**Experiment No 1**

**LIT System: To find the impulse response of a given system, Solution of difference equation, Verification of Sampling Theorem.**

**i)LIT System**

**%Without initializing values**

**%y(n)+0.8y(n-2)+0.6y(n-3)=x(n)+0.7x(n-1)+0.5x(n-2)**

clear all;

close all;

b=input('Enter the coefficients of x ');

a=input('Enter the coefficients of y ');

N=input('Enter the length of the input sequence ');

n=0:1: N;

step=1.^n;

imp=[1,zeros(1,N)];

RES1=filter(b,a,step)

RES2=filter(b,a,imp)

subplot(2,2,1)

stem(n,step)

grid on

xlabel('Time');

ylabel('Amplitude');

title('Step Input ')

subplot(2,2,2)

stem(n,imp)

grid on

xlabel('Time');

ylabel('Amplitude');

title('Impulse Input')

subplot(2,2,3)

stem(n,RES1)

grid on

xlabel('Time');

ylabel('Amplitude');

title('Step Response')

subplot(2,2,4)

stem(n,RES2)

grid on

xlabel('Input');

ylabel('Output Response');

title('Impulse Response')

**Result**

Enter the coefficients of x [1 0.7 0.5]

Enter the coefficients of y [1 0 0.8 0.6]

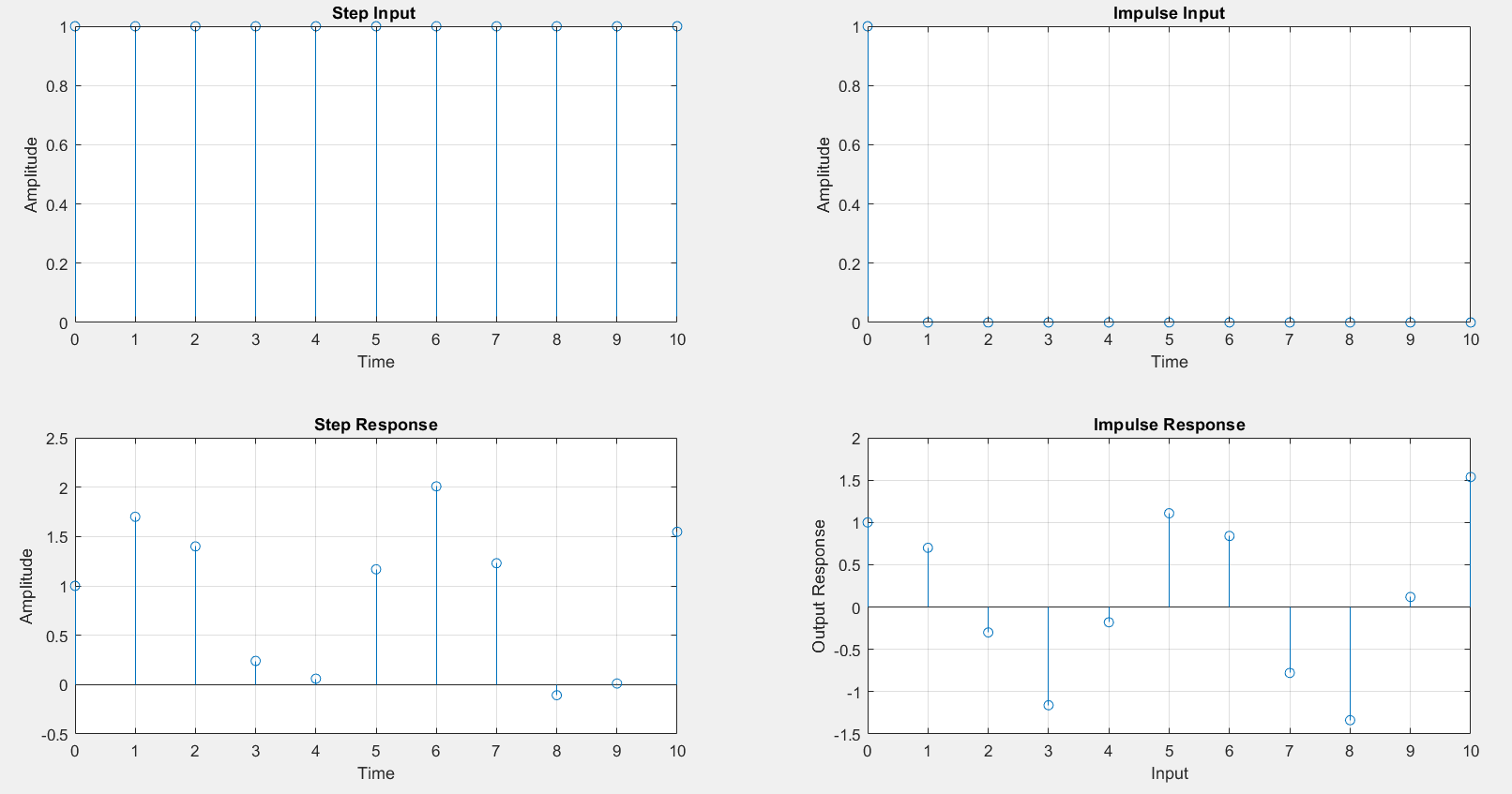
Enter the length of the input sequence 10

RES1 =

1.0000 1.7000 1.4000 0.2400 0.0600 1.1680 2.0080 1.2296 -0.1072 0.0115 1.5480

RES2 =

1.0000 0.7000 -0.3000 -1.1600 -0.1800 1.1080 0.8400 -0.7784 -1.3368 0.1187 1.5365



**Program 2**

**%Initializing Y values**

**%y(n)=1//3x(n)+1/3x(n-1)+1/3x(n-2)+0.95y(n-1)-0.9025y(n-2)**

**%y(-1)=-2, y(-2)=-3**

clear all;

close all;

b=input('Enter the coefficients of x ');

a=input('Enter the coefficients of y ');

N=input('Enter the length of the input sequence ');

n=0:1:N;

step=1.^n;

imp=[1,zeros(1,N)];

Y=[-2 -3];

XIC=filtic(b,a,Y);

RES1=filter(b,a,step,XIC)

RES2=filter(b,a,imp,XIC)

subplot(2,2,1)

stem(n,step)

grid on

xlabel('Time');

ylabel('Amplitude');

title('Step Input ');

subplot(2,2,2)

stem(n,imp)

grid on

xlabel('Time');

ylabel('Amplitude');

title('Impulse Input');

subplot(2,2,3)

stem(n,RES1)

grid on

xlabel('Time');

ylabel('Amplitude');

title('Step Response');

subplot(2,2,4)

stem(n,RES2)

grid on

xlabel('Input');

ylabel('Output Response');

title('Impulse Response');

**Result**

Enter the coefficients of x [1/3 1/3 1/3]

Enter the coefficients of y [1 -0.95 0.9025]

Enter the length of the input sequence 10

RES1 =

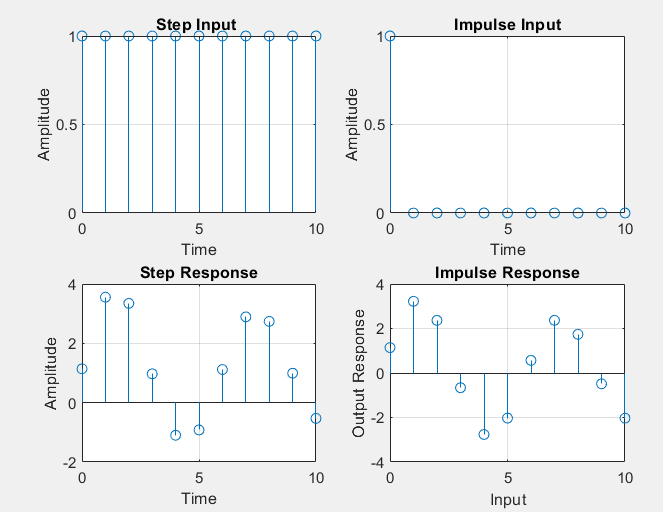
1.1408 3.5555 3.3481 0.9719 -1.0984 -0.9206 1.1167 2.8917 2.7393 0.9925

