***DSP LAB***

***Experiment 1 (PLOTTING)***

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);

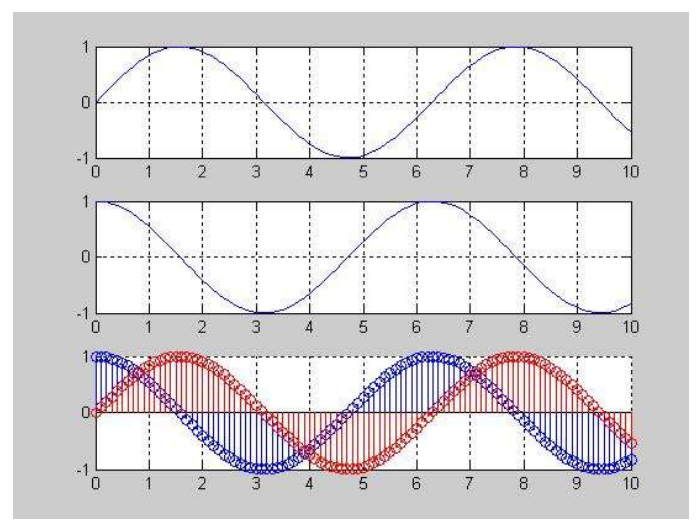
stem (x,z);

grid on;

hold on;

subplot (3,1,3);

stem (x,y, ,'r');



***Experiment 2 (Generating a Signal)***

% Generation of discrete time signals

% 2sin(2πτ-π/2)

T = [-5:0.01:5];

x=2\*sin((2\*pi\*t) - (pi/2));

plot(t,x)

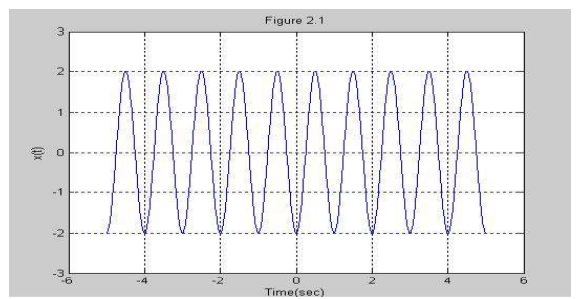
grid on;

axis ([-6 6 -3 3])

ylabel ('x(t)')

xlabel ('Time(sec)')

title ('Figure 2.1')



***Experiment 3 (Generating a Signal)***

% Generation of discrete time signals

n = [-5:5];

x = [0 0 1 1 -1 0 2 -2 3 0 -1];

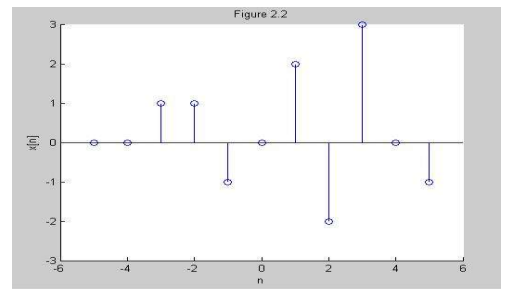
stem (n,x);

axis ([-6 6 -3 3]);

xlabel ('n'); ylabel

('x[n]'); title

('Figure 2.2');



***Experiment 4 (Generating a Signal)***

%Generation of random sequence

n = [0:10];

x = rand (1, length (n));

y = randn (1, length (n));

plot (n,x) ;

grid on;

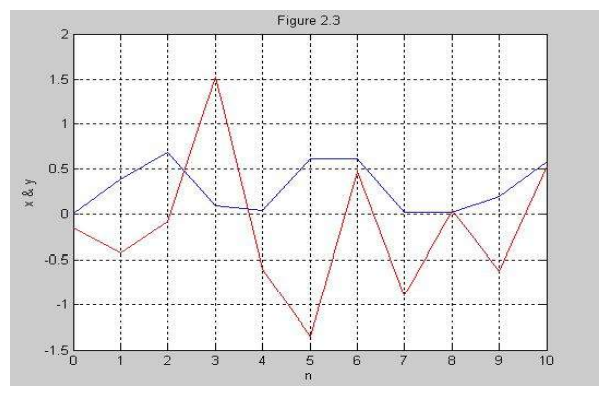
hold on;

plot(n,y,'r');

ylabel ('x & y')

xlabel ('n')

title ('Figure 2.3');



***Experiment 5 (Generating a discrete periodic signal Signal)***

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);

