***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);

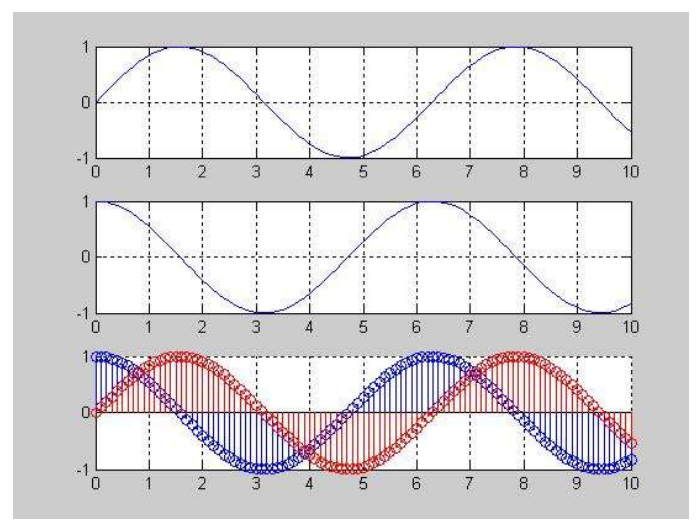
stem (x,z);

grid on;

hold on;

subplot (3,1,3);

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



***Experiment 2 (PLOTTING)***

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);

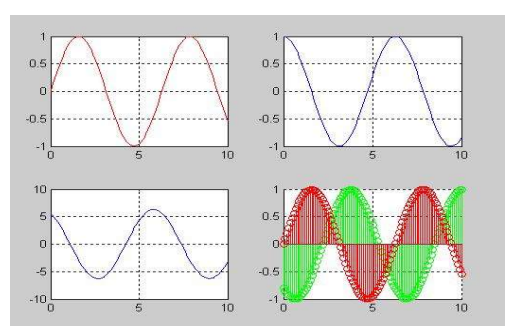
subplot (2,2,4);

stem (x,zz, ,'g');

hold on;

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

grid on;



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

Generation of Signals (Sinusoidal Sequence)

% Generation of sinusoidal signals

% 2sin(2πτ-π/2)

T = [-5:0.01:5];

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

plot(t,x)

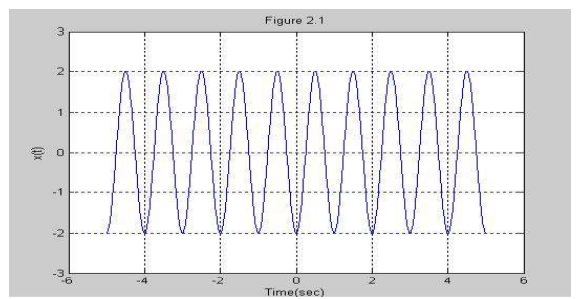
grid on;

axis ([-6 6 -3 3])

ylabel ('x(t)')

xlabel ('Time(sec)')

title ('Figure 2.1')



***Experiment 4 (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)

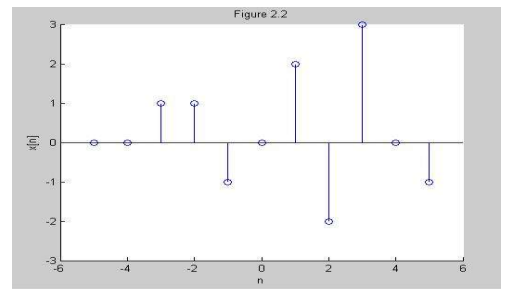
grid on;

axis ([-6 6 -3 3])

ylabel ('x(t)')

xlabel ('Time(sec)')

title ('Figure 2.1')



***Experiment 5 (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;

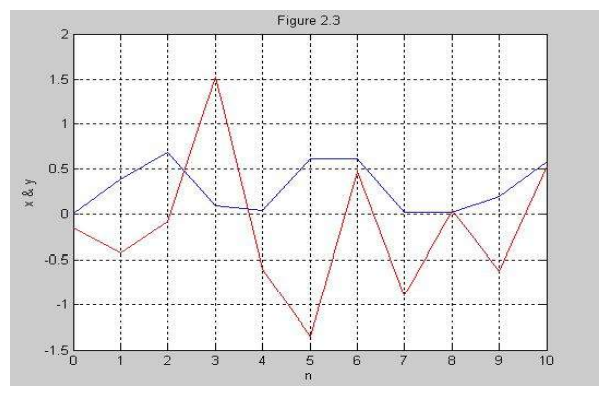
hold on;

plot(n,y,'r');

ylabel ('x & y')

xlabel ('n')

title ('Figure 2.3');



***Experiment 6 (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);

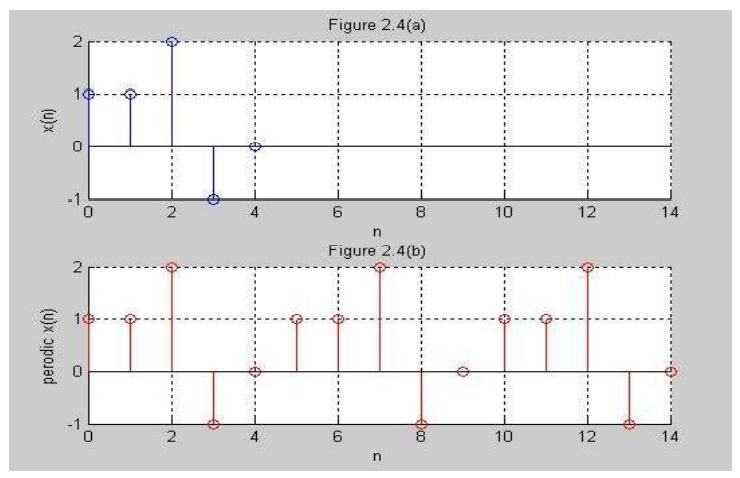
stem (n\_new,xtilde,'r');

grid on;

xlabel ('n');

ylabel ('perodic x(n)');

title ('Figure 2.4(b)');



***Experiment 7 (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;

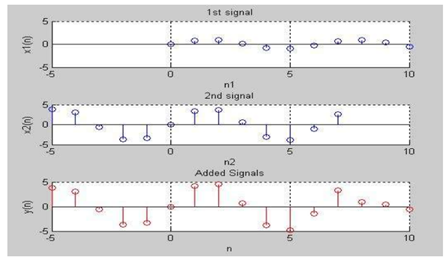
else

m2(i)=0;

end

end

y=m1+m2;



