EXPERIMENT # 1

MATLAB Basics for Communication System Design

Objective

- To understand the use of MATLAB for solving communication engineering problems.
- Learn the basics of MATLAB as used in Analogue Communication.
- To develop understanding of MATLAB environment, commands and syntax.

MATLAB

MATLAB is a powerful tool that is utilized by the engineers and others professionals in development and testing of various projects. It is versatile software, with the help of which you can solve and develop any sort of engineering problem. The name MATLAB stands for MATRIX LABORAORY. All the work done in MATLAB is basically in the form of matrices. Scalars are referred as 1-to-1 matrix and vectors are matrices having more than 1 row and column. MATLAB is programmable and have the same logical, relational, conditional and loop structures as in other programming languages, such as C, Java etc. It's very easy to use MATLAB, all we need is to practice it and become a friend of it.

Summary:

- **Scalars**
- **Vectors**
- **Matrices**
- **Plotting**
- **m-files**
- **functions**

Getting Started:

a) Go to the start button, then programs, MATLAB and then start MATLAB. It is preferred that you have MATLAB2015a. You can then start MATLAB by double clicking on its icon on Desktop, if there is any.

b) The Prompt:

```
>>
```

The operator shows above is the prompt in MATLAB. MATLAB is interactive language like C, Java etc. We can write the commands over here.

c) In MATLAB we can see our previous commands and instructions by pressing the up key. Press the key once to see the previous entry, twice to see the entry before that and so on. We can also edit the text by using forward and back-word keys.

Help in MATLAB

In order to use the built-in help of the MATLAB we use the **help** keyword. Write it on the prompt and see the output.

```
>> help sin
```

Also try

```
>> lookfor sin
```

Scalars

A scalar is a single number. A scalar is stored in the MATLAB as a 1 x 1 matrix. Try these on the prompt.

```
>> A = 2;

>> B = 3;

>> C = A^B

>> C = A*B
```

Try these instructions as well

```
>> C = A+B

>> C = A-B

>> C = A/B

>> C = A\B
```

Note the difference between last two instructions.

Try to implement these two relations and show the result in the provided space

a) $25\ (\ 3^{1/3}\ ) + 2\ (2+9^2) =$ _____

b) $5x^3 + 3x^2 + 5x + 14$ for $x = 3$ is _____

Vectors

Vectors are also called arrays in MATLAB. Vectors are declared in the following format.

```
>> X = [1 2 3 4]

>> Y = [2 5 8 9]
```

Try these two instructions in MATLAB and see the result

>> length (X) = _____

>> size (X) = _____

What is the difference between these two?

Try these instructions and see the results.

>> X.*Y = _____

>> X.^Y = _____

>> X+Y = _____

>> X-Y = _____

>> X./Y = _____

>> X' = _____

Also try some new instructions for this like and notice the outputs in each case.

>> ones (1,4)

>> ones (2,4)

>> ones (4,1)

>> zeros (1,4)

>> zeros (2,4)

There is an important operator, the colon operator (**:**), it is very important operator and frequently used during these labs. Try this one.

>> X = [0:0.1:1]

Notice the result. And now type this

>> length (X)

>> size (X)

What did the first and second number represent in the output of last instruction?

Now try this one.

>>A= [ones(1,3), [2:2:10], zeros(1,3)] What is the length and size of this?

>> Length = _____

Size = _____

Try '*help ones'* and '*help zeros'* as well, and note down its important features.

MATRICES

Try this and see the output.

>> A = [1 2 3;4 5 6;7 8 9]

>> B = [1,2,3;4,5,6;7,8,9]

Is there any difference between the two? Try to implement 2-to-3 matrix and 3-to-2 matrix.

Also take help on **mod, rem, det, inv** and **eye** and try to implement them. Try to use **length** and **size** commands with these matrices as well and see the results.

Try to solve these.

1. 6x + 12y + 4z = 70

   7x – 2y + 3z= 5

   2x + 8y -9z = 64

2. A = [2 3 4 5; 1 8 9 0; 2 3 1 3; 5 8 9 3]

   Solve $6A - 2I + A^2$ =

PLOTTING

Plotting is very important as we have to deal with various type of waves and we have to view them as well.

