PSG INSTITUTE OF TECHNOLOGY AND APPLIED RESEARCH

NEELAMBUR, COIMBATORE-641 062

EE3024 – DIGITAL SIGNAL PROCESSING SYSTEM DESIGN LABORATORY

DEPARTMENT OF ELECTRICAL AND ELECTRONICS ENGINEERING

ACADEMIC YEAR 2023-2024

PSG INSTITUTE OF TECHNOLOGY AND APPLIED RESEARCH

NEELAMBUR, COIMBATORE-641 062

EE3024 – DIGITAL SIGNAL PROCESSING SYSTEM DESIGN LABORATORY

DEPARTMENT OF ELECTRICAL AND ELECTRONICS ENGINEERING

ACADEMIC YEAR 2023-2024

NAME	
REGISTER No	
BRANCH/YEAR	
SEMESTER	

PSG INSTITUTE OF TECHNOLOGY AND APPLIED RESEARCH

NEELAMBUR, COIMBATORE-641 062

BONAFIDE CERTIFICATE

Certified that this is a Bonafide Record of work done by ……...of III year B. E EEE in the **EE3024 – Digital Signal Processing System Design Laboratory** conducted in this institution, as prescribed by Anna University, Chennai, for the odd semester, during the academic year 2023-2024.

Faculty in-charge **Head of the Department**

Date :

University Register Number: _____

Submitted on: _____

Internal Examiner **External Examiner**

TABLE OF CONTENTS

S.No.	Date	Name of the Experiment	Marks	Signature

		Average Marks		

EXP.No: 01	GENERATION OF ELEMENTARY DISCRETE TIME SEQUENCES
Date:	

AIM:

To write a MATLAB program to generate the following elementary discrete time sequences and plots their responses in discrete time domain.

1. Unit step,
2. Unit impulse,
3. Unit ramp,
4. Sinusoidal signal,
5. Cosine wave
6. Parabolic signal
7. Exponential signal(Growing and Decaying)

APPARATUS REQUIRED:

Hardware: PC with Windows 8 or 10

Software: MATLAB™ 2021a

Algorithm:

Step 1: Start the program

Step 2: Get the dimension of "n"

Step 3: Discrete output is obtained for n>= 0 and zeros for all other values.

Step 4: Output is generated in stem format

Step 5: Terminate the process

PROGRAM:

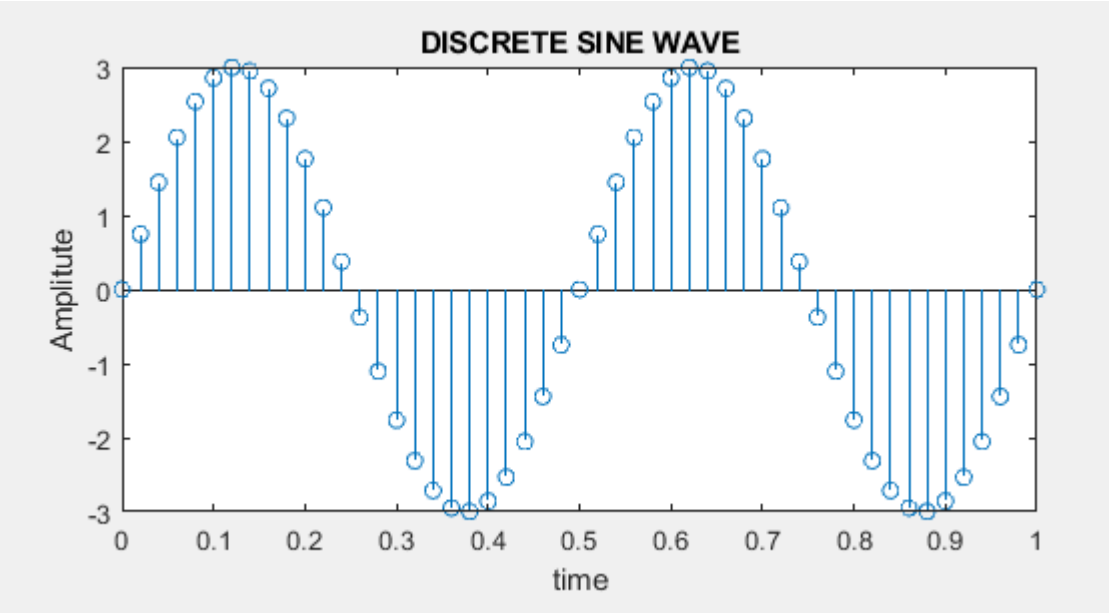
SINUSOIDAL SIGNAL:

```
t=0: 0.02:1
Amp=3
f=2
y=Amp*sin(2*pi*f*t)
subplot(2,2,1)
stem(t,y)
xlabel('time')
ylabel('Amplitute')
title('DISCRETE SINE WAVE')
```