**Experiment 8 (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;

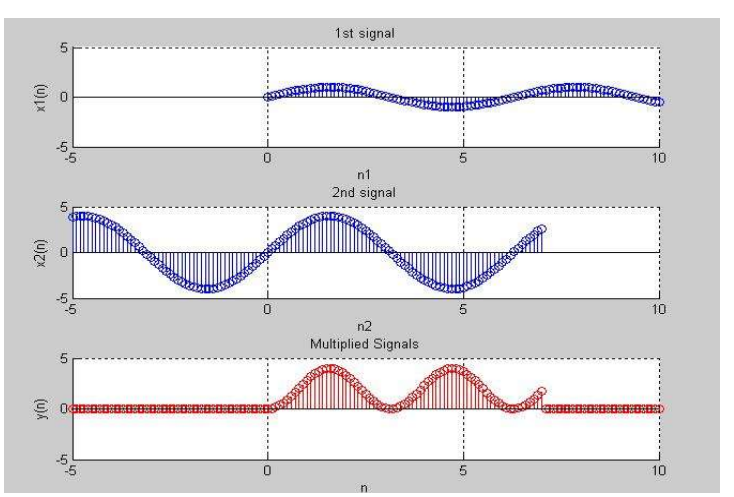
else

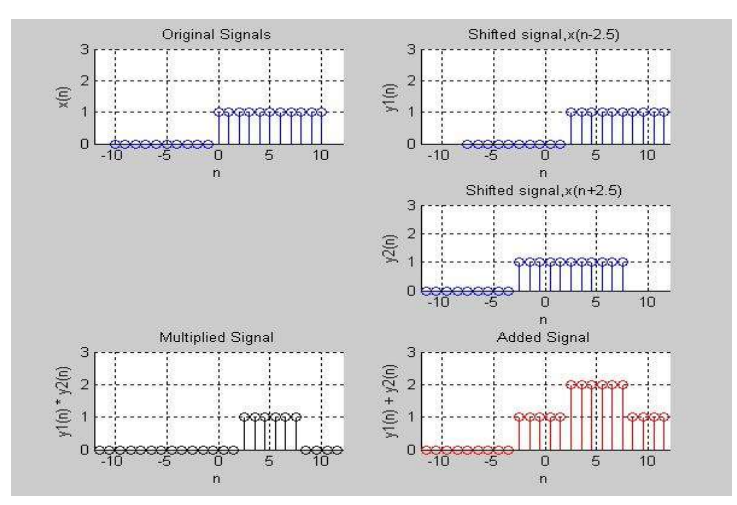
m2(i)=0;

end

end

m=m1.\*m2;





Experiment 9 (piyal) Sunday

Scaling of discrete time signal

This program is for the signal compression when “**α € (-∞ to ∞)**” where α is an integer number and time expansion when “**α € (0 to 1)**” where α is a fractional number.

**Code**

**ScalingSignal**:

close all;

clear all;

clc;

y=[2 0 1 2 3 4 7 8 6 5];

n=-5:4;

value=input("Please enter the scaling factor for the compression : ");

value1=1/2;

%This function is used for find the compression of the signal.

[x1,y1]=scaling\_compression(n,y,value);

%This function is used for calculate the expansion of the signal.

[x2,y2]=scalling\_expansion(n,y,value1);

% Plot the original signal.

subplot(3,1,1);

stem(n,y);

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

title("Representation of original signal");

% Plot the compression signal.

subplot(3,1,2);

stem(x1,y1);

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

title("Representation of Compress Signal signal");

% Plot the expansion signal.

subplot(3,1,3);

stem(x2,y2);

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

title("Representation of Expanded signal");

**scaling\_compression**:

function [x1,y1]=scaling\_compression(n,y,value) %'nx' is the position of the x axis and x is amplitude.

temp=1;

for i=1:length(n)

% This condition check whether this point will added in the new signal or not.

if(rem(n(i),value) == 0)

x1(temp)=n(i)./value; % This line is for the value of the time domain axis

y1(temp)=y(i);

temp=temp+1;

end

end

**scalling\_expansion**:

function [x2,y2]=scallingexpansion(x,y,value1)

num=1/value1;

low=min(x).\*num;

high=max(x).\*num;

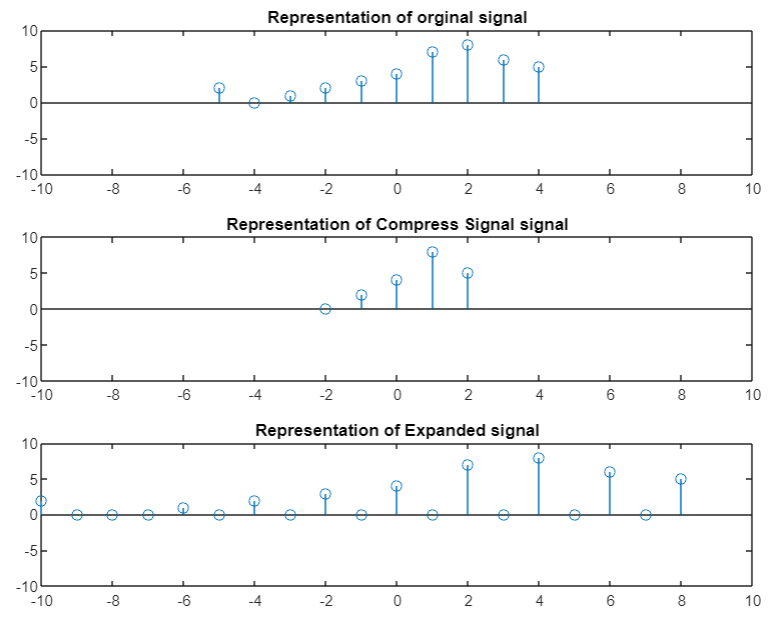
x2=low:high;

y2=zeros(1,length(x2));

for i=1:length(y)

y2((i.\*num)-(num-1)) = y(i);

end



Experiment 10 (priyo) Sunday

**Experiment Name:** Time Reversal (Reflection) of a discrete time signal.

**Matlab Code:**

**Code no.1 :** Time reversal of discrete time signal (replacing t with -t)

%Time reversal of a discrete time signal (replacing t with -t)

close all

clc

t=-4:4;

x=[0 3 5 7 8 10 21 31 0];

figure

subplot(211)

stem(t,x,'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';

%Time Reversed part

subplot(212)

stem(-t,x,'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;

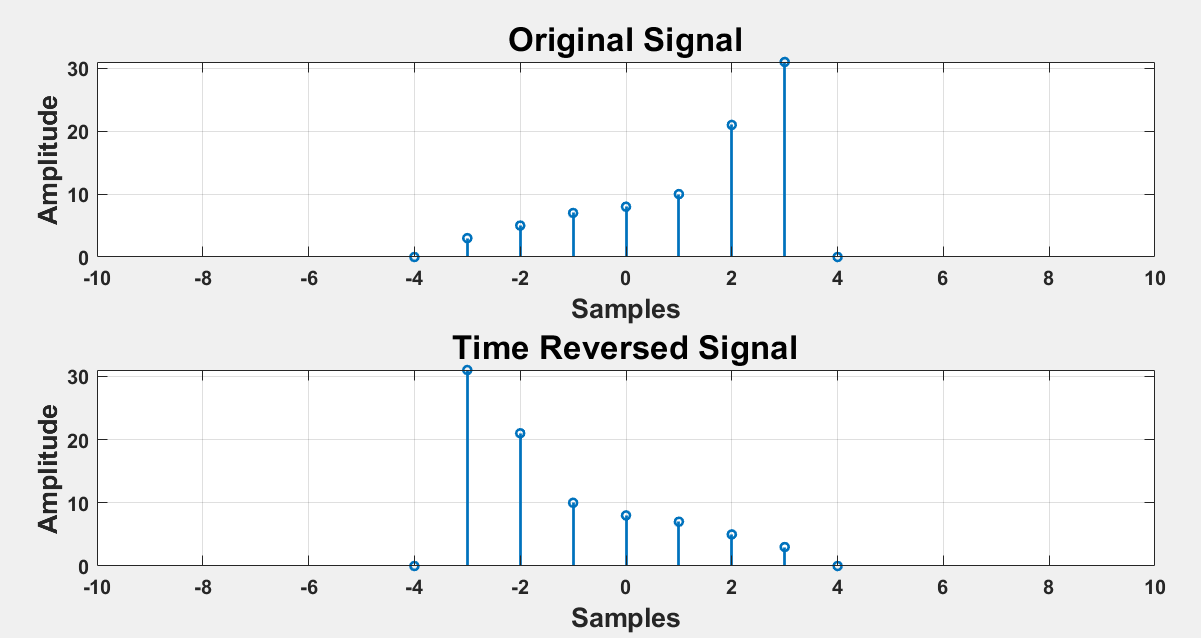
ax.XAxis.FontSize = 15;

ax.XAxis.FontWeight = 'bold';

ax.YAxis.FontSize = 15;

ax.YAxis.FontWeight = 'bold';

