Lab 1: MATLAB Basics

INTRODUCTION

The goal of this lab is to give you a quick start with MATLAB. MATLAB is a powerful computing environment for numeric computation and visualization. It is fun using it, since it is interactive and intuitive. Don't think of learning MATLAB as learning one more programming language. You will master it quickly and it will become second nature.

After this lab, you will be able to write simple MATLAB scripts and functions. You will learn how to express discrete-time signals in MATLAB. You will also learn how to make your code run faster by avoiding loops in your code. Then you will exercise writing a function simulating an echo effect.

GRADING

Each lab is divided into two sections: Pre-Lab (4 points) and In-Lab (6 points). For the Pre-Lab component, 4 points will be awarded for the hand-in answer sheet (must be turned in at when coming to the lab at 1:25 PM.) For the In-Lab component, in addition to showing your results (**running MATLAB scripts, NOT typing individual commands**) to the TA, you also need to be ready to explain your answers to the TA, and be prepared to answer questions.

TIPS FOR A SUCCESSFUL EXPERIENCE IN LAB

- Always complete your pre-lab. Besides earning you credit, this helps prepare you for the tasks in the lab and saves time. Hand it in at the beginning of lab.
- Feel free to work on the lab outside of lab time but remember that you need to attend lab to get your work checked in order to get full credit.
- Always label plots: x-axis, y-axis, title, legend. A plot is meaningless if it is not labeled!
- Always use script and/or function files. Not only does it save you time from reentering commands, but it also helps you (and others) organize and debug.
- Feel free to discuss the labs with your classmates but work should be individually done.

Section 1: ***Pre-Lab***

When you come to the lab, please hand-in your answer sheet only to the problems in this section that ask you to "comment" on some code.

Read the Syllabus on "Instructions for Labs."

Open MATLAB and you will see a prompt like this: >>
Let's start by typing `a=2` and press the Enter key

```
>> a=2
a =
   2
```

You don't have to declare a variable. The basic data type in MATLAB is a matrix of double precision numbers. Think of a variable `a` as a 1 by 1 matrix. Variable names are case-sensitive: `c` is different than `C`.

```
>> b=[1 2 3]
b =
   1   2   3

>> b=[1, 2, 3]
b =
   1   2   3

>> c=[2 3 5]'
c =
   2
   3
   5
```

You noticed the `'` operator, which means *transpose*. `b` is a 1 by 3 matrix, while `c` is a 3 by 1 matrix, so you cannot do certain operations on them:

```
>> b+c
??? Error using ==> +
Matrix dimensions must agree.
```

However, you can do this (why?):

```
>> b*c
ans =
  23
```

How about `c*b`?
We will often refer to a 1 by N matrix as a *row vector*, and N by 1 matrices as *column vectors*.

Be very careful about this next point: MATLAB variables are matrices whose elements are referred to by indexes starting with 1 (not 0 as is usual in most languages). Thus:

```
>> a = [1 3 5];
>> a(3)
ans =
5
```

Type **`whos`** and you will see how much memory you have consumed:

```
>> whos
 Name   Size           Bytes Class

 a     1x1             8 double array
 ans   1x1             8 double array
 b     1x3             24 double array
 c     3x1             24 double array

Grand total is 8 elements using 64 bytes
```

<u>Briefly (no more than one line) comment (on the pre-lab sheet)</u> every line of the following (say what each line is doing, or write down the outcome):

```
a=zeros(1,5)

b=ones(3,2)

c=size(a)

abs([-5.2 , 3])

floor(3.6)

help floor

d=[1:-3.5:-9]

f=d(2)

g=sin(pi/2)
```

```
h=exp(1.0)

K=[1.4, 2.3; 5.1, 7.8]

m=K(1,2)

n=K(:,2)

comp = 3+4i

real(comp)

imag(comp)

abs(comp)

angle(comp)

disp('haha, MATLAB is fun')

j^2

4==4

2==8

3~=5
```

Next we will plot some graphs…

Graphs:

```
>> x=[1:1:5]
x =
   1   2   3   4   5
>> y=[3 5 7 6 8]
y =
   3   5   7   6   8
>> plot(x,y)
```

You should see a graph in a new window. Play around with the controls/settings you can find in that window.