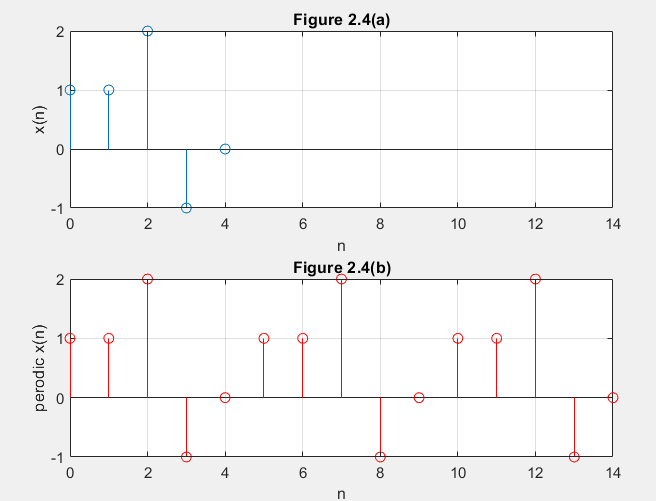
stem (n\_new, xtilde,'r');

grid on;

xlabel ('n');

ylabel ('perodic x(n)');

title ('Figure 2.4(b)');



***Experiment 6 (Generating Square wave) using loop***

clear;

clc;

n = input ('Insert the value of odd n:');

t = 0:.001:1;

sum=0;

for f=1:2:n

w=sin(2\*pi\*f\*t);

sum=sum+w;

end

subplot(1,1,1)

plot(t,sum)

grid on;

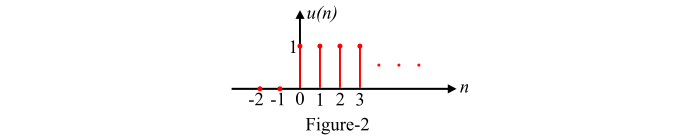
***Experiment 7 (Generating Unit Step Discrete Time Signal)***

**Experiment Name:** Generating and Plotting Unit Step Discrete Time Signal.

**Discrete Time Unit Step Signal:**

It is denoted by u[n]. Mathematically, the discrete-time unit step signal or sequence u[n] is defined as follows –

The graphical representation of the discrete-time unit step signal u[n] is shown in the following figure:



%Generating and Plotting Unit Step Discrete Time Signal.

clc; %clears the command window

clear all; %clears the current variables which are being used

close all; %close programs that are running behind in MATLAB

N=input('Enter the range: ');

n=-N:1:N;

y= [zeros(1,N),1,ones(1,N)];

stem(n,y);

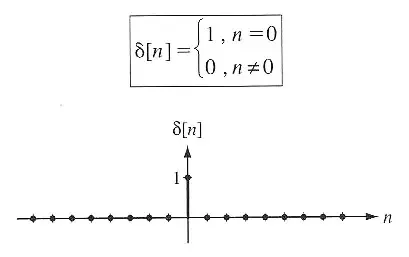
axis([-(N+1) N+1 -0.5 1.5]); % [-x x -y y]

xlabel('Time');

ylabel('Amplitude of Y');

title('Generating Unit Step Function');

***Experiment 8 :*** Unit impulse signal is mathematically defined as,



Code of implementation of impulse signal in Matlab:

%Code of implementation of impulse signal in Matlab:

clc;

clear all;

m1=input('enter the value of x-axis in negative side:');

m2=input('enter the value of x-axis in positive side:');

n=m1:m2;

x=(n==0);%it works as if statement like n=-5:5( 0 0 0 0 0 1 0 0 0 0 0 0)

