = foo(225)
```

- In MATLAB, comments may be made using the '%' symbol as shown above. Please remember to always comment your code in a clear, organized manner :)
- For this lab, I recommend that you save the function files individually, as '*FunctionName*.m'. However, if you want to make a long script for your other code, you can easily separate and run code 'cells' using a double percent sign '%%'. This should draw a line separating sections of your code. If you select a block of code you can then use the green arrow to run it, or right click and select 'Evaluate Current Cell.'

Example 1 (continued)
6) Create a function that takes the digital frequency (ω) of the signal as the input and the signal as the output. Run the function from the terminal.

```
function x = demo2_func(w)

%% generating a sine signal
n = 0:20;
x = sin(w*n);
stem(n,x);

%%
title('A sine signal: sin(2*pi*n/10)');
xlabel('n (Samples)');
ylabel('Amplitude');
```

Exercise 3 (continued)
5) Create a function that takes the period of the signal as the input and the signal as the output. Run the function from the terminal.

5 Cleaning up

Sometimes we all just want a fresh start....

- **close all**: Close all open figures
- **clear all**: Clear all variables in the workspace
- **clc**: Clear the command window

6 Homework - Due 09/14/2015 at 5:00 PM (Submit Online)

1. (a) On a single plot (e.g. if **x** is a complex variable, plot using **plot(real(x),imag(x))**, representing the complex plane, graph:
   i. The complex number $z = 1 - 3j$ (mark the location with an asterisk, i.e. '*') (The commands **figure**, **plot**, **real**,and **imag** may be useful). Also give the magnitude and phase of this number (the commands **sqrt**, **abs**, **angle**, **exp**, and/or **hold** could come in handy).
   ii. The complex number $exp((1 - 0.2j))$ (mark the location with 'o').
   iii. The unit circle, by plotting 1001 complex points (e.g., using the vector [0:1000]) $exp(j\theta_k)$, where θ_k takes on 1001 values from 0 to 2π (The MATLAB command **axis equal** will scale the axes so that this looks like a true circle).

   (b) On the same graph, plot 100 samples (e.g., $n = [0 : 99]$) of the signal $e^{0.03(1+3j)n}$, marking each location with an **'x'**.

   (c) Plot the real and imaginary parts of the signal $e^{0.03(1+3j)n}$, respectively, using the **stem** command. Use the **subplot** command to plot the real and imaginary parts in a single figure. Label the **x-axis** as **'samples'** (**xlabel**) and the **y-axis** as **'real amplitude'** or **'imaginary amplitude'** (**ylabel**) as appropriate.

   (d) Plot the magnitude of the signal $e^{0.03(1+3j)n}$. Use the command **phase** or **unwrap** and **angle** to plot the phase respectively. Compare the difference and explain it.

2. Consider the discrete-time signal

$$x_M[n] = sin(\frac{2\pi Mn}{N}),$$

and assume $N = 14$. For M = 2,5,7 and 12, plot $x_M[n]$on the interval $0 \le n \le 2N - 1$. Use **stem** to create your plots, and be sure to appropriately label your axes. What is fundamental period of each signal (e.g. how many samples before the signal repeats - remember this is not an analog signal)? In general, how can the fundamental period be determined from arbitrary integer values of M and N?

3. (a) Plot 200 samples of the sinc function

$$sinc(t) = \begin{cases} 1, & t = 0, \\ \frac{\sin(\pi t)}{\pi t}, & t \neq 0. \end{cases}$$

on the interval $[-10, 10]$. (Use the **sinc** command or you can write your own sinc function.) What are the locations of the nulls, and what is the width of the main lobe?

   (b) Plot the Dirichlet/periodic sinc function

$$diric(t) = \begin{cases} (-1)^{\frac{t}{2\pi}(n-1)}, & t = 0, \pm 2\pi, \pm 4\pi, \ldots \\ \frac{\sin(nt/2)}{n\sin(t/2)}, & \text{otherwise.} \end{cases}$$

on the interval $[-4\pi, 4\pi]$ for n = 7. (Use the **diric** command or you can write your own diric function.) Also plot the Dirichlet function for $n = 4$. For a given n, what are the locations of the nulls, what is the width of the main lobe, and what is the period?

   (c) What are the similarities and differences between the sinc and the Dirichlet function?

4. (a) In MATLAB, create a 100×100 matrix containing the magnitude of the complex function $f(z) = \frac{z-0.35}{z-0.8e^{\frac{-j\pi}{3}}}$, for values of z over the ranges 0 to 1 in both the real and imaginary parts (that is, for all combinations of $\Re_e(z) = (0, 0.01, 0.02, \ldots, 0.99)$ and $\Im_m(z) = (0, 0.01, 0.02, \ldots, 0.99)$, and plot that using either the command mesh or surf, whichever you prefer. (The **for** and **zeros** commands may be useful.)

(b) In MATLAB, create a 256×256 head phantom using the **phantom** command. Display the image using the **imagesc** command. Plot the 150^{th} row of the created phantom. Then, load the clown image using the command **load clown;**. Display the image using **imagesc**(X) both in the given colormap and in grayscale (the **colormap** command may be helpful). Plot the 150^{th} row of the image. Which values encode black and white, respectively, in the grayscale image? (Note: The phantom and clown are stock images that live somewhere inside of MATLAB. If you want to view a particular image that is not in MATLAB, it needs to be in your current directory)

(c) Write a function to flip an image i) upside-down, and ii) from right to left. Display the effects on the clown image.

(d) Create a random matrix of the same size as the clown image, with values from 0 to 1. Multiply it with the image. Display the results. How do the results look? Try the following types of random matrices: i) gaussian, ii) matrix with only two kinds of values – 0 and 1, but these values are placed at random locations in the matrix.

5. (a) Define a MATLAB vector nx to be the time indices $-3 \leq n \leq 7$ and the MATLAB vector x to be the values of the signal $x[n]$at those samples, where $x[n]$ is given by

$$x[n] = \begin{cases} 2, & n = 0, \\ -1, & n = 2, \\ 1 & n = 3, \\ 3 & n = 4, \\ 0 & \text{otherwise}, \end{cases}$$

Plot this discrete-time sequence by typing **stem(nx,x)**.

(b) Define vectors $y1$ through $y4$ to represent the following discrete-time signals:

$$\begin{aligned} y_1[n] &= x[n-2] \\ y_2[n] &= x[n+1] \\ y_3[n] &= x[-n] \\ y_4[n] &= x[-n+1] \end{aligned}$$

To do this, you should define $y1$ through $y4$ to be equal to x. The key is to define correctly the corresponding index vectors $ny1$ through ny_4. First, you should figure out how the index of a given sample of $x[n]$ changes when transforming to $y_i[n]$. The index vectors need not span the same set of indices as nx, but they should all be at least 11 samples long and include the indices of all nonzero samples of the associated signal.

(c) Generate plots of $y_1[n]$ to $y_4[n]$ using **stem**. Based on your plots, state how each signal is related to the original $x[n]$, e.g. "delayed by 4" or "flipped and advanced by 3".