Department of Electrical and Computer Engineering
School of Engineering
Quchan University of Advanced Technologies

Quchan, Khorasan-Razavi, Iran

DIGITAL COMMUNICATIONS

LAB MANUAL-Part 1

Fall-2016

Prepared by: Dr. Hassan Khani

Date: October, 2016

Last Updated by: Dr. Khani

Date: August, 2016

Approved by the Head of Department: Dr. Hassan Khani

Date: August, 2016

Table of Contents

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 MATLAB7. 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

تمرین ۱) مقادیر عبارات زیر را با استفاده از نرم افزار مطلب محاسبه نمایید.

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

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

c) Solve this quadratic equation using quadratic formula.

تمرین ۲) ریشه های معادله درجه ۲ با ضرایب زیر را توسط نرم افزار مطلب محاسبه نمایید.

a = 2.5, b = 5, c = -6

x=_____ and _____

Vectors

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

تمرین ۳) دستورات زیر را در پنجره فرمان مطلب وارد کرده و جاهای خالی را پر کنید.

>> clear all; X = [1 2 3 4+j]

>> Y = [2 5 8 9]

Try these two instructions in MATLAB and see the results.
>> length (X) = _____

>> size (X) = _____

What is the difference between these two? تفاوت دو دستور بالا چیست؟

_____

Try these instructions and see the results.

>> X.*Y = _____

>> X.^Y = _____

>> X+Y = _____

>> X-Y = _____

>> X./Y = _____

>> X.\Y = _____

>> X′ = _____

>> X.′ =_____

>> exp(Y)=_____

>> sum(Y)=_____

>> 4+Y=_____

>> 4*Y=_____

>> 4./Y=_____

>>Y/5=_____

>> Y.^2+X.^3=_____

>> real(X)=_____

>> imag(X)=_____

>> abs(X)=_____

>> angle(X)=_____

>> conj(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?

عدد اول و دوم در خروجی آخرین دستور چه چیزی را مشخص می کنند؟

Now try this one.
>> A= [ones(1,3), [2:2:10], zeros(1,3)]

What is the length and size of this?

length(A) = _____

size(A) = _____

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. تمرین ۴) دستگاه معادلات زیر را با استفاده از روابط ماتریسی حل کنید.

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

   7x – 2y + 3z= 5

   2x + 8y -9z = 64

تمرین ۵) به ازای ماتریس داده شده حاصل عبارت ماتریسی زیرش را بیابید.

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

   $6A – 2I + A^2$ =

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;
>> 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.