Try these and have a look on the results.

```
>> x = [0:0.1:10];
>> y = sin (x);
>> z = cos (x);
>> subplot (3,1,1);
>> plot (x,y);
>> grid on;
>> subplot (3,1,2);
>> plot (x,z);
>> grid on; hold on;
>> subplot (3,1,3);
```

```
>> stem (x,z);
>> grid on;
>> hold on;
>> subplot (3,1,3);
>> stem (x,y, ,'r');
```

Take **help** on the functions and commands that you don't know. See the difference between the **stem** and **plot**.

See help on plot, figure, grid, hold, subplot, stem and other features of it.

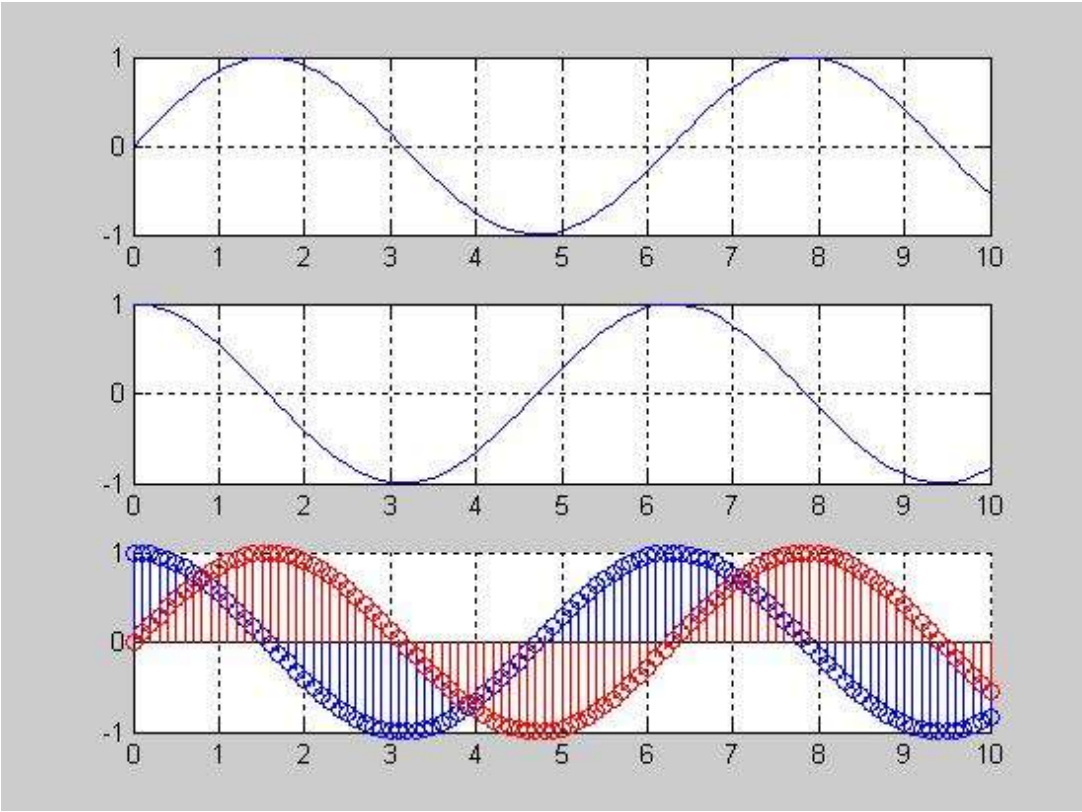

Figure 1.1

M-FILES

MATLAB can execute a sequence of statements stored in disk files. Such files are called M-files because they must have the file type '**.m**'. Lot of our work will be done with creation of m-files.

There are two types of m-files: Script and function files.