<u>Comment each of the following code</u>:

```
figure

stem(x,y)

figure

plot(x,y,'r+')

xlabel('My x-axis')

ylabel('My y-axis')
```

You can copy and paste your graph into Microsoft Word. Select Edit→Copy Figure. Then paste it on MS Word. (Do not turn this question in. Just try it yourself.)

MATLAB Functions:
Besides working interactively with MATLAB by typing commands after the prompt, you can write functions or scripts. Here is a very simple function:

```
function output = myfunction(input)
output = input;
```

This function takes the input, and assigns the same matrix to the output. The input can be a matrix of any size.
Write this 2 line code in a MATLAB's M-editor (or any other text editor), save it as "myfunction.m". To call this function, type:

```
>> out = myfunction([5 4 3])
out =
   5   4   3
```

If MATLAB complains it cannot find this function, you have to make sure the function is under MATLAB's path. To edit the path, use commands such as `addpath` or `rmpath`, or launch an interface with `pathtool` and specify the folder you want to include in the path. The order of the paths is relevant, that is, if two functions from two different folders are both under the path, then the function from the first path will be called.

Section 2: ***In-Lab***

Some problems ask you to write down your answer. Some ask you to save your code in a file. Some ask you to copy and paste the result into Word. To avoid doing the problem again, make sure you know what to show the TA for each problem. You must do the problems sequentially, starting from the first problem.

1. **Without** using MATLAB, write down the result of each expression. Is it a column vector? Is it a row vector? Is it a scalar? Pay special attention to **:**, **;**, and `'`.
   (a) `a=[3 4 5]`
   (b) `b=[3:5]`
   (c) `c=[3:1:5]`
   (d) `d=[3:-1:5]`
   (e) `e=[3; 4; 5]`
   (f) `f=[3 4 5; 6 7 8; 9 1 2]`
   (g) `g=[5:-1:3]`
   (h) `h=zeros(4,1)`
   (i) `z=ones(1,4)`
   (j) `z=z'`
   (k) `k=[2:5:14]`
   (l) `y=a-k`
   (m) `m=a(2)`
   (n) `n=a(4)`   (if it is an error message, what does it say?)
   (o) `o=a-e`   (if it is an error message, what does it say?)
   (p) `p=length(a)`
   (q) `q=length(e)`
   (r) `r=size(a)`
   (s) `f(2,3)`
   (t) `f(6)`
   (u) `f(2:3,:)`
   (v) `inv(f)`
   (w) `f^2`
   (x) `f.^2`

2. How many data points are generated by the command `s=[0:0.1:1]?`
   (a) Estimate and write down the number **without** using MATLAB.
   (b) Verify your answer with MATLAB. Instead of counting one by one, use a MATLAB command you learned in Problem 1 to find out the number. There is more than one way to do it. Write down the command.

3. When you have questions with a command, you don't have to look it up in a book. For example, if you want to know about "if", simply type

   ```
   help if
   ```

   and read the help documentation. Notice that "if" is always accompanied with "end". Also read the help documentation for "for".
   Given a signal `x=[3 8 6 2 7 ]`, write code with **if** and **for** that will clip the elements larger than 5 to 5, and save it in y. In other words, the output y should be [3 5 5 2 5].
   Write the code in an editor and save it as "Lab1Prob3.m". Then you can run this code later by typing "Lab1Prob3" under MATLAB prompt as long as the file is under MATLAB's path. This is called a MATLAB *script*.
4. (a) Write a MATLAB function (by using for loop) that takes an input matrix and outputs a column vector that is a concatenation of each column vector in the input matrix. For example, if the input is [1 2; 3 4], the output should be [1;3;2;4]. Make sure it works with matrix of any size.
   (b) Do the same thing without using a loop
5. (a) Write a MATLAB function (by using for loop) that takes an input matrix and outputs a matrix of the same size but with reversed order columns. For example, if the input is [1 2;3 4], the output should be [2 1; 4 3]. Make sure it works with matrix of any size.
   (b) Write a MATLAB function (by using for loop) that takes an input matrix and outputs a matrix of the same size but with rows of reversed order. For example, if the input is [1 2;3 4], the output should be [3 4; 1 2]. Make sure it works with matrix of any size.
   (c) Do the same thing in (a) and (b) without using loop
   TA CHECK: Show your functions for problem 5 with matrices of size 2 and 5.