**Output for code no.1:**



**Figure 1: Time reversal of discrete time signal code 01 output.**

**Code No.2:** 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(211)

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;

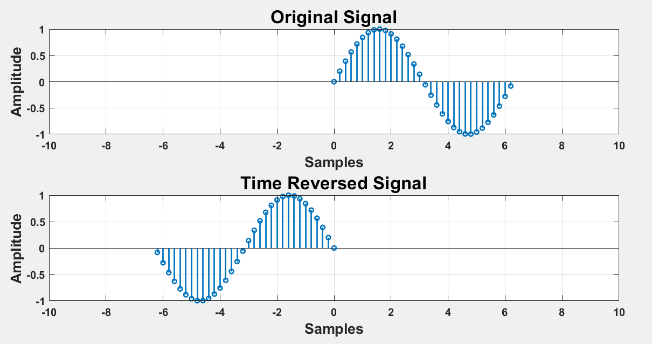
ax.XAxis.FontSize = 15;

ax.XAxis.FontWeight = 'bold';

ax.YAxis.FontSize = 15;

ax.YAxis.FontWeight = 'bold';

**Output for code no.2:**



**Figure 2: Time reversal of discrete time signal code 2 output.**

**Code No.3:** 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;

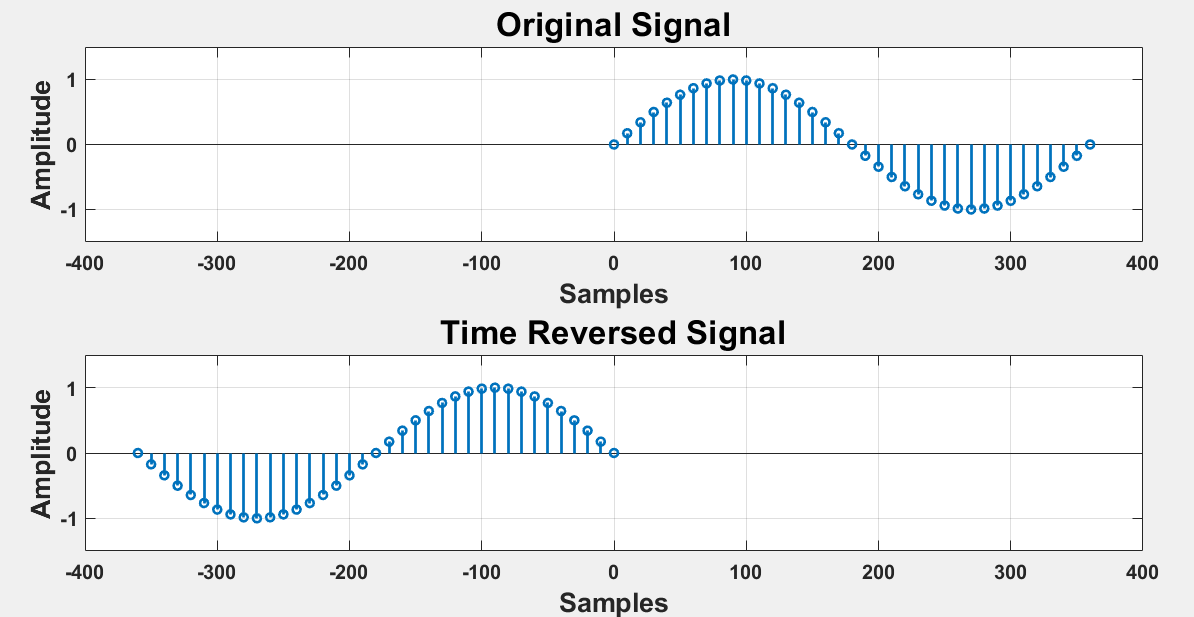
ax.XAxis.FontSize = 15;

ax.XAxis.FontWeight = 'bold';

ax.YAxis.FontSize = 15;

ax.YAxis.FontWeight = 'bold';

**Output of code no.3:**



**Figure 3:Time reversal of discrete time signal code 3 output.**

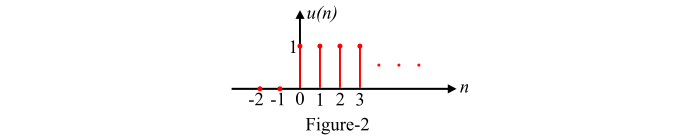
Experiment 11 (mow) Sunday

**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:



**Code:**

%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)];

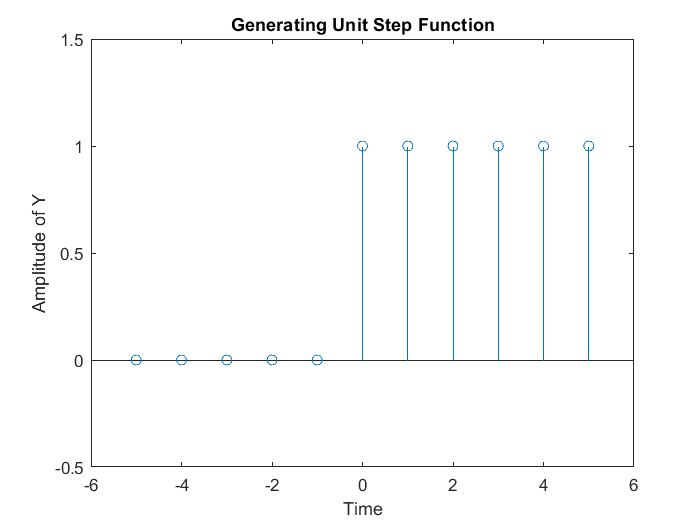
stem(n,y);

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

xlabel('Time');

ylabel('Amplitude of Y');

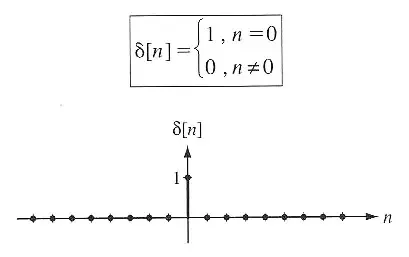
title('Generating Unit Step Function');



Experiment 12 (Faisal) Tuesday

**Experiment name**: Generating and plotting unit impulse discrete time signal.

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)