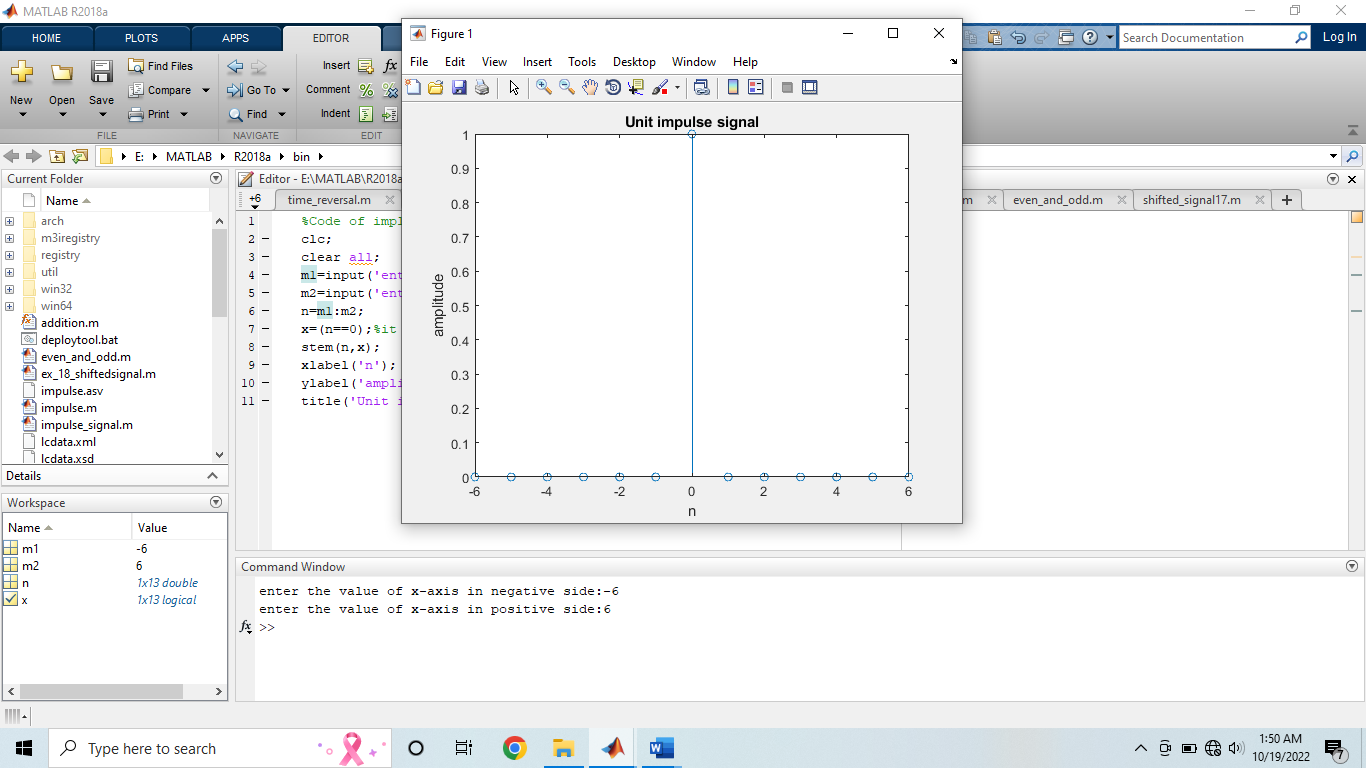
stem(n,x);

xlabel('n');

ylabel('amplitude');

title('Unit impulse signal');

**Output:**

****

The unit impulse can be implemented in different way:

clc;

clear all;

close all;

m1=input('enter the value of x-axis in negative side:');

m2=input('enter the value of x-axis in positive side:');

n=-m1:m2;

d=[zeros(1,m1) 1 zeros(1,m2)];

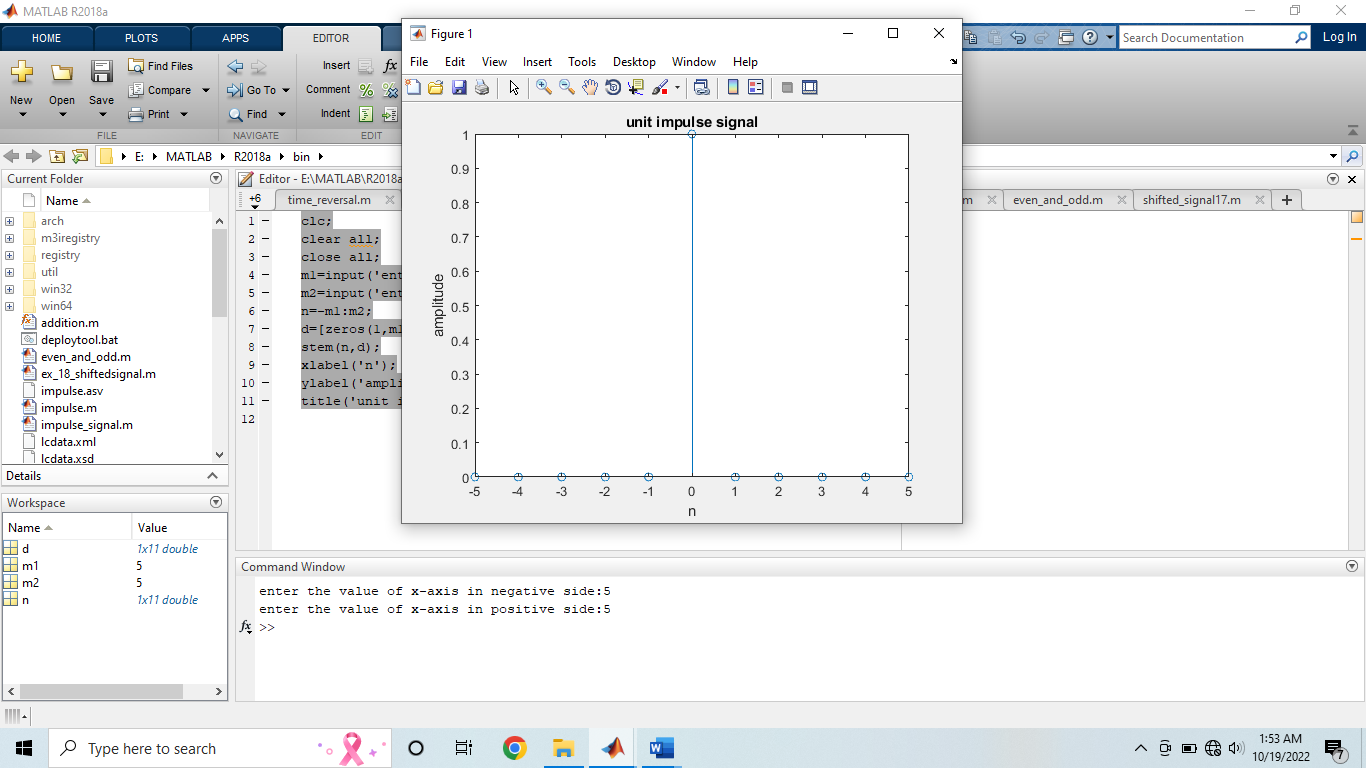
stem(n,d);

