PART-A

EXPERIMENT NO-01: VERIFICATION OF SAMPLING THEOREM

AIM: Verification of Sampling Theorem for sin and cosine inputs for Nyquist rate, under sampling and over sampling condition in time domain using MATLAB

THEORY:

Sampling: Is the process of converting a continuous time signal into a discrete time signal. It is the first step in conversion from analog signal to digital signal.

Sampling theorem: Sampling theorem states that "Exact reconstruction of a continuous time base-band signal from its samples is possible, if the signal is band-limited and the sampling frequency is greater than twice the signal bandwidth".i.e. $f_s > 2W$, where W is the signal bandwidth.

Nyquist Rate Sampling: The Nyquist rate is the minimum sampling rate required to avoid aliasing, equal to the highest modulating frequency contained within the signal. In other words, Nyquist rate is equal to two sided bandwidth of the signal (Upper and lower sidebands) i.e., $f_s = 2W$.To avoid aliasing, the sampling rate must exceed theNyquist rate. i.e.$f_s > f_N$, where $f_N = 2W$.

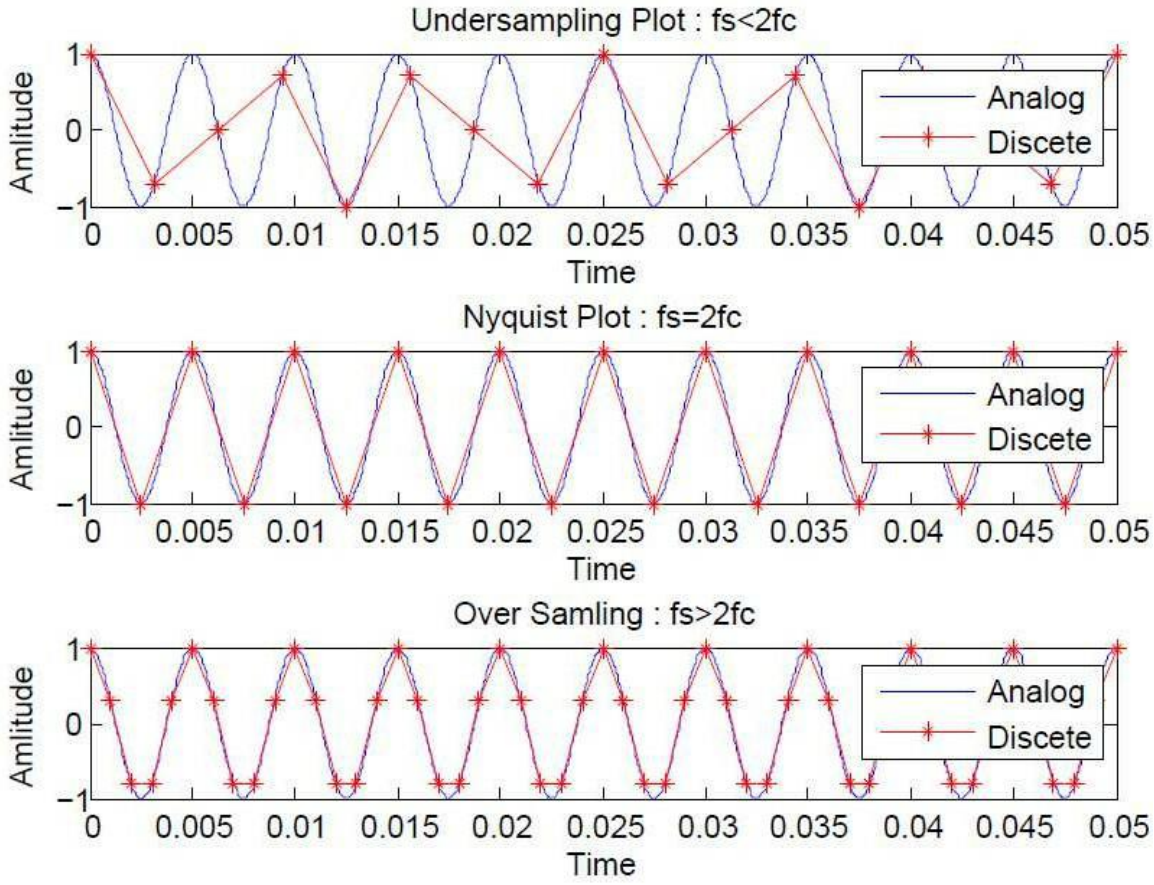
MATLAB PROGRAM TO VERIFY SAMPLING THEOREM:

```
clc
close all
clear all
tfinal = .05;
t = 0:0.00005:tfinal;
fc=input('Enter the Analog Wave Frequency : ');
 xt=cos(2*pi*fc*t);
fs1=1.6*fc;
 n1=0:1/fs1:tfinal;
 xn=cos(2*pi*fc*n1);
 subplot(3,1,1);
plot(t,xt,'b',n1,xn,'r*-');
 title('Undersampling Plot : fs<2fc');
xlabel('Time');
ylabel('Amlitude');
legend('Analog','Discete');
fs2=2*fc;
n2=0:1/fs2:tfinal;
 xn=cos(2*pi*fc*n2);
 subplot(3,1,2);
```

```
plot(t,xt,'b',n2,xn,'r*-');
 title('Nyquist Plot : fs=2fc');
xlabel('Time');
ylabel('Amlitude');
legend('Analog','Discete');
fs3=5*fc;
n3=0:1/fs3:tfinal;
xn=cos(2*pi*fc*n3);
subplot(3,1,3);
plot(t,xt,'b',n3,xn,'r*-');
title('Over Samling : fs>2fc');
xlabel('Time');
ylabel('Amlitude');
legend('Analog','Discete');
```

OUTPUT:

Enter the Analog Wave Frequency : 200



OUTCOME: Sampling theorem for the input analog signal is verified for all three conditions using MATLAB.

EXPERIMENT NO-2: LINEAR CONVOLUTION

AIM: Linear Convolution of two given sequences, for right sided and both sided sequences

THEORY: Convolution is an integral concatenation of two signals. It has many applications in numerous areas of signal processing. The most popular application is the determination of the output signal of a linear time-invariant system by convolving the input signal with the impulse response of the system. Note that convolving two signals is equivalent to multiplying the Fourier transform of the two signals. In linear systems, convolution is used to describe the relationship between three signals of interest: the input signal, the impulse response, and the output signal. In linear convolution length of output sequence is,

length(y(n)) = length(x(n)) + length(h(n)) – 1

Mathematical Formula:

The linear convolution of two continuous time signals x(t) and h(t) is defined by

$$y(t) = x(t) * h(t) = \int_{-\infty}^{\infty} x(\tau)h(t-\tau)d\tau$$

For discrete time signals x(n) and h(n), is defined by

$$y(n) = x(n) * h(n) = \sum_{k=-\infty}^{\infty} x(k)h(n-k)$$

Where x(n) is the input signal and h(n) is the impulse response of the system.

CALCULATION:

x1(n)={ 1, 2, 3, 2, 1, 3, 4 } and x2(n)={ 2, -3, 4, -1, 0, 1 }

x1(n)=δ(n+3)+2δ(n+2)+3δ(n+1)+2δ(n)+ δ(n-1)+3δ(n-2)+4δ(n-3)

x2(n)= 2δ(n+1)-3δ(n)+4δ(n-1)-δ(n-2)+ δ(n-4)

z(n) = [δ(n+3)+2δ(n+2)+3δ(n+1)+2δ(n)+ δ(n+1)+3δ(n+2)+4δ(n+3)] * [2δ(n+1)-3δ(n)+4δ(n-1)-δ(n-2)+ δ(n-4)]

On Simplification, we get

z(n) = [2δ(n+4) + δ(n+3) + 4 δ(n+2) + 2δ(n+1) + 6δ(n)+ 9δ(n-1)+3δ(n-2)+2δ(n-3) + 15δ(n-4) -3δ(n-5)+3δ(n-6)+4δ(n-7)]

z(n) = { 2, 1, 4, 2, 6, 9, 3, 2, 15, -3, 3, 4 }

PROGRAM:LINEAR CONVOLUTION OF TWO SIDED SEQUENCES

```
clc
clear all
close all
x1= input('Enter the first sequence x1(n) = ');      % define first sequence
i1= input('Enter the first index of the first sequence x1(n) = '); % -2;
```

```
disp(x1);
n1=i1: length (x1)+i1-1;% time axis for first sequence
disp(n1);
x1= input('Enter the second sequence x2(n) = ');
disp(x2);
i2= input('Enter the first index of the second sequence x2(n) = '); % -1;
n2=i2: length (x2)+i2-1;
disp(n2);
a=length(x1);                % Compute length of the first sequence
b=length(x2);                % Compute length of the second sequence
ybegin=n1(1)+n2(1);
yend=n1(a)+n2(b);            % calculate the end point of x axis of output
ny=[ybegin:yend];            % define x axis for output
y=conv(x1, x2);              % convolute first and second sequences
disp('Linear Convolution of x1 and x2 is y = :');
disp(y);                     % display the output
% Graphical Display
subplot(2,2,1);% divide the display screen in four sections and choose the first to display
stem(n1,x1);     % plot the first discrete sequence
xlabel('Time index n');      % label x axis
ylabel('Amplitude');         % label y axis
title('Plot of x1');         % graph title
%PLOT OF X2
subplot(2,2,2); %divide the display screen in four sections and choose the second to
display
stem(n2,x2);
xlabel('Time index n');
ylabel('Amplitude');
title('Plot of x2');
subplot(2,1,2);% divide the display screen in four sections and choose the second to display
stem(ny,y);
xlabel('Time index n');
ylabel ('Amplitude');
title('Convolution Output');
```

OUTPUT:

Enter the first sequence x1(n) = [1 2 3 2 1 3 4] Enter

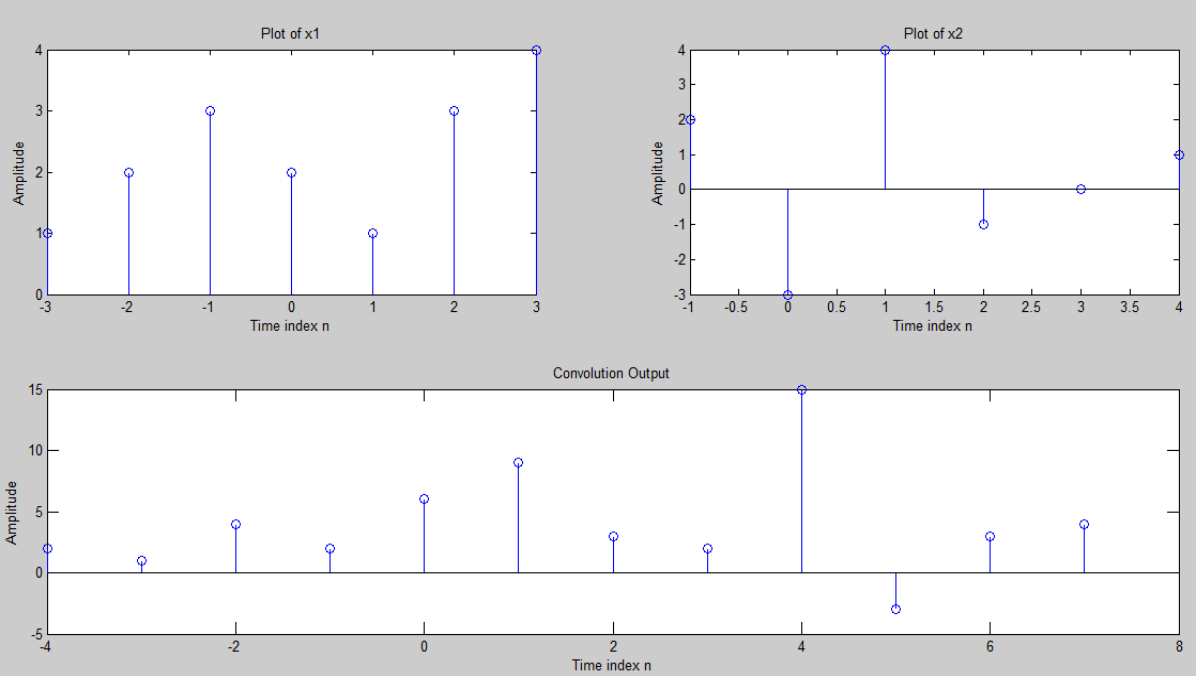
the first index of the first sequence x1(n) = -2 Enter

the second sequence x2(n) = [2 -3 4 -1 0 1] Enter the

first index of the second sequence x2(n) = -1 Linear

convolution of x1 and x2 is y =

2 1 4 2 6 9 3 2 15 -3 3 4



OUTCOME: Linear Convolution for the given sequences is obtained using MATLAB and verified theoretically.

EXPERIMENT NO-3: CIRCULAR CONVOLUTION

AIM: To implement circular convolution of given sequences in time domain using MATLAB and to verify the result theoretically

THEORY: Let x1(n) and x2(n) are finite duration sequences both of length N with DFT's X1(k) and X2(k). Convolution of two given sequences x1(n) and x2(n) is given by the equation,

x3(n) = IDFT[X3(k)] where X3(k) = X1(k) X2(k)

$$x3(n) = \sum_{m=0}^{N-1} x1(m)\, x2((n-m))_N$$

CALCULATION:

Let's take x1(n) = {1, 1, 2, 1} and x2(n) = {1, 2, 3, 4}

x3(0) = x1(m) x2(-m) = x1(0) x2(0) + x1(1) x2(3) + x1(2) x2(2) + x1(3) x2(1)

= 1 + 4 + 6 +2 = 13

x3(1) = x1(m) x2(1-m) = x1(0) x2(1) + x1(1) x2(0) + x1(2) x2(3) + x1(3) x2(2)

= 2 + 1 + 8 + 3= 14

x3(2) = x1(m) x2(2-m) = x1(0) x2(2) + x1(1) x2(1) + x1(2) x2(0) + x1(3) x2(3)

= 3 + 2 + 2+ 4= 11

x3(3) = x1(m) x2(3-m) = x1(0) x2(3) + x1(1) x2(2) + x1(2) x2(1) + x1(3) x2(0)

= 4 + 3 + 4 + 1= 12

The convoluted signal is,

x3(n) = { 13, 14, 11, 12 }

PROGRAM: CIRCULAR CONVOLUTION IN TIME DOMAIN

```
clc; % clear screen
clear all; % clear workspace
close all;
xn= input('Enter the first sequence x(n) = ');
hn=input('Enter the second sequence h(n) = ');
l1 = length(xn);
l2 = length(hn);
N = max(l1,l2);
xn = [xn, zeros(1,N-l1)];
 hn = [hn, zeros(1,N-l2)];.
 n1=0:length(xn)-1;
n2=0:length(hn)-1;
```

```
for n=0:N-1;
y(n+1) = 0;
   for k=0:N-1
   i = mod((n-k),N);
   y(n+1) =y(n+1)+hn(k+1)*xn(i+1);
   end ;
end;
disp('Circular convolution in Time Domain = ');
disp(y);
ny=0:N-1;
subplot(2,2,1);
stem(n1,xn);
xlabel('Time Index n');
ylabel('Amplitude x(n)');
title('Plot of x(n)');
subplot(2,2,2);
stem(n2,hn);
xlabel('Time Index n');
ylabel('Amplitude h(n)');
title('Plot of h(n)');
subplot(2,1,2);
stem(ny,y);
xlabel('Time Index n');
ylabel('Amplitude y(n)');
 title('Circular Convoluted Output y(n)');
```

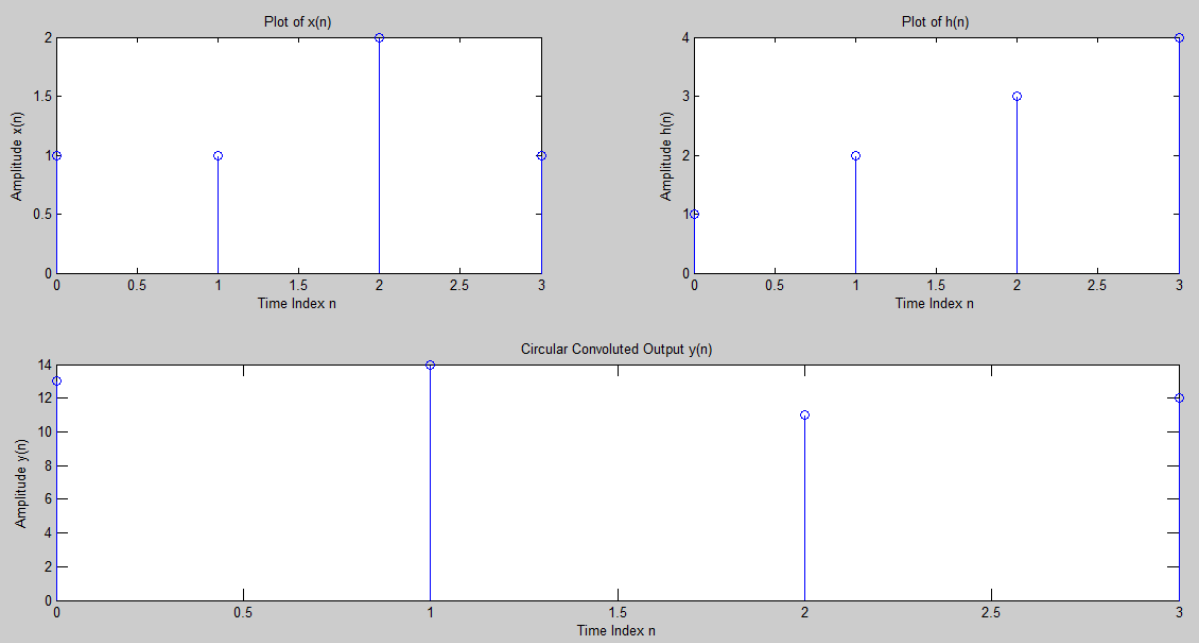
OUTPUT:

Enter the first sequence x(n) = [1 1 2 1]

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

Circular convolution in Time Domain =

    13   14   11   12



OUTCOME: Circular Convolution for the given sequences is obtained using MATLAB and verified theoretically.

EXPERIMENT NO-04: AUTOCORRELATION

AIM: To find Autocorrelation of the given sequence using an inbuilt MATLAB function "XCORR" and to verify its properties.

THEORY: Correlation: Correlation determines the degree of similarity between two signals. If the signals are identical, then the correlation coefficient is 1; if they are totally different, the correlation coefficient is 0, and if they are identical except that the phase is shifted by exactly 180^0 (i.e. mirrored), then the correlation coefficient is -1.

Autocorrelation: The Autocorrelation of a sequence is correlation of a sequence with itself.

The autocorrelation of a sequence x(n) is defined by,

$$R_{xx}(k) = \sum_{n=-\infty}^{\infty} x(n)\, x(n-k) \qquad k = 0, \pm1, \pm2, \pm3, \ldots$$

Where k is the shift parameter

Or equivalently

$$R_{xx}(k) = \sum_{n=-\infty}^{\infty} x(n+k)\, x(n) \qquad k = 0, \pm1, \pm2, \pm3, \ldots$$

PROPERTIES OF AUTOCORRELATION:

1. $R_{xx}(0)$ = Energy(x)
2. Autocorrelation function is **symmetric.** i.e. $R_{xx}(m) = R_{xx}(-m)$

CALCULATION:

x(n) ={ 3, 4, 5, 6 }

$$R_{xx}(k) = \sum_{n=-\infty}^{\infty} x(n)\, x(n-k)$$

Put k=0 in the above equation, we get

$$R_{xx}(0) = \sum_{n=-\infty}^{\infty} x(n)\, x(n) = 9 + 16 + 25 + 36 = 86$$

Put k=1 in the above equation, we get

$$R_{xx}(1) = \sum_{n=-\infty}^{\infty} x(n)\, x(n-1) = 0 + 12 + 20 + 30 = 62$$

Put k=2 in the above equation, we get

$$R_{xx}(2) = \sum_{n=-\infty}^{\infty} x(n)\, x(n-2) = 0 + 0 + 15 + 24 + 0 = 39$$

Put k=3 in the above equation, we get

$$R_{xx}(3) = \sum_{n=-\infty}^{\infty} x(n)\, x(n-3) = 0 + 0 + 0 + 18 + 0 + 0 + 0 = 18$$

Put k=-1 in the above equation, we get

$$R_{xx}(-1) = \sum_{n=-\infty}^{\infty} x(n)\, x(n+1) = 0 + 12 + 20 + 30 = 62$$

Put k=-2 in the above equation, we get

$$R_{xx}(-2) = \sum_{n=-\infty}^{\infty} x(n)\, x(n+2) = 0 + 0 + 15 + 24 + 0 = 39$$

Put k=-3 in the above equation, we get

$$R_{xx}(-3) = \sum_{n=-\infty}^{\infty} x(n)\, x(n+3) = 0 + 0 + 0 + 18 + 0 + 0 + 0 = 18$$

Therefore $R_{xx}(k)$ = [18 39 62 86 62 39 18]

PROGRAM: AUTO CORRELATION

```
clc
clear all
close all
x=input('Enter the input sequence x(n):');
a1=length(x);
n1=0:a1-1;
[Rxx, lag]=xcorr(x);
Rxx=round(Rxx);
disp('Auto correlation sequence r(n) is :');
disp(Rxx);
%disp(lag);
subplot(2,1,1);
stem(n1,x);
xlabel('Time index n');
ylabel('Amplitude x(n)');
title('Tnput sequence x(n)');
subplot(2,1,2);
stem(lag,Rxx);
xlabel('Time index n');
ylabel('Amplitude r(n)');
title('Auto Correlation Output Sequence r(n)');
% Property 1: Rxx(0)gives the energy of the signal
Energy = sum(x.^2); % calculate the energy of input signal
```

```
center_index= ceil(length(Rxx)/2);
Rxx_0=Rxx(center_index);

if Rxx_0==Energy
disp('Rxx(0) gives energy -- proved');
else
disp('Rxx(0) gives energy -- not proved');
end
% Property 2: Rxx is even
Rxx_Right = Rxx(center_index:1:length(Rxx));
Rxx_left = Rxx(center_index:-1:1);
if Rxx_Right == Rxx_left
disp('Rxx is even'); % display the result
else
disp('Rxx is not even'); % display the result
end
```

OUTPUT:

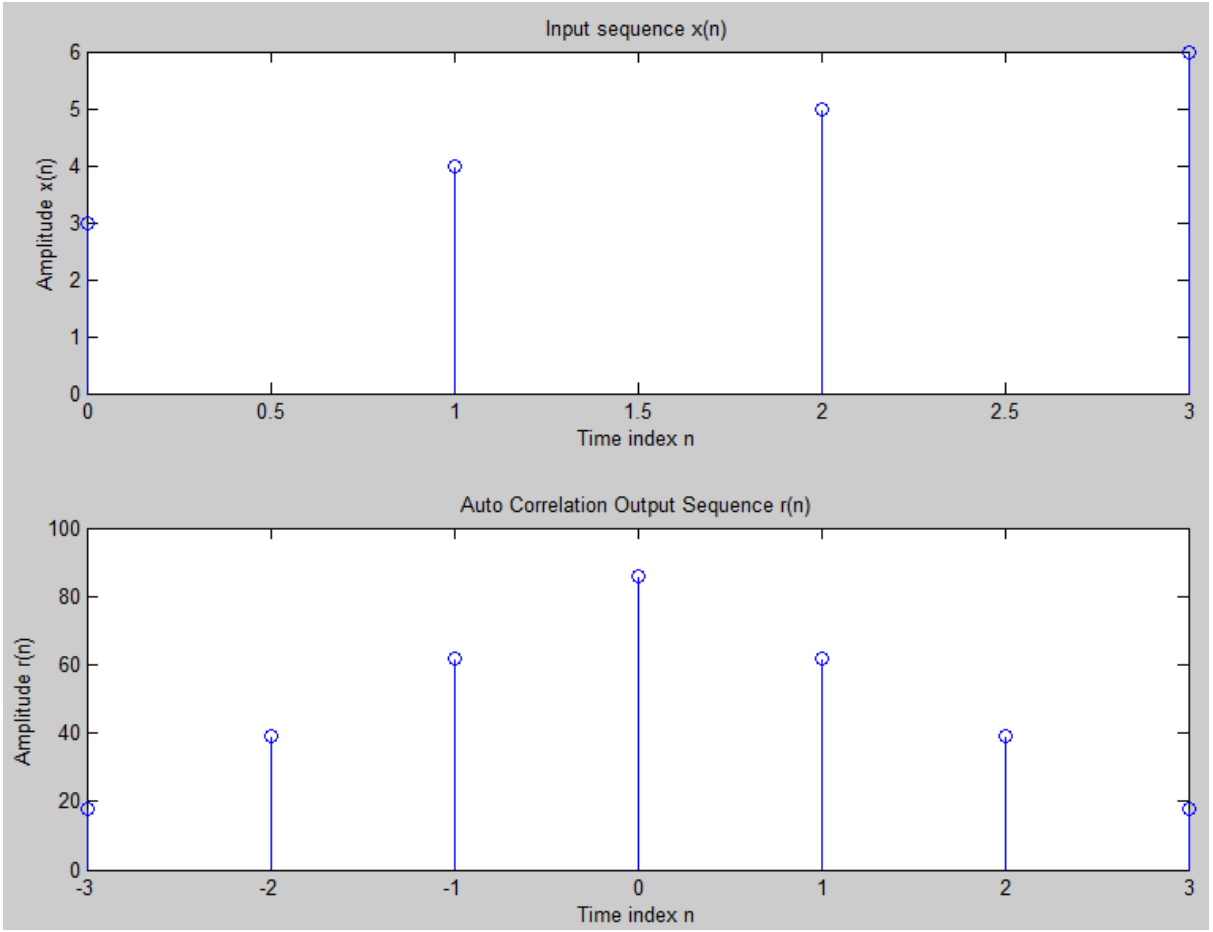
```
Enter the input sequence x(n): [ 3 4 5 6 ]

Auto correlation sequence r(n) is :

   18     39     62     86     62     39     18

Rxx(0) gives energy -- proved

Rxx is even
```



OUTCOME: Autocorrelation for the given sequence and its properties are verified using MATLAB is obtained.

EXPERIMENT NO-5: CROSS-CORRELATION

AIM: Cross-correlation of a given sequences and verification of its properties using an inbuilt MATLAB function "XCORR"**.**

THEORY: When two independent signals are compared, the procedure is known as cross correlation. It is given by,

$$R_{xy}(k) = \sum_{n=-\infty}^{\infty} x(n)\, y(n-k) \qquad k = 0, \pm1, \pm2, \pm3, \ldots$$

Where k is the shift parameter

Or equivalently

$$R_{xy}(k) = \sum_{n=-\infty}^{\infty} y(n+k)\, x(n) \qquad k = 0, \pm1, \pm2, \pm3, \ldots$$

Comparing above two equations, we find that, $R_{xy}(k) = R_{yx}(-k)$, Where$R_{yx}(-k)$ is the folded version of $R_{xy}(k)$ about k = 0. So, we can write Cross correlation of the two sequences is given by,

$$R_{xy}(k) = \sum_{n=-\infty}^{\infty} x(n)y[-(k-n)]_k \qquad k = 0, \pm1, \pm2, \pm3, \ldots$$

$$R_{xy}(k) = x(k) * y(-k) \qquad (1)$$

Equation (1) shows that cross correlation is the essentially the convolution of two sequences in which one of the sequences has been reversed.

PROPERTIES OF CROSS CORRELATION:

1. $R_{xy}(k)$ is always a real valued function which may be a positive or negative.
2. $R_{xy}(-k) = R_{yx}(k)$
3. When $R_{xy}(k) = 0$, x(n) and y(n) are said to be uncorrelated or they said to be statistically independent.
4. $R_{xy}(k)$ may not be necessarily have a maximum at k=0 nor $R_{xy}(k)$ an even function.

CALCULATION:

Put K=0 in the above equation, we get

x(n) ={ 1,2,3,4} y(n)={1,2,1,2}

$$R_{xy}(k) = \sum_{n=-\infty}^{\infty} x(n)\, y(n-k)$$

$k = 0, \pm 1, \pm 2, \pm 3, \ldots$

$$R_{xy}(0) = \sum_{n=-\infty}^{\infty} x(n)\, y(n) \qquad k = 0, \pm 1, \pm 2, \pm 3, \ldots$$

$$R_{xy}(0) = 1 + 4 + 3 + 8 = 16$$

Put K=1 in the above equation, we get

$$R_{xy}(1) = 2 + 6 + 4 = 12$$

Put K=2 in the above equation, we get

$$R_{xy}(2) = 3 + 8 = 11$$

Put K=3 in the above equation, we get

$$R_{xy}(3) = 4 = 4$$

Put K= -1 in the above equation, we get

$$R_{xy}(-1) = 2 + 6 + 2 = 13$$

Put K= -2 in the above equation, we get

$$R_{xy}(-2) = 1 + 4 = 5$$

Put K= -3 in the above equation, we get

$$R_{xy}(-3) = 2 = 2$$

Therefore $R_{xy} = [\ 2\ 5\ \ 10\ \ 16\ \ 12\ \ 11\ \ 4\ ]$

PROGRAM: CROSS CORRELATION USING XCORR

```
clc; % clear screen
clear all; % clear workspace
close all; % close all figure windows
x = input('Enter the first sequence x(n) =');
y = input('Enter the second sequence y(n) =');
r = xcorr(x,y);
disp('Cross Correlation Output = ');
disp(r); % display the output
n1 = length(x)-1;
t1 = 0:n1;
subplot(2,2,1);
stem(t1,x);
xlabel('n');
ylabel('x(n)');
title('plot of x(n)');
n2 = length(y)-1; %
t2 = 0:n2;
subplot(2,2,2);
stem(t2,y);
xlabel('n')
```

```
ylabel('y(n)');
title('plot of y(n)');
N = max(n1,n2);
k = -N:N;
subplot(2,1,2);
stem(k,r);
xlabel('n');
ylabel('r(n)');
title('cross correlation output');
x= input(“seq1’);)
y = input(‘seq2’);
Rxy = xcorr(x,y);
Ryx=xcorr(y,x);
Rxy1 = fliplr(Rxy);
if Rxy1 = Ryx
disp(‘Rxy(-k) = Ryx(k) - proved’); else
disp(‘Not proved’);
end
```

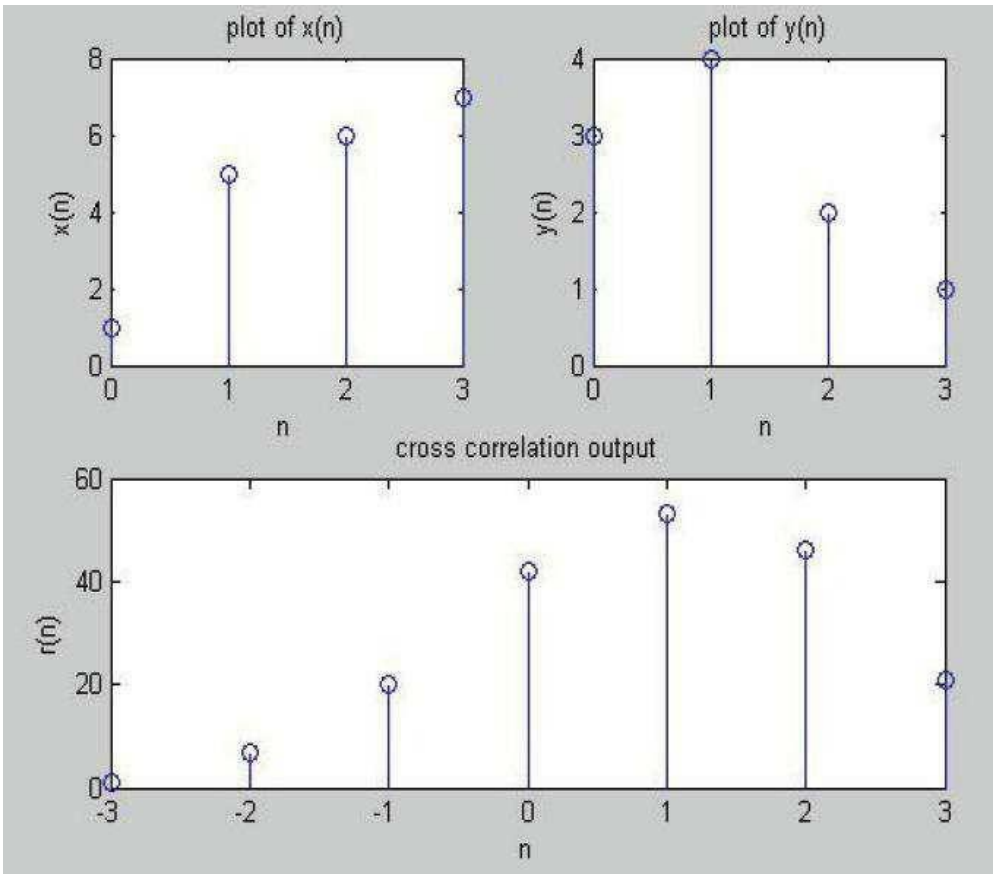
OUTPUT:

Enter the first sequence x(n) = [1 5 6 7]

Enter the second sequence y(n) = [3 4 2 1]

Cross Correlation Output =

1 7 20 42 53 46 21



OUTCOME: Cross-correlation of the sequence is found and properties of cross correlation are verified.

EXPERIMENT NO-6: IMPULSE RESPONSE OF A GIVEN LTI SYSTEM

AIM: To Solve a given difference equation **and find the impulse response of a given LTI system** using an inbuilt MATLAB function "IMPZ"**.**

THEORY:

A General n^{th} order Difference equations looks like,

$y[n] + a_1y[n-1] + a_2y[n-2] + \ldots + a_Ny[n-N] = b_0x[n] + b_1x[n-1] + \ldots. + b_Mx[n-M]$

Here y[n] is "Most advanced" output sample and y[n-m] is "Least advanced" output sample.

The difference between these two index values is the order of the difference equation.

Here we have: n-(n-N) = N

We can rewrite the above equation as, $y[n]+ \Sigma a_i y[n-i] = \Sigma b_i\, x[n-i]$

y[n] becomes, $y[n] = -\Sigma a_i y[n-i] + \Sigma b_i\, x[n-i]$

EXAMPLE:

y[n] – 0.5y[n-1] = x[n] Taking Z-Transform

we get $Y(z)-0.5Y(z)z^{-1} = X(z)$

$$\frac{Y(z)}{X(z)} = H(z) = \frac{1}{1 - 0.5z^{-1}}$$

$$= 1 + 0.5z^{-1} + 0.25z^{-2} + 0.125z^{-3} + 0.0625z^{-4} + \ldots$$

Therefore h(n)= [1, 0.5, 0.25, 0.125, 0.0625]

PROGRAM: IMPULSE RESPONSE USING IMPZ FUNCTION

```
clc
close all
clear all
N=input('Enter the length of Impulse response required : ');
x=input('Enter the x(n) coefficient array : '); %example[1]
y=input('Enter the y(n) coefficient array : ');
h=impz(x,y,N); %Impulse Sequence computation
disp('The coefficients of h(n) is :');
disp(h);
%Plot the Impulse response required
n=0:N-1; %Time Vector for plotting
stem(n,h);
title('Impulse Response');
xlabel('Time Index n');
ylabel('Amplitude');
```

OUTPUT:

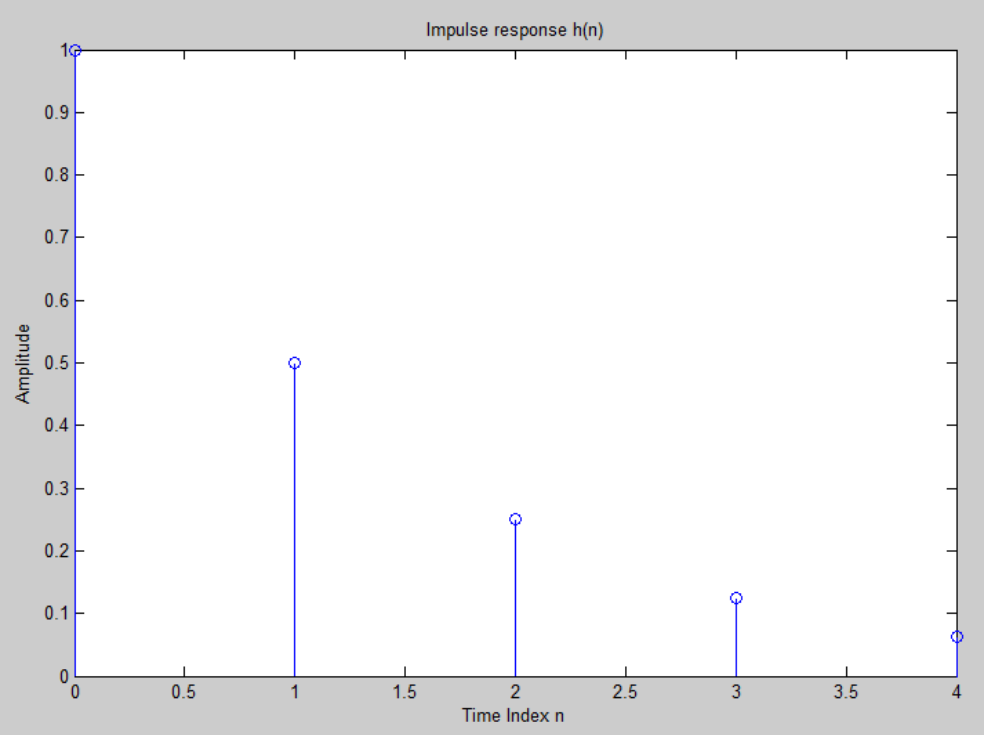
Enter the length of Impulse response required: 5

Enter the x(n) coefficient array : 1

Enter the y(n) coefficient array : 1 -0.5

The coefficients of h(n) is :

1 0.5 0.25 0.125 0.0625



OUTCOME: Impulse Response for the given difference equation (response of the filter) using MATLAB is obtained.

EXPERIMENT NO-7: SOLUTION TO A GIVEN DIFFERENCE EQUATION

AIM: To solve the given difference equation(response of the filter) with and without initial conditions using an inbuilt MATLAB functions "FILTER and FILTIC"

THEORY:

A General n^{th} order Difference equations looks like,

$y[n] + a_1y[n-1] + a_2y[n-2] + \ldots + a_Ny[n-N) = b_0x[n] + b_1x[n-1] + \ldots. + b_Mx[n-M]$

Here y[n] is "Most advanced" output sample and y[n-m] is "Least advanced" output sample.

The difference between these two index values is the order of the difference equation.

Here we have: n-(n-N) = N

We can rewrite the above equation as, $y[n]+ \Sigma a_i y[n-i] = \Sigma b_i\ x[n-i]$

y[n] becomes, $y[n] = -\Sigma a_i y[n-i] + \Sigma b_i\ x[n-i]$

SOLUTION OF DIFFERENCE EQUATION WITHOUT INITIAL CONDITIONS

EXAMPLE:

y(n)+2y(n-1)+3y(n-2)=x(n)+3x(n-1)+2x(n-2)

Given x(n)={ 2, 2, 3 }

The response y(n) computed value is

2 4 -1

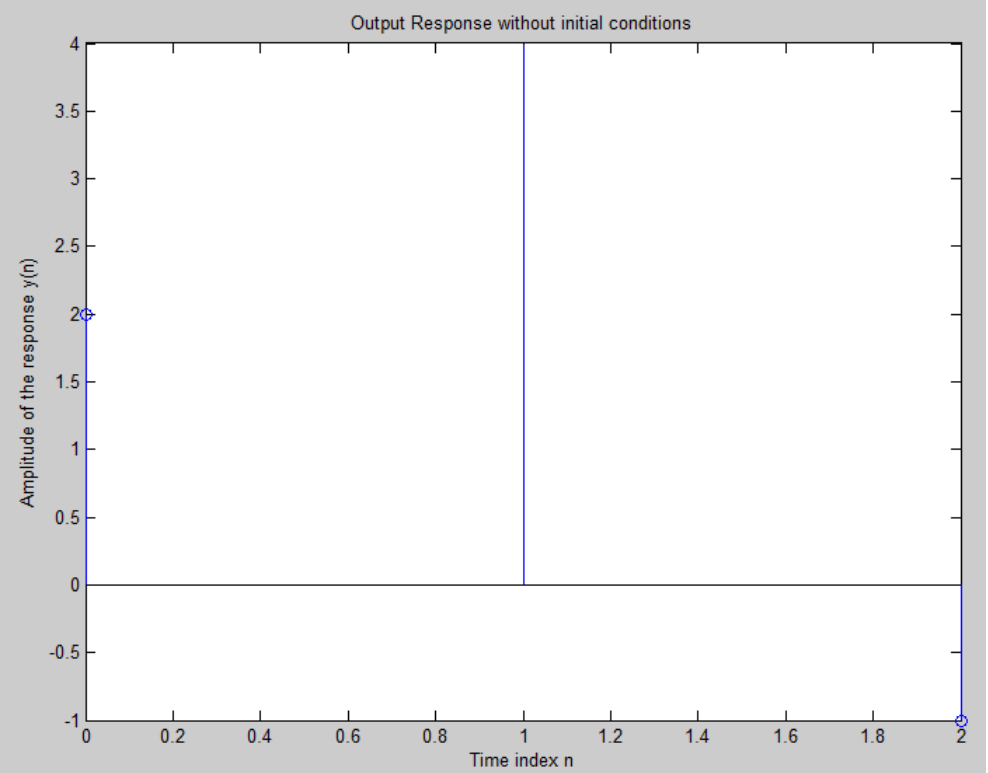
PROGRAM:

```
clc
close all
clear all
b=[1 3 2]
a=[1 2 3]
x=[2 2 3]
z=filter(b,a,x);            % calculate the response of the system
disp('Output coefficient without initial conditions');
disp(z);
n=0:2;
stem(n,z);
xlabel('Time index n');
ylabel('Amplitude of the response y(n)');
title('Output Response without initial conditions');
```

OUTPUT:

Output coefficient without initial conditions

2 4 -1



OUTCOME: Response for the given difference equation(response of the filter) without initial conditions using MATLAB is obtained.

EXPERIMENT NO-8: N-POINT DFT COMPUTATION

AIM: To compute N-point DFT of a given sequence and to plot magnitude and phase spectrum.

THEORY: Discrete Fourier Transform is a powerful computation tool which allows us to evaluate the Fourier Transform $X(e^{j\omega})$ on a digital computer or specially designed digital hardware. Since $X(e^{j\omega})$ is continuous and periodic, the DFT is obtained by sampling one period of the Fourier Transform at a finite number of frequency points. Apart from determining the frequency content of a signal, DFT is used to perform linear filtering operations in the frequency domain.

The sequence of N complex numbers $x_0,\ldots,x_{N-1}$ is transformed into the sequence of *N* complex numbers $X_0,\ldots,X_{N-1}$ by the DFT according to the formula:

$$X(k) = \sum_{n=0}^{N-1} x(n) e^{\frac{-j2\,kn}{N}} \quad ; k = 0, 1, 2, \ldots N-1$$

EXAMPLE:

Let us assume the input sequence x[n] = [1 1 0 0]

We have,

$$X(k) = \sum_{n=0}^{N-1} x(n) e^{\frac{-j2\ kn}{N}} \quad ; k = 0, 1, 2, \ldots N-1$$

For k = 0,

$$X(0) = \sum_{n=0}^{N-1} x(n) \quad ; k = 0, 1, 2, \ldots N-1$$

X(0) = x(0) + x(1) + x(2) + x(3)

X(0) = 1+1+0+0 = 2

For k = 1,

X(1) = 1 + cos(π/2) - jsin(π/2)

X(1) = 1 – j

For k = 2

$X(2) = x(0) + x(1)\, e^{-j\omega} + x(2)\, e^{-j2\omega} + x(3)\, e^{-j3\omega}$

X(2) = 1 + cos π – jsin π

X(2) = 1-1 = 0

For k = 3,

$= x(0) + x(1)\, e^{-j3\omega/2} + x(2)\, e^{-j3\omega} + x(3)\, e^{-j9\omega/2}$

$X(3) = 1 + \cos(3\pi/2) - j\sin(3\pi/2)$

$X(3) = 1 + j$

The DFT of the given sequence is,

$X(k) = \{ 2, 1- j, 0, 1+j \}$

To find Magnitude of X(k):

Magnitude= $(a^2+b^2)^{1/2}$

Where a and b are real and imaginary parts respectively

To fine Phase of X (k):

Phase=$\tan^{-1}(b/a)$

PROGRAM: N POINT DFT USING STANDERD EQUATION

```
clc
clear all
close all
x=input('Enter the input sequence:');
N=length(x);
for k=0:N-1
   xk(k+1)=0;
   for n=0:N-1
      xk(k+1)=xk(k+1)+x(n+1)*exp(-i*2*pi*n*k/N);
   end;
end;
disp('DFT of the input sequence:');
disp(xk);
n=0:N-1;
subplot(3,1,1);
stem(n,x);
xlabel('Time Index n');
ylabel('Amplitude');
title('Input sequence');
subplot(3,1,2);
stem(n,abs(xk));
% disp('Magnitude of xk :');
% disp(abs(xk));
xlabel('Time Index k');
ylabel('Amplitude');
title('Magnitude Spectrum');
subplot(3,1,3);
stem(n,angle(xk));
% disp('Phase angle of xk :');
% disp(abs(xk));
```

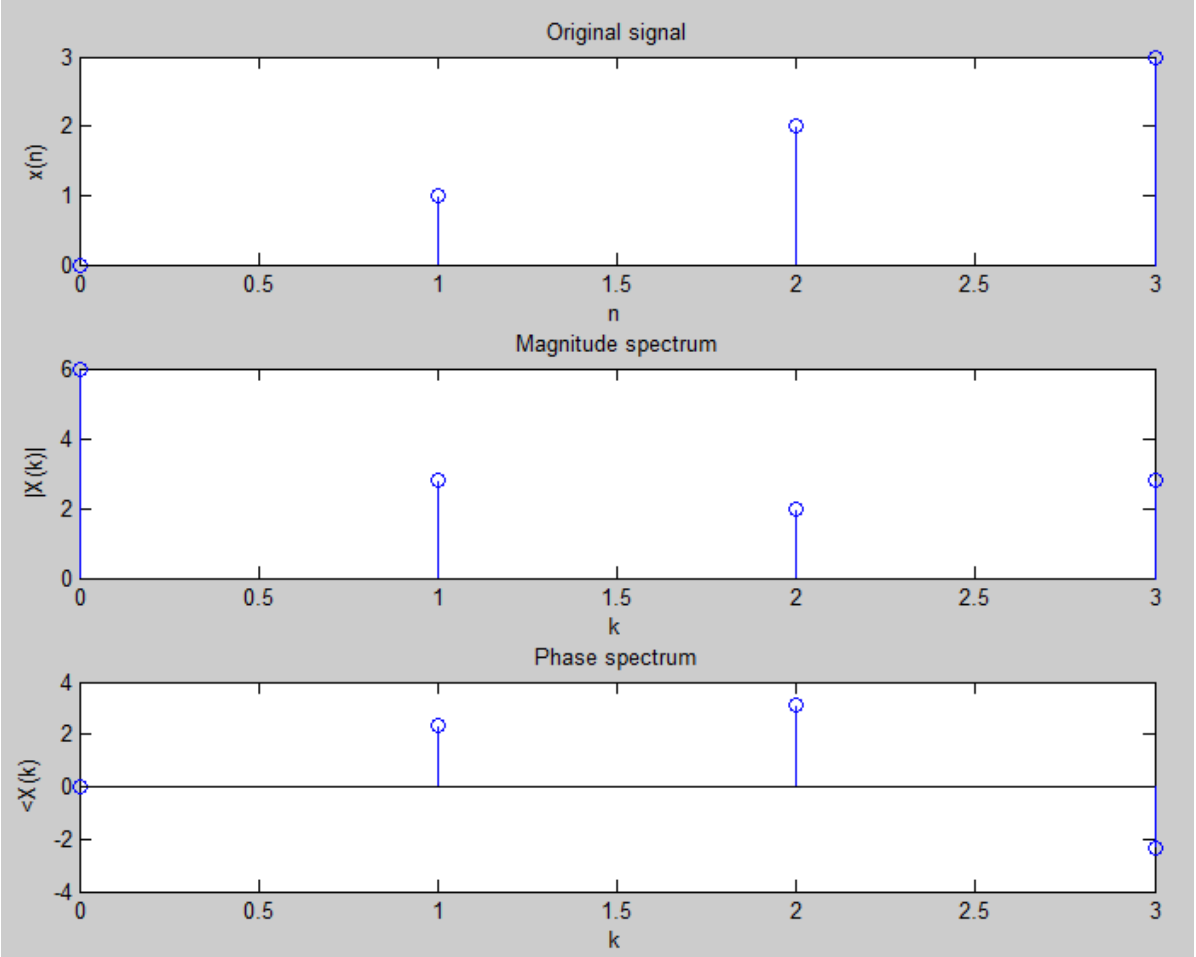
```
xlabel('Time Index k');
ylabel('Amplitude');
title('Phase Spectrum');
```

OUTPUT:

Enter the input sequence x(n) = [0 1 2 3]

N point DFT of x(n) is =

6.0000 -2.0000 + 2.0000i -2.0000 -2.0000 - 2.0000i



OUTCOME: DFT of the given sequence is found and the magnitude and phase response are plotted using MATLAB.

EXPERIMENT NO-9: LINEAR CONVOLUTION USING DFT AND IDFT

AIM: To find the Linear convolution of a given sequence using the inbuilt MATLAB functions "FFT and IFFT" for DFT and IDFT and verify the result using the function "CONV".

THEORY: Convolution is an integral concatenation of two signals. It has many applications in numerous areas of signal processing. The most popular application is the determination of the output signal of a linear time-invariant system by convolving the input signal with the impulse response of the system. Note that convolving two signals is equivalent to multiplying the Fourier Transform of the two signals.

MATHEMATICAL FORMULA:

The linear convolution of two continuous time signals x(t) and h(t) is defined by

$$\infty$$

$$y(t) = x(t) * h(t) = \int_{-\infty}^{\infty} x(\tau)h(t-\tau)d\tau$$

For discrete time signals x(n) and h(n), is defined by

$$y(n) = x(n) * h(n) = \sum_{k=-\infty}^{\infty} x(k)h(n-k)$$

Where x(n) is the input signal and h(n) is the impulse response of the system.

CALCULATION:

x1(n) = {1, 1, 2}

x2(n) = {1, 2}

For linear convolution,

Length N = Length(x1) + Length(x2) - 1

N = 3 + 2 – 1 = 4

Convolution of two sequences x1(n) and x2(n)

is, x3(n) = IDFT[X3(k)]

x3(n) = IDFT[X1(k) X2(k)]

Where, X1(k) = DFT [x1(n)] and X2(k) = DFT [x2(n)]

$$X1(k) = \sum_{n=0}^{N-1} x1(n)e^{\frac{-j2\ kn}{N}} \quad ; k = 0, 1, 2, \ldots N-1$$

Given x1(n) = {1, 1, 2} and N=4

X1(0) = 1 + 1 + 2 = 4

X1(1) = 1 – j – 2 = -1 – j

X1(2) = 1 – 1 + 2 = 2

X1(3) = 1 + j – 2 = -1 + j

Therefore X1(k) = {4, -1 – j, 2, -1 + j}

Similarly

$$X2(k) = \sum_{n=0}^{N-1} x2(n) e^{\frac{-j2\ kn}{N}} \quad ; k = 0, 1, 2, \ldots N-1$$

Given x2(n) = {1, 2} and N=4

X2(0) = 1 + 2 = 3

X2(1) = 1 + 2(-j) = 1 - j2

X2(2) = 1 + 2(-1) = -1 X2(3) = 1

+ 2(j) = 1 + j2

Therefore X2(k) = {3, 1-j2, -1, 1+j2} We

know that,

X3(k) = X1(k) • X2(k)

X3(k) = { 12, -3+j, -2, -3-j }

Convolution of two given sequences is,

x3(n) = IDFT[X3(k)]

$$x3(k) = \frac{1}{N}\sum_{k=0}^{N-1} X3(k) e^{\frac{j2\ kn}{N}} \quad ; n = 0, 1, 2, \ldots N-1$$

x3(0)= (1/4) [12 – 3 +j -2 -3 –j] = 1

x3(1)= (1/4) [12 + (-3 + j) j + (-2) (-1) + (-3 - j) (-j)] = 3

x3(2)= (1/4) [12 + (-3 + j) (-1) + (-2) (1) + (-3 - j) (-1)] = 4

x3(3)= (1/4) [12 + (-3 + j) (-j) + (-2) (-1) + (-3 - j) (j)] = 4

Therefore Convoluted sequence of two given sequences is,

x3(n) = {1, 3, 4, 4}

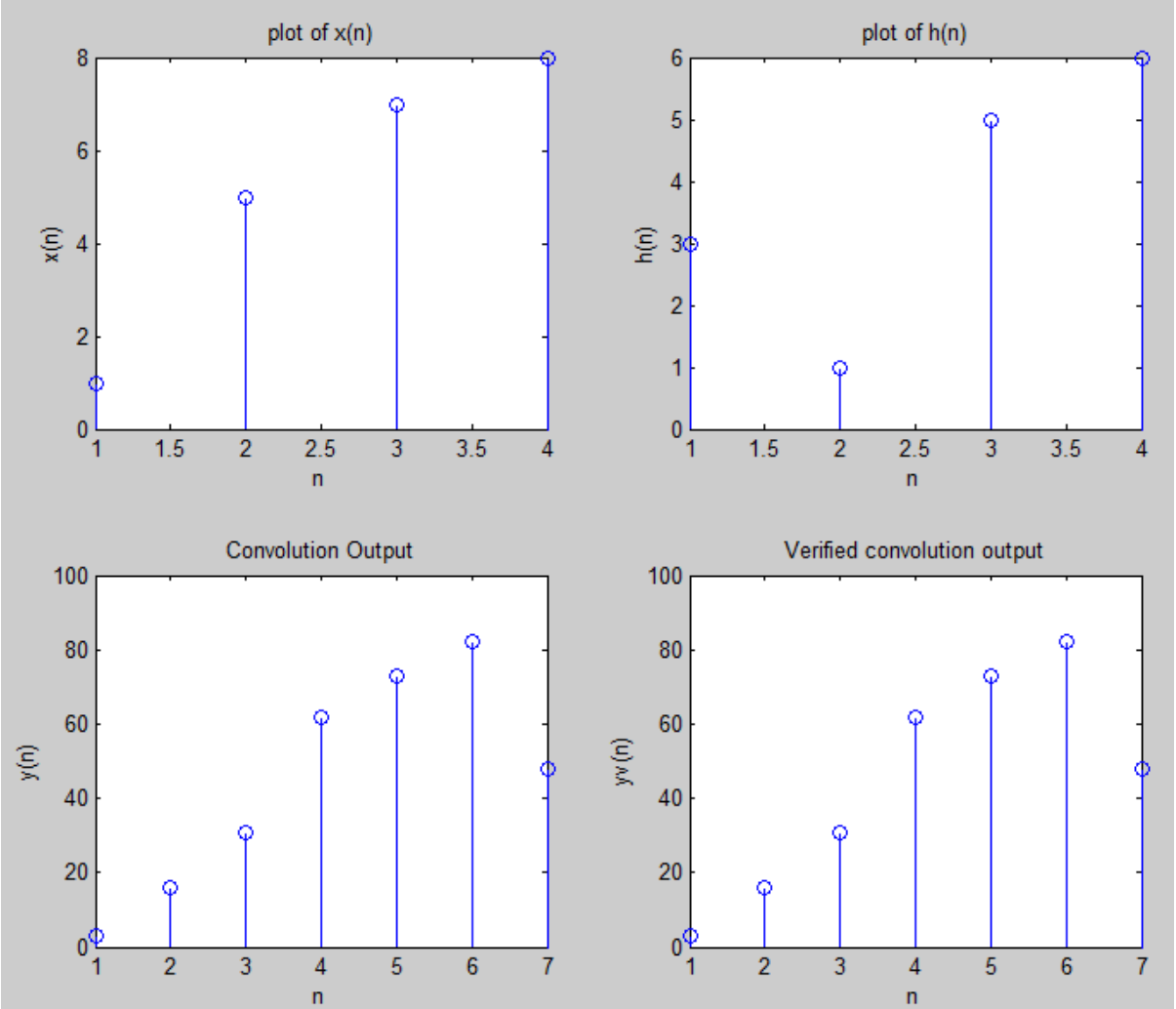
PROGRAM: LINEAR CONVOLUTION USING DFT AND IDFT

```
clc; % clear screen
clear all;
close all;
xn = input('Enter the first sequence x(n) = ');
hn = input('Enter the second sequence h(n) = ');
N = length(xn)+length(hn)-1; % length of output
```

```
Xk = fft(xn,N);
Hk = fft(hn,N);
Yk = Xk.*Hk;
yn = ifft(Yk,N);
disp('Linear convolution of x(n) and h(n) =');
disp(yn);
subplot(2,2,1);
stem(xn);
xlabel('n');
ylabel('x(n)');
title('plot of x(n)');
subplot(2,2,2);
stem(hn);
xlabel('n');
ylabel('h(n)');
title('plot of h(n)');
subplot(2,2,3);
stem(yn);
xlabel('n');
ylabel('y(n)');
title('Convolution Output');
yv =conv(xn,hn);
disp('Convolution in time domain using conv function = ');
disp(yv);
subplot(2,2,4);
stem(yv);
xlabel('n');
ylabel('yv(n)');
title('Verified convolution output');
```

OUTPUT:

```
Enter the first sequence x(n) = [ 1  5 7 8 ]
Enter the second sequence h(n) = [ 3 1 5 6 ]
Linear convolution of x(n) and h(n) =
    3.0000    16.0000    31.0000    62.0000    73.0000    82.0000    48.0000
Convolution in time domain using conv function =
     3    16    31    62    73    82    48
```



OUTCOME: Linear convolution of two given sequences found using DFT and IDFT and the results are verified.

EXPERIMENT NO-10: CIRCULARCONVOLUTION USING DFT AND IDFT

AIM: To find the **Circular** convolution of a given sequence using the inbuilt MATLAB functions "FFT and IFFT" for DFT and IDFT and verify the result using the function "CONV".

THEORY: Let x1(n) and x2(n) are finite duration sequences both of length N with DFT's X1(k) and X2(k). Convolution of two given sequences x1(n) and x2(n) is given by,

x3(n) = IDFT[X3(k)]

x3(n) = IDFT[X1(k) X2(k)]

Where, X1(k) = DFT [x1(n)], X2(k) = DFT [x2(n)]

EXAMPLE:

Let x1(n) = {1, 1, 2, 1} and x2(n) = {1, 2, 3, 4}

$$X1(k) = \sum_{n=0}^{N-1} x1(n) e^{\frac{-j2\,kn}{N}} \quad ; k = 0, 1, 2, \ldots N-1$$

Given x1(n) = {1, 1, 2, 1} and N=4

X1(0) = 1 + 1 + 2 + 1 = 5

X1(1) = 1 – j – 2 + j = -1

X1(2) = 1 – 1 + 2 - 1 = 1

X1(3) = 1 + j - 2 - j = -1

X1(k) = {5, -1, 1, -1}

Now,

$$X2(k) = \sum_{n=0}^{N-1} x2(n) e^{\frac{-j2\,kn}{N}} \quad ; k = 0, 1, 2, \ldots N-1$$

Given x2(n) = {1, 2, 3, 4} and N=4

X2(0) = 1 + 2 + 3 + 4 = 10

X2(1) = 1 + 2(-j) + 3(-1) + 4(j) = -2 + j2

X2(2) = 1 + 2(-1) + 3(1) + 4(-1) = -2

X2(3) = 1 + 2(j) + 3(-1) + 4(-j) = -2 – j2

X2(k) = {10, -2+j2, -2, -2-j2}

We know that,

X3(k) = X1(k) • X2(k)

X3(k) = {50, 2 – j2, -2, 2 + j2}

Convolution of two given sequences is, x3(n) = IDFT[X3(k)]

$$x3(k) = \frac{1}{N}\sum_{k=0}^{N-1} X3(k)e^{\frac{j2\ kn}{N}} \quad ; n = 0, 1, 2, \ldots N-1$$

x3(0)= (1/4) [50 + 2 - j2 – 2 + 2 + j2] = 13

x3(1)= (1/4) [50 + (2 - j2) j + (-2) (-1) + (2 + j2) (-j)] = 14

x3(2)= (1/4) [50 + (2 - j2) (-1) + (-2) (1) + (2 + j2) (-1)] = 11

x3(3)= (1/4) [50 + (2 - j2) (-j) + (-2) (-1) + (2 + j2) (j)] = 12

Convoluted sequence of two given sequences is,

x3(n) = { 13, 14, 11, 12 }

PROGRAM: CIRCULAR CONVOLUTION USING DFT AND IDFT

```
clc; % clear screen
clear all; % clear work space
close all; % close all figure windows
xn = input('enter the first sequence x(n) = ');
hn = input('enter the second sequence h(n) = ');
N = max(length(xn),length(hn));
Xk = fft(xn,N);
Hk = fft(hn,N);
Yk = Xk.*Hk;
yn = ifft(Yk,N);
disp('Circular convolution of x(n) and h(n) =');
disp(yn);
subplot(2,2,1);
stem(xn);
xlabel('n');
ylabel('x(n)');
title('plot of x(n)');
subplot(2,2,2);
stem(hn);
xlabel('n');
ylabel('h(n)');
title('plot of h(n)');
subplot(2,1,2);% stem(yn);
xlabel('n');
ylabel('y(n)');
title('Circular convolution Output');
```

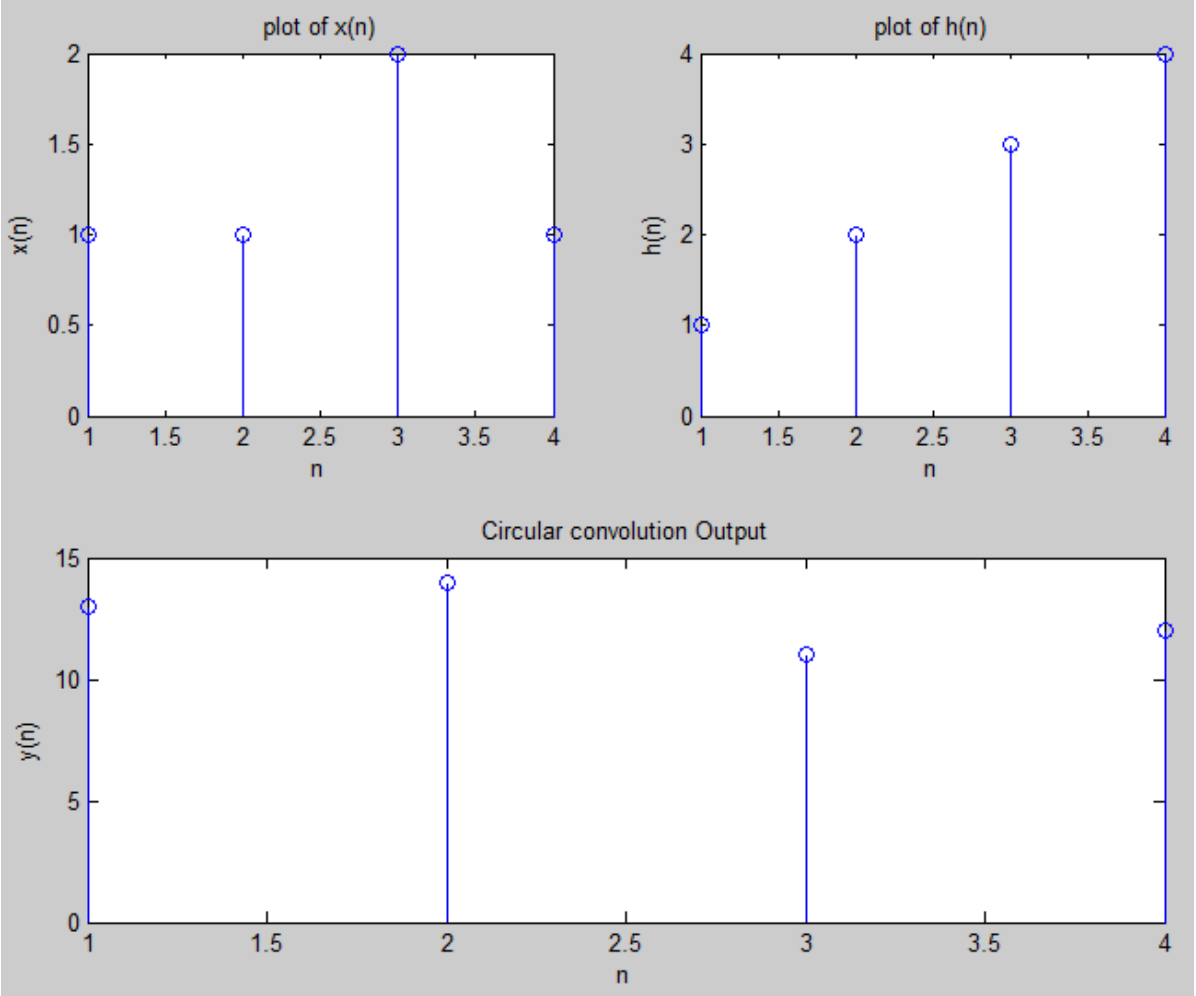
OUTPUT:

Enter the first sequence x(n) = [1 1 2 1]

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

Circular convolution of x(n) and h(n) =

13 14 11 12



OUTCOME: Circular convolution of two given sequences found using DFT and IDFT and the results are verified.

EXPERIMENT NO-11:- DESIGN AND IMPLEMENTATION OF FIR FILTER

AIM: To design the FIR filter by different windowing techniques and using the inbuilt MATLAB function“FIR1”.

THEORY: The FIR filters are of non-recursive type, whereby the present output sample is depending on the present input sample and previous input samples. The transfer function of a FIR causal filter is given by,

$$H(z) = \sum_{n=0}^{N-1} h(n) z^{-n}$$

Where h(n) is the impulse response of the filter.

The Fourier transform of h(n) is

$$H(e^{j\omega}) = \sum_{n=0}^{N-1} h(n) e^{-j\omega n}$$

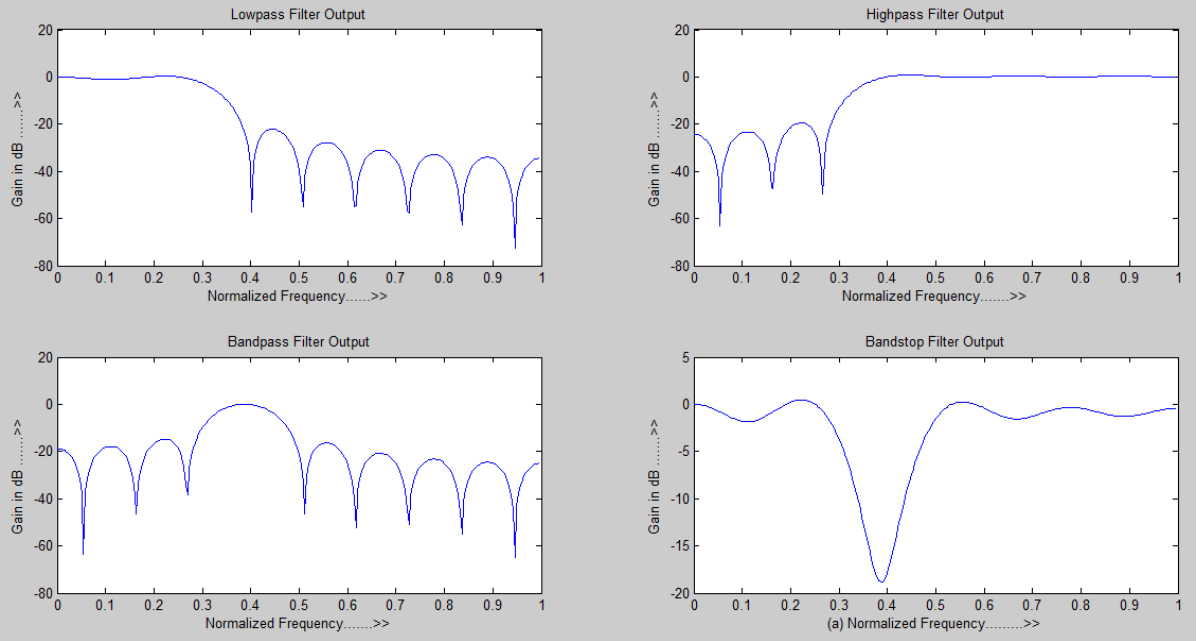
PROGRAM:DESIGN AND IMPLEMENTATION OF FIR FILTER USING RECTANGULAR WINDOW

```
clc
close all
clear all
rp=[0.05];
rs=[0.04];
fp=[1500];
fs=[2000];
f=[9000];
wp=2*(fp/f);
ws=2*(fs/f);
num= -20*log10(sqrt(rp*rs))-13;
dem=14.6*(fs-fp)/f;
n=ceil(num/dem);
disp('Order of the filter is :');
disp(n);
n1=n+1;
if(rem(n,2)~=0)
   n1=n;
   n=n-1;
end
y=rectwin(n1);
%Lowpass Filter
b=fir1(n,wp,y);
[h,g]=freqz(b,1,256);
```

```
m=20*log10(abs(h));
subplot(2,2,1);
plot(g/pi,m); % plot the response
title('Lowpass Filter Output');
xlabel('Normalized Frequency      >>');
ylabel('Gain in dB       >>');
%Highpass       Filter
b=fir1(n,wp,'high',y);
[h,g]=freqz(b,1,256);
m=20*log10(abs(h));
subplot(2,2,2);
plot(g/pi,m);
title('Highpass Filter Output');
xlabel('Normalized Frequency      >>');
ylabel('Gain in dB       >>');
%Bandpass Filter
wn=[wp ws];
b=fir1(n,wn,y);
[h,g]=freqz(b,1,256);
m=20*log10(abs(h));
subplot(2,2,3);
plot(g/pi,m);
title('Bandpass Filter Output');
xlabel('Normalized Frequency      >>');
ylabel('Gain in dB       >>');
%Bandstop       Filter
b=fir1(n,wn,'stop',y);
[h,g]=freqz(b,1,256);
m=20*log10(abs(h));
subplot(2,2,4);
plot(g/pi,m);
title('Bandstop Filter Output');
xlabel('(a) Normalized Frequency        >>');
ylabel('Gain in dB       >>');
```

OUTPUT:

Order of the filter is:



OUTCOME: Design and implementation of FIR filter for the given specifications is done and the desired frequency response is obtained using Rectangular window

.

PROGRAM: DESIGN IMPLEMENTATION OF FIRFILTERUSING HAMMING WINDOW

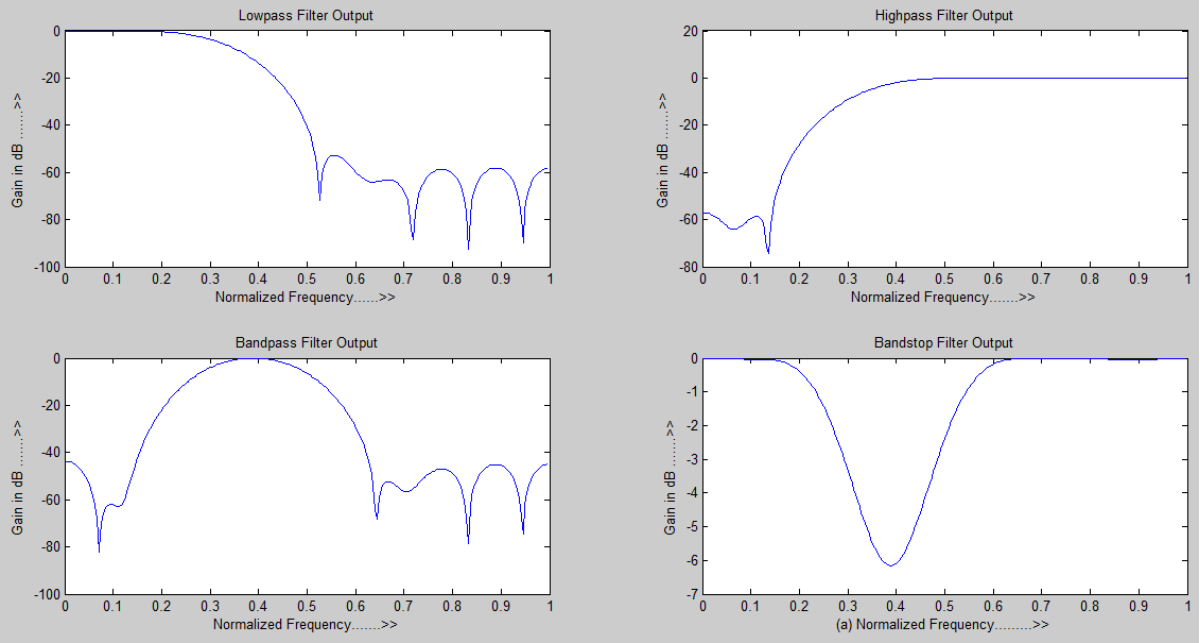
```
clc
close all
clear all
rp=[0.05];
rs=[0.04];
fp=[1500];
fs=[2000];
f=[9000];
wp=2*(fp/f);
ws=2*(fs/f);
num= -20*log10(sqrt(rp*rs))-13;
dem=14.6*(fs-fp)/f;
n=ceil(num/dem);
disp('Order of the filter is :');
disp(n);
n1=n+1;
if(rem(n,2)~=0)
    n1=n;
    n=n-1;
end
```

```
y=hamming(n1);
```

```
b=fir1(n,wp,y);
[h,g]=freqz(b,1,256);
m=20*log10(abs(h));
subplot(2,2,1);
plot(g/pi,m);
title('Lowpass Filter Output');
xlabel('Normalized Frequency      >>');
ylabel('Gain in dB       >>');
b=fir1(n,wp,'high',y);
[h,g]=freqz(b,1,256);
m=20*log10(abs(h));
subplot(2,2,2);
plot(g/pi,m);
title('Highpass Filter Output');
xlabel('Normalized Frequency       >>');
ylabel('Gain in dB       >>');
%Bandpass Filter
wn=[wp ws];
b=fir1(n,wn,y);
[h,g]=freqz(b,1,256);
m=20*log10(abs(h));
subplot(2,2,3);
plot(g/pi,m);
title('Bandpass Filter Output');
xlabel('Normalized Frequency       >>');
ylabel('Gain in dB       >>');
b=fir1(n,wn,'stop',y);
[h,g]=freqz(b,1,256);
m=20*log10(abs(h));
subplot(2,2,4);
plot(g/pi,m);
title('Bandstop Filter Output');
xlabel('(a) Normalized Frequency        >>');
  ylabel('Gain in dB     >>');
```

OUTPUT:

```
Order of the filter is:
   18
```



OUTCOME: Design and implementation of FIR filter for the given specifications is done and the desired frequency response is obtained using Hamming window.

EXPERIMENT NO-12: DESIGN AND IMPLEMENTATION OF IIR FILTER

AIM: Design and implementation of Butterworth IIR Digital filters to meet given specifications using the MATLAB functions BUTTORD and BUTTER.

THEORY: Basically digital filter is a linear time-invariant discrete time system.

Infinite Impulse Response(IIR) filter: IIR filters are of recursive type, whereby the present output sample depends on the present input, past input samples and output samples.The impulse response h(n) for a realizable filter is,h(n) = 0 for n_0. And for stability, it must satisfy the condition,

$$\sum_{n=0}^{\infty} |h(n)| < \infty$$

EXAMPLE:

Let's design an analog Butterworth lowpass filter.

Steps to design an analog Butterworth lowpass filter.

1. Get the pass band and stop band edge frequencies
2. Get the pass band and stop band ripples
3. Get the sampling frequency
4. From the given specifications find the order of the filter N.
5. Round off it to the next higher integer.
6. Find the transfer function H(s) for Ωc = 1rad/sec for the value of N.
7. Calculate the value of cutoff frequency Ωc
8. Find the transfer function Ha(s) for the above value of Ωc by substituting s→ (s/Ωc) in H(s).

PROGRAM:DESIGN AND IMPLEMENTATION OF BUTTERWORTH FILTER

Design: Step 1:

$$w_p = \frac{2 f_p}{F_s} = \frac{2 * 500}{2000} = 0.5 \text{ rad}$$

$$w = \frac{2 f_s}{F_s} = \frac{2 * 750}{2000} = 0.75 \text{ rad}$$

Step 2: T=1

$$\Omega_p = \frac{2}{T}\tan\frac{w_p}{2} \qquad \Omega_s = \frac{2}{T}\tan\frac{w_s}{2}$$

$$\Omega_p = 2\tan\frac{0.5\ \text{rad}}{2} \qquad \Omega_s = 2\tan\frac{0.75\ \text{rad}}{2}$$

$\Omega_p = 2 \frac{rad}{sec}$ $\Omega_s = 4.828 \frac{rad}{sec}$

Step 3: order of filter

$$N \geq \frac{\log \frac{10^{0.1rp} - 1}{10^{0.1rs} - 1}}{2 \log \frac{\Omega_p}{\Omega_s}} \qquad N \geq \frac{\log \frac{10^{0.301} - 1}{10^{1.5} - 1}}{2 \log 4.828}$$

$N \geq 1.941$, so N = 2

Step 4: cut off frequency

$$\Omega_c = \frac{\Omega_s}{(10^{0.1rs} - 1)^{\frac{1}{2N}}} = 2.052 \frac{rad}{sec}$$

Step 5: poles

$$s_k = \pm \Omega_c (N + 2k + 1) 2N$$

Where K=0 to N-1

Therefore s0=-1.45+j1.45; s1=-1.45-j1.45

$$H_a(s) = \frac{\Omega_c^2}{(s - s_0)(s - s_1)} = \frac{4.2107}{(s + 1.45 - j1.45)(s + 1.45 + j1.45)} = s^2 + 2.9s + 4.205$$

PROGRAM:

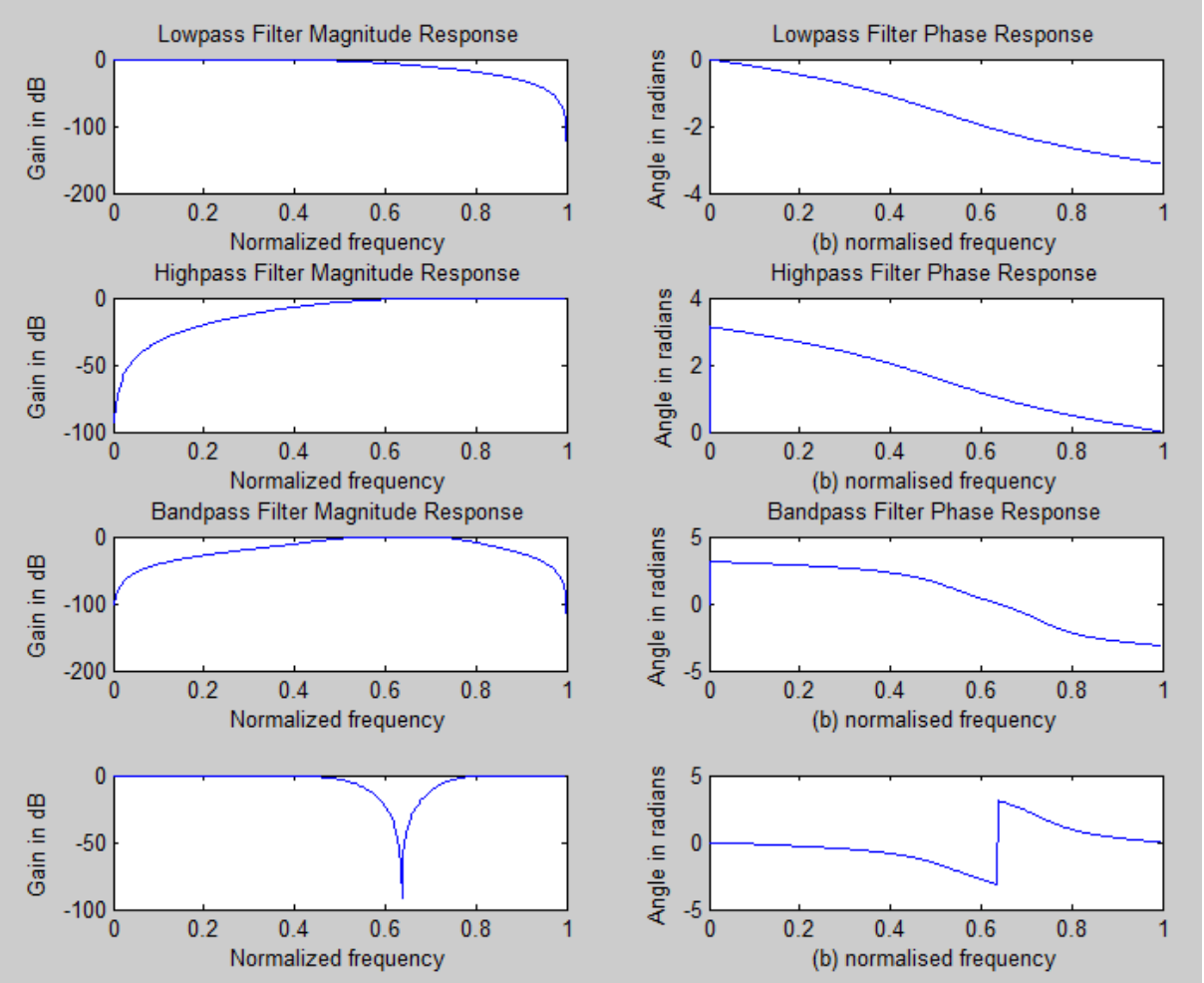
```
clc clear all close all
rp=input('Enter the passband ripple in dB: ');
rs=input('Enter the stopband ripple in dB: ');
wp=input('Enter the passband frequency: ');
ws=input('Enter the stopband frequency: ');
fs=input('Enter the sampling frequency: ');
w1=2*wp/fs;
w2=2*ws/fs;
[n,wn]=buttord(w1,w2,rp,rs);
disp('Order of the filter N =');
disp(n);
[b,a]=butter(n,wn);
w=0:0.01:pi;
[h,g]=freqz(b,a,w);
m=20*log10(abs(h));
an=angle(h);
subplot(4,2,1);
plot(g/pi,m);
ylabel('Gain in dB');
xlabel('Normalized frequency');
title('Lowpass Filter Magnitude Response');
```

```
subplot(4,2,2);
plot(g/pi,an);
ylabel('Angle in radians');
xlabel('(b) normalised frequency');
title('Lowpass Filter Phase Response');
w1=2*wp/fs;
w2=2*ws/fs;
[n,wn]=buttord(w1,w2,rp,rs);
[b,a]=butter(n,wn,'high');
w=0:0.01:pi;
[h,g]=freqz(b,a,w);
m=20*log10(abs(h));
an=angle(h);
subplot(4,2,3);
plot(g/pi,m);
ylabel('Gain in dB');
xlabel('Normalized frequency');
title('Highpass Filter Magnitude Response');
subplot(4,2,4);
plot(g/pi,an);
ylabel('Angle in radians');
xlabel('(b) normalised frequency');
title('Highpass Filter Phase Response');
w1=2*wp/fs;
w2=2*ws/fs;
[n]=buttord(w1,w2,rp,rs);
wn=[w1 w2];
[b,a]=butter(n,wn,'bandpass');
w=0:0.01:pi;
[h,g]=freqz(b,a,w);
m=20*log10(abs(h));
an=angle(h);
subplot(4,2,5);
plot(g/pi,m);
ylabel('Gain in dB');
xlabel('Normalized frequency');
title('Bandpass Filter Magnitude Response');
subplot(4,2,6);
plot(g/pi,an);
ylabel('Angle in radians');
xlabel('(b) normalised frequency');
title('Bandpass Filter Phase Response');
w1=2*wp/fs;
w2=2*ws/fs;
```

```
[n]=buttord(w1,w2,rp,rs);
wn=[w1 w2];
[b,a]=butter(n,wn,'stop');
w=0:0.01:pi;
[h,g]=freqz(b,a,w);
m=20*log10(abs(h));
an=angle(h);
subplot(4,2,7);
plot(g/pi,m);
ylabel('Gain in dB');
xlabel('Normalized frequency');
subplot(4,2,8);
plot(g/pi,an);
ylabel('Angle in radians');
xlabel('(b) normalised frequency');
```

OUTPUT:

```
Enter the passband ripple in dB: 3.01
Enter the stopband ripple in dB: 15
Enter the passband frequency: 500
Enter the stopband frequency: 750
Enter the sampling frequency: 2000
Order of the filter N =
    2
```



OUTCOME: Design and implementation of IIR filter for the given specifications is done and the desired frequency response is obtained