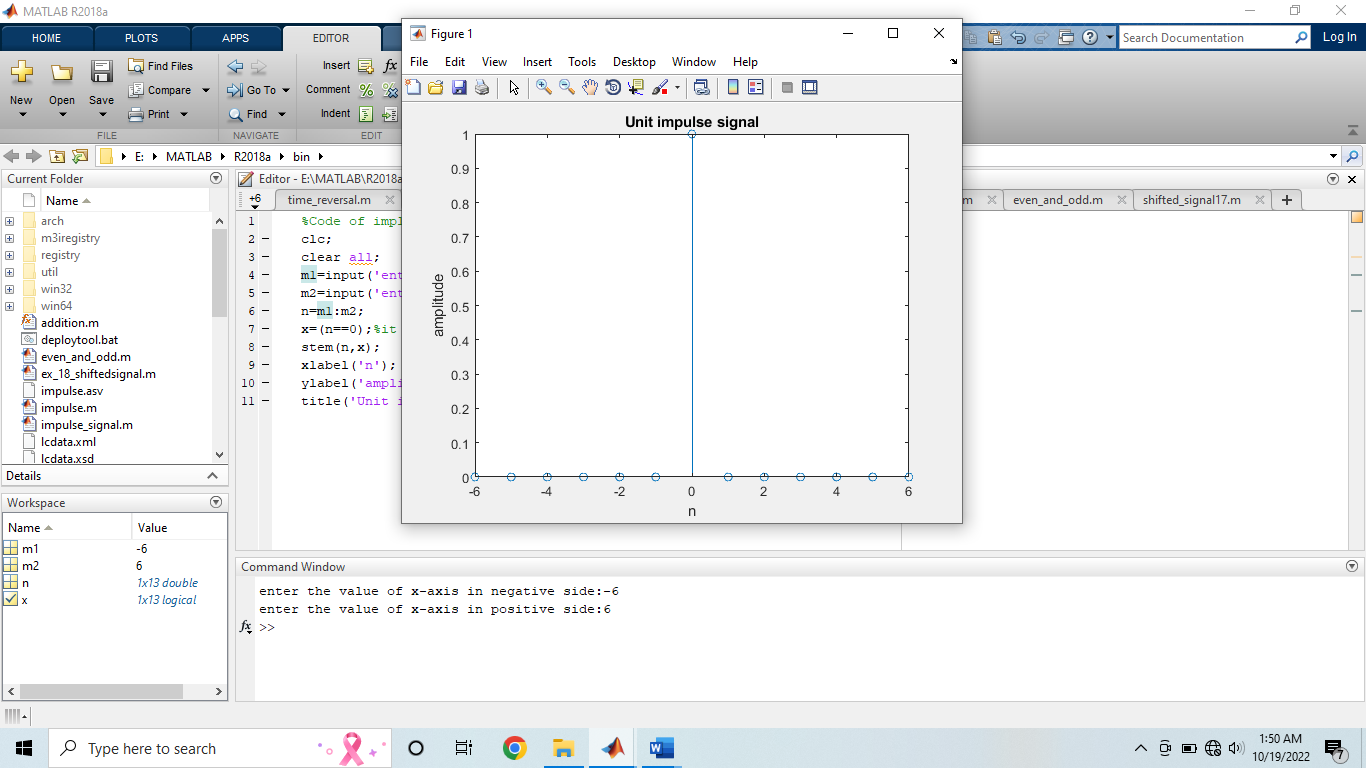
stem(n,x);

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)];

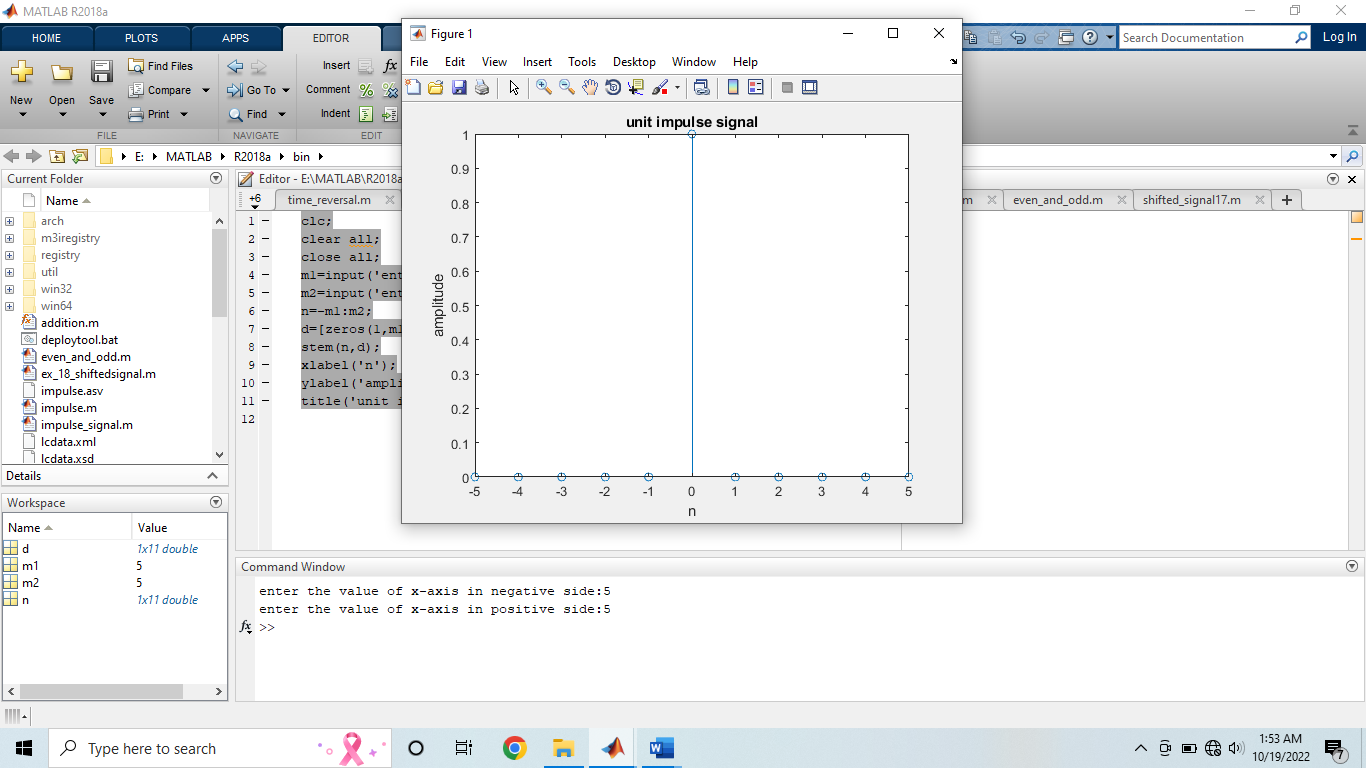
stem(n,d);

xlabel('n');

ylabel('amplitude');

title('unit impulse signal');

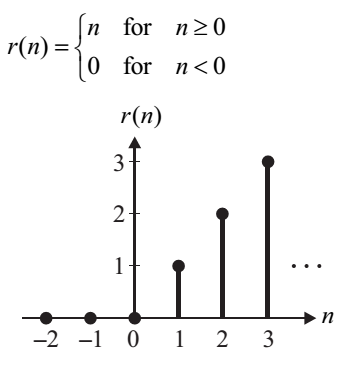
**Output:**

****

Experiment 13 (Sushmita) Tuesday

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:**

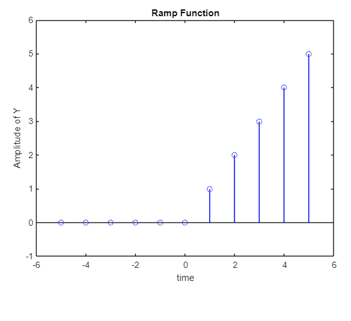
Enter lower limit

-5

Enter upper limit

5

**Output:**



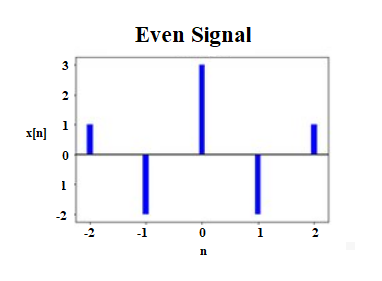
Experiment 14 (Shafi)

Even and odd part of discrete time signal

A signal is said to be an even signal if it is symmetrical about the vertical axis or time origin, i.e.,

x(-t) = x(t) for all t, … continuous time signal

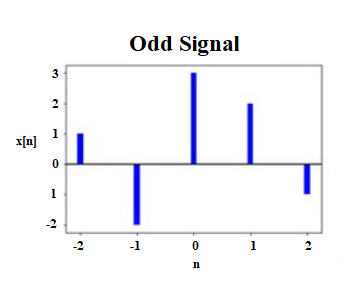
x[-n] = x[n] for all n, … discrete time signal