xlabel('n');

ylabel('amplitude');

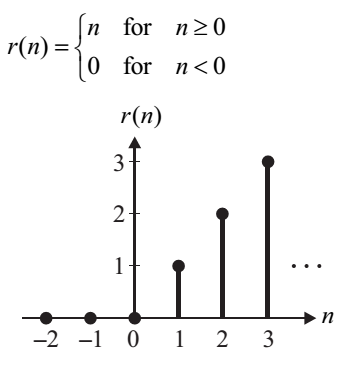
title('unit impulse signal');

**Output:**

****

**Experiment 9** Generating and plotting ramp discrete time signal.

The discrete time unit ramp signal is that function which starts from n = 0 and increases linearly. It is denoted by r(n). It is signal whose amplitude varies linearly with time n. mathematically; the discrete time unit ramp sequence is defined as –



**Code:**

close all;

clear all;

clc;

n1= input ('Enter lower limit');

n2= input ('Enter upper limit');

n= n1: 1: n2;

x=n.\*[n>=0];

stem (n, x, 'b');

axis([(n1-1) (n2+1) -1 (n2+1)]); % -x,x,-y,y

title (' Ramp Function ');

xlabel ('time');

ylabel ('Amplitude of Y');

**Input:**

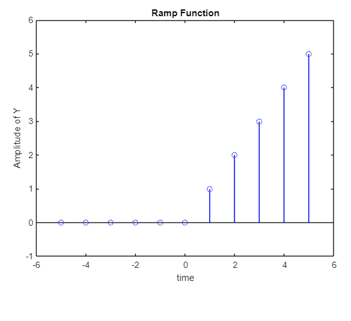
Enter lower limit

-5

Enter upper limit

5

**Output:**



**Experiment 10 (**Time reversal using a discrete sinusoidal function [use of fliplr( ) and values of x-axis(angle) in radian)

%Time reversal using a function (sinusoidal function angle in radian)

close all

clc

t1=0:0.2:2\*pi; %values of x-axis in radian

x1=sin(t1); %values of y-axis

x2=fliplr(x1); %fliplr() -> this function gives the flipped result;

%lr means left right ...flipud() ud means up down

t2= -fliplr(t1); % time values must be flipped and negated

subplot(2,1,1)

stem(t1,x1,'LineWidth',2)

xlim([-10 10])

title('\bf\fontsize{25}Original Signal')

xlabel('\bf\fontsize{20}Samples')

ylabel('\bf\fontsize{20}Amplitude')

grid on;

ax = gca;

ax.XAxis.FontSize = 15;

ax.XAxis.FontWeight = 'bold';

ax.YAxis.FontSize = 15;

ax.YAxis.FontWeight = 'bold';

subplot(212)

stem(t2,x2,'LineWidth',2)

xlim([-10 10])

title('\bf\fontsize{25}Time Reversed Signal')

xlabel('\bf\fontsize{20}Samples')

ylabel('\bf\fontsize{20}Amplitude')

grid on;