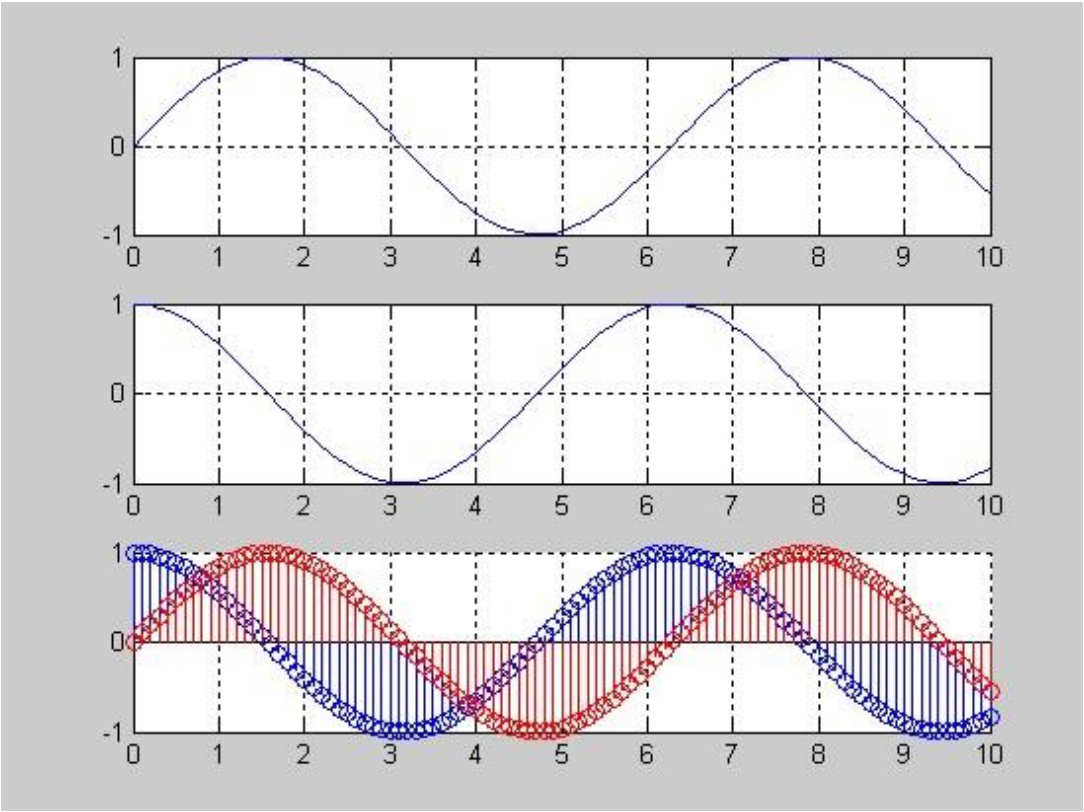


Figure 1.1

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
=pi/2;
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;
```

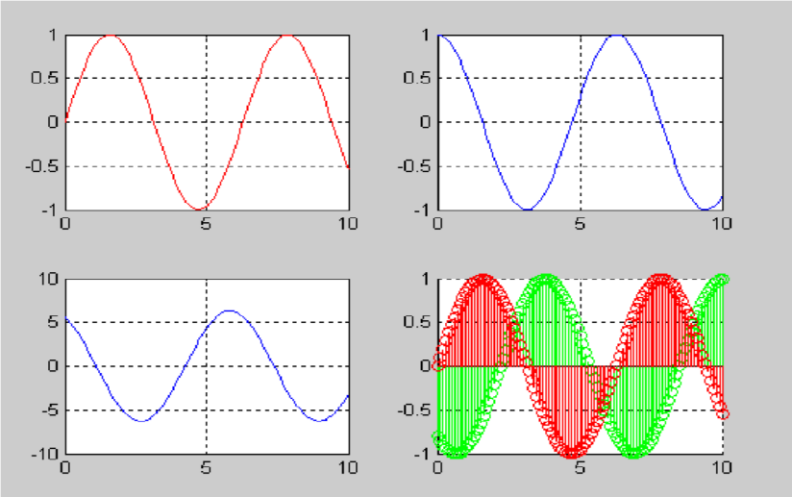


Figure 1.2

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

Try to develop a function that will compute the maximum and minimum of two numbers.

تمرین ۶) تابعی را ایجاد کنید که یک بردار را به عنوان ورودی گرفته و حاصل جمع تمامی عناصر و همچنین حاصل ضرب تمامی عناصر آنرا به ترتیب به خروجی اول و خروجی دوم ارسال نماید. (راهنمایی: از دستور length و حلقه for استفاده نمایید.)

تمرین ۷) تابعی ایجاد کنید که یک بردار را به عنوان ورودی گرفته و میانگین عناصر آنرا به خروجی می فرستد. (راهنمایی: از دستور length و حلقه for استفاده نمایید.)

تمرین ۸) دستور sum را بر روی چند ماتریس اعمال کرده و استنباط خود را از نتایج حاصل بیان کنید. با استفاده از استنباط به دست آمده بیان کنید برای محاسبه حاصل جمع تمامی عناصر ماتریس چگونه می توان از دستور sum استفاده نمود؟

Experiment # 2

Communication Signals: Generation and Interpretation

Objective

- To use MATLAB for generation of different signals important in communication theory.
- Learn the basics of signals and their operations as used in Analog Communication.
- To develop understanding of communication signals and their properties.

Generation of Signals

Signals are represented mathematically as a function of one or more independent variables. We will generally refer to the independent variable as time. Therefore we can say a signal is a function of time. Write these instructions in m-file as execute to see the result.

Sinusoidal Sequence:

```
% Example 2.1
% Generation of continuous time signals
% x(t)=2sin( 2πt-π/2)
t=[-5:0.01:5];
x=2*sin((2*pi*t)-(pi/2));
plot(t,x);
grid on;
axis([-6 6 -3 3])
ylabel ('x(t)')
xlabel ('Time(sec)')
title ('Figure 2.1')
```

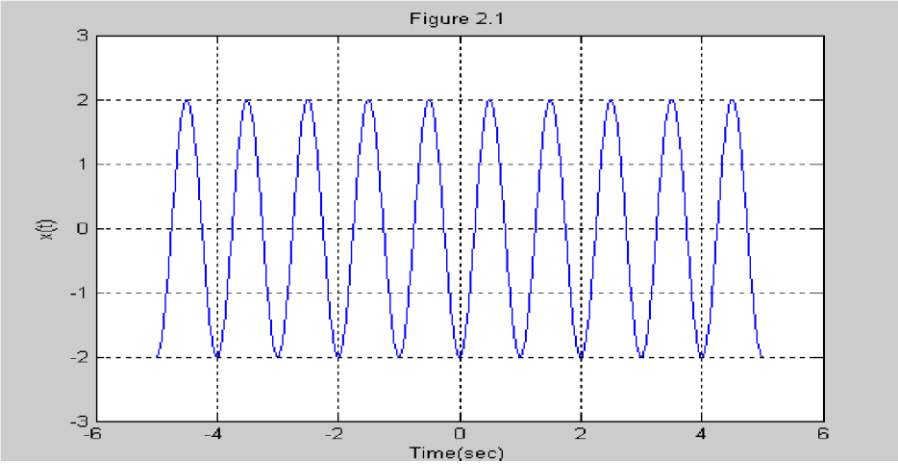


Figure 2.1

See the output, change the phase shift value and observe the differences.

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]; y=sign(n);
subplot(2,1,1);
stem (n,x);
axis ([-6 6 -3 3]);
xlabel ('n'); ylabel
('x[n]');
title ('Figure 2.2.1');
subplot(2,1,2);
stem (n,y);
axis ([-6 6 -3 3]);
xlabel ('n'); ylabel
('y[n]');
title ('Figure 2.2.2');
```

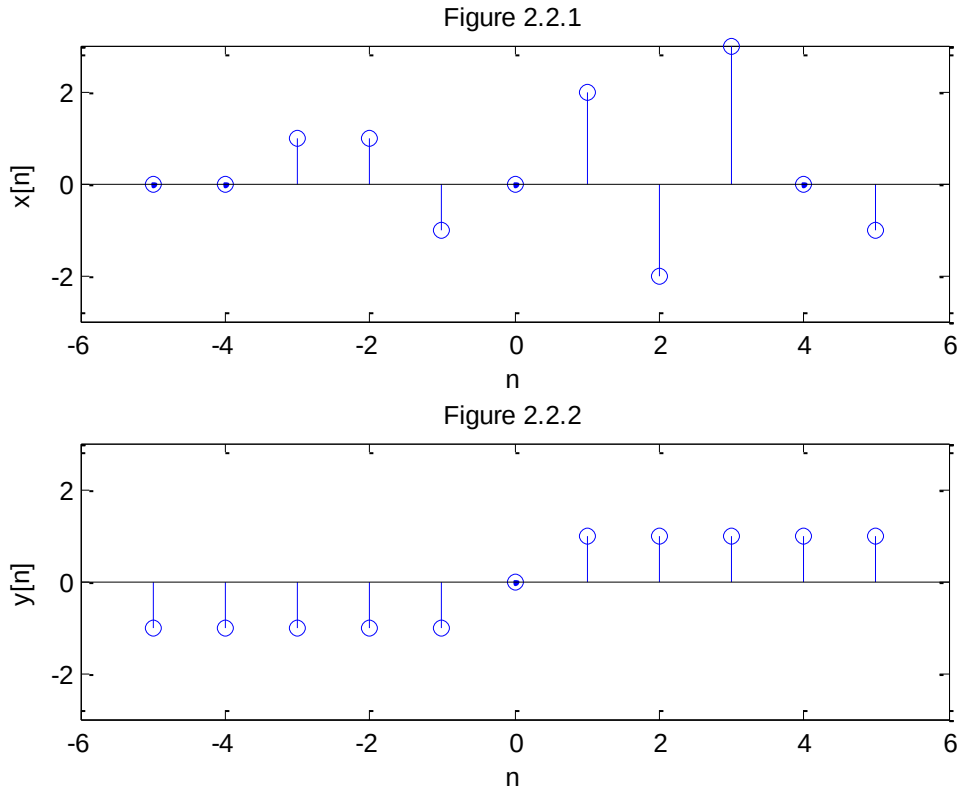


Figure 11.2

Impulse Sequence:

A unit impulse sequence is defined as:

$$\delta[n] = \begin{cases} 1, & n = 0 \\ 0, & n \neq 0 \end{cases}$$

We are making a function named impseq 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) = δ(n-n0); n1<=n,n0 <= n2
% [x,n] = impseq(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];
```

Unit Step Sequence:

It is defined as:

$$u[n] = \begin{cases} 1, & n \geq 0 \\ 0, & n < 0 \end{cases}$$

The MATLAB code for step 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 < z1) | (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];
```

Real Valued Exponential Sequence:

It is defined as:

$$x(n) = a^n, \text{ for all } n;\ a \in real\ numbers$$

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

```
n = [-5:10];
x = (0.9).^n;
x1=(-0.9).^n;
y=(1.2).^n;
y1=(-1.2).^n;
subplot(2,2,1);
stem (n,x);
axis ([-10 10 -3 3]);
xlabel ('n'); ylabel
('x[n]');
title ('Figure 2.2.1');
subplot(2,2,2);
stem (n,y);
axis ([-10 10 -8 8]);
xlabel ('n'); ylabel
('y[n]');
title ('Figure 2.2.2');
subplot(2,2,3);
stem (n,x1);
axis ([-10 10 -3 3]);
xlabel ('n'); ylabel
('x1[n]');
title ('Figure 2.2.3');
subplot(2,2,4);
stem (n,y1);
axis ([-10 10 -8 8]);
xlabel ('n'); ylabel
('y1[n]');
title ('Figure 2.2.4');
```

Observe the result.

Complex Valued Exponential Sequence:

It is define as:

$$x(n) = e^{(a + jb)\, n} \text{ , for all n}$$

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. They depend upon the parameters like probability density function or their statistical moments. In MATLAB two types of random sequences are available. See the code below:

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

And

```
>> 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')
```

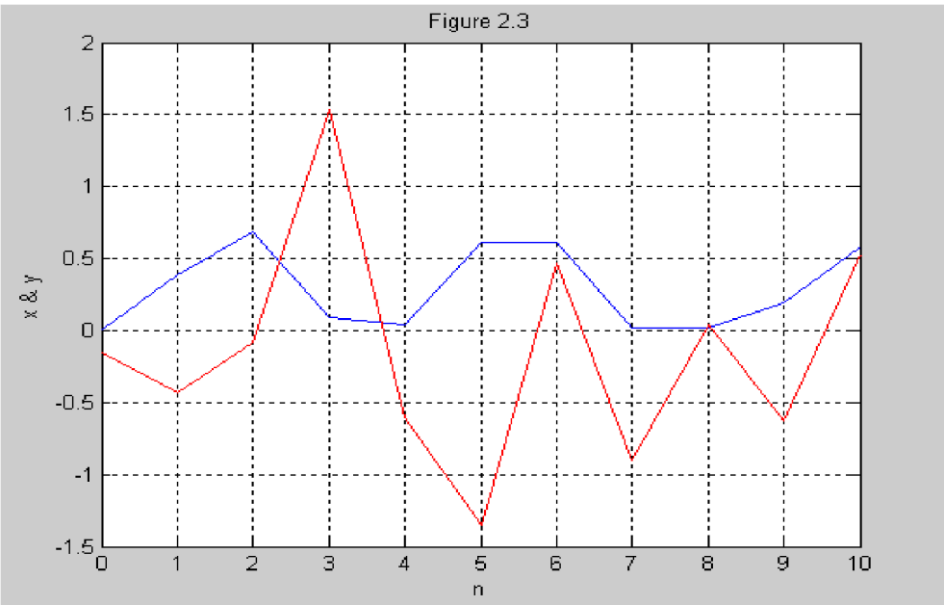


Figure 2.3

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)');
```

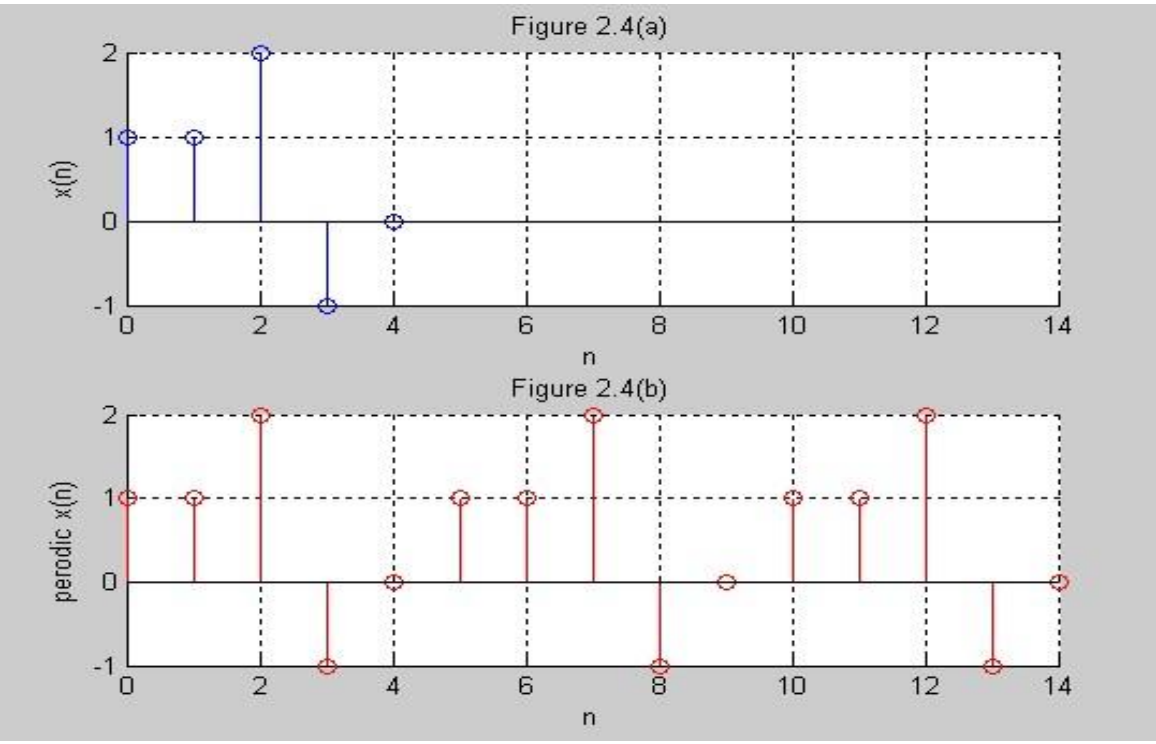


Figure 2.4

SIGNALS OPERATIONS:

Signal Addition

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

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

The length of x1 and x2 should be equal. See the MATLAB code below:

```
function [y,n] = sigadd(x1,n1,x2,n2) % implement y(n) = x1(n) + x2 (n)

% [y,n] = sigadd (x1,n1,x2,n2)

% y = sum sequence over n, which include n1 and n2

% x1 = first sequence over n1

% x2 = second sequence over n2 (n2 can be different from n1)

n = min(min(n1),min(n2)): max(max(n1),max(n2));       %duration of y(n)

y1 = zeros(1,length(n));                                    % initialization

y2 = y1;

y1(find((n>=min(n1))&(n<=max(n1))==1))=x1;              % x1 with duration of y

y2(find((n>=min(n2))&(n<=max(n2))==1))=x2;          % x2 with duration of y y =

y1 + y2;
```

See example of signal addition below

```
% Example 2.5
% signal addition using sigadd function
% [y,n] = sigadd(x1,n1,x2,n2)
clear;
clc;
n1 = [0:10];
x1 = sin (n1); n2 =
[-5:7];
x2 = 4*sin(n2);
[y,n] = sigadd(x1,n1,x2,n2);
subplot (3,1,1); stem
(n1,x1);
grid on;
axis ([-5 10 -5 5]);
xlabel ('n1');
```

```
ylabel ('x1(n)');
title ('1st signal');
subplot (3,1,2);
stem (n2,x2);
grid on;  hold on;
axis ([-5 10 -5 5]);
xlabel ('n2');
ylabel ('x2(n)');
title ('2nd signal');
subplot (3,1,3);
stem (n,y,'r');
grid on;
axis ([-5 10 -5 5]);
xlabel ('n');
ylabel ('y(n)');
title ('Added Signals');
```

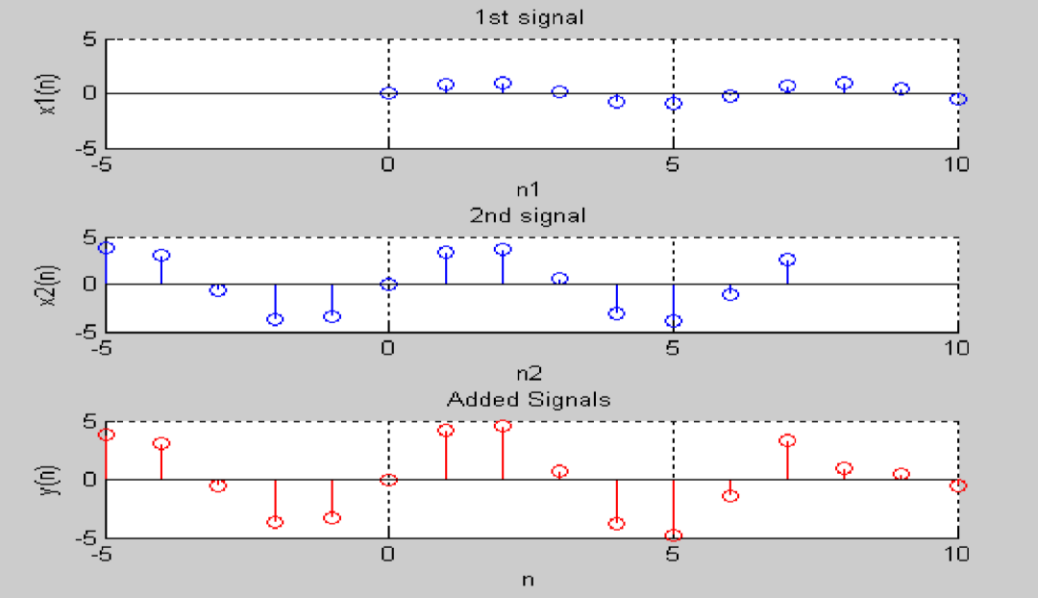


Figure 2.5

Signal Multiplication:

The multiplication of two signals is basically sample by sample multiplication or you can say dot multiplication. By definition it is

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

It is implemented by the array operator ' **.*** ' that we studied in last lab. A signal multiplication function is developed that is similar to the sigadd function. See the code below:

```
function [y,n] = sigmult (x1,n1,x2,n2)
% implement y(n) = x1(n) * x2 (n)
% [y,n] = sigmult (x1,n1,x2,n2)
% y = product sequence over n, which include n1 and n2
% x1 = first sequence over n1
% x2 = second sequence over n2 (n2 can be different from n1)
n = min(min(n1),min(n2)):max(max(n1),max(n2));   %duration of y(n)
y1 = zeros(1,length(n));                          % initialization
y2 = y1;

y1(find((n>=min(n1))&(n<=max(n1))==1))=x1;          % x1 with duration of y
y2(find((n>=min(n2))&(n<=max(n2))==1))=x2;          % x2 with duration of y y =
y1 .* y2;
```

See the example below:

```
% Example 2.6
% signal multiplication using sigmult function
clear;
clc;
n1 = [0:10];
x1 = sin (n1); n2 =
[-5:7];
x2 = 4*sin (n2);
[y,n] = sigmult(x1,n1,x2,n2);
subplot (3,1,1);
stem (n1,x1);
grid on;
axis ([-5 10 -5 5]);
xlabel ('n1');
ylabel ('x1(n)');
title ('1st signal');
```