OUTPUT:

SINE SIGNAL

Number of samples=10



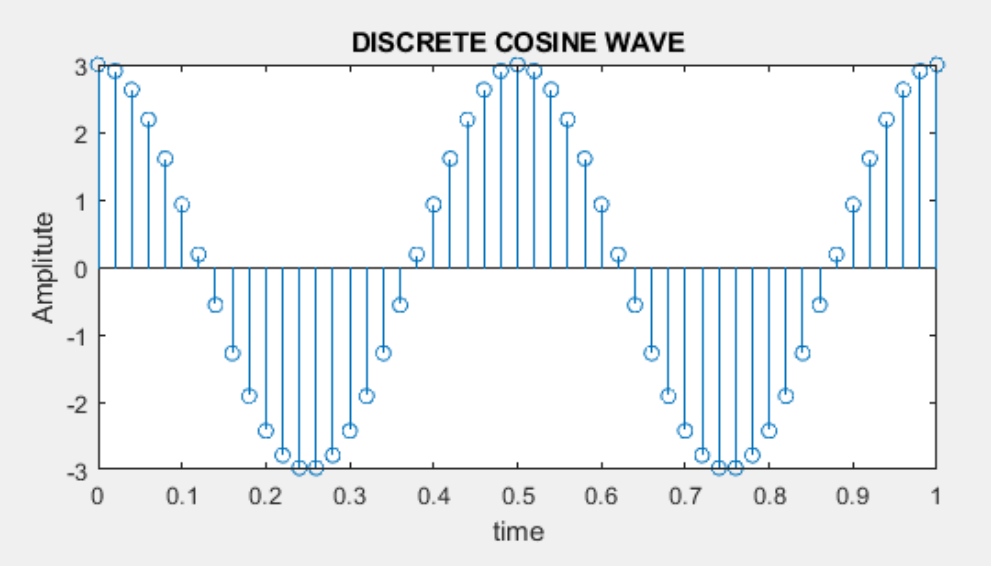
COSINE SIGNAL:

```
z=Amp*cos(2*pi*f*t)
subplot(2,2,2)
stem(t,z)
xlabel('time')
ylabel('Amplitute')
title('DISCRETE COSINE WAVE')
```

OUTPUT:

COSINE SIGNAL

Number of samples=10



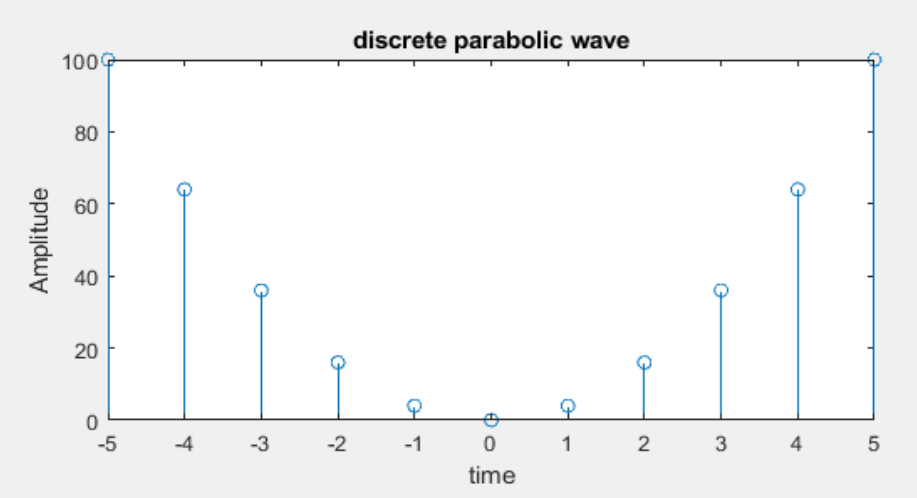
PARABOLIC SIGNAL:

```
x=-5:5
y=4*x.^2
subplot(2,2,3)
stem(x,y)
xlabel('time')
ylabel('Amplitude')
title('discrete parabolic wave')
```

OUTPUT:

PARABOLA SIGNAL

Number of samples=10

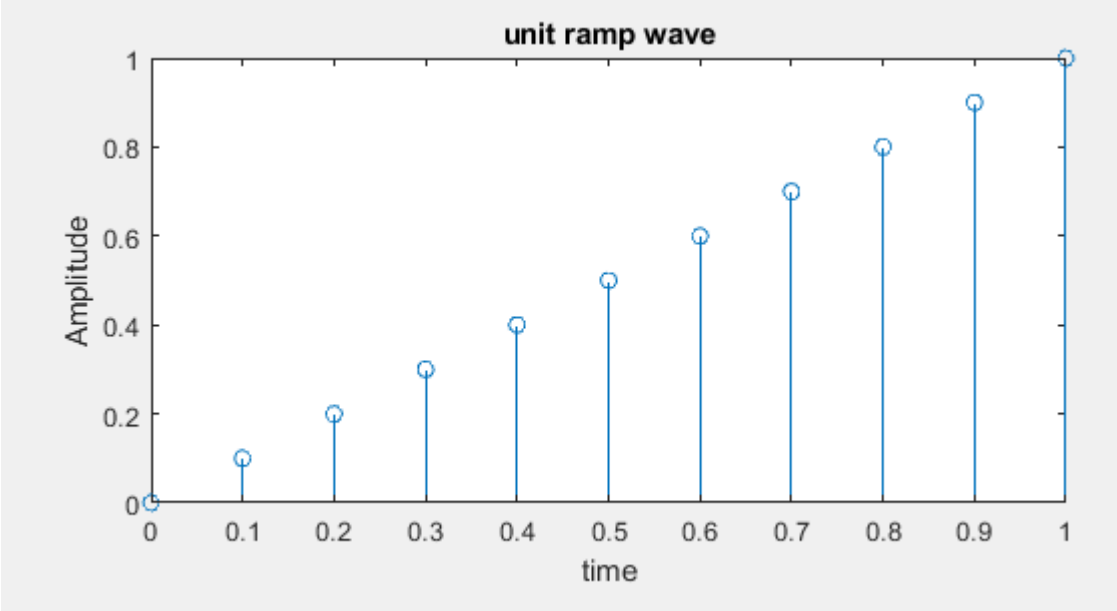


RAMP SIGNAL:

```
T=0:0.1:1
y=T
subplot(2,2,4)
stem(T,y)
xlabel('time')
ylabel('Amplitude')
title('discrete ramp wave')
```

OUTPUT:

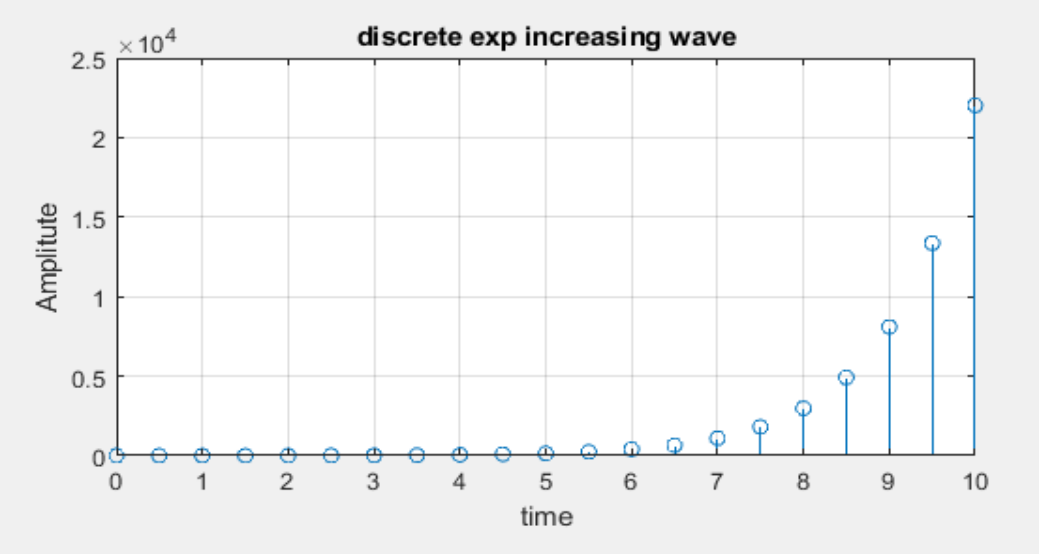
UNIT RAMP SIGNAL
Number of samples=10



EXPONENTIAL INCREASING SIGNAL:

```
x=0:0.5:10
op1=exp(x)
subplot(2,2,1)
stem(x,op1)
xlabel('time')
ylabel('Amplitute')
title('discrete exp increasing wave')
grid on
```

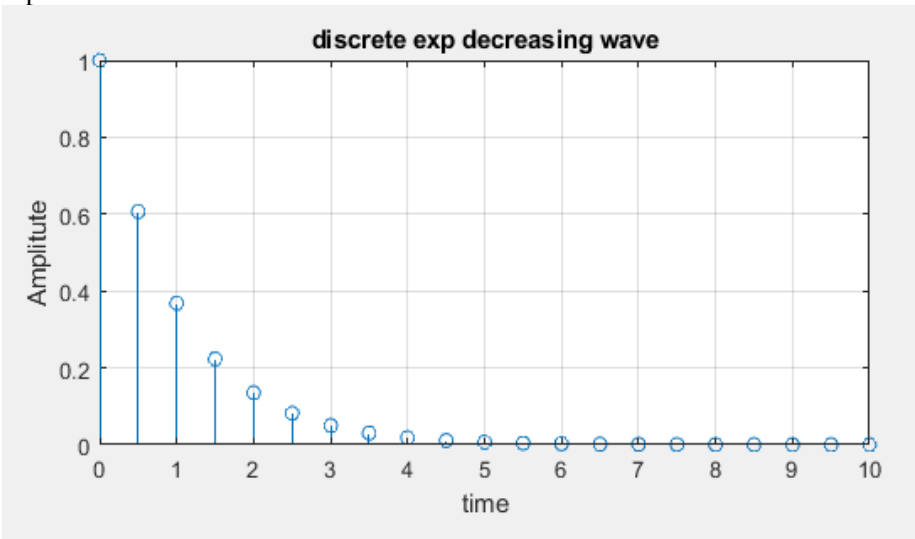
OUTPUT:
EXPONENTIALLY GROWING SIGNAL
Number of samples=10



EXPONENTIAL DECREASING SIGNAL:

```
op2=exp(-x)
subplot(2,2,2)
stem(x,op2)
xlabel('time')
ylabel('Amplitute')
title('discrete exp decreasing wave')
grid on
```

OUTPUT:
EXPONENTIALLY DECAYING SIGNAL
Number of samples=10

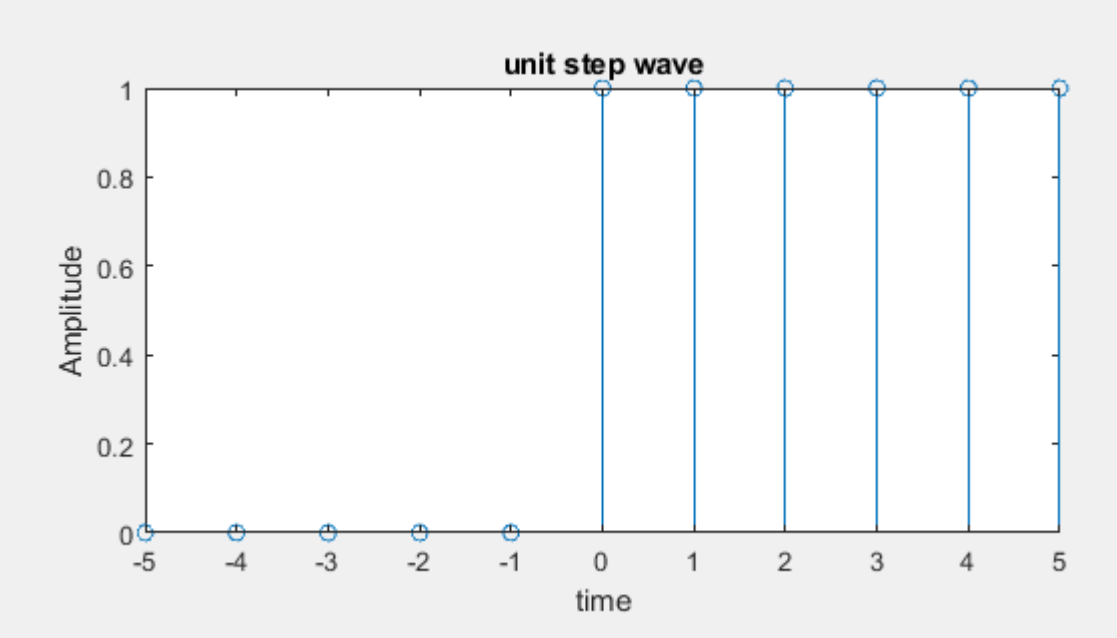


UNIT STEP SIGNAL:

```
x=-5:5;
y=x>=0
subplot(2,2,3)
stem(x,y)
xlabel('time')
ylabel('Amplitude')
title('discrete step wave')
```

OUTPUT:

UNIT STEP SIGNAL
Number of samples=10



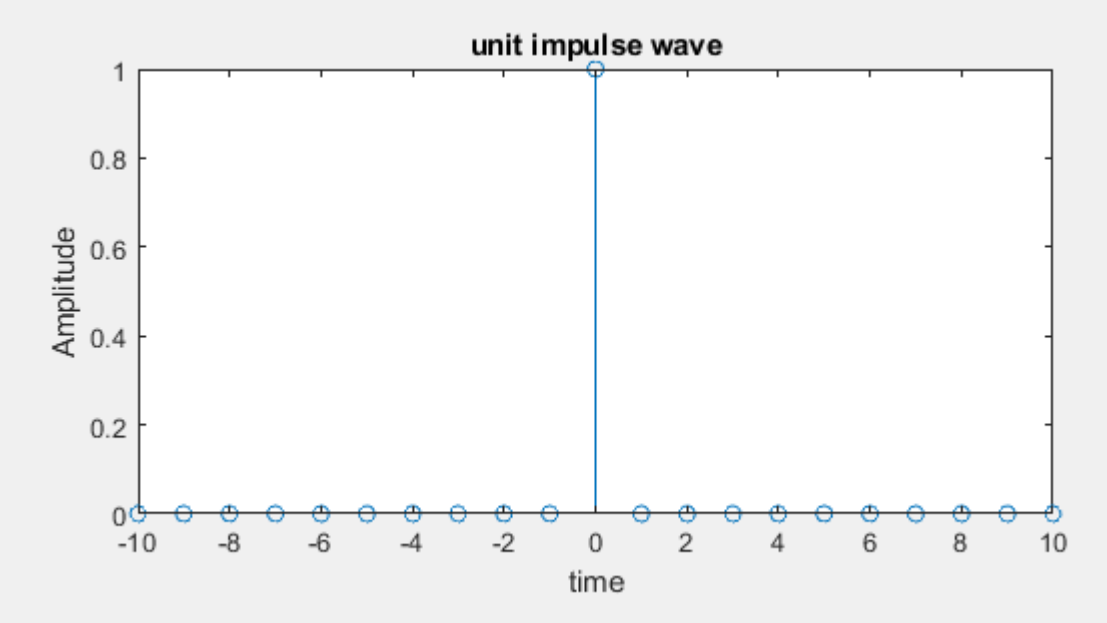
IMPULSE SIGNAL:

```
X=-10:10
Y=X==0
subplot(2,2,4)
stem(X,Y)
xlabel('time')
ylabel('Amplitude')
title('discrete impulse wave')
```

OUTPUT:

UNIT IMPULSE SIGNAL

Number of samples=20



VIVA / POST-LAB QUESTIONS:

1. **Define Stem, Plot, Plot3, fplot, ezplot, linspace, flyplr, grid, mesh and legend.**
    - Stem is for discrete plot.
    - Plot is for continuous plot.
    - Plot3 produces graph in 3d manner.
    - **fplot** plots a function between specified limits.
    - **ezplot**(fun) plots the expression fun(x) over the default domain $-2\pi < x < 2\pi$
    - linspace generates equally spaced vectors in given range.
    - B = **flyplr**(A) returns A with columns flipped in the left-right direction, that is, about a vertical axis.
    - Grid produces basic grid lines in the graph based on the values.
    - **mesh**(X,Y,Z) draws a wireframe **mesh** with color determined by Z
    - **legend** associates strings with the objects in the axes in the same order that they are listed in the axes Children property.