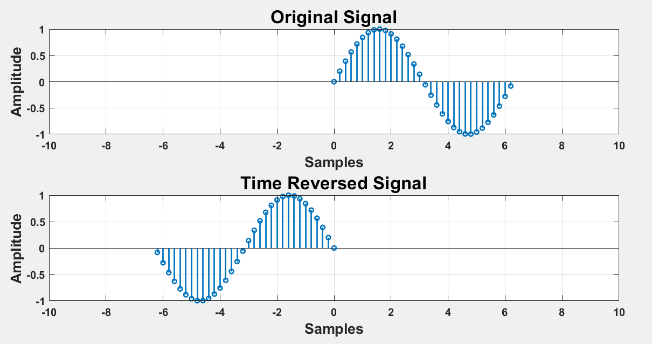
ax = gca;

ax.XAxis.FontSize = 15;

ax.XAxis.FontWeight = 'bold';

ax.YAxis.FontSize = 15;

ax.YAxis.FontWeight = 'bold';



**Experiment 11** Time reversal using a discrete sinusoidal function [use of fliplr()and values of x-axis(angle) in degree]

%Time reversal using a function (sinusoidal function angle in degree)

close all

clc

t1=0:10:360; %values of x-axis in degree

x1=sind(t1); % values of y axis

x2=fliplr(x1); %fliplr() -> this function gives the flippefd result;

%lr means left right ...flipud() ud means up down

t2= -fliplr(t1); % time values must be flipped and negated

subplot(211)

stem(t1,x1,'LineWidth',2)

xlim([-400 400])

ylim([-1.5 1.5])

title('\bf\fontsize{25}Original Signal')

xlabel('\bf\fontsize{20}Samples')

ylabel('\bf\fontsize{20}Amplitude')

grid on;

ax = gca;

ax.XAxis.FontSize = 15;

ax.XAxis.FontWeight = 'bold';

ax.YAxis.FontSize = 15;

ax.YAxis.FontWeight = 'bold';

subplot(212)

stem(t2,x2,'LineWidth',2)

xlim([-400 400])

ylim([-1.5 1.5])

title('\bf\fontsize{25}Time Reversed Signal')

xlabel('\bf\fontsize{20}Samples')

ylabel('\bf\fontsize{20}Amplitude')

grid on;

ax = gca;

ax.XAxis.FontSize = 15;

ax.XAxis.FontWeight = 'bold';

ax.YAxis.FontSize = 15;

ax.YAxis.FontWeight = 'bold';

***Experiment 12 (Signal Addition)***

**addition.m ->**

clear all;

clc;

x1=[-5 -4 -3 -2 -1 0];

y1=[2 5 4 6 3 5];

x2=[-2 -1 0 1 2];

y2=[8 9 2 5 6];

% Draw the second signal.

subplot(3,1,1);

stem(x1,y1);

grid on;

grid minor;

axis([-10 10 -8 8]);

% Draw the second signal.

subplot(3,1,2);

stem(x2,y2);

grid on;

grid minor;

axis([-10 10 -8 8]);

n=min(min(x1),min(x2)):1:max(max(x1),max(x2));

% This function is for the addition the two signal .

[y] = add\_function(n,x1,x2,y1,y2);

% This is for the plot the added signal.

subplot(3,1,3);

stem(n,y);

grid on;

grid minor;

axis([-10 10 -8 8]);

**add\_function.m ->**

function[y] = add\_function(n,x1,x2,y1,y2)

m1=zeros(1,length(n));

m2=zeros(1,length(n));

temp=1;

for i=1:length(n)

if(n(i)>=min(x1) & n(i)<=max(x1))

m1(i)=y1(temp);

temp=temp+1;

else

m1(i)=0;

end

end

temp=1;

for i=1:length(n)

if(n(i)>=min(x2) & n(i)<=max(x2))

m2(i)=y2(temp);

temp=temp+1;

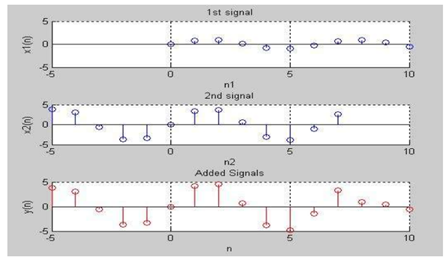
else

m2(i)=0;

end

end

y=m1+m2;



**Experiment 13 (Signal Multiplication)**

Multiplicaton.m ->

clc;

clear all;

close all;

x1=[0:0.1:10];

y1=sin(x1);

x2=[-5:0.1:7];

y2=4\*sin(x2);

% This plot is for the plotting the graph of (x1,y1).

subplot(3,1,1);

stem(x1,y1);

grid on;

grid minor;

axis([-5 10 -5 5]);