-0.5293

RES2 =

1.1408 3.2221 2.3647 -0.6615 -2.7626 -2.0275 0.5671 2.3686 1.7383 -0.4862

-2.0307



**%For the given interval**

**%y(n)-y(n-1)+0.9y(n-2)=x(n) for all n**

**%n=-5:5**

%For the given interval

%y(n)-y(n-1)+0.9y(n-2)=x(n) for all n

%n=-5:5

clear all;

close all;

b=input('Enter the coefficients of x ');

a=input('Enter the coefficients of y ');

n=-5:5;

step=[zeros(1,5) ones(1,5)];

imp=[zeros(1,5) 1 zeros(1,5)];

RES1=filter(b,a,step)

RES2=filter(b,a,imp)

subplot(2,2,1)

stem(step)

grid on

xlabel('Time');

ylabel('Amplitude');

title('Step Input ');

subplot(2,2,2)

stem(imp)

grid on

xlabel('Time');

ylabel('Amplitude');

title('Impulse Input');

subplot(2,2,3)

stem(RES1)

grid on

xlabel('Time');

ylabel('Amplitude');

title('Step Response');

subplot(2,2,4)

stem(RES2)

grid on

xlabel('Time');

ylabel('Amplitude');

title('Impulse Response');

**Result**

Enter the coefficients of x 1

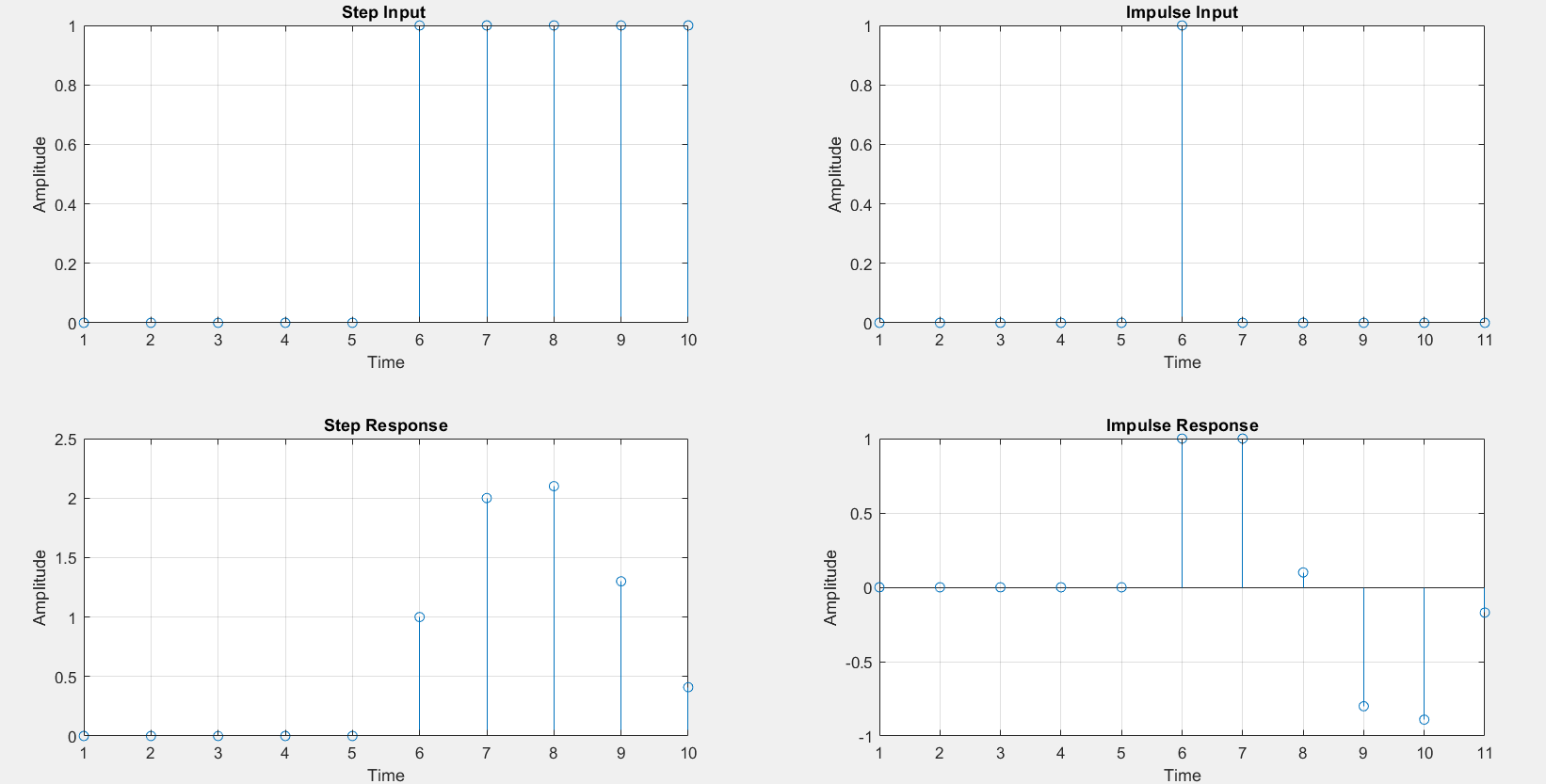
Enter the coefficients of y [1 -1 0.9]

RES1 =

0 0 0 0 0 1.0000 2.0000 2.1000 1.3000 0.4100

RES2 =

0 0 0 0 0 1.0000 1.0000 0.1000 -0.8000 -0.8900 -0.1700

****

**ii) Sampling Theorem**

close all;

clear all;

t=0:0.001:0.1;

f1=input('Enter the Input Frequency1 = ');

f2=input('Enter the Input Frequency2 = ');

%Input Signal

y=cos(2\*pi\*f1\*t) + cos(2\*pi\*f2\*t);

fm=max(f1,f2);

subplot(4,1,1)

plot(t,y);

grid on;

title('Input Sinusoidal Signal');

xlabel('Time(s)');

ylabel('Amplitude(V)');

%Under Sampling

fs1=fm;

ts1=1/fs1;

tx1=0:ts1:0.1;

y1=cos(2\*pi\*f1\*tx1) + cos(2\*pi\*f2\*tx1)

subplot(4,2,3)

stem(tx1,y1);

grid on;

title('Sinusoidal Signal sampled at fs=fm Hz');

xlabel('Time(s)');

ylabel('Amplitude(V)');

subplot(4,2,4)

plot(tx1,y1);

grid on;

title('Recovered Signal sampled at fs=fm Hz');

xlabel('Time(s)');

ylabel('Amplitude(V)');

%Right Sampling

fs2=2\*fm;

ts2=1/fs2;

tx2=0:ts2:0.1

y2=cos(2\*pi\*f1\*tx2) + cos(2\*pi\*f2\*tx2)

subplot(4,2,5)

stem(tx2,y2);

grid on;

title('Sinusoidal Signal sampled at fs=2\*fm Hz');

xlabel('Time(s)');

ylabel('Amplitude(V)');

subplot(4,2,6)

plot(tx2,y2);

grid on;

title('Recovered Signal sampled at fs=2\*fm Hz');

xlabel('Time(s)');

ylabel('Amplitude(V)');

%Over Sampling

fs3=3\*fm;

ts3=1/fs3;

tx3=0:ts3:0.1;

y3=cos(2\*pi\*f1\*tx3) + cos(2\*pi\*f2\*tx3)

subplot(4,2,7)

stem(tx3,y3);

grid on;

title('Sinusoidal Signal sampled at fs=3\*fm Hz');

xlabel('Time(s)');

ylabel('Amplitude(V)');

subplot(4,2,8)

plot(tx3,y3);

grid on;

title('Recovered Signal sampled at fs=3\*fm Hz');