Script Files

We can use script files in order to write long programs such as one on the previous page. A script file may contain any command that can be entered on the prompt. Script files can have any name but they should be saved with ".m" extension. In order to excurse an m-file from the prompt, just type its name on the prompt. You can make an m-file by typing **edit** on the prompt or by clicking on the file then new and m-file. See an example of m-file. Write it and see the results.

```
% This is comment
% A comment begins with a percent symbol
% The text written in the comments is ignored by the MATLAB
% comments in your m-files.

clear;
clc;
x = [0:0.1:10];
y = sin (x);
subplot (2,2,1);
plot (x,y, ,'r');
grid on;
z = cos (x);
subplot (2,2,2);
plot (x,z);
grid on;
w = 90;
yy = 2*pi*sin
(x+w)
subplot (2,2,3);
plot (x,yy);
grid on;
zz = sin (x+2*w);
subplot (2,2,4);
stem (x,zz, ,'g');
hold on;
stem (x,y, ,'r');
grid on;
```

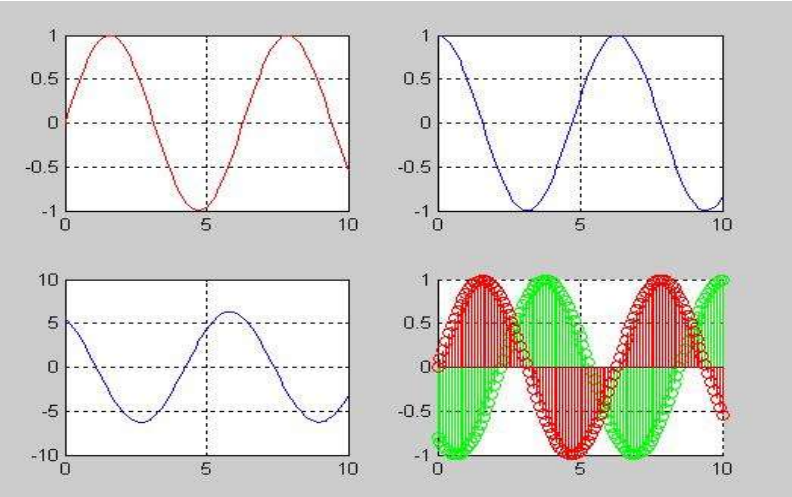

Figure 1.2

Function Files

MATLAB have many built-in functions including trigonometry, logarithm, calculus and hyperbolic functions etc. In addition we can define our own functions and we can use built-in functions in our functions files as well. The function files should be started with the function definition and should be saved with the name of function. The general format of the function file is

Function [output_variables] = function name (input_variables)

See the following example and implement it.

```
% this is a function file

% this function computes the factorial of a number function [y] = my_func (x)

y = factorial (x);
```

POST LAB

Discrete Time Sequences:

See the example below:

```
 % Example 2.2
% Generation of discrete time signals
n = [-5:5];
x = [0 0 1 1 -1 0 2 -2 3 0 -1];
stem (n,x);
axis ([-6 6 -3 3]);
xlabel ('n'); ylabel
('x[n]'); title
('Figure 2.2');
```

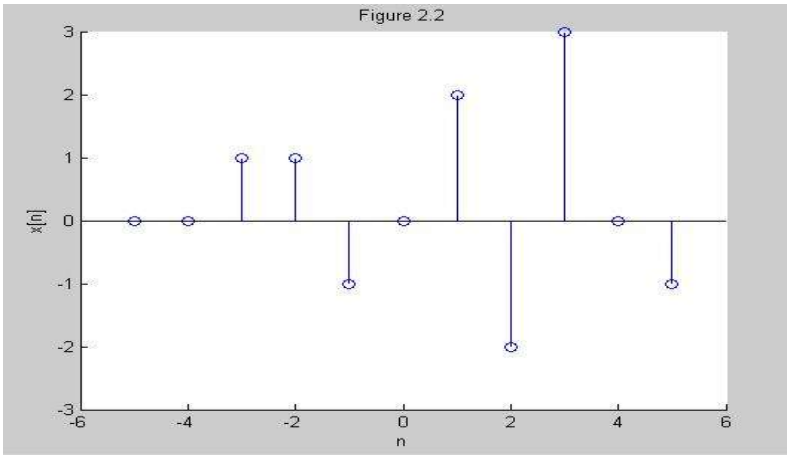

Figure 2.2

Unit Impulse Sequence:

A unit impulse sequence is defined as

Delta (n) = 1 n = 0

= 0 n ≠ 0

We are making a function named imseq and we further use this function in next experiments of this lab. The MATLAB code is given below:

```
function [x,n] = impseq(n0,n1,n2)

% Generates x(n) = delta (n-n0); n1<=n,n0 <= n2
% x[x,n] = imseq(n0,n1,n2)
% n0 = impulse position, n1 = starting index, n2 = ending index
    if ((n0 < n1) | (n0 > n2) | (n1 > n2))
        Error('arguments must satisfy n1 <= n0 <= n2')
        end
        n = [n1:n2];
        % x = [zeros(1,(n0-n1)),1,zeros(1,(n2-n0))];

        x = [(n-n0) == 0];
        stem(n,x)
```

Unit Step Sequence:

It is defined as

u(n) = 1 n ≥ 0

0 n ≤ 0

The MATLAB code for stem sequence function is given below:

```
function [x,n] = stepseq(n0,n1,n2)
% Generates x(n) = u(n-n0); n1 <= n, n0<=n2
% [x,n] = stepseq(n0,n1,n2)
if ((n0 < n1) | (n0 > n2) | (n1 > n2))
    error('arguments must satisfy n1 <= n0 <= n2')
end
n = [n1:n2];
% x = [zeros(1,(n0-n1)),ones(1,(n2-n0+1))];
x = [(n-n0) >= 0];
stem(n,x)
```

Real Valued Exponential Sequence:

It is define as:

x (n) = an, for all n; a € Real numbers

We require an array operator " **.^** " to implement a real exponential sequence. See the MATLAB code below

```
>> n = [0:10];
>>x = (0.9).^n;
```

Observe the result

Complex Valued Exponential Sequence:

It is define as:

x(n) = e (a + jb) n , for all n

Where **a** is called the attenuation and **b** is the frequency in radians. It can be implemented by following MATLAB script.

```
>> n = [0:10];
>> x = exp ((2+3j)*n);
```

Random Sequence:

Many practical sequences cannot be described by the mathematical expressions like above, these are called random sequences. In MATLAB two types of random sequences are available. See the code below:

```
>>rand (1,N)
```

>> randn (1,N)

The above instruction generates a length **N** random sequence whose elements are uniformly distributed between [0,1]. And the last instruction, **randn** generates a length **N** Gaussian random sequence with mean 0 and variance 1. Plot these sequences.

% example 2.3

```
%Generation of random sequence
n = [0:10];
x = rand (1, length (n));
y = randn (1, length (n));
plot (n,x) ;
grid on;
hold on;
plot(n,y,'r');
ylabel ('x & y')
xlabel ('n')
title ('Figure 2.3')
```

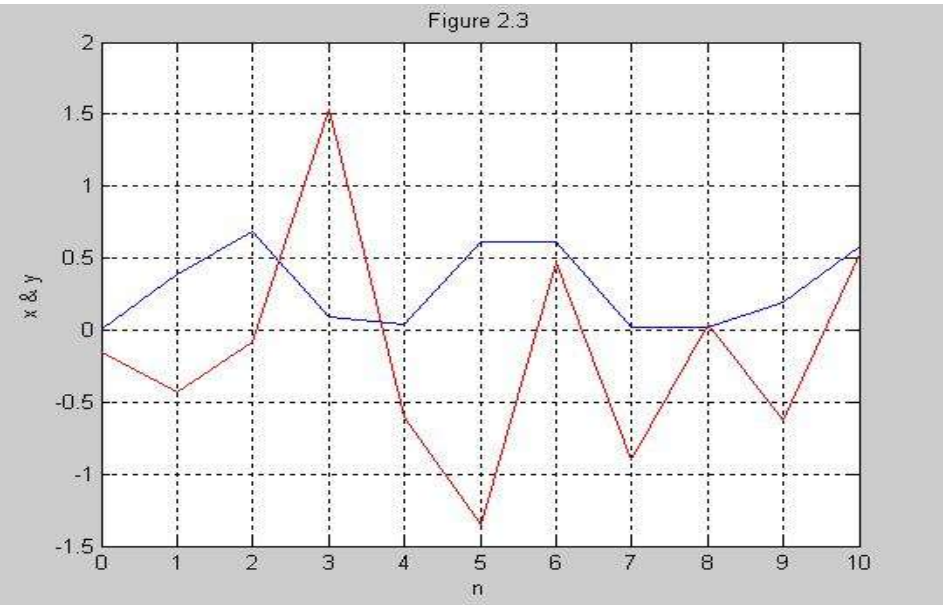

Figure 2.3

Periodic Sequences:

A sequence is periodic if it repeats itself after equal interval of time. The smallest interval is called the fundamental period. Implement code given below and see the periodicity.

