EE-3221 MATLAB INTRODUCTION

Goal – Become familiar with MATLAB and its ability to manipulate and plot discrete signals (sequences of numbers).

Background – MATLAB is an industry-standard software package for processing lists of data in matrix or vector form. It can perform DSP operations quickly, even on large data sets. It is a programming language and is very picky regarding syntax. Within MATLAB you will give names to lists of numbers and then perform mathematical operations on those numbers.

Procedure – Run MATLAB on your laptop. You will see a main Command Window – this is where your work will generally take place. Set the Current Directory (top, middle of the screen) to a folder where your EE3221 lab work will be saved. In the Command Window, notice the >> prompt. Also, the Help menu can be very useful.

Let's begin by generating a sequence of time values that are separated by 0.1s in the interval 0 to 1 s, and save this sequence as a vector named "t."
t = 0 : 0.1 : 1

You can display the result of each element in the sequence (vector) with the following:
t(1)
t(2)
t(3)

NOTE: MATLAB SPECIFIES THAT THE FIRST ELEMENT IN THE VECTOR IS ELEMENT #1, NOT #0, WHICH CONFLICTS WITH THE NORMAL DEFINITIONS IN THE DSP WORLD. Think of matrix elements instead. That is, t(1) = 0, and t(2) = 0.1000, etc. THERE IS NO element t(0).

You can determine the size, or how many elements are in a vector, using the following function:
size(t)

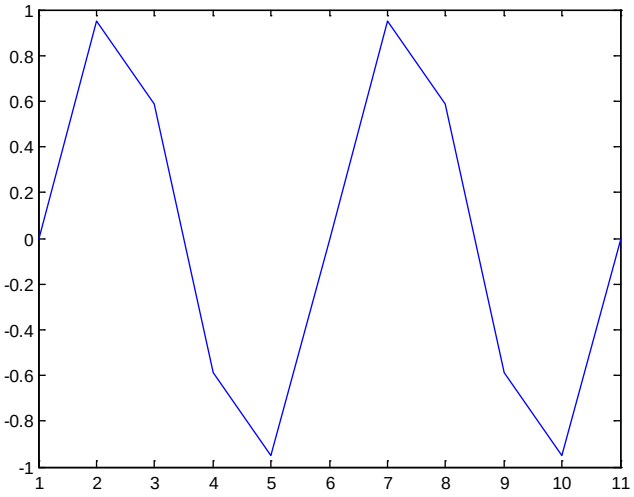
1. *What are your results?*

Finally, it is often better to declare the "time sampling period" by declaring it as a variable in your code. For example:
Ts = 0.1; % sampling period
t = 0 : Ts : 1;

Basic Plots

Oftentimes we want to plot a time-valued function such as x(t) = sin(2πf·t). Note that t is not an integer, but rather a continuous variable. We can create a plot by setting t to discrete values and evaluating x(t) at those values. For example, using the vectors created in the previous section enter the following:
f =2 % frequency in Hz
x = sin(2 * pi * f*t)

Next we can plot the sine function:
plot(x) % note that the samples are shown connected by line segments



2. *Does this plot closely resemble sin(4πt), which is a function of the continuous variable t? Why or why not? What are the values on the horizontal axis? Explain your answer.*

We can modify the plot with the following:
plot(t, x)
xlabel('t')
ylabel('sin(4\pi*t)')
title('Basic Sine Wave Plot')

3. *What has changed and why?*

In class we will typically use stem plots to represent discrete sequences. For example, try:
stem(x)

4. *What has changed? How would you label the horizontal axis? What will happen if we type stem(t, x) on the command line?*

Multiple Plots

In this section you will learn how to plot multiple functions on one plot or a series of plots. The MATLAB commands you will use are **hold** and **subplot().** We will illustrate the concept by plotting two different sinusoids on the same plot. The frequency of the sinusoids will be 11 Hz and 12 Hz.

Let's begin by decreasing our sampling period to 1000 samples per second. What will Ts be equal to?

Ts = <?>
t = 0:Ts:1;

5. *What is the size of the vector t?*

To set f1 =11 and f2 = 12, we can execute:
f1 = <?>
f2 = <?>

x1 = sin(2*pi*f1*t);
x2 = <?>**;**

The hold command is used to hold the contents of the plot for plotting additional sequences.
plot(t,x1)
hold on
plot(t,x2)

Axes labels and a plot name can also be added.

You can turn off the hold for future plots. This is accomplished with:
hold off

Next we create aligned subplots in a single plotting window. For example,
subplot(4,1,1)

This will create a window consisting of four plots in a single column. The syntax is subplot(num_row, num_col, plot_number). Let's plot x1, x2, x1 and x2, and x1+x2 on the four different plots, complete with axes labels.

```
subplot(4,1,1)
plot(t,x1)
xlabel('t')
ylabel('x1(t)')
subplot(4,1,2)
plot(t,x2)
xlabel('t')
ylabel('x2(t)')
subplot(4,1,3)
plot(t,x1)
hold on
plot(t,x2)
xlabel('t')
ylabel('x1(t), x2(t)')
subplot(4,1,4)
plot(t,x1+x2)
xlabel('t')
ylabel('x1(t)+x2(t)')
subplot(4,1,1)
title('<your name> Plots')
```

Optional Challenge: Re-do the above replacing f1 and f2 with a 2-element column vector f (use a semicolon to separate values within square braces to make a column vector: [f1; f2]). Then, f*t is a matrix (2×1 times 1×N yields 2×N by rules of matrix multiplication) and only one assignment statement to x is needed. To extract rows of x, use x(1,:) and x(2,:). Also, try plot(t,x) when x is a 2×N matrix. `sum(x)` will add the 2 row-vectors. Type `help sum` or `doc sum` for more information on the sum function.