A signal is said to be an odd signal if it is anti-symmetrical about the vertical axis or time origin, i.e.,

x(-t) = -x(t) for all t, … continuous time signal

x[-n] = -x[n] for all n, … discrete time signal



**Code 1:**

%Even and Odd Signal

n1 = 0:6;

%x1 value is from n = 0 to n = 6

x1 = [1 1 1 1 1 1 0];

n2 = -fliplr(n1);

%after flipping, n2 = 6 5 4 3 2 1

% then after multiply, n2 = -6 -5 -4 -3 -2 -1 0

n = min(min(n1),min(n2)) :max(max(n1),max(n2));

% n = min(0, -6) : max(6, 0), so n ranges from -6 to 6

% but for n = -6 to n = -1, no data value is defined, that's why the

% following two lines of code

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

%y1 = 0 for n = [-6, 6] but we want y1 value to

% be 0 for the interval [-6, -1] for the rest of the interval [0, 6], y1

% value will be same as x1

y1((n>=min(n1)) & (n<=max(n1))) = x1();

% now y1 = [0 0 0 0 0 0 1 1 1 1 1 1 0]

x=y1;

% x[n] = y1

xe = 0.5\*(x + fliplr(x));%fliplr(x) is x[-n], so xe = (1/2)\*(x[n] + x[-n])

xo = 0.5\*(x - fliplr(x));%fliplr(x) is x[-n], so xe = (1/2)\*(x[n] - x[-n])

subplot(3,1,1)

stem(n,x);

title('Input signal');

subplot(3,1,2)

stem(n,xe);

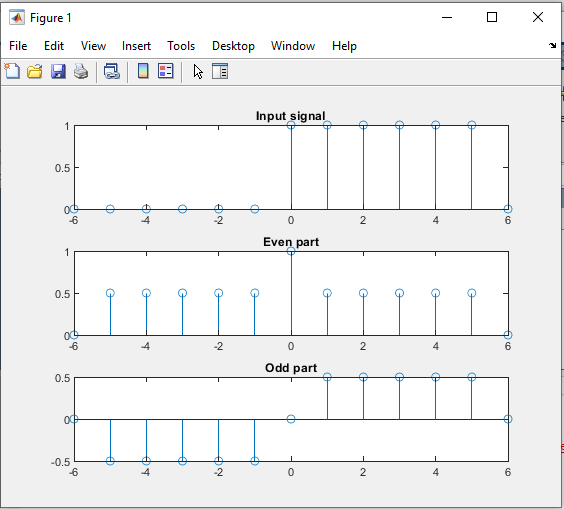
title('Even part');

subplot(3,1,3)

stem(n,xo);

title('Odd part')

**Output:**



**Code 2:**

clc

clear all

close all

n1=input('Enter the time sample range of x');

x=input('Enter the sequence');

n2=-fliplr(n1);

y=fliplr(x);

u=min(min(n1),min(n2));

t=max(max(n1),max(n2));

r=u:1:t; %r = [-3 -2 -1 0 1 2 3]

z1=[];

temp=1;

for i=1:length(r)

if(r(i)<min(n1) || r(i)>max(n1))

z1=[z1 0];

else

z1=[z1 x(temp)];

temp=temp+1;

end

end

z2=[];

temp=1;

for i=1:length(r)

if(r(i)<min(n2) || r(i)>max(n2))

z2=[z2 0];

else

z2=[z2 y(temp)];

temp=temp+1;

end

end

z3=(z1+z2)/2;%Even part

z=(z1-z2)/2;%Odd part

subplot(3,1,1);

stem(r,z1);

title('Original signal');

subplot(3,1,2);

stem(r,z3);

title('Even signal');

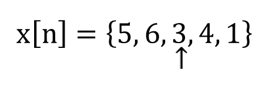
subplot(3,1,3);

stem(r,z);

title('Odd signal');

Experiment 15 (Faisal-B180305127- 5 MARKS)

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



**Code**:

%experiment-15:find the even and odd component of the given signal x[n]={5,6,3,4,1}

clc;

clear all;

close all;

%plotting the original signal

x1=[-2 -1 0 1 2 ];

y1=[5 6 3 4 1];

subplot(2,2,1);

stem(x1,y1);

xlabel('n');

ylabel('amplitude');

title('orginal signal:x[n]');

%plotting the reversed signal

x2=fliplr(-x1);

y2=fliplr(y1);

subplot(2,2,2);

stem(x2,y2);

xlabel('n');

ylabel('amplitude');

title('reversed signal:x[-n]');

%equivalent the axis both signal

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

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

%equation of even and odd part of the signal

xe=0.5\*(m1+m2);

xo=0.5\*(m1-m2);

%plotting the even signal

subplot(2,2,3);

stem(n,xe);

xlabel('n');

ylabel('amplitude');

title('even signal:xe[n]');

%plotting the odd signal

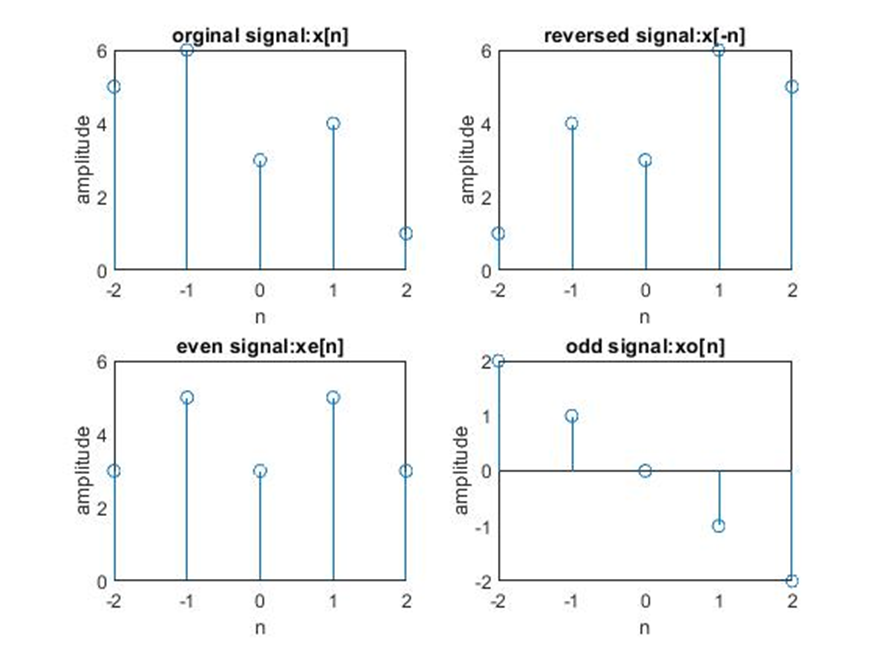
subplot(2,2,4);

stem(n,xo);

xlabel('n');

ylabel('amplitude');

title('odd signal:xo[n]');



**Output:**

Experiment No. 16 (B170305010-5 marks)

Find the even and odd components of the discrete-time signal x(n),

Code Section

% Exp: 16

% ID: B170305010

Clc;

clear all;

close all;

x1=-5:5;

y1=[0 0 0 0 0 1 1 1 1 1 1];

% This portion is for the orginal Signal Construction.

subplot(4,1,1);

stem(x1,y1);

grid on;

grid minor;

xlabel("N");

ylabel("x[N]");

title("Orginal Signal");

% This portion is for the reverse Signal Construction.

x2=-1\*x1;

subplot(4,1,2);

stem(x2,y1);

grid on;

grid minor;

xlabel("N");

ylabel("X[N]");

title("Reverse Signal");

% This line calculate the new range for the signal

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

%This is for the set the amplitude value y1

y1=(n>=0&n<=max(x1));

%This is for the set the amplitude value for y2

y2=(n>=min(x2)&n<=0);

% Calculate the even signal.