% Example 2.4

```
% Generation of periodic sequences
```

```
n = [0:4];
x = [1 1 2 -1 0];
subplot (2,1,1);
stem (n,x);
grid on;
axis ([0 14 -1 2]);
xlabel ('n');
ylabel ('x(n)');
title ('Figure 2.4(a)');
xtilde = [x,x,x];
length_xtilde = length (xtilde);
n_new = [0:length_xtilde-1];
subplot (2,1,2);
stem (n_new,xtilde,'r');
grid on;
xlabel ('n');
ylabel ('perodic x(n)');
title ('Figure 2.4(b)');
```

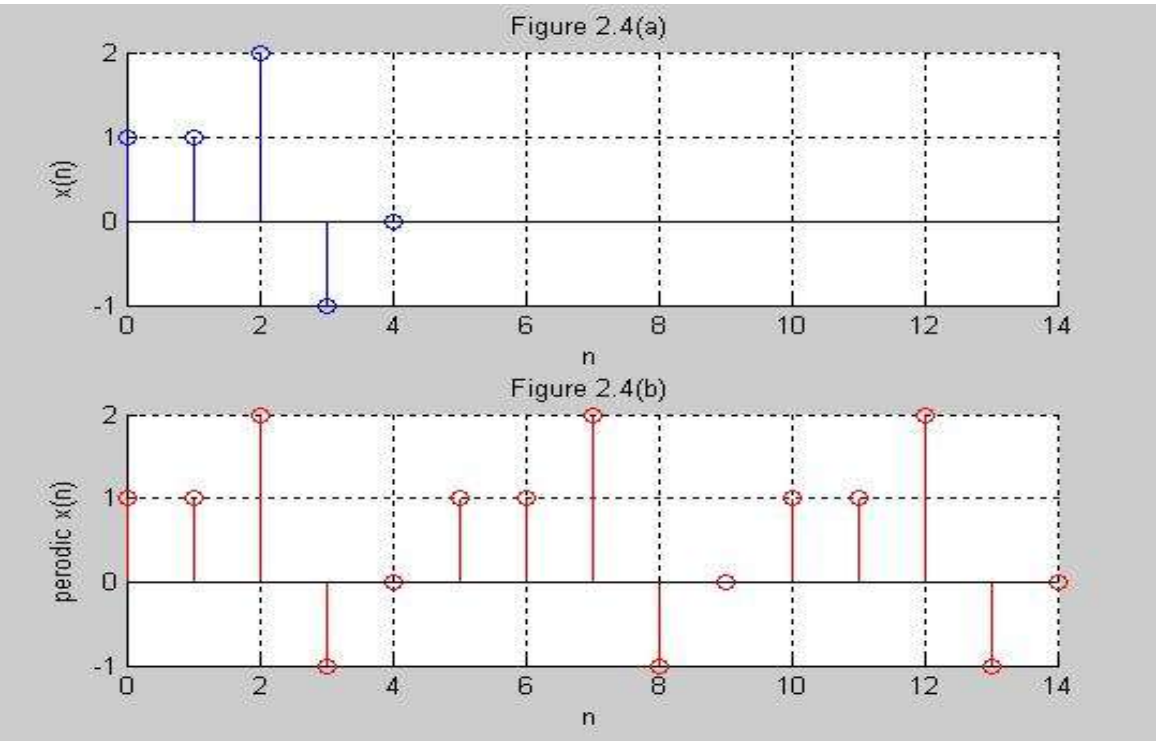

Figure 2.4

SIGNALS OPERATIONS:

Signal Addition

This is basically sample by sample addition. The definition is given below:

{x1(n)} + {x2(n)} = {x1(n) + x2(n)}