Introduction to MATLAB Usage in Signals & Systems.

Aim: To study the various MATLAB commands:

1. Commands related to Basic Matrix Operations.
2. Commands related to signals and systems functions.
3. Commands related to controls functions.

Theory:

The name MATLAB stands for MATrix LABoratory. MATLAB is a high-performance language for technical computing. It integrates computation, visualization, and programming environment. Furthermore, MATLAB is a modern programming language environment: it has sophisticated data structures, contains built-in editing and debugging tools, and supports object-oriented programming. These factors make MATLAB an excellent tool for research. It is used extensively in signal processing and control engineering for analysis and design. There are many different toolboxes available which extend the basic functions of MATLAB into different application area. Follow the following tutorial step by step and complete the lab.

1. MATLAB Basics

MATLAB is started by clicking the mouse on the MATLAB desktop icon or my launching it from the start menu. Once MATLAB is launched, we typically see 2 windows pop up. One housing the command window and the other is the HELP window. The command window is where we can type '*MATLAB commands*' which will be explained in great details shortly. After each MATLAB command, the "return" or "enter" key must be depressed.

Definition of Variables

Variables are assigned numerical values by typing the expression directly, for example, typing:

```
>> a = 1+2
```

yields:

```
a =
     3
```

The answer will not be displayed when the semicolon is put at the end of an expression, for example, type:

```
>> a = 1+2;
```

Here the answer is not displayed once the return key is depressed; rather, MATLAB displays the prompt again and a blinking cursor indicates that the command was successfully executed and MATLAB is ready to accept the next command.

MATLAB utilizes the following arithmetic operators:

+ addition

- subtraction

* multiplication

/ division

^ power operator

' transpose

A variable can be assigned using a formula that utilizes these operators and either numbers or previously defined variables. For example, since the variable 'a' was defined previously, the following expression is valid

```
>> b = 2*a;
```

To determine the value of a previously defined quantity, type the variable by itself:

```
>> b
```

yields:

```
b =
     6
```

If your expression does not fit on one line, use an ellipsis (three or more periods at the end of the line) and continue on the next line.

```
>> c = 1+2+3+...
5+6+7;
```

There are several predefined variables which can be used at any time, in the same manner as user defined variables:

`i` sqrt(-1)

`j` sqrt(-1)

`pi` 3.1416...

For example,

```
>> y= 2*(1+4*j)
```

yields: `y=`

```
2.0000 + 8.0000i
```

There are also a number of predefined functions that can be used when defining a variable. Some common functions that are used are:

`abs` magnitude of a number (absolute value for real numbers)

`angle` angle of a complex number, in radians

`cos` cosine function, assumes argument is in radians

`sin` sine function, assumes argument is in radians

`exp` exponential function

For example, with `y` defined as above,

```
>> c = abs(y)
```

yields: `c =`

```
8.2462
```

```
>> c = angle(y)
```

yields: `c =`

```
1.3258
```

With `a`=3 as defined previously,

```
>> c = cos(a)
```

yields: `c =`

```
-0.9900
```

```
>> c = exp(a)
```

yields: `c =`

```
20.0855
```

Note that `exp` can be used on complex numbers. For example, with `y = 2+8i` as defined above,

```
>> c = exp(y)
```

yields: `c =`

```
-1.0751 + 7.3104i
```

which can be verified by using Euler's formula:

Definition of Matrices

MATLAB is based on matrix and vector algebra; even scalars are treated as 1x1 matrices. Therefore, vector and matrix operations are as simple as common calculator operations.

Matrix can be defined in two ways. The first method is used for arbitrary elements:

```
>> v = [1 3 5 7];
```

creates a 1x4 matrix with elements 1, 3, 5 and 7. Note that commas could have been used in place of spaces to separate the elements.

Additional elements can be added to the matrix:

```
>> v(5) = 8;
```

yields the matrix `v = [1 3 5 7 8]`. Previously defined vectors can be used to define a new

vector. For example, with `v` defined above

```
>> a = [9 10];
>> b = [v a];
```

creates the vector `b = [1 3 5 7 8 9 10]`.

The second method used for creating matrices' is by using the colon (:) operator:

```
>> t = 0:.1:10;
```

creates a 1x101 vector with the elements 0, .1, .2, .3,...,10. Note that the middle number defines the increment. If only two numbers are given, then the default increment used is 1:

```
>> k = 0:10;
```

creates a 1x11 vector with the elements 0, 1, 2, ..., 10.

Matrices are defined by entering the elements row by row:

```
>> M = [1 2 4; 3 6 8];
```

creates the 2x3 matrix

M = [1 2 4

3 6 8]

There are a number of special matrices that can be defined:

null matrix: `M = [];`

n x m matrix of zeros: `M = zeros(n,m);`

n x m matrix of ones: `M = ones(n,m);`

n x n identity matrix: `M = eye(n);`

A particular element of a matrix can be assigned:

```
>> M(1,2) = 5;
```

places the number 5 in the first row, second column.