y3=0.5.\*(y1+y2);

% Calculating odd signal.

y4=0.5.\*(y1-y2);

% This portion is for the print the even signal.

subplot(4,1,3);

stem(n,y3);

grid on;

grid minor;

xlabel("N");

ylabel("X[N]");

title("Even Signal");

% This portion is for the print the odd signal.

subplot(4,1,4);

stem(n,y4);

grid on;

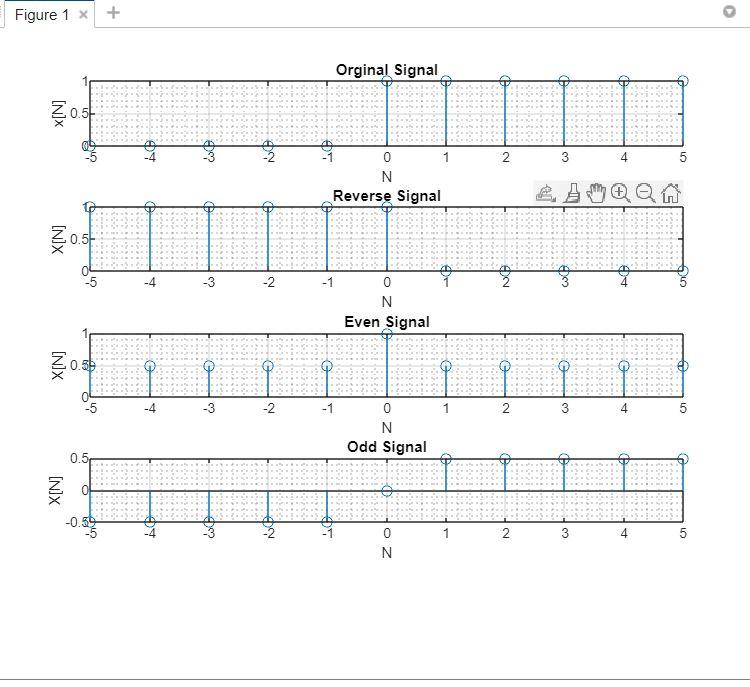
grid minor;

xlabel("N");

ylabel("X[N]");

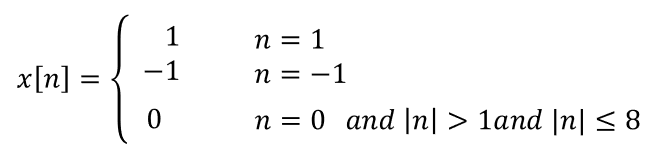
title("Odd Signal");

Output



Experiment 17 (B160305003- 5 marks)

Given the Discrete time signal x[n]. Draw the signal Find y[n]=x[-n] and then find z[n]=x[n]+x[-n].



**Code:**

clc;

clear all;

close all;

n1=-8:8

a1=[zeros(1,7) -1 0 1 zeros(1,7)];

subplot(3,1,1);

stem(n1,a1);

xlabel('n');

ylabel('Amplitude');

title('orginal signal:x[n]');

%generating the reversed signal of x[n]

n2=fliplr(-n1);

a2=fliplr(a1);

subplot(3,1, 2);

stem(n2,a2);

xlabel('n');

ylabel('Amplitude');

title('reversed signal:x[-n]');

%find Z[n]=x[n]+x[-n]

s=a1+a2;

subplot(3,1, 3);

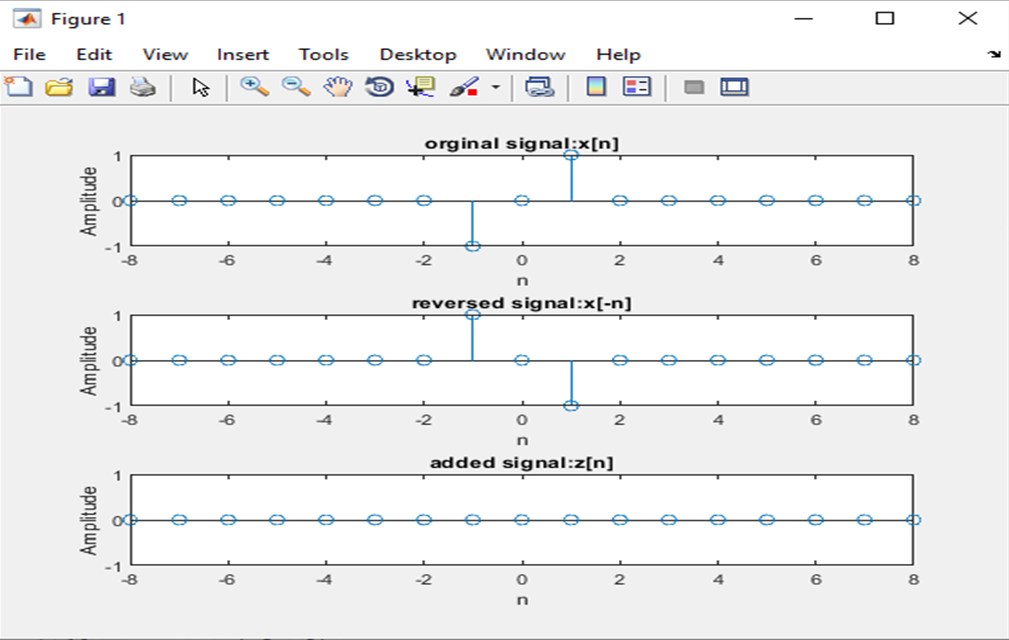
stem(n1,s);

xlabel('n');

ylabel('Amplitude');

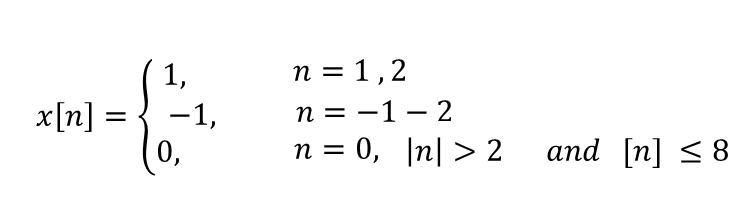
title('added signal:z[n]');

Output:



Experiment 18 (B160305004 -5MARKS)

Given,



Find the time shifted signal y[n]=x[n+3]

%shifted signal for y[n]=x[n+3]

clc;

clear all;

close all;

n=-8:8;

d=[ 0 0 0 0 0 0 -1 -1 0 1 1 0 0 0 0 0 0 ];% discrete value of x-axis

subplot(2,1,1);

stem(n,d);%plotting the orginal signal

axis([-15 15 -2 2]);

grid on;

xlabel('n');

ylabel('Amplitude');

title('orginal signal:x[n]');

m=n-3;%how many times shifted

subplot(2,1,2);

stem(m,d);% plotting the shifted signal

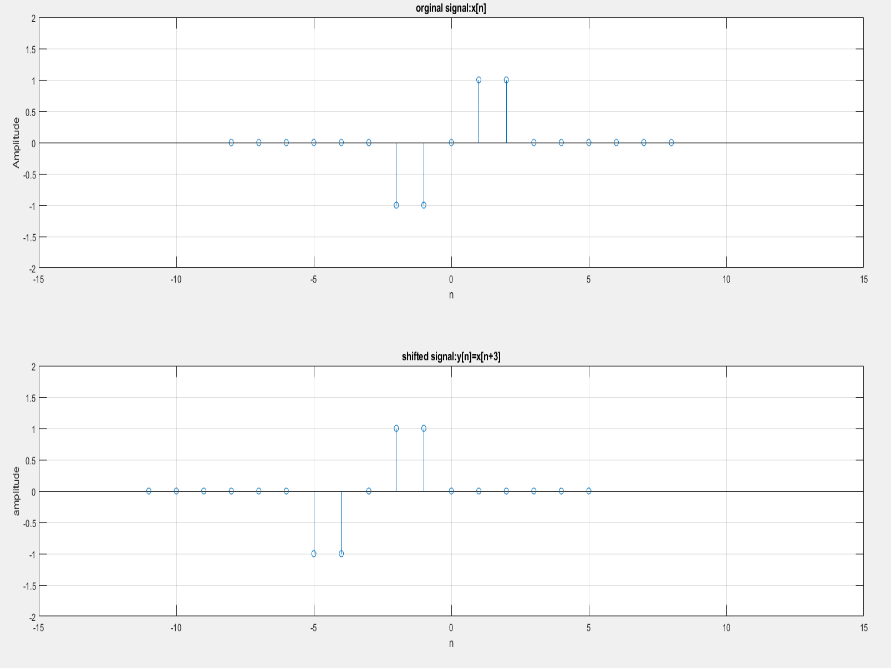
axis([-15 15 -2 2]);

grid on;

xlabel('n');

ylabel('amplitude');

title('shifted signal:y[n]=x[n+3]')