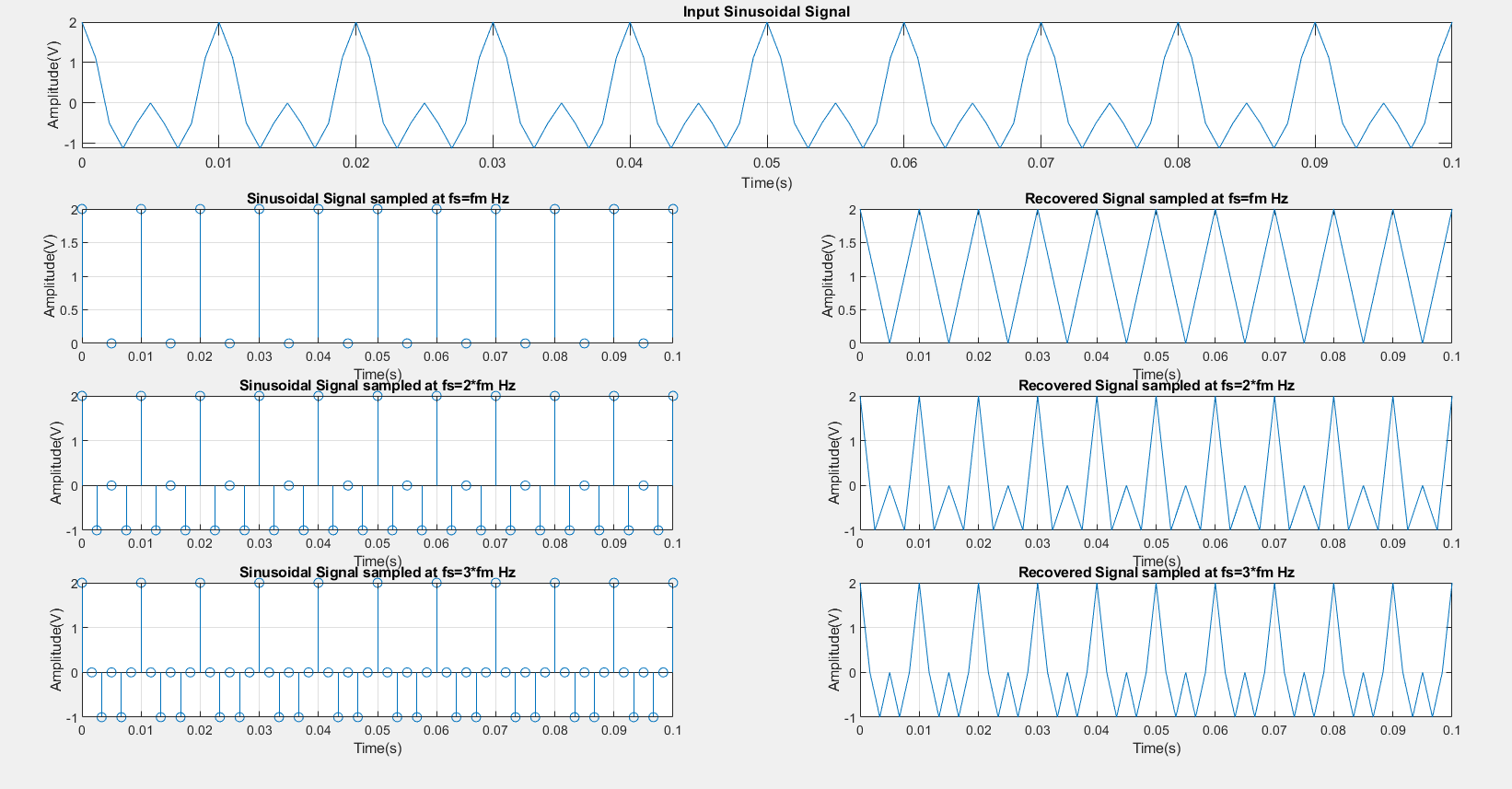
xlabel('Time(s)');

ylabel('Amplitude(V)');

**Result**

Enter the Input Frequency1 = 100

Enter the Input Frequency2 = 200

****

**Experiment No 2**

**ii) Linear Convolution**

clear all;

close all;

x=input('Enter the first input sequence x[n] ');

h=input('Enter the first input sequence h[n] ');

Lx=length(x);

Lh=length(h);

len=Lx+Lh -1;

for n=1:len

y(n)=0;

for k=1:Lx

if((n-k)>=0&(n-k)<Lh)

y(n)=y(n)+x(k).\*h(n-k+1);

end

end

end

disp('Linear Convolution of x[n] & h[n] is ')

disp(y)

**Result**

Enter the first input sequence x[n] [1 2 3 4]

Enter the first input sequence h[n] [1 0 1]

Linear Convolution of x[n] & h[n] is

1 2 4 6 3 4

**ii) Circular Convolution**

clear all;

close all;

x1=input('Enter the first input sequence x1[n] ');

x2=input('Enter the first input sequence x2[n] ');

Lx1=length(x1);

Lx2=length(x2);

len=max(Lx1,Lx2);

if Lx1<len

x1=[x1,zeros(len-Lx1)];

else

x2=[x2,zeros(len-Lx2)];

end

for n=1:len

y(n)=0;

for k=1:len

i=n-k+1;

if(i<=0)

i=i+len;

end

y(n)=y(n)+x2(k)\*x1(i);

end

end

disp('Circular Convolution of x1[n] & x2[n] is ')

disp(y)

**Result**

Enter the first input sequence x1[n] [1 2 3 4]

Enter the first input sequence x2[n] [1 0 1]

Circular Convolution of x1[n] & x2[n] is

4 6 4 6

**iii) Autocorrelation**

clear all;

close all;

x=input('Enter the input sequence x[n]');

Lx=length(x)-1;

h=fliplr(x);

rxx=conv(x,h);

disp('Auto Corelation of x[n] is rxx[n]')

disp(rxx)

%Verification using builtin function xcorr()

z=xcorr(x,x);

disp('Auto Corelation of x[n] using builtin function is z[n]')

disp(z)

%Auto correlation using for loop

len=2\*Lx+1;

for n=1:len

Rxx(n)=0;

for k=1:Lx+1

if((n-k)>=0 & (n-k)<=Lx)

Rxx(n)=Rxx(n)+x(k).\*h(n-k+1);

end

end

end

disp('Auto Corelation of x[n] is Rxx[n]')

disp(Rxx)

%Ploting the graph

a=0:Lx;

subplot(2,2,1)

stem(a,x)

title('Input Sequence x[n]')

xlabel('Samples')

ylabel('Amplitude')

b=(-Lx):Lx;

subplot(2,2,2)

stem(b,Rxx)

title('Auto Correlation rxx[n]')

xlabel('Samples')

ylabel('Amplitude')

subplot(2,2,3)

stem(b,z)

title('Auto Correlation using builtin function z[n]')

xlabel('Samples')

ylabel('Amplitude')

subplot(2,2,4)

stem(b,z)

title('Auto Correlation Rxx[n]')

xlabel('Samples')

ylabel('Amplitude')

**Result**

Enter the input sequence x[n][1 2 3 4]

Auto Corelation of x[n] is rxx[n]

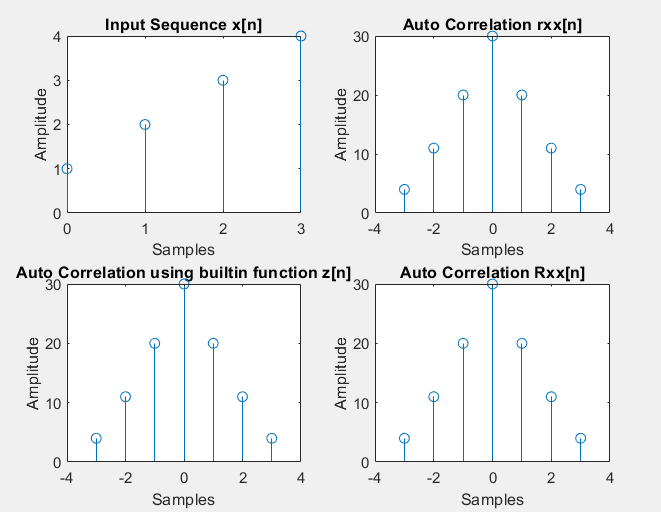
4 11 20 30 20 11 4

Auto Corelation of x[n] using builtin function is z[n]

4.0000 11.0000 20.0000 30.0000 20.0000 11.0000 4.0000

Auto Corelation of x[n] is Rxx[n]

4 11 20 30 20 11 4

****

**iv) Cross Correlation**

clear all;

close all;

x=input('Enter the input sequence x[n]');

Lx=length(x)-1;

h=input('Enter the second sequence h[n]');

Lh=length(h)-1;

y=fliplr(h);

Rxy=conv(x,y);

disp('Cross Correlation is Rxy[n]')

disp(Rxy)

%Verification using builtin function xcorr()

z=xcorr(x,h);

disp('Cross Correlation using builtin function is z[n]')

disp(z)