6. Using MATLAB, generate the column vector [1 4 9 16 … N*N]' where N = 1234500
   (a) by using for loop (compute each element by scalar multiplication within a for loop). Save it as Lab1Prob4.m.
   (b) without using a loop
   (c) Which one is faster? (timing can be performed with `tic` and `toc`) If you don't see a difference, use a larger N. This tells you to avoid for loops with proper MATLAB built-in commands for faster computation.

7. Using MATLAB, generate the following discrete-time signals.

   ```
   vect1 = [0 pi/4 2*pi/4 3*pi/4 4*pi/4 5*pi/4 6*pi/4
   7*pi/4]
   ```

```
vect2 = cos(vect1)
```

The input of many built-in functions can be vectors. Let `vect3` be the transpose of `vect1`. Generate `vect4 = cos(vect3)`. Is `vect4` the same as `vect2`? Can you compute `vect4 – vect2` using MATLAB? Why or why not?

8. As seen in problem 3, a discrete-time signal can be represented by a vector. In this problem, we will generate a discrete-time impulse. Create a row vector of length 100. Make the 5th element value one, and everywhere else value zero. Plot it and copy the result to Word.

9. Execute the following code one by one.

```
stem(1:5,[3 6 2 4 1])
plot(1:5,[3 6 2 4 1])
plot(1:5,[3 6 2 4 1],'r+')
```

If you feel uncomfortable with the display, try the **axis** command as follows:

```
axis([-1 10 0 7])
```

Explain briefly what it does.

10. Plot all the signals in problem 9 in a single window (figure) with the commands **`subplot`** and **`stem`** (for the first), and **`plot`** (for the second and third). Label each subplot with the command **`title`**. Type **`help subplot`** and **`help title`** if you need help. Write the code in an editor and save as “Lab1Prob8.m”. Copy the figure to Word.

11. Here we will practice writing a simple function. Write a function that takes as input a column vector, and generates as output a column vector that is the concatenation of the vector to itself. For example, if input is `[5 4 3]'`, then output is `[5 4 3 5 4 3]'`. If the input is not of size N by 1, display a message saying “This function works only for column vectors”. Name the function as L1Prob9 and save it as L1Prob9.m.
Test the function with a scalar, a 1 by 5 row vector, a 5 by 1 column vector, and a 5 by 5 matrix.

12. Please generate the following sample signals with 100 Hz sampling rates for 10s:
(a) $\cos(2\pi t)$
(b) $e^{-0.9t}\cos(2\pi t)$
(c) 1 Hz square wave with a 50% duty cycle

13. Given a discrete-time signal contained in a vector **`x`**, write a MATLAB function **`my_echo(x,d,a)`** to generate a vector that includes the original signal plus an echo occurring **`d`** seconds after the start of the original signal and having amplitude that is **`a`** times as large as that of the original one. The sampling frequency is fixed as 100 Hz, and the first sample occurs at time `t = 0`. You might need to concatenate the original signal and/or its delayed copy with zeros to avoid length mismatches. Remember that echoes are additive. Do not use any **for** or **while** loops. Name the function as "my_echo.m". Then execute

```
out=my_echo([1:15000]', 2400, 0.7);
plot(1:length(out),out)
```

and copy the graph to Word. Also run your echo function on each of the three sample signals from problem 12, plot the results, and copy the plots into Word. **TA CHECK: Show the results of running the echo function on the sample signals from problem 12. Discuss why this function can model an echo? What can be done to make it a better echo model, in other words, what might be lacking?**