Experiment 19 (B170305017)

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');

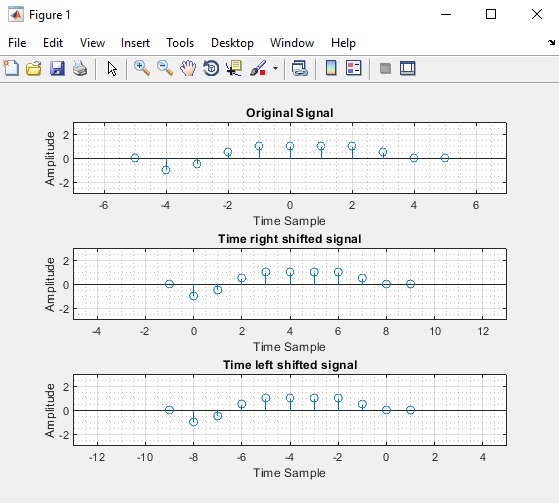
ylabel('Amplitude');

title('Time left shifted signal');

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

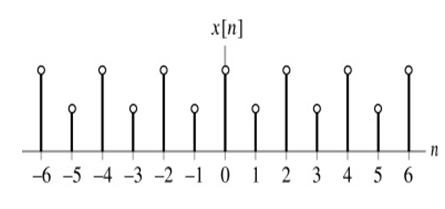
grid on;

grid minor;



Experiment 20 (B180305037-5 MARKS)

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



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

Solution:

close all;

clear all;

clc;

n=-6:6;

y=[1 0.5 1 0.5 1 0.5 1 0.5 1 0.5 1 0.5 1];

value=2;

temp=1;

for i=1:length(n)

if(rem(n(i),value)==0)

x1(temp)=n(i)./value;

y1(temp)=y(i);

temp=temp+1;

end;

end

subplot(2,1,1);

stem(n,y,'r');

xlabel("Time domain");

ylabel("Amplitude");

grid on;

axis([-10 10 -2 2]);

title("Original signal");

subplot(2,1,2);

stem(x1,y1,'g');

xlabel("Time domain");

ylabel("Amplitude");

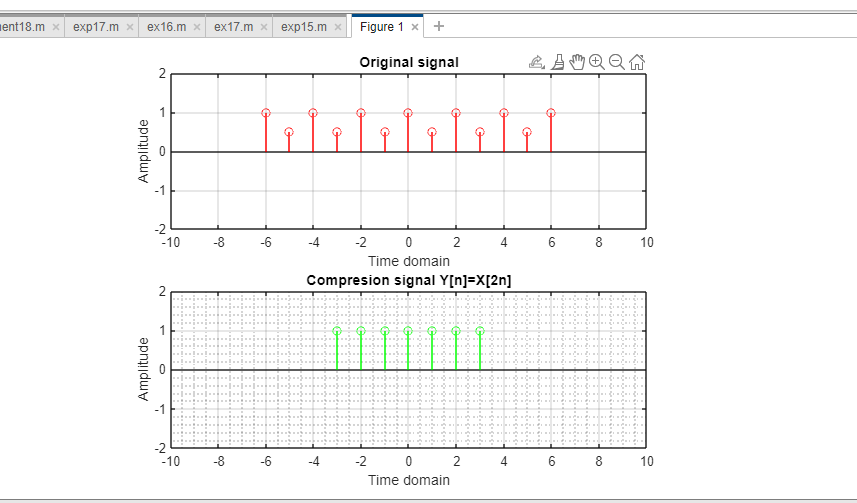
grid on;

grid minor;

axis([-10 10 -2 2]);

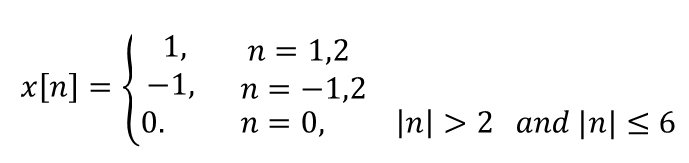
title("Compresion signal Y[n]=X[2n]");

Signal:



Experiment 21 (B180305002-5 MARK)

A discrete time signal x(n) is given by



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

Code:

clc;

close all;

%orginal signal

n=-6:6;

y=[0 0 0 0 -1 -1 0 1 1 0 0 0 0];

subplot(3,1,1);

stem(n,y);

axis([-10 10 -2 2]);

xlabel('n');

ylabel('amplitude');

title('x[n]');

%shifting the given signal

n1=n-3;

subplot(3,1,2);

stem(n1,y);

axis([-10 10 -2 2]);

value=2;

temp=1;

for i=1:length(n1)

if(rem(n1(i),value)== 0)

x1(temp)=n1(i)./ value;

y1(temp)=y(i);

temp=temp+1;

end

end

%final signal

subplot(3,1,3);

stem(x1,y1);

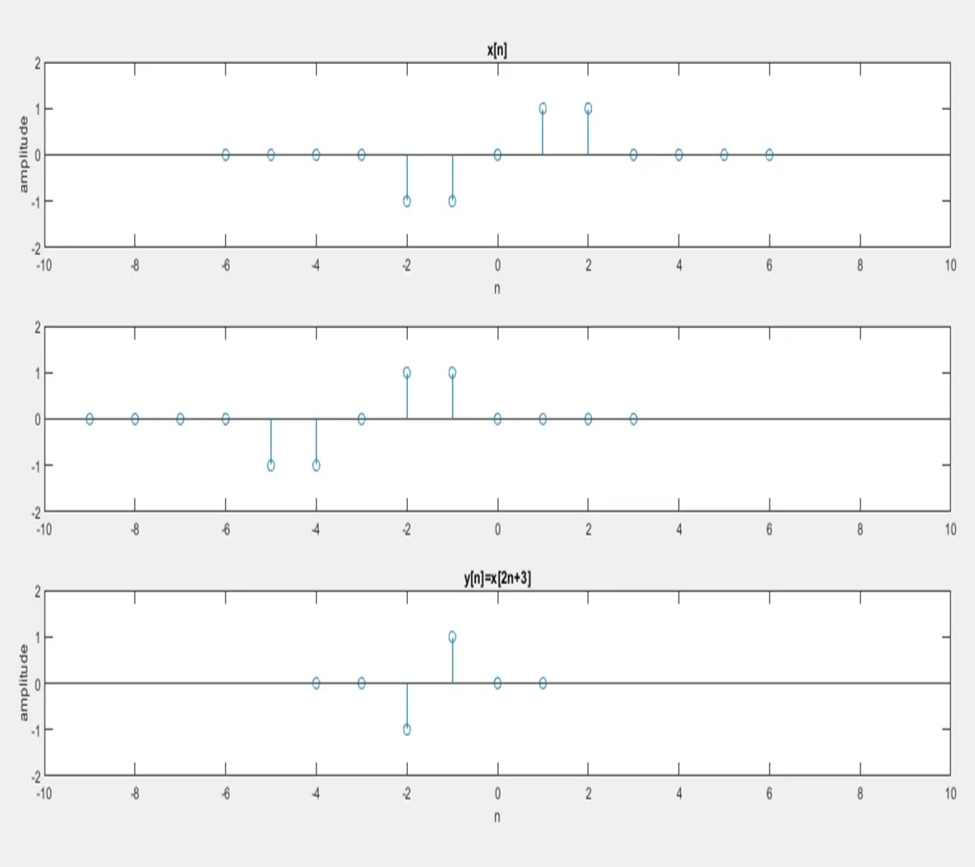
axis([-10 10 -2 2]);

xlabel('n');

ylabel('amplitude');

title('y[n]=x[2n+3]');