```
subplot (3,1,2); stem (n2,x2);
grid on; hold on;
axis ([-5 10 -5 5]);
xlabel ('n2'); ylabel ('x2(n)');
title ('2nd signal');
subplot (3,1,3); stem (n,y,'r');
grid on;
axis ([-5 10 -5 5]);
xlabel ('n');
ylabel ('y(n)');
title ('Multiplied Signals');
```

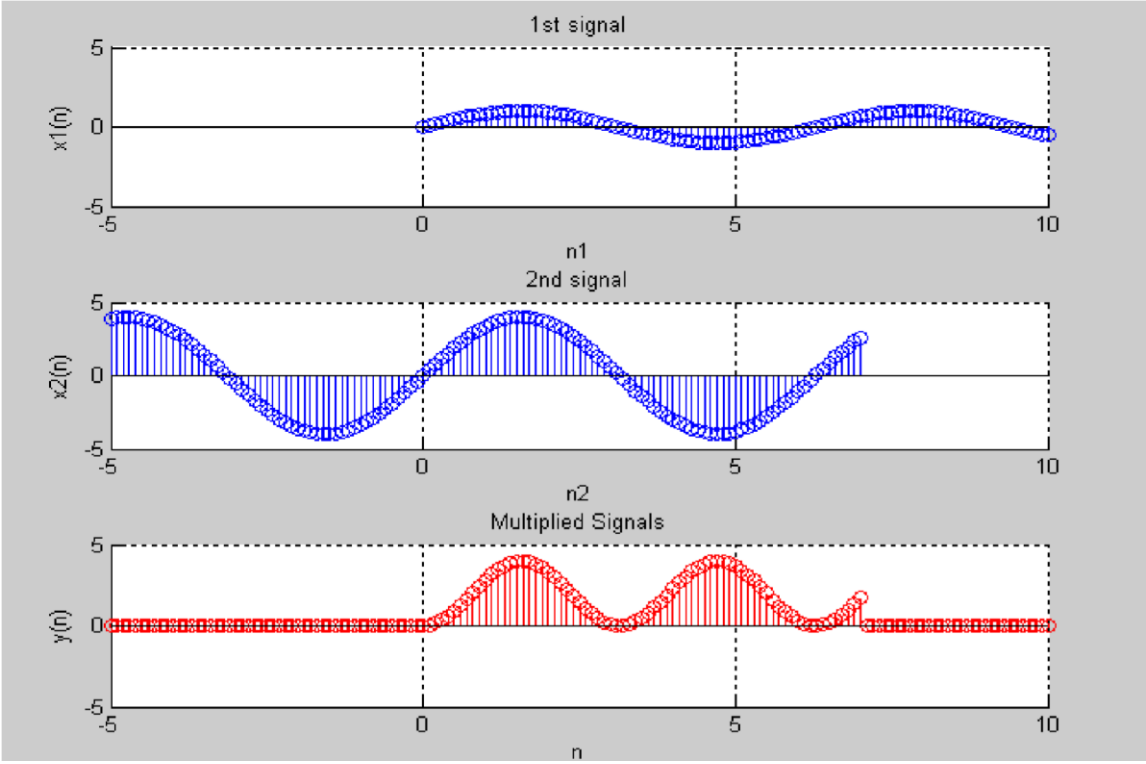


Figure 2.6

POST LAB

Write MATLAB code to plot these signals:

a. $x[n] = 2\sin(3n) + 2\cos(3n)$

b. $x[n] = u[n] + 4\cos(3n)$

c. $x[n] = n[u(n) - u(n-10)] + 10e^{-0.3(n-10)}[u(n-10)-u(n-20)]$

You are not allowed to multiply impulse sequences with a number. Implement this by using **impseq**, **stepseq** and **sigadd** functions.

Experiment # 3

Communication Signals: Operations

Objective

- To learn the use of MATLAB for different operations on signals.
- To develop a thorough understanding of communication signals and their basic operations as used in Analogue Communication.

SUMMARY

- Signal operations (Scaling, Shifting, Folding, Sample Summation, Sample product, Energy, Even and Odd sequences, Convolution)

SIGNAL OPERATIONS:

1. Scaling:

In this operation the samples of the signal is multiplied by a scalar α. The mathematical operator ***** is used for the implementation of the scaling property.

α{x(n)} = { α x(n)}

```
>> [x,n] = stepseq (-1,-5,5);

>> a = 2;

>> y = a*x;

>> subplot (2,1,1);

>>stem (n,x);grid on;

>> subplot (2,1,2);

>> stem (n,y, 'r');

 >> grid on;
```

2. Shifting

In this operation, each sample of the signal is shifted by **k** to get a shifted signal. By definition:

y(n) = {x (n-k)}

In this operation there is no change in the array or vector x, that contains the samples of the signal. Only n is changed be adding **k** to each element. The function is given below:

```
function [y,n] = sigshift (x,m,n0) % implement y(n) = x(n-n0)
% x = samples of original signal
% m = index values of the signal
% n0 = shift amount , may be positive or negative
% [y,n] = sigshift(x,m,n0) n = m+n0;
y = x;
```

See the example of above function:

```
% Example 3.1
% This example demonstrate the use of stepseq, sigshift, sidadd & sigmult function
clc;
clear;
%----------------------------------------
[x,n] = stepseq (0,-10,10);
subplot (3,2,1);
stem (n,x);
axis ([-12 12 0 3]);
grid on;
xlabel ('n');
ylabel ('x(n)');
title ('Original Signals');
%----------------------------------------
[y1,n1] = sigshift(x,n,2.5);
subplot (3,2,2);
stem (n1,y1);
axis ([-12 12 0 3]);
grid on;
```

```
xlabel ('n');

ylabel ('y1(n)');

title ('Shifted signal,x(n-2.5)');

%-----------------------------------------------

[y2,n2] = sigshift (x,n,-2.5);
subplot (3,2,4);

stem (n2,y2);

axis ([-12 12 0 3]);

grid on;

xlabel ('n');

ylabel ('y2(n)');

title ('Shifted signal,x(n+2.5)');

   %-----------------------------------------------

 [y_add,n_add] = sigadd(y1,n1,y2,n2);
subplot (3,2,6);

stem (n_add,y_add,'r');

axis ([-12 12 0 3]);

grid on;

xlabel ('n');

ylabel ('y1(n) + y2(n)');

title ('Added Signal');