% This plot is for the plotting the graph of (x2,y2);

subplot(3,1,2);

stem(x2,y2);

grid on;

grid minor;

axis([-5 10 -5 5]);

% This line is use for find out the new range of the signal.

n=min(min(x1),min(x2)):0.1:max(max(x1),max(x2));

[m]=mul\_function(n,x1,y1,x2,y2);

%This plot is for the plotting the graph of (n,y) multiplicated signal.

subplot(3,1,3);

stem(n,m,'r');

grid on;

grid minor;

axis([-5 10 -5 5]);

mul\_function.m ->

function[m]=mul\_function(n,x1,y1,x2,y2)

m1=zeros(1,length(n));

m2=m1;

% This loop is use for the fill the loop m1.

temp=1;

for i=1:length(n)

if(n(i)>=min(x1) & n(i)<=max(x1))

m1(i)=y1(temp);

temp=temp+1;

else

m1(i)=0;

end

end

% This loop is use for the fill the loop m2.

temp=1;

for i=1:length(n)

if(n(i)>=min(x2) & n(i)<=max(x2))

m2(i)=y2(temp);

temp=temp+1;

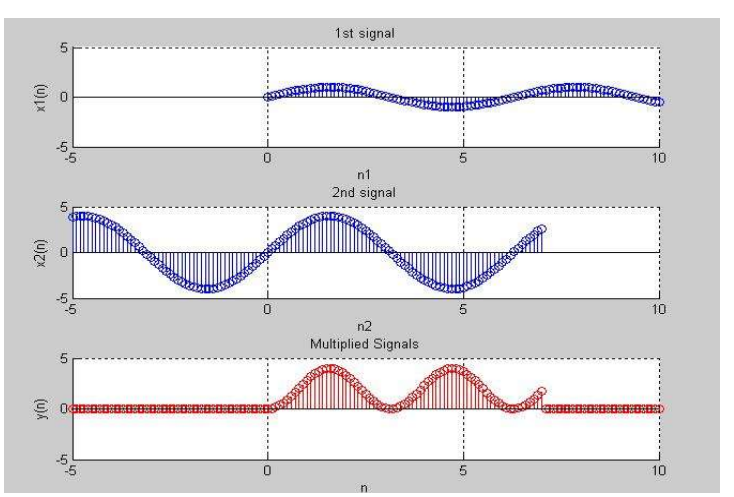
else

m2(i)=0;

end

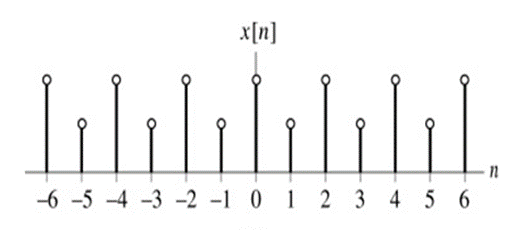
end

m=m1.\*m2;



**Experiment 14 (Time Scaling)**

A discrete time signal x(n) is shown in figure.



Sketch the signal x[n], the sketch y[n]=x[n/2].

Solution:-

close all;

clear all;

clc;

start\_value = input('Enter the start value: ');%-6

end\_value = input('Enter the end value: ');%6

n1 = start\_value:end\_value;

y=input("Enter the values of signal = "); %[1 0.5 1 0.5 1 0.5 1 0.5 1 0.5 1 0.5 1]

index=1;

n2=(2\*start\_value):(2\*end\_value);

for i=1:length(n2)

x1(i)=n2(i);

if(rem(n2(i),2)==0)

y1(i)=y(index);

index=index+1;

else

y1(i)=0;

end

end

subplot(2,1,1);

stem(n1,y,'r');

xlabel("Time");

ylabel("Amplitude");

grid on;

grid minor;

axis([(start\_value-1) (end\_value+1) -2 2]);

title("Original signal Y[n]=X[n]");

subplot(2,1,2);

stem(x1,y1,'b');

xlabel("Time");

ylabel("Amplitude");