%Cross correlation using for loop

len=Lx+Lh+1;

for n=1:len

rxx(n)=0;

for k=1:Lx+1

if((n-k)>=0 & (n-k)<=Lh)

rxx(n)=rxx(n)+x(k).\*y(n-k+1);

end

end

end

disp('Cross Correlation of x[n] is Z[n]')

disp(rxx)

a=0:Lx;

subplot(3,2,1)

stem(a,x)

title('First Input Sequence x[n]')

xlabel('Samples')

ylabel('Amplitude')

b=0:Lh;

subplot(3,2,2)

stem(b,h)

title('Second Input Sequence y[n]')

xlabel('Samples')

ylabel('Amplitude')

c=(-Lx):Lx;

subplot(3,2,3)

stem(c,Rxy)

title('Cross Correlation Rxy[n] using builtin function xcorr()')

xlabel('Samples')

ylabel('Amplitude')

subplot(3,2,4)

stem(c,z)

title('Cross Correlation using builtin function xcorr() z[n]')

xlabel('Samples')

ylabel('Amplitude')

subplot(3,2,5:6)

stem(c,rxx)

title('Cross Correlation rxx[n]')

xlabel('Samples')

ylabel('Amplitude')

**Result**

Enter the input sequence x[n][1 2 3 4]

Enter the second sequence h[n][4 3 2 1]

Cross Correlation is Rxy[n]

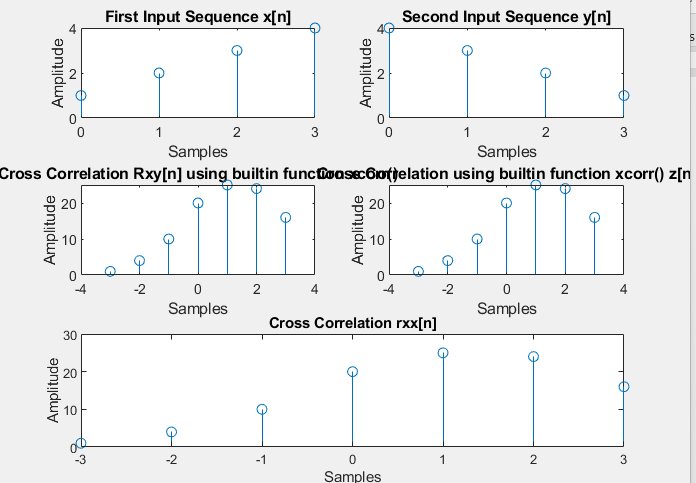
1 4 10 20 25 24 16

Cross Correlation using builtin function is z[n]

1.0000 4.0000 10.0000 20.0000 25.0000 24.0000 16.0000

Cross Correlation of x[n] is Z[n]

1 4 10 20 25 24 16



**Experiment No 3**

**To find DFT and IDFT of a sequence**

close all;

clear all;

x=input(‘Enter the input sequence x[n]=')

L=length(x)

N=input('Enter the length of the DFT sequence N =');

if(N<L)

disp('N should be greater than L')

else

%Wn=-j\*2\*pi/N;

for k=0:N-1

X(k+1)=0;

for n=0:L-1

X(k+1)=X(k+1)+x(n+1)\*exp(-j\*2\*pi\*n\*k/N);

end

end

disp('DFT of x[n] is X(K)=')

disp(X)

end

%Verification

disp('DFT using built in function')

Y=fft(x,N)

disp(Y)

%Inverse DFT

for n=0:N-1

y(n+1)=0;

for k=0:L-1

y(n+1)=y(n+1)+X(k+1)\*exp(j\*2\*pi\*n\*k/N);

end

y(n+1)=y(n+1)/N;

end

disp('IDFT of X(K)is y(n)=')

disp(y)

%Verification

disp('IDFT using built in function ')

Z=ifft(X)

disp(Z)

a=0:L-1;

subplot(3,2,1)

stem(a,x)

grid on

title('Input Sequence x[n]')

xlabel('Samples')

ylabel('Values')

b=0:N-1;

subplot(3,2,2)

stem(b,X)

grid on

title('DFT Sequence X(K)')

xlabel('Samples')

ylabel('Values')

subplot(3,2,3)

stem(b,abs(X))

grid on

title('DFT magnitude')

xlabel('Samples')

ylabel('Values')

subplot(3,2,4)

stem(b,angle(X))

grid on

title('DFT phase angle')

xlabel('Samples')

ylabel('Values')

subplot(3,2,5)

stem(y)

grid on

title('IDFT sequence y[n]')

xlabel('Samples')

ylabel('Values')

**Result:**

Enter the input sequence x[n]=[1 2 3 4]

x =

1 2 3 4

L =

4

Enter the length of the DFT sequence N =4

DFT of x[n] is X(K)=

10.0000 + 0.0000i -2.0000 + 2.0000i -2.0000 - 0.0000i -2.0000 - 2.0000i

DFT using built in function

Y =

10.0000 + 0.0000i -2.0000 + 2.0000i -2.0000 + 0.0000i -2.0000 - 2.0000i

10.0000 + 0.0000i -2.0000 + 2.0000i -2.0000 + 0.0000i -2.0000 - 2.0000i

IDFT of X(K)is y(n)=

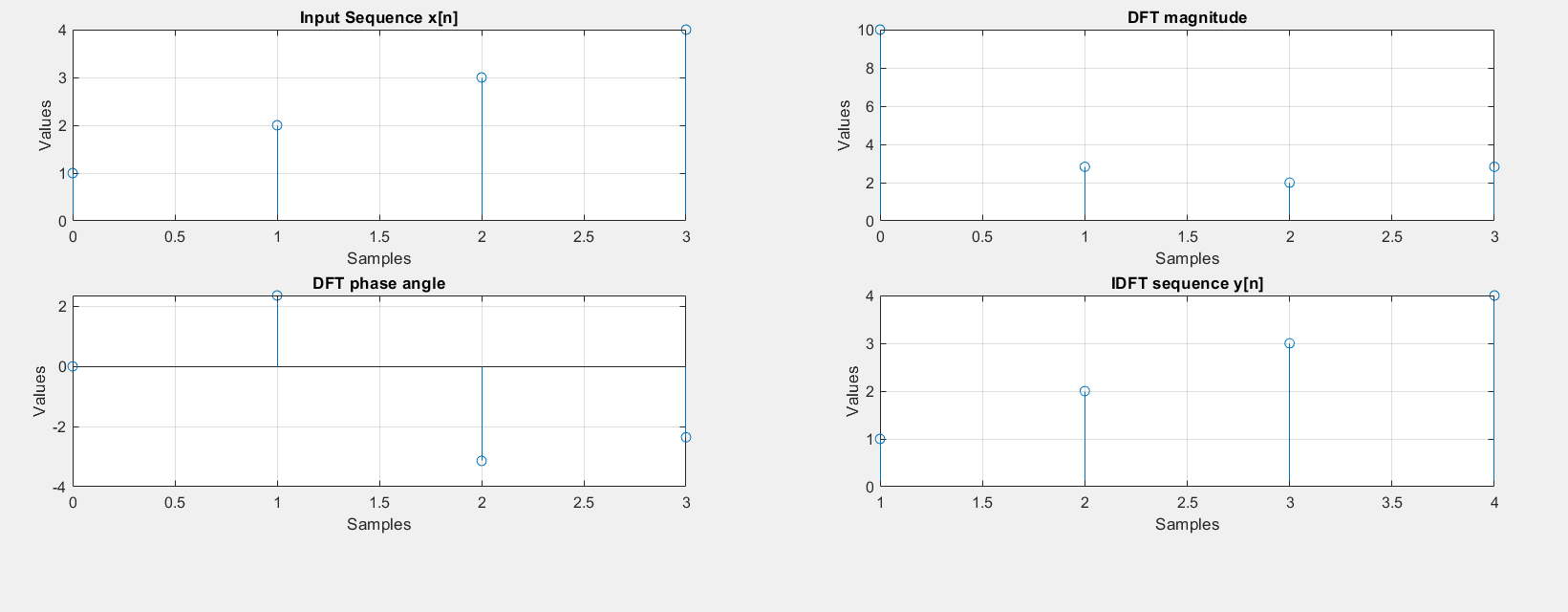
1.0000 - 0.0000i 2.0000 - 0.0000i 3.0000 - 0.0000i 4.0000 + 0.0000i

IDFT using built in function

Z =

1.0000 - 0.0000i 2.0000 - 0.0000i 3.0000 + 0.0000i 4.0000 + 0.0000i

1.0000 - 0.0000i 2.0000 - 0.0000i 3.0000 + 0.0000i 4.0000 + 0.0000i



**ii) To find FFT and IFFT of a sequence**

close all;

clear all;

x=input('Enter the input Sequnce x[n]');

L=length(x);

N=input('Enter the length of the FFT sequence N =');

if(N<L)

disp('N should be greater than L')

else

x=[x zeros(1,N-L)];

end

%Plotting the Input Sequence

d=0:N-1;

subplot(3,2,1)

stem(d,x)

title('Input Sequence x[n]');

%To alter the input sequence ie x[0] x[2] x[1] x[3]

x=bitrevorder(x);

M=log2(N);

h=1;

for stage=1:M

for index=0:(2^stage):N-1

for n=0:(h-1)

pos=n+index+1;

pow=(2^(M-stage)\*n);

w=exp((-i)\*(2\*pi)\*pow/N);

a=x(pos)+x(pos+h).\*w;

b=x(pos)-x(pos+h).\*w;

x(pos)=a;

x(pos+h)=b;

end

end

h=2\*h;

end

y=x

disp(y);

%Plotting the FFT Sequence

subplot(3,2,2)

stem(d,abs(y))

grid on

title('FFT magnitude')

xlabel('Samples')

ylabel('Values')

subplot(3,2,3)

stem(d,angle(y))

grid on

title('FFT phase angle')

xlabel('Samples')

ylabel('Values')

y=bitrevorder(y);

h=1;

for stage=1:M

for index=0:(2^stage):N-1

for n=0:(h-1)

pos=n+index+1;

pow=(2^(M-stage)\*n);

w=exp((i)\*(2\*pi)\*pow/N);

a=y(pos)+y(pos+h).\*w;

b=y(pos)-y(pos+h).\*w;

y(pos)=a;

y(pos+h)=b;

end

end

h=2\*h;

end

z=y/N

disp(z)

%Plotting the IFFT Sequence

subplot(3,2,4)

stem(d,z)

title('IFFT Sequence z[n]');

**Result:**

Enter the input Sequnce x[n][1 2 3 4]

Enter the length of the FFT sequence N =4

y =

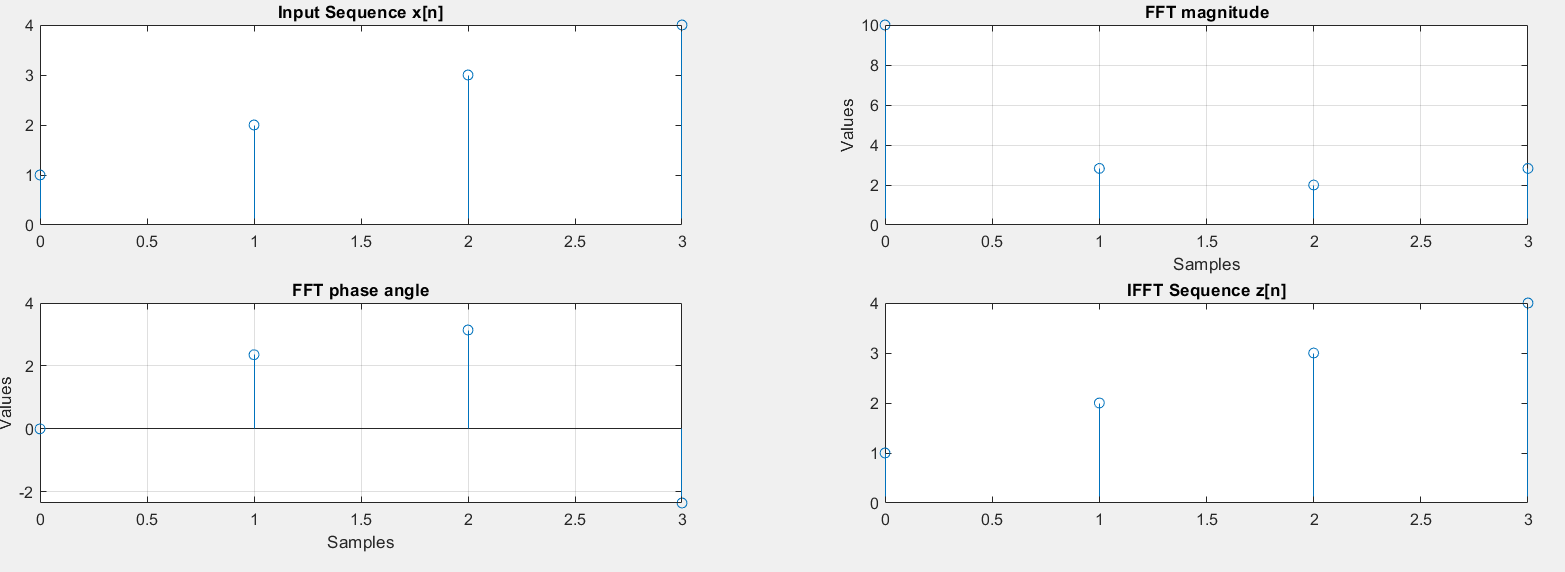
10.0000 + 0.0000i -2.0000 + 2.0000i -2.0000 + 0.0000i -2.0000 - 2.0000i

10.0000 + 0.0000i -2.0000 + 2.0000i -2.0000 + 0.0000i -2.0000 - 2.0000i

z =

1.0000 + 0.0000i 2.0000 + 0.0000i 3.0000 + 0.0000i 4.0000 - 0.0000i

1.0000 + 0.0000i 2.0000 + 0.0000i 3.0000 + 0.0000i 4.0000 - 0.0000i

****

**iii)To find Linear and Circular Convolution using FFT algorithm**

**Linear Convolution**

clear all;

close all;

x1=input('Enter the first input sequence x1[n] ');

x2=input('Enter the second input sequence x2[n] ');

Lx1=length(x1);

Lx2=length(x2);

N=Lx1+Lx2-1;

X1=fft(x1,N);

X2=fft(x2,N);

Y=X1.\*X2;

y=ifft(Y,N);

disp('Linear Convolution of x1[n] and x2[n] is y[n]= ')

disp(y)

%Verification

z=conv(x1,x2);

disp('Linear Convolution of x1[n] and x2[n] using builtin function is z[n]= ')

disp(z)

a=0:Lx1-1;

subplot(2,2,1)

stem(a,x1)

title('Input Sequence x1[n]')

xlabel('Samples')

ylabel('Values')

b=0:Lx2-1;

subplot(2,2,2)

stem(b,x2)

title('Input Sequence x2[n]')

xlabel('Samples')

ylabel('Values')

c=0:N-1;

subplot(2,2,3)

stem(c,y)

title('Linear Convolution of x1[n] and x2[n]')

xlabel('Samples')

ylabel('Values')

d=0:N-1;

subplot(2,2,4)

stem(c,z)

title('Linear Convolution of x1[n] and x2[n]using builtin function')

xlabel('Samples')

ylabel('Values')

**Result:**

Enter the first input sequence x1[n] [1 2 3 4]

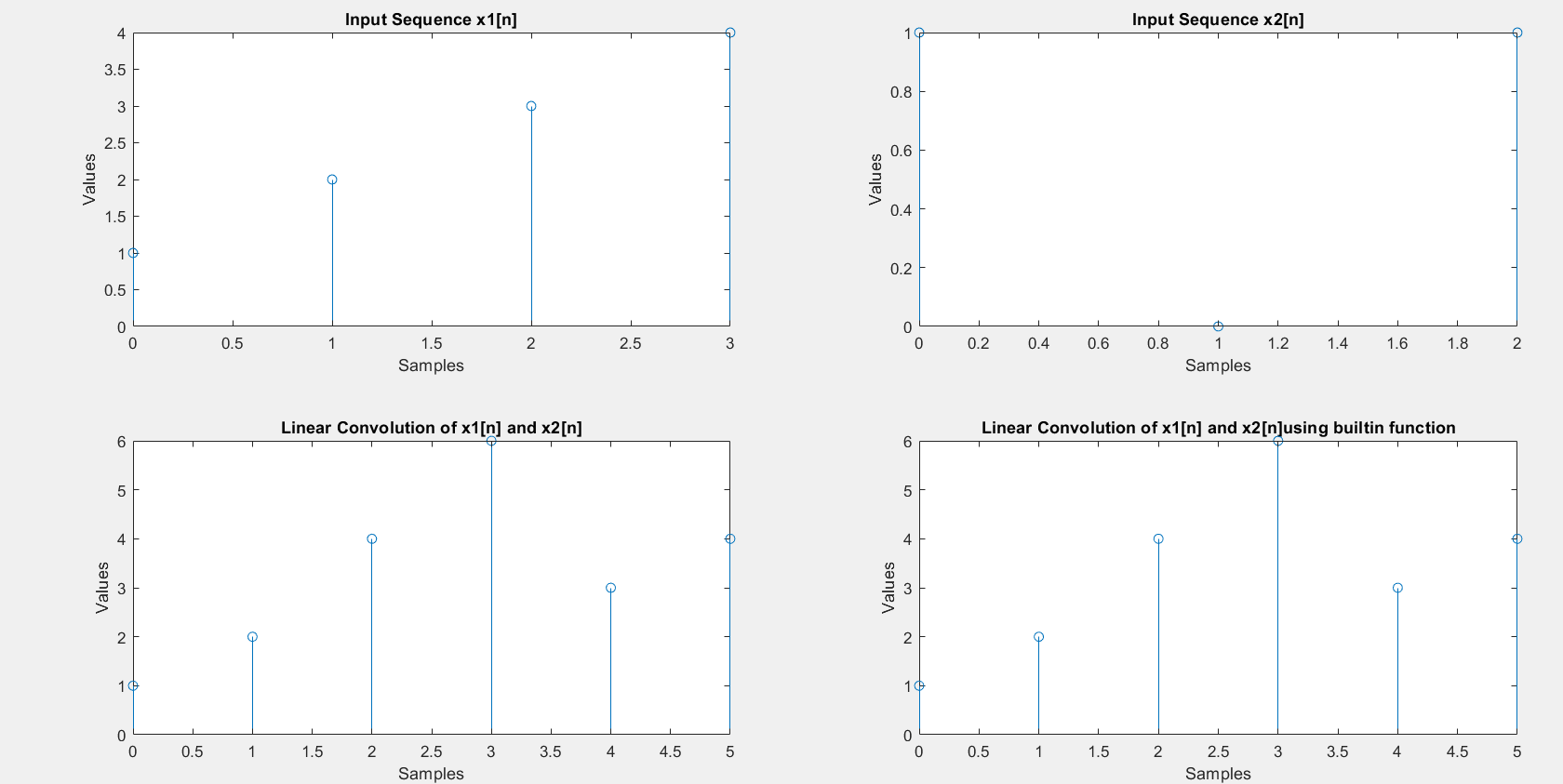
Enter the second input sequence x2[n] [1 0 1]

Linear Convolution of x1[n] and x2[n] is y[n]=

1 2 4 6 3 4

Linear Convolution of x1[n] and x2[n] using builtin function is z[n]=

1 2 4 6 3 4



**Circular Convolution**

clear all;

close all;

x1=input('Enter the first input sequence x1[n] ');

x2=input('Enter the second input sequence x2[n] ');

Lx1=length(x1);

Lx2=length(x2);

N=max(Lx1,Lx2);

X1=fft(x1,N);

X2=fft(x2,N);

Y=X1.\*X2;

y=ifft(Y,N);

disp('Circular Convolution of x1[n] and x2[n] is y[n]= ')

disp(y)

%Verification

z=cconv(x1,x2,N);

disp('Circular Convolution of x1[n] and x2[n] using builtin function is z[n]= ')

disp(z)

a=0:Lx1-1;

subplot(2,2,1)

stem(a,x1)

title('Input Sequence x1[n]')

xlabel('Samples')

ylabel('Values')

b=0:Lx2-1;

subplot(2,2,2)

stem(b,x2)

title('Input Sequence x2[n]')

xlabel('Samples')

ylabel('Values')

c=0:N-1;

subplot(2,2,3)

stem(c,y)

title('Circular Convolution of x1[n] and x2[n]')

xlabel('Samples')

ylabel('Values')

d=0:N-1;

subplot(2,2,4)

stem(c,z)

title('Circular Convolution of x1[n] and x2[n]using builtin function')

xlabel('Samples')

ylabel('Values')

**Result:**

Enter the first input sequence x1[n] [1 2 3 4]

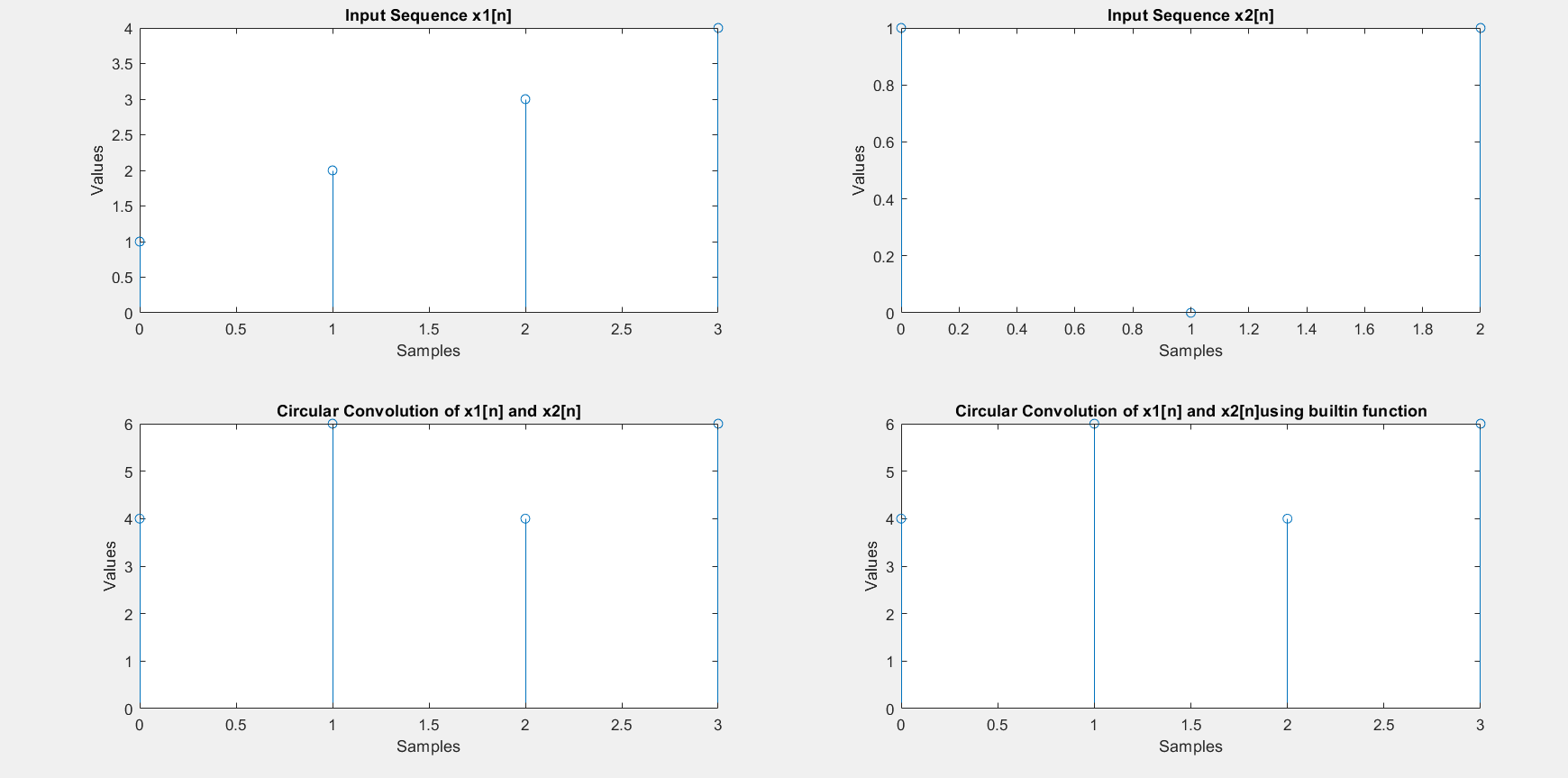
Enter the second input sequence x2[n] [1 0 1]

Circular Convolution of x1[n] and x2[n] is y[n]=

4 6 4 6

Circular Convolution of x1[n] and x2[n] using builtin function is z[n]=

4 6 4 6



**iv)To find Linear and Circular Convolution using FFT algorithm**

**Linear Convolution**

clear all;

close all;

x1=input('Enter the first input sequence x1[n] ');

x2=input('Enter the second input sequence x2[n] ');

Lx1=length(x1);

Lx2=length(x2);