   %-----------------------------------------------

[y_mul,n_mul] = sigmult(y1,n1,y2,n2);
subplot (3,2,5);
stem (n_mul,y_mul,'k');
axis ([-12 12 0 3]);
grid on;
xlabel ('n');
ylabel ('y1(n) * y2(n)');
title ('Multiplied Signal');
%-----------------------------------------------
```

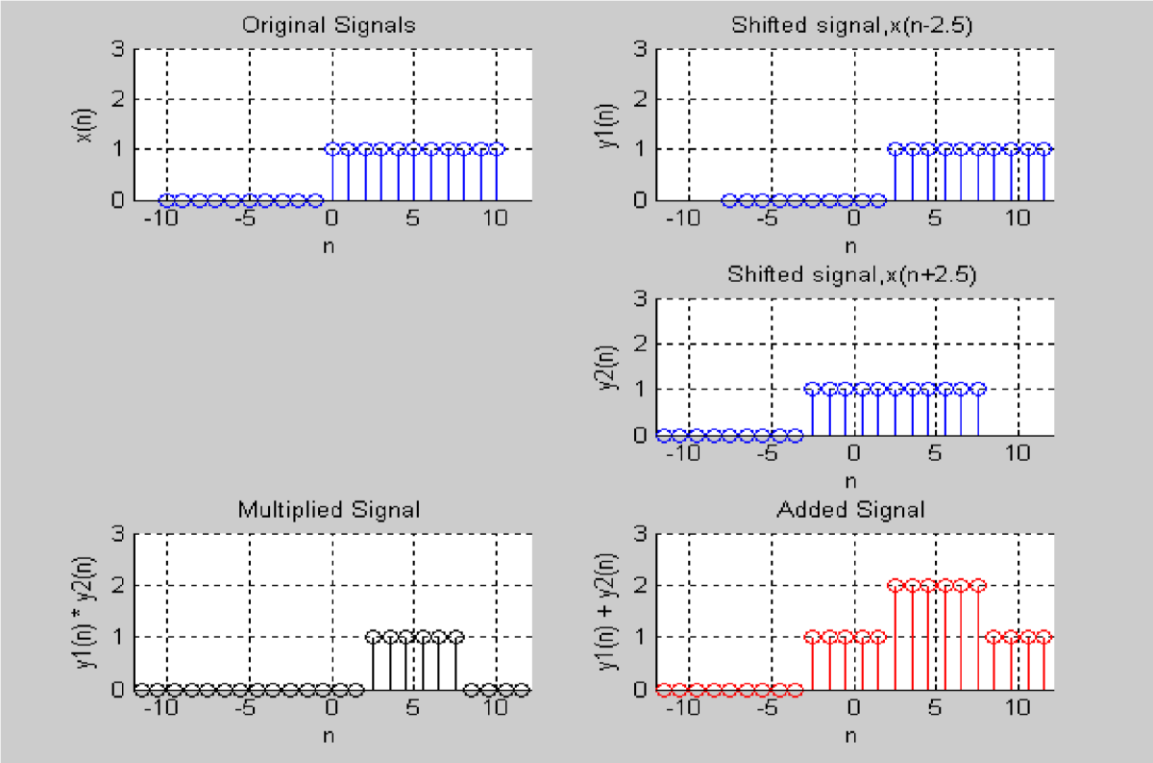


Figure 12.1

3. Folding:

In this operation each sample of **x(n)** is flipped around **n=0** to obtain a folded signal **y(n)**.

$$y(n) = \{x(-n)\}$$

In MATLAB, this function is implemented by using a built-in function **fliplr(x)** and **–fliplr(x).** Its implementation is given below:

```
function [y,n] = sigfold(x,n)

% implements y(n) = x(-n)

% [y,n] = sigfold(x,n)

% x = samples of the original signal  %
n = indexes of the original signal y =
fliplr(x); n = -fliplr (n);
```

Do its example by yourself from any example signals.

4. Sample Summation:

This operation is different from **sigadd** function. In this operation we add all the sample values of any signal **x(n)** between any two of its index values. By definition

$$\sum \mathbf{x(n) = x(n1) +\ldots\ldots\ldots+x(n2)}$$

In MATLAB it is implemented by the **sum(x(n1:n2))** command. See the code below for the demonstration of above function.

```
>> [x,n] = stepseq (5,0,10)

>> sum(x(2:7))
```

5. Sample Product:

This operation also differs from the **sigmult** function. It implies the sample values over the range **n1:n2**. It is implemented by the **prod(x(n1:n2))**. See the code below.

```
>> x = [0 1 2 3 4 5]

>> prod(x(2:5))
```

6. Energy:

The energy of any signal x is computed by the mathematical relation:

$$\mathbf{E_x = \sum x(n)\, x^*(n) = \sum |\, x(n)\, |^2}$$

Where the subscript ***** is used for complex conjugate of the signal x. The energy of the finite duration signal is computed in MATLAB as.

```
>> Ex = sum (x.*conj(x));
```

Or

```
>> Ex = sum (abs(x).^2);
```

7. Even and Odd Sequence:

A real valued sequence $x_e(n)$ is called even if the following condition satisfies.

$$\mathbf{x_e(-n) = x_e(n)}$$

Similarly a signal is said to be an odd signal if

$$\mathbf{x_o(-n) = -x_o(n)}$$

See the example below:

```
% example 3.2

% Generation of even and odd signals
n1 = [0:0.01:1];
x1 = 2*n1;
```

```
n2 = [1:0.01:2];

x2 = -2*n2+4;

n = [n1,n2];

x = [x1,x2];

%Even Signal

[xe,ne] = sigfold(x,n); %Plotting of original signal

subplot (3,1,1);

plot (n,x);

axis ([-4 4 0 2.5]);

grid on;

%Plotting of original signal + even signal

subplot (3,1,2);

plot (n,x/2,ne,xe/2);

axis ([-4 4 0 2.5]);

grid on;

% Plotting of original signal + odd signal
xo = -xe;
no = ne;
subplot (3,1,3);
plot (n,x/2,no,xo/2);
axis ([-4 4 -2.5 2.5]);
grid on;
```

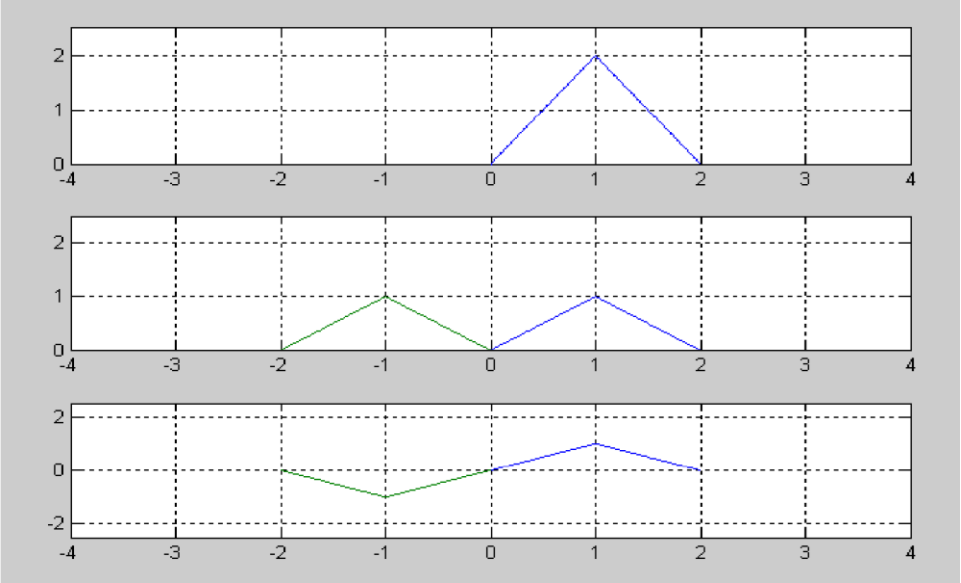


Figure 12.2

The above example shows to develop the even and odd signals from a given signal. Now we are going to develop a function to compute the even and odd signals for ourselves. See the code of function file below:

```
function [xe,xo,m] = evenodd (x,n)

% Decomposes a real function into its even and odd parts

% [xe,xo,m] = evenodd(x,n)

% xe = even signal

% xo = odd signal

% m = indexes

% x = original signal

% n = indexes for original signal

if any(imag(x)~=0)

error('x is not a real sequence')

end

m = -fliplr(n);

m1 = min([m,n]); m2
= max([m,n]); m =
m1:m2;
```

```
nm = n(1)-m(1);

n1 = 1:length(n);

x1 = zeros(1,length(m));
x1(n1+nm) = x;

x = x1;

xe = 0.5*(x+fliplr(x));

xo = 0.5*(x-fliplr(x));
```

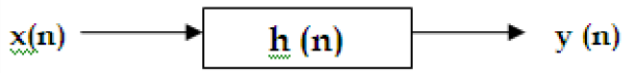
Now change the example 3.2 code to implement the same example with this function.

8. Convolution:

The convolution is very important operation as far the system as their impulse responses are concern. It is mathematically defines as:

y (n) = x(n) * h(n)

Where **h(n)** is the impulse response of the system. The above definition is best depicted by the following diagram.



In MATLAB convolution is implemented by the following instructions.

```
>> x = [1 5 3 9 1 2 3 8 5 -3 0 4];

>> h = [1 0 2 3];

>> y = conv(x,h);
```

A function is developed which will evaluate convolution in a more precise form and also calculate the indexes to help us plot the sequences.

```
function [y,ny] = conv_m(x,nx,h,nh)

% Modified convolution routine for signal processing

% [y,ny] = conv_m(x,nx,h,nh)

% [y,ny] = convolution result

% x = original signal

% nx = index values

% h = impulse response signal

% nh = index values for impulse response
```

```
nyb = nx(1) + nh(1);
nye = nx(length(x)) + nh(length(h)); ny
= [nyb:nye];