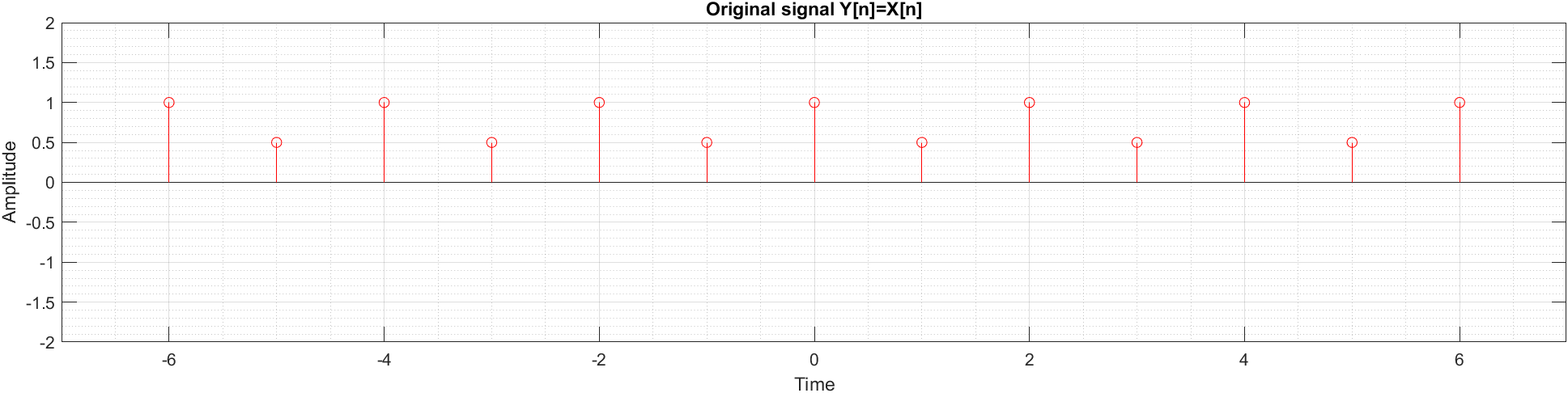
grid on;

grid minor;

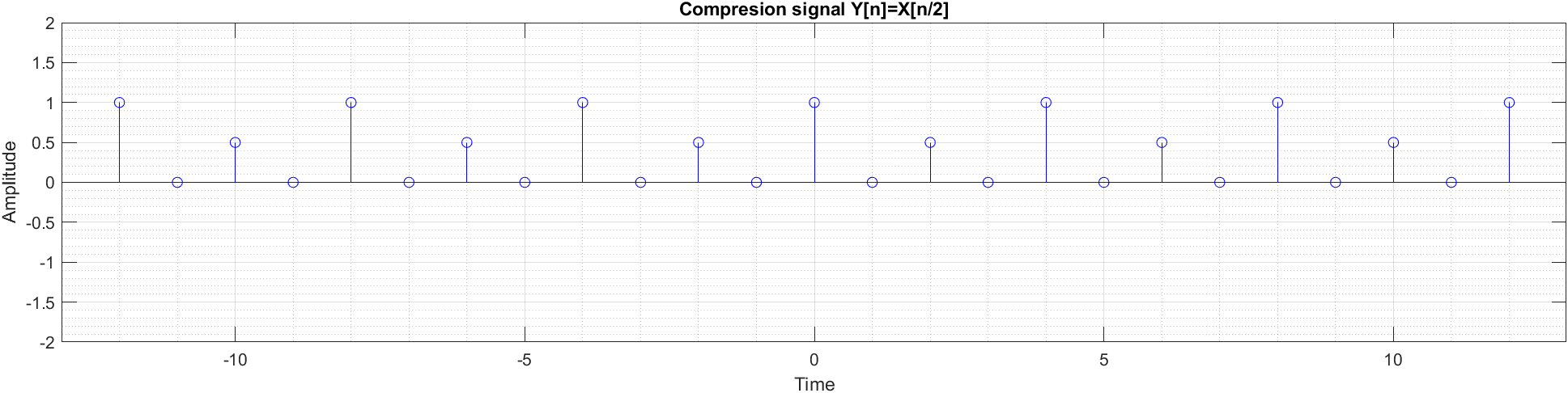
axis([(2\*start\_value-1) (2\*end\_value+1) -2 2]);

title("Compresion signal Y[n]=X[n/2]");

Original signal Y[n]=X[n]: -



Compressed signal Y[n]=X[n/2]:-



**(Important Note: You have to write the program both for compression and expansion)**

**Experiment 15:** A discrete time signal x(n) is shown in figure. Sketch the signal x[n], y[n]=x[n-4] and x[n+4], derived from x[n].