N=Lx1+Lx2-1;

X1=FFT\_L(x1,N);

X2=FFT\_L(x2,N);

Y=X1.\*X2;

y=IFFT\_L(Y,N);

disp('Linear Convolution of x1[n] & x2[n] is ')

disp(y)

**Result**

Enter the first input sequence x1[n] [1 2 3 4]

Enter the second input sequence x2[n] [1 1 1 1]

Linear Convolution of x1[n] & x2[n] is

1.0000 - 0.0000i 3.0000 - 0.0000i 6.0000 - 0.0000i 10.0000 + 0.0000i 9.0000 + 0.0000i 7.0000 + 0.0000i 4.0000 + 0.0000i 0.0000 - 0.0000i

**Circular convolution**

clear all;

close all;

x1=input('Enter the first input sequence x1[n]');

x2=input('Enter the first input sequence x2[n]');

Lx1=length(x1);

Lx2=length(x2);

N=max(Lx1,Lx2);

if Lx1<N

x1=[x1,zeros(N-Lx1)];

else

x2=[x2,zeros(N-Lx2)];

end

X1=FFT\_L(x1,N);

X2=FFT\_L(x2,N);

Y=X1.\*X2;

y=IFFT\_L(Y,N);

disp('Circular Convolution of x1[n] & x2[n] is ')

disp(y)

%Verification

z=cconv(x1,x2,N);

disp('Circular Convolution of x1[n] and x2[n] using builtin function is z[n]= ')

disp(z)

**Result**

**Enter the first input sequence x1[n][1 2 3 4]**

**Enter the first input sequence x2[n][1 0 1]**

**Circular Convolution of x1[n] & x2[n] is**

**4 6 4 6**

**Circular Convolution of x1[n] and x2[n] using builtin function is z[n]=**

**4 6 4 6**

**Functions**

**FFT**

function y=FFT\_L(x,N)

L=length(x);

M=nextpow2(N);

R=rem(N,2);

if(R~=0)

x=[x zeros(1,(2^M)-L)];

end

%To alter the input sequence ie x[0] x[2] x[1] x[3]

x=bitrevorder(x);

h=1;

N=2^M;

for stage=1:M

for index=0:(2^stage):N-1

for n=0:(h-1)

pos=n+index+1;

pow=(2^(M-stage)\*n);

w=exp((-i)\*(2\*pi)\*pow/N);

a=x(pos)+x(pos+h).\*w;

b=x(pos)-x(pos+h).\*w;

x(pos)=a;

x(pos+h)=b;

end

end

h=2\*h;

end

y=x;

**IIFT**

function z=IFFT\_L(y,N)

L=length(y);

M=nextpow2(N);

R=rem(N,2);

if(R~=0)

y=[y zeros(1,(2^M)-L)];

end

y=bitrevorder(y);

h=1;

N=2^M;

for stage=1:M

for index=0:(2^stage):N-1

for n=0:(h-1)

pos=n+index+1;

pow=(2^(M-stage)\*n);

w=exp((i)\*(2\*pi)\*pow/N);

a=y(pos)+y(pos+h).\*w;

b=y(pos)-y(pos+h).\*w;

y(pos)=a;

y(pos+h)=b;

end

end

h=2\*h;

end

z=y/N;

**Experiment No 4**

**i) To find the 2N point DFT using N point DFT**

close all;

clear all;

v=input('Enter the Input sequence v[n]');

N=length(v)/2

for n=0:N-1

g(n+1)=v((2\*n)+1);

h(n+1)=v((2\*n)+2);

end

[G,H]=myN\_Point(g,h);

for k=0:(2\*N)-1

w(k+1)=exp((-i\*pi\*k)/N);

end

m=0;

for k=1:2

for n=0:N-1

V(m+n+1)=G(n+1)+(w(m+n+1)\*H(n+1));

end

m=4;

end

disp('2N -Point DFT using N point DFT is V(K)')

disp(V)

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

function[G,H]=myN\_Point(g,h)

N=length(g);

%x[n]= g[n]+jh[n]

for i=0:N-1

x(i+1)=g(i+1)+h(i+1)\*j;

end

%Finding DFT of x[n] i.e X[K]

X =mydftusingfft(x,N);

%Finding Conjugate of X(K) i.e X\*(K)

Z=conj(X);

%X\*[(N-K)N]

%n=0:N-1

%Z(mod(-n,N)+1)

for k=0:N-1

n=N-k;

if(n==N)

y(k+1)=Z(k+1);

else

y(k+1)=Z(n+1);

end

end

G=zeros(1,N);

H=zeros(1,N);

for k=1:N

G(k)=(1/2)\*(X(k)+y(k));

H(k)=(X(k)-y(k))/(2\*j)

end

end

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

function y=mydftusingfft(x,N)

%to alter the input sequence is X[0] x[1] x[3]

x=bitrevorder(x);

% To find n from 2^n i.e No of stages

%p=nextpow2(N);

M=log2(N);

h=1;

for stage = 1:M

for index = 0:(2^stage):N-1

for n=0:(h-1)

pos =n+index+1;

pow= (2^(M-stage)\*n);

w=exp((-i)\*(2\*pi)\*pow/N);

a=x(pos)+x(pos+h).\*w;

b=x(pos)-x(pos+h).\*w;

x(pos)=a;

x(pos+h)=b;

end

end

h=2\*h;

end

y=x;

end

**Result**

Enter the Input sequence v[n] [1 2 2 2 0 1 1 1]

N =

4

H =

6 0 0 0

H =

6.0000 + 0.0000i 1.0000 - 1.0000i 0.0000 + 0.0000i 0.0000 + 0.0000i

H =

6.0000 + 0.0000i 1.0000 - 1.0000i 0.0000 + 0.0000i 0.0000 + 0.0000i

H =

6.0000 + 0.0000i 1.0000 - 1.0000i 0.0000 + 0.0000i 1.0000 + 1.0000i

2N -Point DFT using N point DFT is V(K)

10.0000 + 0.0000i 1.0000 - 2.4142i -2.0000 + 0.0000i 1.0000 - 0.4142i -2.0000 - 0.0000i 1.0000 + 0.4142i -2.0000 + 0.0000i 1.0000 + 2.4142i

**ii)To find the N point DFT of two sequences using single N point DFT(Using FFT algorithm to find DFT)**

close all;

clear all;

g=input('Enter the first sequence g[n]');

h= input('Enter the second sequence h[n]');

N=length(g)

for i=0:N-1

x(i+1)=g(i+1)+h(i+1)\*j;

end

disp(x)

%Finding DFT of x[n] i.e X[K]

X =mydftusingfft(x,N)

%Finding Conjugate of X(K) i.e X\*(K)

Z=conj(X)

%X\*[(N-K)N]

%n=0:N-1

%Z(mod(-n,N)+1)

for k=0:N-1

n=N-k;

if(n==N)

y(k+1)=Z(k+1);

else

y(k+1)=Z(n+1);

end

end

disp(y)

G=zeros(1,N);

H=zeros(1,N);

for k=1:N

G(k)=(1/2)\*(X(k)+y(k));

H(k)=(X(k)-y(k))/(2\*j);

end

disp('N Point DFT of two sequences using N point DFT')

disp(G)

disp(H)

function y=mydftusingfft(x,N)

%to alter the input sequence is X[0] x[1] x[3]

x=bitrevorder(x);

% To find n from 2^n i.e No of stages

%p=nextpow2(N);

M=log2(N);

h=1;

for stage = 1:M

for index = 0:(2^stage):N-1

for n=0:(h-1)

pos =n+index+1;

pow= (2^(M-stage)\*n);

w=exp((-i)\*(2\*pi)\*pow/N);

a=x(pos)+x(pos+h).\*w;

b=x(pos)-x(pos+h).\*w;

x(pos)=a;

x(pos+h)=b;

end

end

h=2\*h;

end

y=x;

end

**Result:**

Enter the first sequence g[n] [1 2 0 1]

Enter the second sequence h[n] [ 2 2 1 1]