Output :

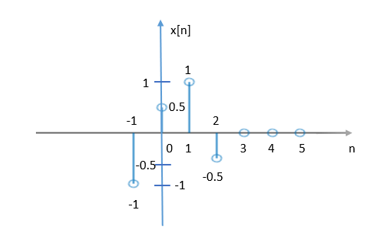


Experiment 22 (B180305034 Naib Uddin-CGPA 3.79-No Marks)

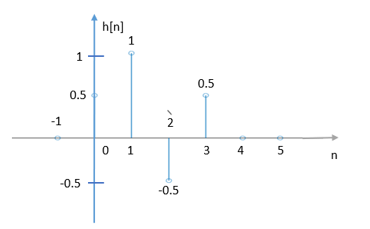
The input x[n] of a LTI system:

1. x[k].h[n-k]

2. y[n]= summation(-infi,+infi) x[k].h[n-k]



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)

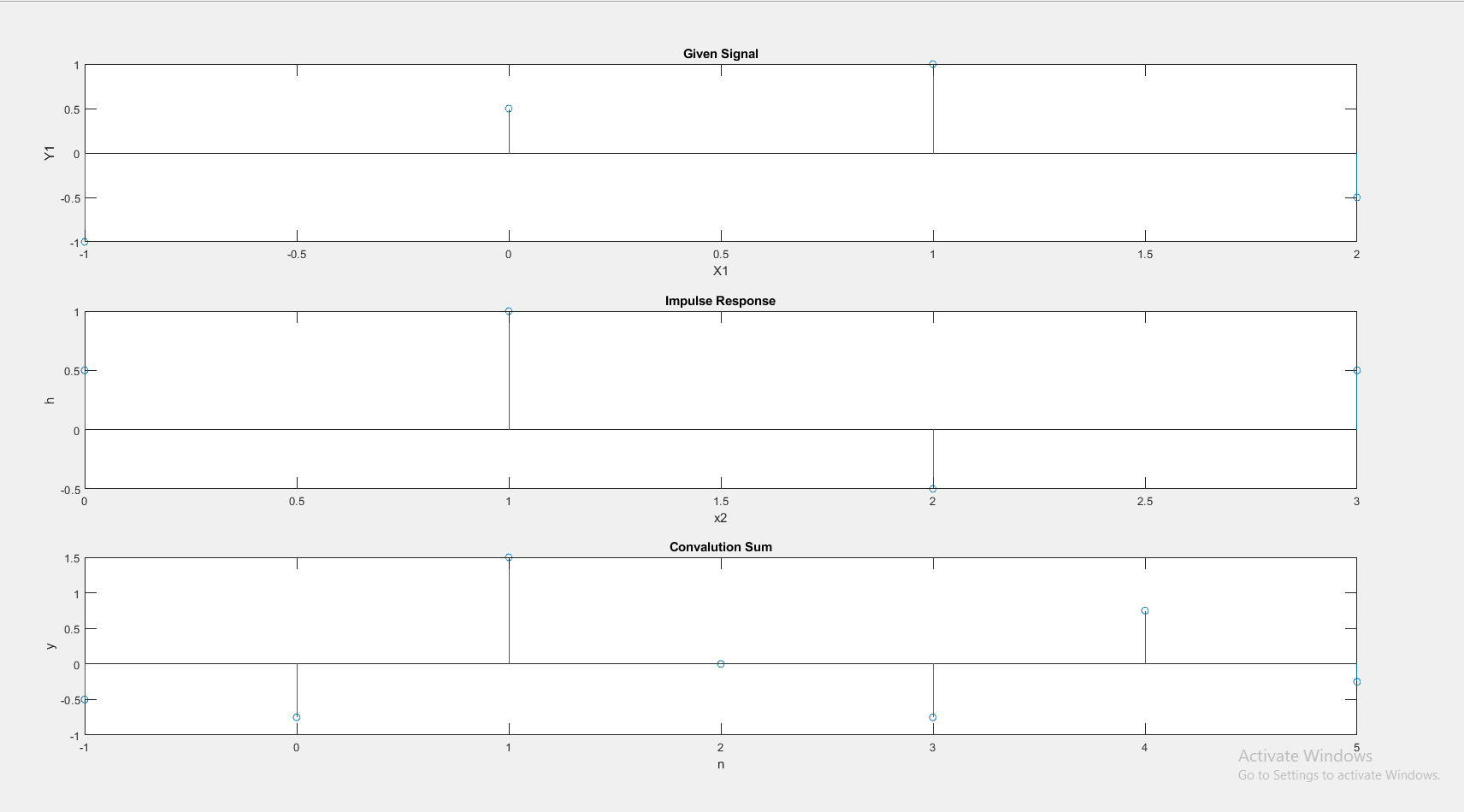
m1=min(x1)+min(x2);

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

n=m1:m2;

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

**Output:**



Experiment 23 (Sampling and Aliasing)

Code:

clc;

clear all;

close all;

frequency =input('Enter the frequency for the signal:\n');

fprintf("Enter the frequency and it must be greater or less than Nyquist frequency\n");

oversampling=input('');

%fprintf("Enter the UnderSampling frequency and it must be Less than %d\n",2\*frequency);

%undersampling=input('');

Time\_Period = 1/frequency;

tmin=-0.05;

tmax=0.05;

time = linspace(tmin,tmax,400);

amplitude = cos(2\*pi\*frequency\*time);

subplot(4,1,1);

plot(time,amplitude);

grid on; grid minor;

xlabel("Time");

ylabel("Amplitude");

title("orginal Signal");

%Nyquist rate Sampling Part.

nyquist\_frequency = 2\*frequency;

time1=tmin:(1/nyquist\_frequency):tmax;

amplitude1 = cos(2\*pi\*frequency\*time1);

subplot(4,1,2);

plot(time,amplitude);

hold on;

grid on; grid minor;

plot(time1,amplitude1);

title("NyQuist Sampling");

hold off;

if oversampling>nyquist\_frequency

%OverSampling Part.

time1=tmin:(1/oversampling):tmax;

amplitude1 = cos(2\*pi\*frequency\*time1);

subplot(4,1,3);

plot(time,amplitude);

hold on;

plot(time1,amplitude1);

grid on; grid minor;

xlabel("Time");

ylabel("Amplitude");

title("OverSampling");

hold off;

else

%OverSampling Part.

time1=tmin:(1/oversampling):tmax;

amplitude1 = cos(2\*pi\*frequency\*time1);

subplot(4,1,3);

plot(time,amplitude);

hold on;

plot(time1,amplitude1);

grid on; grid minor;

xlabel("Time");

ylabel("Amplitude");

title("UnderSampling");

hold off;

end

Output:

