**NATIONAL INSTITUTE OF TECHNOLOGY CALICUT**

**DEPARTMENT OF ELECTRONICS AND COMMUNICATION ENGINEERING**

****

**EC 4091 : DIGITAL SIGNAL PROCESSING LABORATORY**

EXPERIMENT No. 1

Submitted by,

Anjali Venugopal B150071EC

Anju Anand B150303EC

Ankit Kumar B150912EC

Aravind Krishnan B150829EC

**Experiment No. 1**

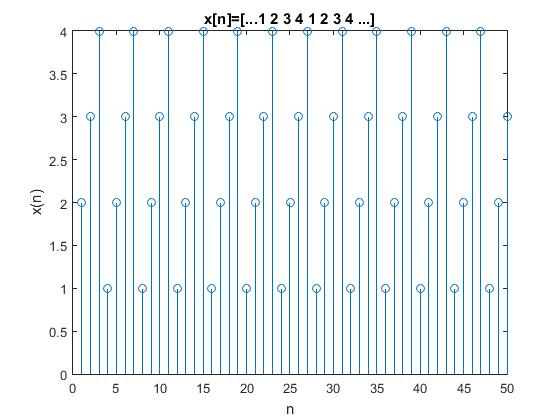
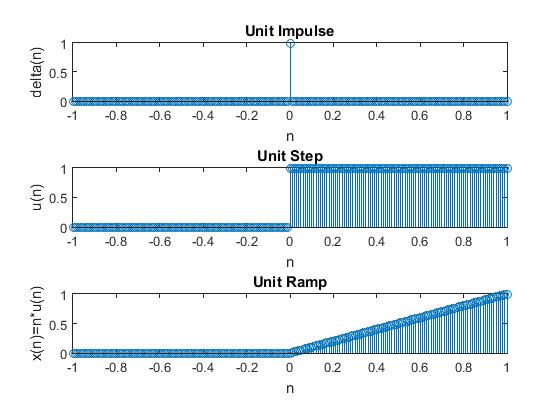
**AIM**

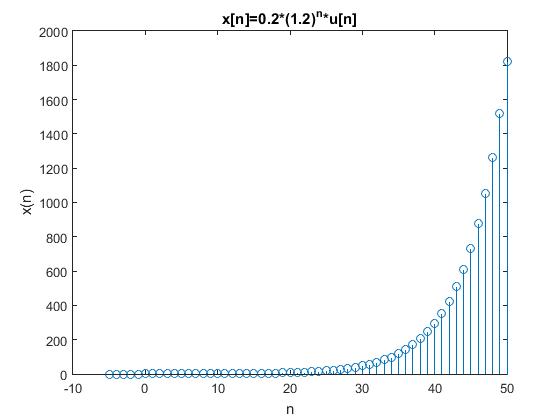
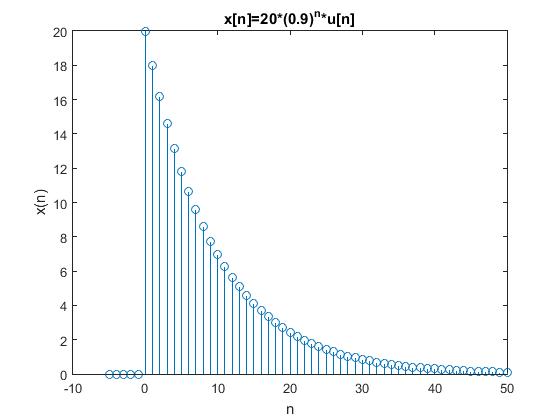
Write MATLAB codes for each question.

**THEORY**

**Frequency calculation in discrete domain :**

The frequency of x[n] is the smallest integer k such that x[n+kN]=x[n].





**MATLAB CODE**

%Generate unit impulse, unit step, unit ramp functions.

clc;

t=(-1:0.01:1);

imp = t==0;

figure;

subplot(3,1,1);

stem(t,imp)

title('Unit Impulse');

xlabel('n');

ylabel('delta(n)');

unitstep= t>=0;

subplot(3,1,2);

stem(t,unitstep);

title('Unit Step');

xlabel('n');

ylabel('u(n)');

ramp=t.\*unitstep;

subplot(3,1,3);

stem(t,ramp);

title('Unit Ramp');

xlabel('n');

ylabel('x(n)=n\*u(n)');

%x[n]=[...1 2 3 4 1 2 3 4 ...]

n=1:1:50;

m=mod(n,4)+1;

figure;

stem(n,m);

title('x[n]=[...1 2 3 4 1 2 3 4 ...]');

xlabel('n');

ylabel('x(n)');

%x[n]=20\*(0.9)^n\*u[n]

t=(-5:50);

unitstep= t>=0;

x=20.\*(0.9).^t.\*unitstep;

figure;

stem(t,x);

title('x[n]=20\*(0.9)^n\*u[n]');

xlabel('n');

ylabel('x(n)');

%x[n]=0.2\*(1.2)^n\*u[n]

x=(0.2).\*(1.2).^t.\*unitstep;

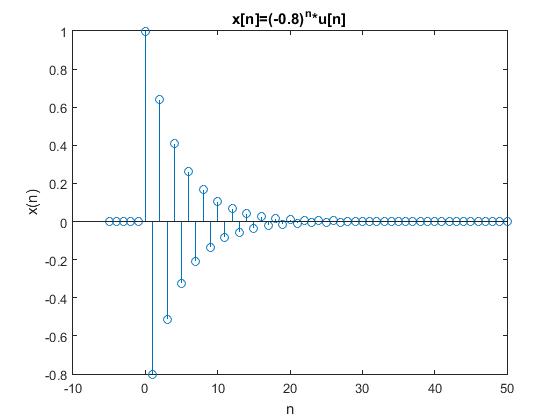
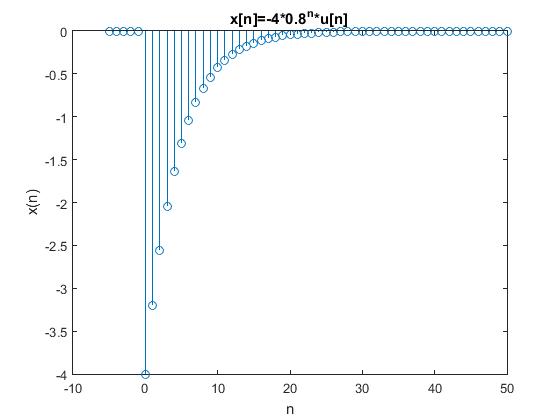
figure;

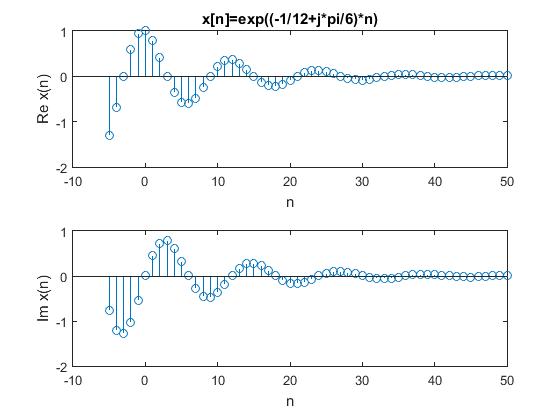
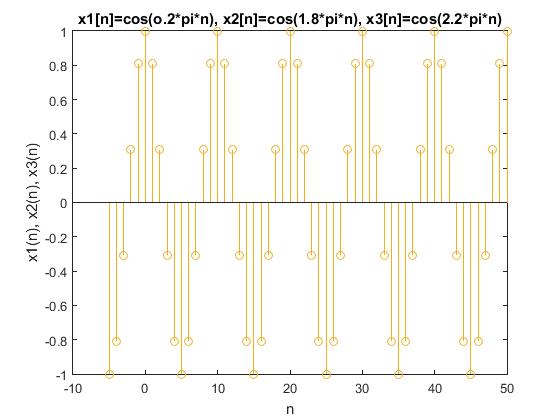
stem(t,x);

title('x[n]=0.2\*(1.2)^n\*u[n]');

xlabel('n');

ylabel('x(n)');

%x[n]=(-0.8)^n\*u[n]

x=(-0.8).^t.\*unitstep;

figure;

stem(t,x);

title('x[n]=(-0.8)^n\*u[n]');

xlabel('n');

ylabel('x(n)');

%x[n]=-4\*0.8^n\*u[n]

x=(-4).\*(0.8).^t.\*unitstep;

figure;

stem(t,x);

title('x[n]=-4\*0.8^n\*u[n]');

xlabel('n');

ylabel('x(n)');

%x[n]=e^((-1/12+j\*pi/6)\*n)

h=-1/12+j\*pi/6;

t=(-5:50);

x=exp(h\*t);

figure;

subplot(2,1,1);

stem(t,real(x));

title('x[n]=exp((-1/12+j\*pi/6)\*n)');

xlabel('n');

ylabel('Re {x(n)}');

subplot(2,1,2);

stem(t,imag(x));

xlabel('n');

ylabel('Im {x(n)}');

%x1[n]=cos(o.2\*pi\*n)

%x2[n]=cos(1.8\*pi\*n)

%x3[n]=cos(2.2\*n)

figure;

stem(t,cos(0.2\*pi\*t));

hold on

stem(t,cos(1.8\*pi\*t));

hold on

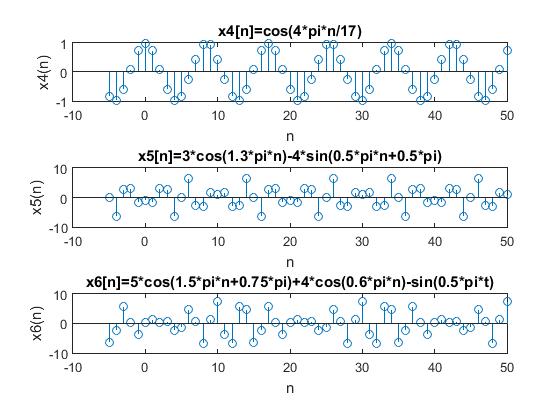
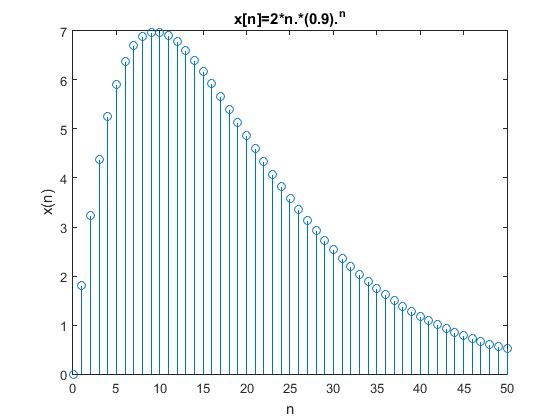
stem(t,cos(2.2\*pi\*t));

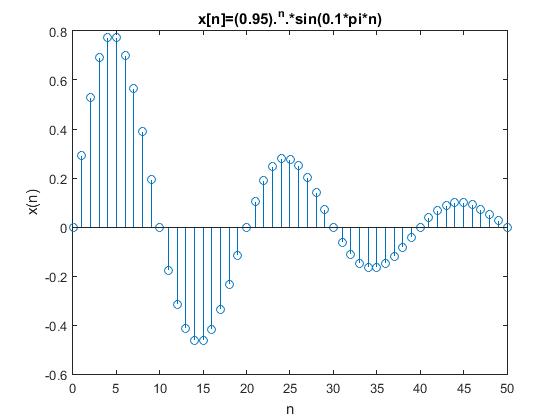
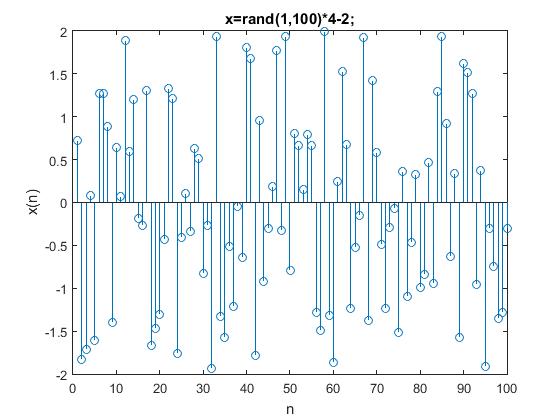
title('x1[n]=cos(o.2\*pi\*n), x2[n]=cos(1.8\*pi\*n), x3[n]=cos(2.2\*pi\*n)');

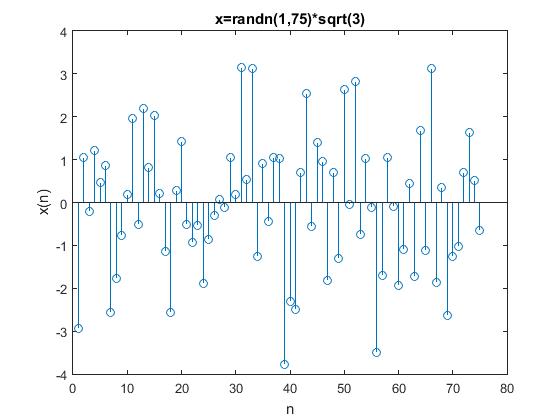
xlabel('n');

ylabel('x1(n), x2(n), x3(n)');

hold off



%x4[n]=cos(4\*pi\*n/17)

%x5[n]=3\*cos(1.3\*pi\*n)-4\*sin(0.5\*pi\*n+0.5\*pi)

%x6[n]=5\*cos(1.5\*pi\*n+0.75\*pi)+4\*cos(0.6\*pi\*n)-sin(0.5\*pi\*t)

%Verify frequency

figure;

subplot(3,1,1);

stem(t,cos(4\*pi\*t/17));

title('x4[n]=cos(4\*pi\*n/17)');

xlabel('n');

ylabel('x4(n)');

subplot(3,1,2);

stem(t,3\*cos(1.3\*pi\*t)-4\*sin(0.5\*pi\*t+0.5\*pi));

title('x5[n]=3\*cos(1.3\*pi\*n)-4\*sin(0.5\*pi\*n+0.5\*pi)');

xlabel('n');

ylabel('x5(n)');

subplot(3,1,3);

stem(t,5\*cos(1.5\*pi\*t+0.75\*pi)+4\*cos(0.6\*pi\*t)-sin(0.5\*pi\*t));

title('x6[n]=5\*cos(1.5\*pi\*n+0.75\*pi)+4\*cos(0.6\*pi\*n)-sin(0.5\*pi\*t)');

xlabel('n');

ylabel('x6(n)');

n=0:50;

%x[n]=2\*n.\*(0.9).^n

figure;

stem(n,2\*n.\*(0.9).^n);

title('x[n]=2\*n.\*(0.9).^n');

xlabel('n');

ylabel('x(n)');

%x[n]=(0.95).^n.\*sin(0.1\*pi\*n)

figure;

stem(n,(0.95).^n.\*sin(0.1\*pi\*n));

title('x[n]=(0.95).^n.\*sin(0.1\*pi\*n)');

xlabel('n');

ylabel('x(n)');

%rand and randn

x=rand(1,100)\*4-2;

y=randn(1,75)\*sqrt(3);

figure;

stem(x);

title('x=rand(1,100)\*4-2;');

xlabel('n');

ylabel('x(n)');

figure;

stem(y);

title('x=randn(1,75)\*sqrt(3)');

xlabel('n');

ylabel('x(n)');

mean(y) % -0.0668

var(y) % 2.5802

**OBSERVATIONS AND INFERENCES**

The plots generated by x1[n]=cos(o.2\*pi\*n),x2[n]=cos(1.8\*pi\*n),x3[n]=cos(2.2\*pi\*n) were found to be the same. The cosines are in discrete domain and their frequencies are the same, in this case.

The frequencies calculated for x4[n]=cos(4\*pi\*n/17),x5[n]=3\*cos(1.3\*pi\*n)-4\*sin(0.5\*pi\*n+0.5\*pi),x6[n]=5\*cos(1.5\*pi\*n+0.75\*pi)+4\*cos(0.6\*pi\*n)-sin(0.5\*pi\*t) were verified with the MATLAB figure and found to be equal.

x4[n]=cos(4\*pi\*n/17)

frequency = 17

x5[n]=3\*cos(1.3\*pi\*n)-4\*sin(0.5\*pi\*n+0.5\*pi)

frequency = gcd (20,4) = 20

x6[n]=5\*cos(1.5\*pi\*n+0.75\*pi)+4\*cos(0.6\*pi\*n)-sin(0.5\*pi\*t)

frequency = gcd (4,10,4) = 20

Given a random variable following uniform distribution with values in the range [0,1], it was expanded to the interval [-2,2] by scaling and shifting the variable.

Given a random variable following Gaussian distribution with mean 0 and variance 1, a random variable of variance 3 was constructed by scaling each value by .

**RESULT**

The MATLAB codes for each question was written.

**NATIONAL INSTITUTE OF TECHNOLOGY CALICUT**

**DEPARTMENT OF ELECTRONICS AND COMMUNICATION ENGINEERING**

****

**EC 4091 : DIGITAL SIGNAL PROCESSING LABORATORY**

EXPERIMENT No. 2

Submitted by,

Anjali Venugopal B150071EC

Anju Anand B150303EC

Ankit Kumar B150912EC

Aravind Krishnan B150829EC

**Experiment No. 2**

**AIM**

Write MATLAB codes for each question.

**THEORY**

**Convolution :**

A convolution is an integral that expresses the amount of overlap of one function g as it is shifted over another

|  |  |  |  |
| --- | --- | --- | --- |
| f*g | = | int_(-infty)^inftyf(tau)g(t-tau)dtau |  |
| http://mathworld.wolfram.com/images/equations/Convolution/Inline15.gif | = | int_(-infty)^inftyg(tau)f(t-tau)dtau |  | |

For complex-valued functions *f*, *g* defined on the set **Z** of integers, the **discrete convolution** of *f* and *g* is given by:

http://pages.jh.edu/~signals/discreteconv2/conv.gif

**Overlap and Add :**

The overlap-add method is an efficient way to calculate the discrete convolution of a very long signal x[n] with a finite impulse response (FIR) filter h[n]. The concept is to divide the sequence into multiple sub-sequences of length L each and perform convolution to each part separately.

Algorithm :

1. l=0 y[n]=0
2. xl[n]=x[n+l L]; n=0 to L-1, 0 elsewhere
3. yl[n]=xl[n]\*h[n]
4. y[n]=y[n] + yl[n-l L]
5. l=l+1 go to step 2

**Overlap and Save :**

Algorithm :

1. l=0 y[n]=0
2. x[n]=[ (m-1) zeros; x[n] ]
3. xl[n]=L+(m-1) long overlapping segments of x[n]
4. yl[n]= xl[n]\*h[n]
5. Discard first and last (m-1) values from yl[n]
6. Append to y[n]

**INBUILT MATLAB FUNCTIONS USED :**

1. fliplr

Syntax :

B = fliplr(A)

Description :

B = fliplr(A) returns A with its columns flipped in the left-right direction (that is, about a vertical axis). If A is a row vector, then fliplr(A) returns a vector of the same length with the order of its elements reversed. If A is a column vector, then fliplr(A) simply returns A. For multidimensional arrays, fliplr operates on the planes formed by the first and second dimensions.

1. conv

Syntax :

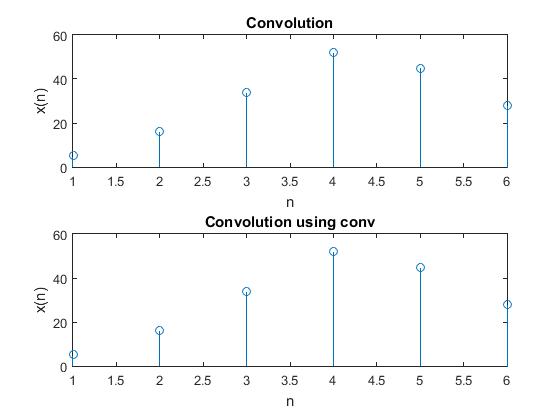
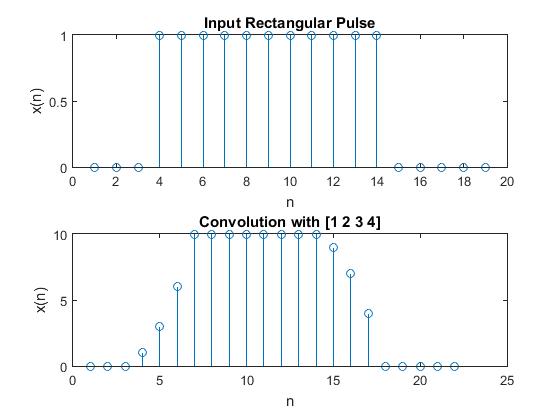
w = conv(u,v)

w = conv(u,v,shape)

Description :

w = conv(u,v) returns the convolution of vectors u and v. If u and v are vectors of polynomial coefficients, convolving them is equivalent to multiplying the two polynomials.

w = conv(u,v,shape) returns a subsection of the convolution, as specified by shape. For example, conv(u,v,'same') returns only the central part of the convolution, the same size as u, and conv(u,v,'valid') returns only the part of the convolution computed without the zero-padded edges.

**MATLAB CODE :**

%convolution

x1=[1 2 3 4];

h=[5 6 7];

h1=fliplr(h);

y=zeros(1,length(x1)+length(h)-1);

x=[zeros(1,length(h)-1),x1,zeros(1,length(h)-1)];

for i=1:length(y)

y(i)=0;

for j=1:length(h)

y(i)=y(i)+h1(j)\*x(i+j-1);

end

end

y

kl=conv(x1,h)

figure;

subplot(2,1,1);

stem(y)

title('Convolution');

xlabel('n');

ylabel('x(n)');

subplot(2,1,2);

stem(kl)

title('Convolution using conv');

xlabel('n');

ylabel('x(n)');

%convolution, rectangular pulse input

h=[1 2 3 4];

x1= [0,0,0, ones(1,11), zeros(1,5)];

subplot(2,1,1);

stem(x1);

title('Input Rectangular Pulse');

xlabel('n');

ylabel('x(n)');

h1=fliplr(h);

y=zeros(1,length(x1)+length(h)-1);

x=[zeros(1,length(h)-1),x1,zeros(1,length(h)-1)];

for i=1:length(y)

y(i)=0;

for j=1:length(h)

y(i)=y(i)+h1(j)\*x(i+j-1);

end

end

kl=conv(x1,h)

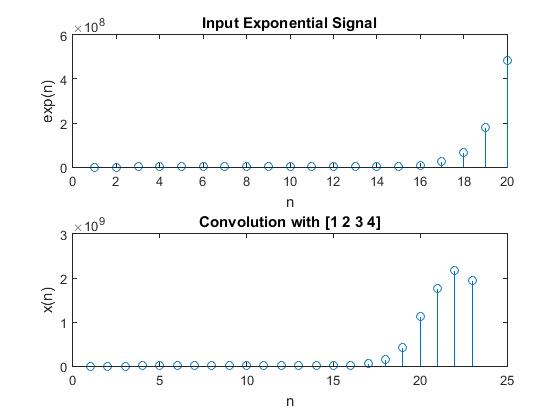
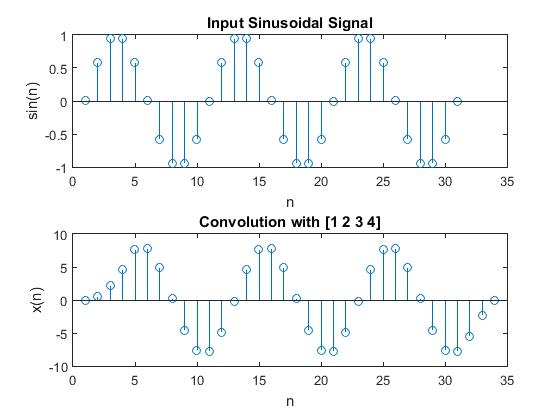
subplot(2,1,2);

stem(y);

title('Convolution with [1 2 3 4]');

xlabel('n');

ylabel('x(n)');

%convolution, exponential signal input

h=[1 2 3 4];

n=1:20;

x1=exp(n);

subplot(2,1,1);

stem(x1);

title('Input Exponential Signal');

xlabel('n');

ylabel('exp(n)');

h1=fliplr(h);

y=zeros(1,length(x1)+length(h)-1);

x=[zeros(1,length(h)-1),x1,zeros(1,length(h)-1)];

for i=1:length(y)

y(i)=0;

for j=1:length(h)

y(i)=y(i)+h1(j)\*x(i+j-1);

end

end

kl=conv(x1,h)

subplot(2,1,2);

stem(y);

title('Convolution with [1 2 3 4]');

xlabel('n');

ylabel('x(n)');