y = conv(x,h);
```

POST LAB

a. x(n) = u(n) – u(n-5). Decompose into even and odd components and plot them.

b. The impulse response of LTI system is h (n) = delta (n-2), if the input to this system is a arbitrary sequence x(n) of length 10, then plot the original and the convolved outputs of the system. What is the change if the h(n) = x(n) and input signal is now the previous impulse response of the system.

c. n = [-2:2]; x1 = [3,2,1,-2,-3]; x2 = [1,1,1,1,1];

Implement y = x1*x2

Appendix A: Lab Evaluation Criteria

1. Experiments and their reports		50%
a. Experiment	60%	
b. Lab report	40%	
3. Final evaluation		50%
a. Project Implementation	60%	
b. Project report and quiz	40%	

Notice:

Copying and plagiarism of lab reports is a serious academic misconduct. First instance of copying may entail ZERO in that experiment. Second instance of copying may be reported to DC. This may result in awarding FAIL in the lab course.

Appendix B: Safety around Electricity

In all the Electrical Engineering (EE) labs, with an aim to prevent any unforeseen accidents during conduct of lab experiments, following preventive measures and safe practices shall be adopted:

- Remember that the voltage of the electricity and the available electrical current in EE labs has enough power to cause death/injury by electrocution. It is around 50V/10 mA that the "cannot let go" level is reached. "The key to survival is to decrease our exposure to energized circuits."
- If a person touches an energized bare wire or faulty equipment while grounded, electricity will instantly pass through the body to the ground, causing a harmful, potentially fatal, shock.
- Each circuit must be protected by a fuse or circuit breaker that will blow or "trip" when its safe carrying capacity is surpassed. If a fuse blows or circuit breaker trips repeatedly while in normal use (not overloaded), check for shorts and other faults in the line or devices. Do not resume use until the trouble is fixed.
- It is hazardous to overload electrical circuits by using extension cords and multi-plug outlets. Use extension cords only when necessary and make sure they are heavy enough for the job. Avoid creating an "octopus" by inserting several plugs into a multi-plug outlet connected to a single wall outlet. Extension cords should ONLY be used on a temporary basis in situations where fixed wiring is not feasible.
- Dimmed lights, reduced output from heaters and poor monitor pictures are all symptoms of an overloaded circuit. Keep the total load at any one time safely below maximum capacity.
- If wires are exposed, they may cause a shock to a person who comes into contact with them. Cords should not be hung on nails, run over or wrapped around objects, knotted or twisted. This may break the wire or insulation. Short circuits are usually caused by bare wires touching due to breakdown of insulation. Electrical tape or any other kind of tape is not adequate for insulation!
- Electrical cords should be examined visually before use for external defects such as: Fraying (worn out) and exposed wiring, loose parts, deformed or missing parts, damage to outer jacket or insulation, evidence of internal damage such as pinched or crushed outer jacket. If any defects are found the electric cords should be removed from service immediately.
- Pull the plug not the cord. Pulling the cord could break a wire, causing a short circuit.
- Plug your heavy current consuming or any other large appliances into an outlet that is not shared with other appliances. Do not tamper with fuses as this is a potential fire hazard. Do not overload circuits as this may cause the wires to heat and ignite insulation or other combustibles.
- Keep lab equipment properly cleaned and maintained.
- Ensure lamps are free from contact with flammable material. Always use lights bulbs with the recommended wattage for your lamp and equipment.
- Be aware of the odor of burning plastic or wire.
- ALWAYS follow the manufacturer recommendations when using or installing new lab equipment. Wiring installations should always be made by a licensed electrician or other qualified person. All electrical lab equipment should have the label of a testing laboratory.
- Be aware of missing ground prong and outlet cover, pinched wires, damaged casings on electrical outlets.
- Inform Lab engineer / Lab assistant of any failure of safety preventive measures and safe practices as soon you notice it. Be alert and proceed with caution at all times in the laboratory.
- Conduct yourself in a responsible manner at all times in the EE Labs.

- Follow all written and verbal instructions carefully. If you do not understand a direction or part of a procedure, ASK YOUR LAB ENGINEER / LAB ASSISTANT BEFORE PROCEEDING WITH THE ACTIVITY.
- Never work alone in the laboratory. No student may work in EE Labs without the presence of the Lab engineer / Lab assistant.
- Perform only those experiments authorized by your teacher. Carefully follow all instructions, both written and oral. Unauthorized experiments are not allowed.
- Be prepared for your work in the EE Labs. Read all procedures thoroughly before entering the laboratory. Never fool around in the laboratory. Horseplay, practical jokes, and pranks are dangerous and prohibited.
- Always work in a well-ventilated area.
- Observe good housekeeping practices. Work areas should be kept clean and tidy at all times.
- Experiments must be personally monitored at all times. Do not wander around the room, distract other students, startle other students or interfere with the laboratory experiments of others.
- Dress properly during a laboratory activity. Long hair, dangling jewelry, and loose or baggy clothing are a hazard in the laboratory. Long hair must be tied back, and dangling jewelry and baggy clothing must be secured. Shoes must completely cover the foot.
- Know the locations and operating procedures of all safety equipment including fire extinguisher. Know what to do if there is a fire during a lab period; “Turn off equipment, if possible and exit EE lab immediately.”

Appendix C: Guidelines on Preparing Lab Reports

Each student will maintain a lab notebook for each lab course. He will write a report for each experiment he performs in his notebook. A format has been developed for writing these lab reports.

Lab Report Format

1. **Introduction**: Introduce area explored in the experiment.
2. **Objective**: What are the learning goals of the experiment?
3. **Design/Measurements**: In your own words write how the experiment is performed. Include the circuit diagram with explanation.
4. **Issues**: Technical issues which were faced during the performance of the experiment and how they were resolved?
5. **Conclusions**: What conclusions can be drawn from experiment?
6. **Application**: Suggest a real world application where this exercise may apply.
7. Answers to post lab questions (if any).

Sample Lab Report:

Introduction

An RC circuit is a first order circuit that utilizes a capacitor as an energy storage element whereas a resistor as an energy wastage element. RC circuits are building blocks of electronic devices and their thorough understanding is important in comprehending advance engineering systems such as transistors and transmission lines.

An RC circuit can be operated with both DC and AC sources. In this lab we study transient response of RC circuits with a square wave as a DC source. During the DC operation of an RC circuit the voltage across the capacitor or the resistor show energy storing (capacitor charging) and dissipating (capacitor discharging via resistor) mechanisms of the circuit. The capacitor charging or discharging curves then lead to determine time constant of the circuit where the time constant signifies time required by the RC circuit to store or waste energy.

Objective:

To study transient response of a series RC circuit

Measurements:

The circuit used for the experiment is shown in Fig. 1.

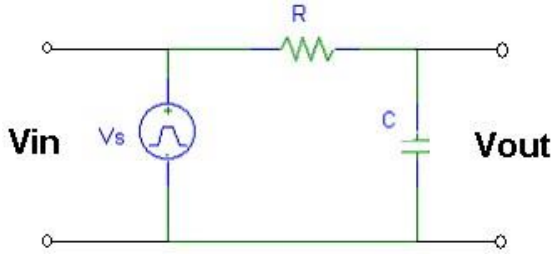


Fig.1. Circuit used in the experiment

Both input (a square wave) and output (voltage across capacitor) waveforms are monitored on an oscilloscope. The capacitor charging is observed during "on" part of the square waveform whereas the capacitor discharging is observed during "off" part of the square waveform (Fig. 2). We measure the time constant from the capacitor charging or discharging curve. While keeping the capacitor value constant, we also measure time constants with various resistor values (Table I).

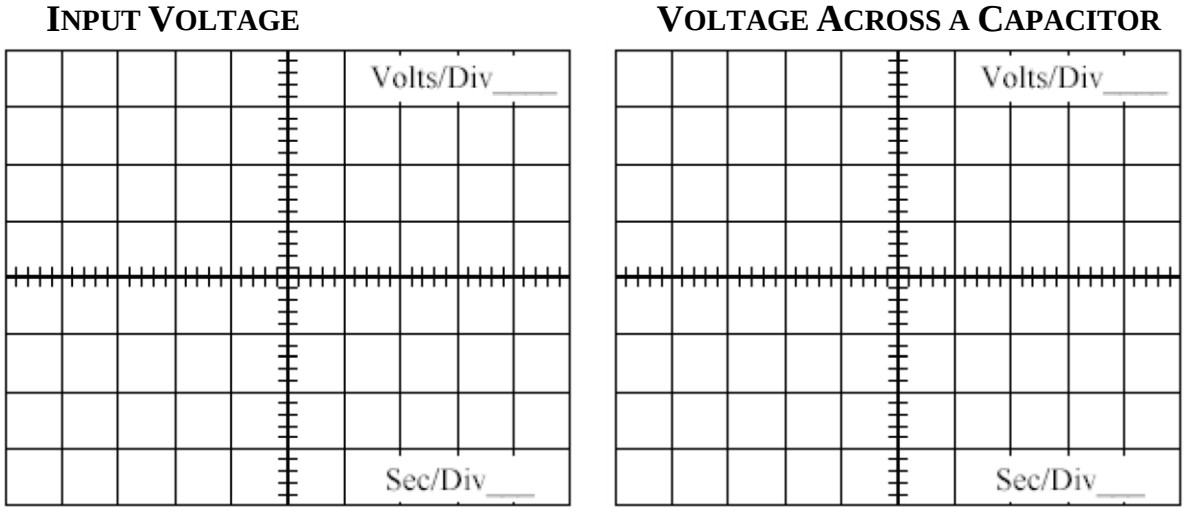


Fig. 2. Input and Output waveforms

TABLE I. Time constant as a function of the resistor values

Resistance (Nominal)	270	330	470	1 k	2.2 k	3.3 k
Time constant (Calculated)						
Time constant (Measured)						

Conclusions:

From the measurements following conclusions can be drawn:

a) The capacitor charging and discharging curves are exponential.
b) The time constant is directly proportional to the resistor value.

Both of the above conclusions are also easily verifiable by solving differential equation for the RC circuit.

Applications:

An RC circuit can be employed for a camera flash. The capacitor discharges through the flash light during a picture taking event.