2. **Define impulse signal, sinusoidal signal and ramp signal.**
    - The **impulse** that is referred to in the term **impulse** response is generally a short-duration time-domain **signal**.

- A **sinusoidal signal** is the only periodic **signal** where it retains its wave shape when added to another **sinusoidal signal** of the same frequency with arbitrary initial phase and amplitude.
- The **ramp function** is a unary real **function**, whose graph is shaped like a **ramp**.

3. Define unit step and exponential signal

- **Unit step**: A **signal** with magnitude one for time greater than zero. We can assume it as a dc **signal** which got switched on at time equal to zero.
- The complex **exponential** is a complex valued **signal** that simultaneously encapsulates both a cosine **signal** and a sine **signal** by posting them on the real and imaginary components of the complex **signal**.

4. Write a program and plot to continuous time signal $x(t)=2\sin\pi t$ with a interval $0 \leq t \leq 2$

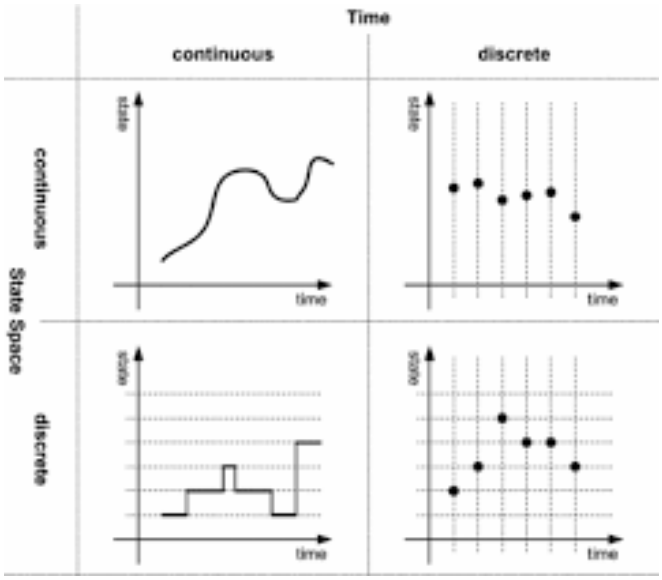
```
t=0:0.1:2;
y=2*sin(pi*t);
plot(t,y);
```

5. Compare Continuous Time System & Discrete Time System with neat sketch.

The **difference between discrete** and **continuous** data can be drawn clearly on the following grounds: ...

Discrete data is countable while **continuous** data is measurable.

Discrete data contains distinct or separate values. On the other hand, **continuous** data includes any value within range.



RESULT:

Thus the MATLAB programs for unit step, unit impulse, unit ramp, sinusoidal signal, cosine, exponential signals were generated and their responses were plotted in discrete domain successfully.

EXP.No: 02	LINEAR AND CIRCULAR CONVOLUTIONS
Date:	

AIM:

To write a MATLAB program to obtain the linear and circular convolutions for the input sequence(s) and plot their response in discrete time domain.

APPARATUS REQUIRED:

Hardware **:** PC with Windows 8 or 10

Software **:** MATLAB™ 2021a

ALGORITHM:

Step 1: Start the program.

Step 2: Get the sequence x(n).

Step 3: Get the range of the sequence x(n).

Step 4: Get the sequence h(n).

Step 5: Get the range of the sequence h(n).

Step 6: Find the convolution between the sequences x(n) and h(n)

Step 7: Plot the convoluted sequence y(n).

Step 8: Terminate the process.