**Solution:**

clc;

clear;

n = -5:5;

x= [0 -1 -.5 .5 1 1 1 1 .5 0 0]

subplot(3,1,1);

stem (n,x);

xlabel('Time Sample');

ylabel('Amplitude');

title('Original Signal');

axis([-7 7 min(x)-2 max(x)+2]);

grid on;

grid minor;

m = n+4;

subplot(3,1,2);

stem (m,x);

xlabel('Time Sample');

ylabel('Amplitude');

title('Time right shifted signal');

axis([-7-2+4 7+2+4 min(x)-2 max(x)+2]);

grid on;

grid minor;

l = n-4;

subplot(3,1,3);

stem (l,x);

xlabel('Time Sample');

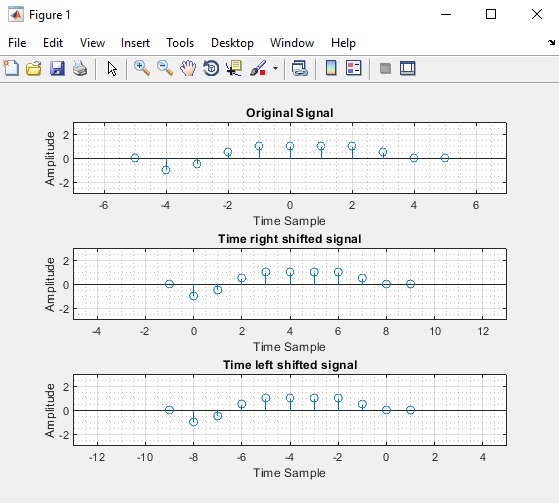
ylabel('Amplitude');

title('Time left shifted signal');

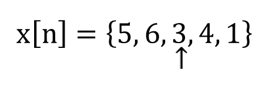
axis([-7-2-4 7+2-4 min(x)-2 max(x)+2]);

grid on;

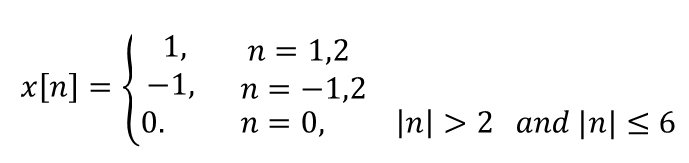
grid minor;



**Experiment 16:** Find the even and odd components of the discrete-time signal x(n), where,

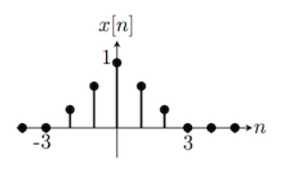


**Experiment 17:** A discrete time signal x(n) is given by



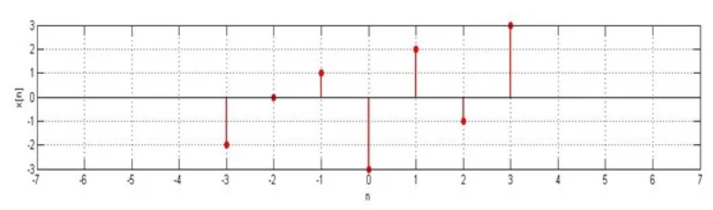
Sketch, y[n]=x[2n+3].

**Experiment 17:** Given the signal

****

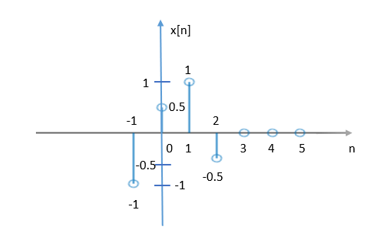
Find y[n]=x[2n] and y[n]=x[n/2]

**Experiment 18:** Given the signal

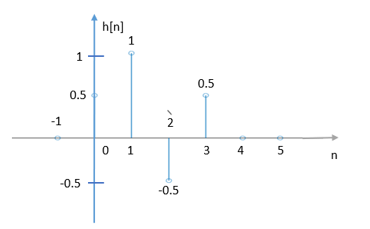
****

**Find y[n]=x[n-3] and z[n]=x[n+2]**

**Experiment 19:** The input x[n] of a LTI system,



The impulse response of the system:



Find out y[n].

**Code:**

**Convolution.m**

clc;

clear all;

close all;

x1=[-1 0 1 2];

y1=[-1 0.5 1 -0.5];

x2=[0 1 2 3 ];

h=[0.5 1 -0.5 0.5];

[n y]=func\_convalution(x1,y1,x2,h);

subplot(3,1,1);

stem(x1,y1);

xlabel('X1');

ylabel('Y1');

title("Given Signal");

subplot(3,1,2);

stem(x2,h);

xlabel('x2');

ylabel('h');

title("Impulse Response");

subplot(3,1,3);

stem(n,y);

xlabel('n');

ylabel('y');

title("Convalution Sum");

**func\_convaluation.m:**

function[n y]=func\_convalution(x1,y1,x2,h)

m1=min(x1)+min(x2);

m2=max(x1)+max(x2);

n=m1:m2;

y=conv(y1,h); % build in function

**Output:**