%convolution, sinusoidal signal input

clc;

clear var;

h=[1 2 3 4];

n=0:30;

x1=sin(2\*pi\*n/10);

subplot(2,1,1);

stem(x1);

title('Input Sinusoidal Signal');

xlabel('n');

ylabel('sin(n)');

h1=fliplr(h);

y=zeros(1,length(x1)+length(h)-1);

x=[zeros(1,length(h)-1),x1,zeros(1,length(h)-1)];

for i=1:length(y)

y(i)=0;

for j=1:length(h)

y(i)=y(i)+h1(j)\*x(i+j-1);

end

end

kl=conv(x1,h)

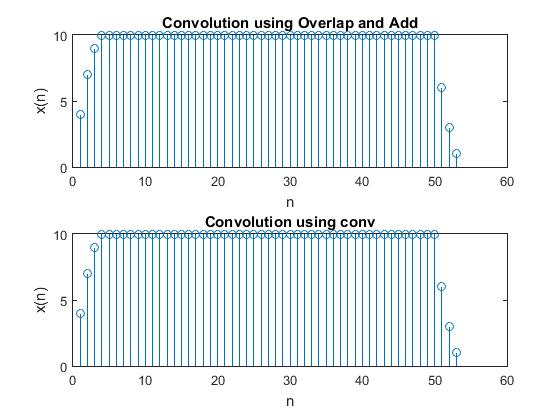
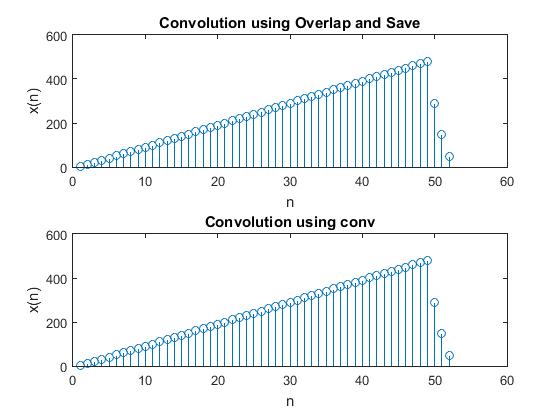
subplot(2,1,2);

stem(y);

title('Convolution with [1 2 3 4]');

xlabel('n');

ylabel('x(n)');

%Overlap and add

clc;

clear var;

x=ones(1,50);

h=[4 3 2 1];

l=5;

m=length(h);

last=zeros(1,length(h)-1);

y=zeros(1,length(h)+length(x)-1);

for i=1:length(x)/l

xb=x((i-1)\*l+1:l\*i);

yb=conv(xb,h);

yb(1:m-1)=yb(1:m-1)+last;

last=yb(l+1:l+m-1);

y((i-1)\*l+1:l\*i)=yb(1:l);

end

y(length(x)+1:length(x)+m-1)=last;

w=conv(x,h);

figure;

subplot(2,1,1);

stem(y);

title('Convolution using Overlap and Add');

xlabel('n');

ylabel('x(n)');

subplot(2,1,2);

stem(w);

title('Convolution using conv');

xlabel('n');

ylabel('x(n)');

%Overlap and save

clc;

clear var;

x=(1:49);

h=[4 3 2 1];

N=10;

m=length(h);

l=N-(m-1);

y=zeros(1,length(h)+length(x)-1);

pre=zeros(1,m-1);

for i=1:(length(x))/l

xb=[pre,x((i-1)\*l+1:l\*i)];

yb=conv(xb,h);

y((i-1)\*l+1:l\*i)=yb(m:m+l-1);

pre=xb(l+m-1-(m-1)+1:l+m-1);

end

n=length(xb)+length(h)-1;

y(length(x)+1:length(x)+m-1)=yb(n-(m-1)+1:n);

w=conv(x,h);

figure;

subplot(2,1,1);

stem(y);

title('Convolution using Overlap and Save');

xlabel('n');

ylabel('x(n)');

subplot(2,1,2);

stem(w);

title('Convolution using conv');

xlabel('n');

ylabel('x(n)');

**OBSERVATIONS AND INFERENCES**

Large sequences of data will require a sizeable amount of memory to hold the values for convolution in the traditional method. This also increases delay in the system. Block convolution is implemented to forego this. It takes only small sequences at a time.

**RESULT**

Convolution algorithm was written and the result was verified with that of the in-built MATLAB function, conv. A series of inputs (rectangular pulse, exponential signal, sinusoidal signal) was convoluted with a given LTI system response h[n] and the output was verified with conv function.

Block convolution was implemented using overlap and add and overlap and save methods and verified using conv function.