THEORY:

Convolution is a mathematical operation used to express the relation between input and output of an LTI system. It relates input, output and impulse response of an LTI system as

$$y(n)=x(n)*h(n)$$

Where

y (n) = output of LTI

x (n) = input of LTI

h (n) = impulse response of LTI

Discrete Convolution $y(n)=x(n)*h(n)$

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

By using convolution we can find zero state response of the system.

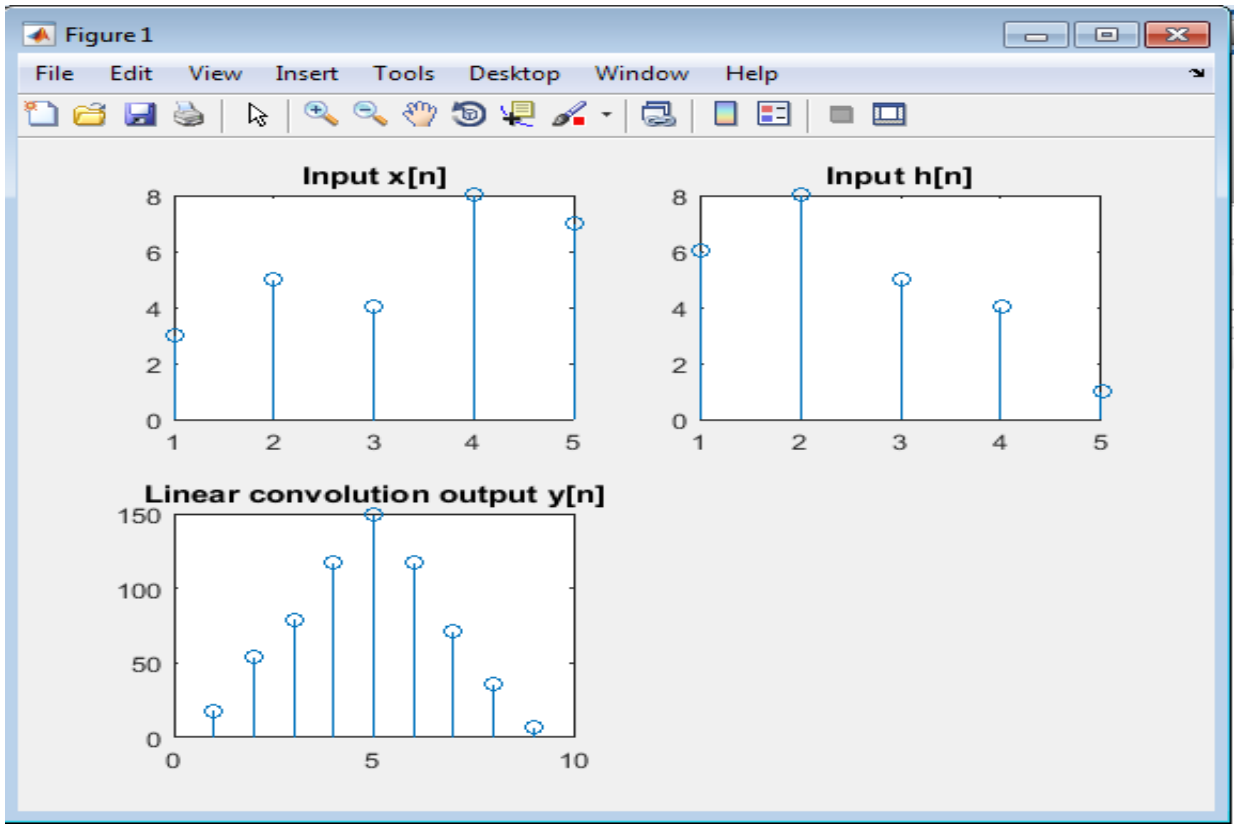
PROGRAM:

LINEAR CONVOLUTION:

```
x=input('Enter x[n]:')
h=input('Enter h[n]:')
m=length(x);
n=length(h);
X=[x,zeros(1,n)];
H=[h,zeros(1,m)];
For i=1:n+m-1
Y(i)=0;
for j=1:m
if(i-j+1>0)
Y(i)=Y(i)+X(j)*H(i-j+1);
end
end
end
Y
subplot(2,2,1)
stem(x);
title('Input x[n]');
subplot(2,2,2);
stem(h);
title('Input h[n]')
subplot(2,2,3);
stem(Y);
title('Linear convolution output y[n]')
```