Operations and functions that were defined for scalars in the previous section can also be used on matrices. For example,

```
>> a = [1 2 3];
>> b = [4 5 6];
>> c = a + b
```

yields:

```
c =
     5 7 9
```

Functions are applied element by element. For example,

```
>> t = 0:10;
>> x = cos(2*t);
```

creates a vector `x` with elements equal to cos(2t) where t = 0, 1, 2, ..., 10.

Operations that need to be performed element-by-element can be accomplished by preceding the operation by a ".". For example, to obtain a vector `x` that contains the elements of x(t) = tcos(t) at specific points in time, you cannot simply multiply the vector `t` with the vector `cos(t)`. Instead you multiply their elements together:

```
>> t = 0:10;
>> x = t.*cos(t);
```

General Information

Matlab is case sensitive so "`a`" and "`A`" are two different names.

Comment statements are preceded by a "`%`".

On-line help for MATLAB can be reached by typing `help` for the full menu or typing `help`

followed by a particular function name or M-file name. For example:

>> `help cos` gives help on the cosine function.

The number of digits displayed is not related to the accuracy. To change the format of the display, type `format short e` for scientific notation with 5 decimal places, `format long e` for scientific notation with 15 significant decimal places and `format bank` for placing two significant digits to the right of the decimal.

The commands `who` and `whos` give the names of the variables that have been defined in the

workspace. The command `length(x)` returns the length of a vector `x` and `size(x)` returns the dimension of the matrix `x`.

M-files

M-files are macros of MATLAB commands that are stored as ordinary text files with the extension "m", that is *filename*.m. An M-file can be either a function with input and output variables or a list of commands.

MATLAB requires that the M-file must be stored either in the working directory. The working directory is displayed right above the command window. All .m files must be stored here.

As example of an M-file that defines a function, create a file in your working directory named yplusx.m that contains the following commands:

```
function z = yplusx(y,x)
z = y + x;
```

Once the file is created and saved as yplusx.m, the following commands are typed from within MATLAB command window.

```
>> x = 2;
>> y = 3;
>> z = yplusx(y,x)
```

MATLAB M-files are most efficient when written in a way that utilizes matrix or vector operations.

Loops and if statements are available, but should be used sparingly since they are computationally inefficient.

An example of the use of the command `for` is

```
>>   for k=1:10,
x(k) = cos(k);
end
```

This creates a 1x10 vector x containing the cosine of the positive integers from 1 to 10. This operation is performed more efficiently with the commands

```
>> k = 1:10;
>> x = cos(k);
```

which utilizes a function of a vector instead of a for loop. An `if` statement can be used to define conditional statements. An example is

```
>> if(a <= 2),
b = 1;
elseif(a >=4)
b = 2;
else
b = 3;
end
```

The allowable comparisons between expressions are >=, <=, <, >, ==, and ~=.

EXERCISES

I) For these exercises, use the command window to enter your code and execute them. Record your results and also very briefly describe the functionality of the code in each case.

1) >> A = [1 3 4 5]; B = [2 6 7 8];

   >> W = A + B
   >> X = A - B
   >> Y = A .* B
   >> Z = A ./ B
   >> er = A * B

2) >> A = [1:3;4:6]; B = [1:2;1 2;3 -1];

   >> W = A * B
   >> X = B * A
   >> er = A .* B

3) >> t = [-9:0.1:9];
   >> y=cos(t);
   >> subplot(1,2,1); plot(y);

```
>> subplot(1,2,2) plot(t,y)
```

4)
```
>> x=[zeros(1,10) [0:0.1:1] cos(0:0.1:2*pi) zeros(1,10)];
>> plot(x)
```

II) Write a program (ex2.m) to assign the following expressions to a variable A and then to print out the value of A in the command window.

a) $(3+4)/(5+6)$ b) $2\pi^2$ c) $\sqrt{2}$

d) $(0.0000123 + 5.67\times10^{-3}) \times 0.4567\times10^{-4}$

III) Celsius temperatures can be converted to Fahrenheit by multiplying by 9, dividing by 5, and adding 32. Create a function "celtofahren(C)" that implements this formula. When the function is called from the command window, it should display the equivalent temperature in Fahrenheit.

IV) Set up a vector called N with five elements: 1, 2, 3, 4, 5. Using vector N, write MATLAB commands that will result in X having these values:

a) 2, 4, 6, 8, 10 b) 1/2, 1, 3/2, 2, 5/2

c) 1, 1/2, 1/3, 1/4, 1/5 d) 1, 1/4, 1/9, 1/16, 1/25

V) A supermarket conveyor belt holds an array of groceries. The price of each product (in Rs) is [6 12 5 13] ; while the quantity of each product is [3 2 1 5]. Use MATLAB to calculate the total bill.

VI) Input 2 matrices A and B. Check if both matrices have the same dimension. If yes, find the sum of the two matrices and display the result. If no, display an appropriate message.

[Hint: Use disp() function to display a message.]