Complex Numbers

MATLAB works with complex numbers. Use 1i or 1j to represent the imaginary unit and, 2.4+3.4j, for example, to represent a complex number. (MATLAB also defaults to allowing just i and j to represent the imaginary unit, but avoid using them since they can be redefined, whereas 1i and 1j cannot be redefined.)

1i*1i

6. *Does this display the proper result? What will 1i*1i*1i display as?*

Let's consider a discrete signal $x(n) = 4e^{-j\frac{\pi}{4}n}$ for n = 0, 1,…,7 to illustrate complex valued sequences. Notice that the independent variable n has discrete values (specifically, integer values), and therefore, x(n) as a discrete signal. One interpretation is that x(n) is a sampled version of the corresponding continuous signal. Notice, however, that the sampling period Ts is not explicit. Let's examine the signal in MATLAB:

```
figure                          % open a new figure window
n = <?>:<?>:<?>                 % specify the limits of n; if the step size is 1, it can be omitted
x = 4*exp(-1i*pi/4*n)           % it is okay to use 1j instead
subplot(4,1,1)
stem(n,real(x))
subplot(4,1,2)
stem(n,imag(x))
subplot(4,1,3)
stem(n,abs(x))
subplot(4,1,4)
stem(n,angle(x))
```

7. *Add appropriate titles to the plots. What mathematical identities do they represent (hint: what identity specifies the relationship between the complex exponential and the sine and cosine)? Can you give equivalent mathematical expressions for the plots?*

Element-wise Operations

As you have seen, MATLAB works with vectors and matrices. Operations such as addition and multiplication follow the usual rules for vectors and matrices, as you learned or will learn in MA-383 Linear Algebra.

In DSP, we will frequently need to perform element-wise operations on discrete sequences. To do the operations of multiplication, division, and raising to a power in an element-wise fashion, you precede the operator with a period.

Consider the signals x1(n) = n and x2(n) = (-1/2)n + 1. Let's demonstrate element-wise operations:

```
n = -4:1:4
x1 = n
x2 = (-1/2)*n + 1
x3 = x1.*x2
x4 = (-1/2)*n.^2 + n
```

8. *Create a plot (using subplot and appropriate axes labels and titles) showing the four signals. What can you conclude from these plots?*

Finding the Sequence Energy

We conclude this lab by finding the energy in a sequence. We will use several different methods in order to illustrate additional concepts in MATLAB.

The energy in a sequence is defined as $E = \sum_{n=n1}^{n2} x(n) \cdot x^*(n)$. This will be discussed in more detail in class.

Let's use MATLAB to find the energy in the sequence $x(n) = 4e^{-j\frac{\pi}{4}n}$ from n1 = 0 to n2 = 7.

```
n1 = 0
n2 = 7
n = n1:n2
x = 4*exp(-1j*pi/4*n)
x_mag = x.*conj(x)
E = sum(x_mag)
```

9. *What is the result?*

Scripts and Functions

We have been working exclusively from the command line prompt (**>>**). However, we can write scripts and functions that are saved for future re-use.

A SCRIPT file is an external file that contains a series of MATLAB commands. By typing the filename at the command line, MATLAB will execute the commands in the script file. Script files have a filename extension of ".m" and are often called "M-files".

Select File>New>Script from the menu bar. This will open an editing window where you can enter the script. Type the following:

```
% script to compute the energy in 4*exp(-j*pi/4*n) from n = 0 to 7.
n = 0:7;
x = 4*exp(-1j*pi/4*n);
x_mag = x.*conj(x);
E = sum(x_mag);
```

Save the file with an appropriate name and enter the following:

```
>>dir
```

You should see the script file in the list. If not, you will need to copy the script file to the default directory in MATLAB (or change your "Current folder" to the location of the file). Next run the script by typing the name at the command line prompt.

```
<name_of_your_file>
E
```

10. *What is the result?*

Finally, we can create a function to perform this operation. Select File>New>Function. A new editing window will open and you should see the following:

```
function [ output_args ] = Untitled( input_args )
%UNTITLED Summary of this function goes here
%   Detailed explanation goes here

end
```

Edit this text with the following:

```
function E = energy(input_sequence)
mag_input = input_sequence.*conj(input_sequence);

E = sum(mag_input);
end
```

and save after editing. Let's test the function:

```
>>n1 = 0;
>>n2 = 7;
>>n = n1:n2;
>>x = 4*exp(-1i*pi/4*n);
>>E = energy(x)
```

11. *What is the result?*