N =

4

1.0000 + 2.0000i 2.0000 + 2.0000i 0.0000 + 1.0000i 1.0000 + 1.0000i

X =

4.0000 + 6.0000i 2.0000 + 0.0000i -2.0000 + 0.0000i 0.0000 + 2.0000i

Z =

4.0000 - 6.0000i 2.0000 - 0.0000i -2.0000 + 0.0000i 0.0000 - 2.0000i

4.0000 - 6.0000i 0.0000 - 2.0000i -2.0000 + 0.0000i 2.0000 - 0.0000i

N Point DFT of two sequences using N point DFT

4.0000 + 0.0000i 1.0000 - 1.0000i -2.0000 + 0.0000i 1.0000 + 1.0000i

6.0000 + 0.0000i 1.0000 - 1.0000i 0.0000 + 0.0000i 1.0000 + 1.0000i

**Experiment 5**

**Sectioned Convolution: Overlap Save and Overlap Add Method for long Duration Sequences**

**Overlap save**

clc;

clear all;

close all;

x=input('Enter 1st Sequence X(n)= ');

h=input('Enter 2nd Sequence H(n)= ');

N=input('Enter length of each block N = ');

% Code to plot X(n)

subplot(2,2,1);

stem(x,'blue');

xlabel ('n---->');

ylabel ('Amplitude ---->');

title('X(n)');

%Code to plot H(n)

subplot(2,2,2);

stem(h,'black');

xlabel ('n---->');

ylabel ('Amplitude ---->');

title(' H(n)');

% Code to perform Convolution using Overlap Save Method

lx=length(x);

lh=length(h);

m=lh-1;

x=[zeros(1,m) x zeros(1,N)];

h=[h zeros(1,N-lh)];

L=N-lh+1;

k=floor(lx/L);

for i=0:k

y=x(1,i\*L+1:i\*L+N);

q=cconv(y,h,N)

%q=C\_Conv(y,h); %Call the C\_Conv function.

p(i+1,:)=q;

end

p1=p(:,lh:N)';

p=p1(:)'

% Representation of the Convoled Signal

subplot(2,2,3:4);

stem(p,'red');

xlabel ('n---->');

ylabel ('Amplitude ---->');

title('Convoled Signal');

**Output**

**Enter 1st Sequence X(n)= [3,-1,0,1,3,2,0,1,2,1]**

**Enter 2nd Sequence H(n)= [1, 1,1]**

**Enter length of each block L = 8**

**p =**

**3 2 2 0 4 6 5 3 3 4 3 1**



**Overlap add**

close all;

clear all;

x=input('Enter First Sequence x[n]= ');

h=input('Enter Second Sequence h[n]= ');

N=input('Enter length of each block N = ');

Lx=length(x);

M=length(h);

L=N-M+1;

K=ceil(Lx/L)

R=rem(Lx,L);

%Padding zeros to input sequences to make length equal to N

if R>0

x=[x zeros(1,L-R)]

end

h=[h zeros(1,N-M)]

%Initialising the Output

y=zeros(N,K);

%Padding zeros to Input sequence at the end of the sequence

z=zeros(1,M-1);

%To perform Circular Convolution of two input sequences

for i=0:K-1

Xn=x(L\*i+1:L\*i+L);

Xi=[Xn z];

u(i+1,:)=cconv(Xi,h,N) %u(i+1,:)=C\_Conv(Xi(i,:),h);

end

Y=u';

M1=M-1;

p=L+M1;

for i=1:K-1

u(i+1,1:M-1)=u(i,p-M1+1:p)+u(i+1,1:M-1);

end

z1=u(:,1:L)'

y1=(z1(:))'

y=[y1 u(K,(M:N))]

%Ploting the Input Sequences

subplot (2,2,1);

stem(x);

title('First Sequence x[n]');

xlabel ('Samples');

ylabel ('Amplitude');

subplot (2,2,2);

stem(h);

title('Second Sequence h[n]');

xlabel ('Samples');

ylabel ('Amplitude');

%Plotting of the Convoled Signal

subplot (2,2,3:4);

stem(y);

title ('Convolved Signal');

xlabel ('Samples');

ylabel ('Amplitude');

**Result**

**Enter First Sequence x[n]= [1 2 -1 2 3 -2 -3 -1 1 1 2 -1]**

**Enter Second Sequence h[n]= [1 2]**

**Enter length of each block N = 4**

**K =**

**4**

**h =**

**1 2 0 0**

**u =**

**1 4 3 -2**

**u =**

**1 4 3 -2**

**2 7 4 -4**

**u =**

**1 4 3 -2**

**2 7 4 -4**

**-3 -7 -1 2**

**u =**

**1 4 3 -2**

**2 7 4 -4**

**-3 -7 -1 2**

**1 4 3 -2**

**z1 =**

**1 0 -7 3**

**4 7 -7 4**

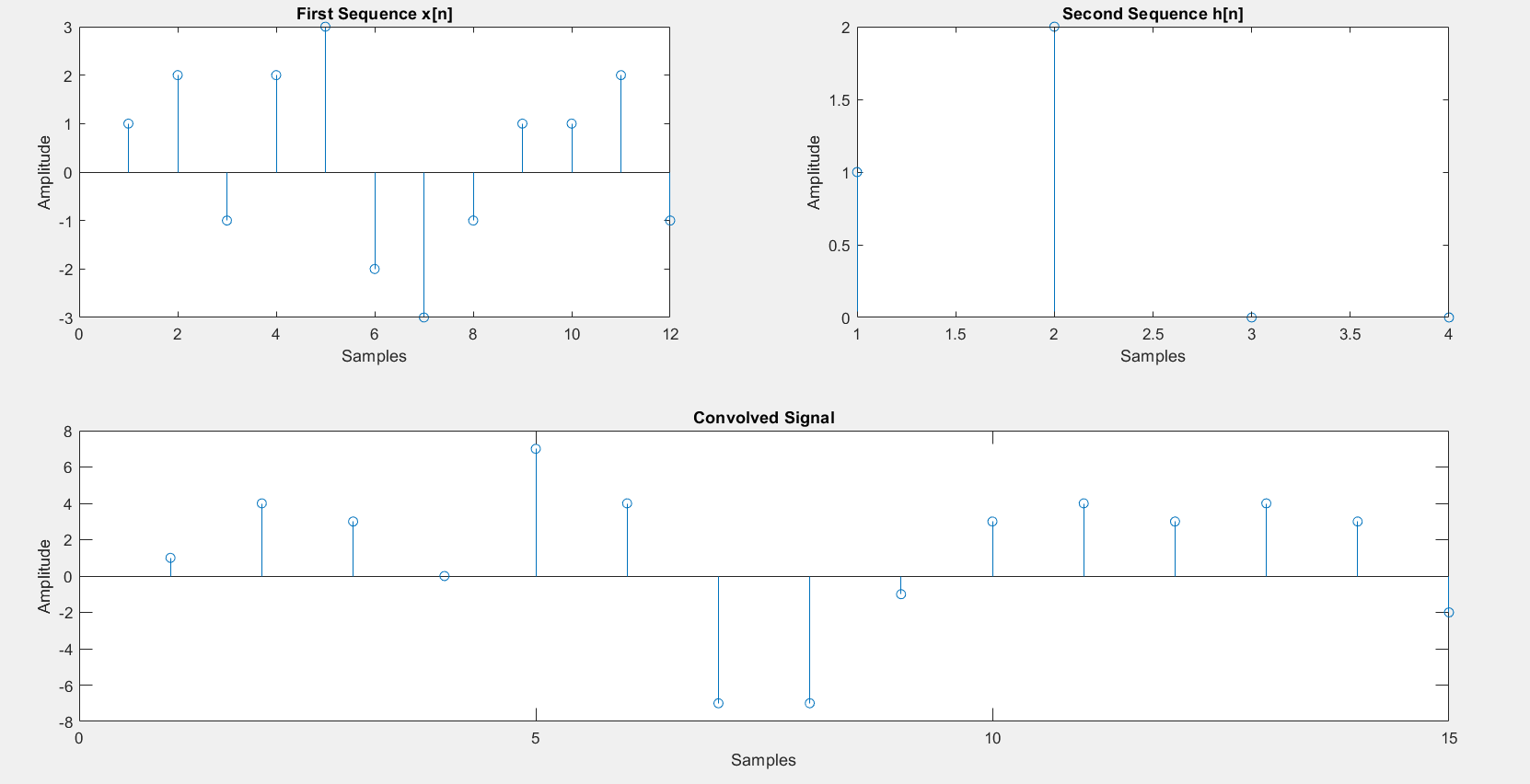
**3 4 -1 3**

**y1 =**

**1 4 3 0 7 4 -7 -7 -1 3 4 3**

**y =**

**1 4 3 0 7 4 -7 -7 -1 3 4 3 4 3 -2**

****