OUTPUT (Linear Convolution)

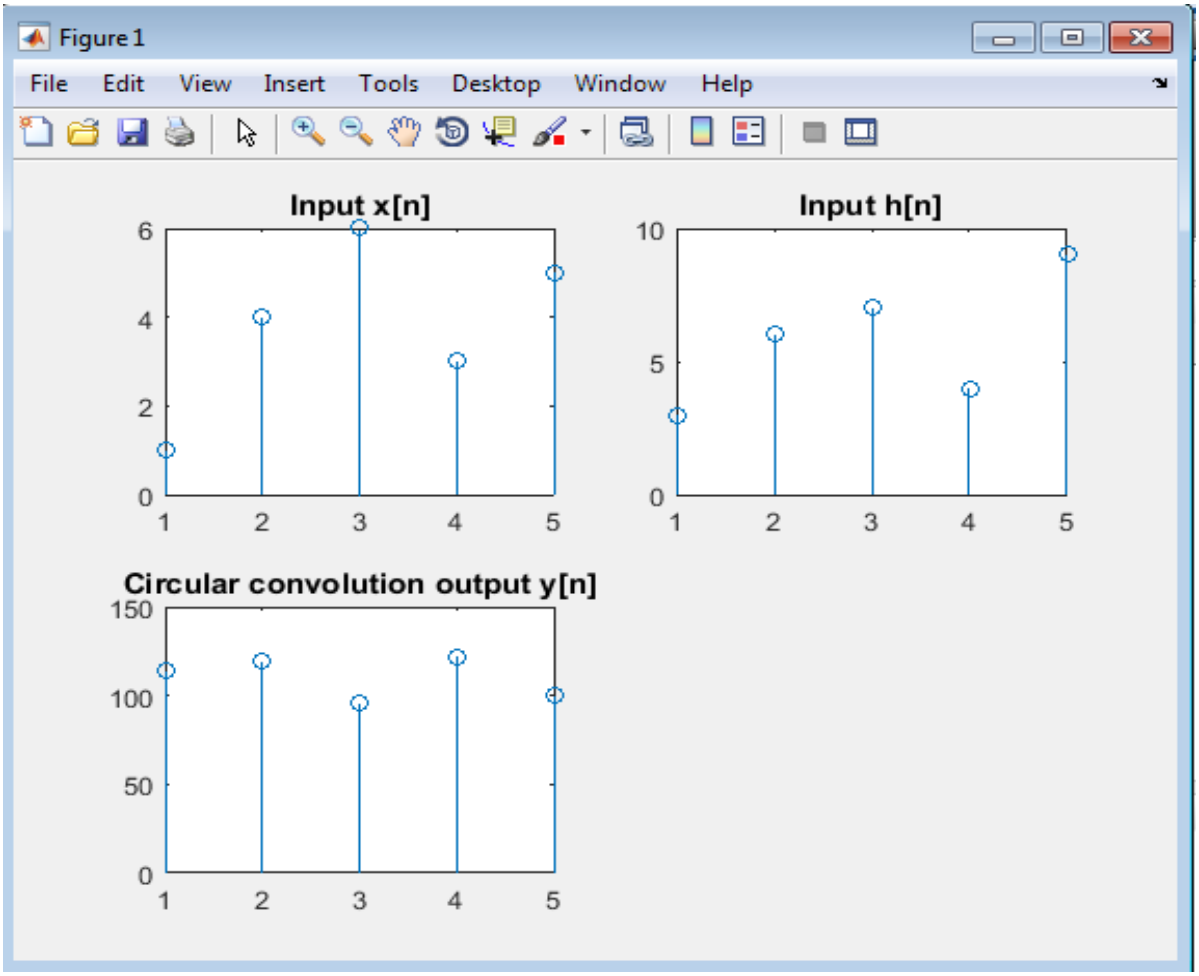
```
Command Window
>> Lconv
Enter x[n]:[3 5 4 8 7]

x =

     3     5     4     8     7

Enter h[n]:[6 8 5 4 1]

h =

     6     8     5     4     1


Y =

    18    54    79   117   149   117    71    36     7
```



CIRCULAR CONVOLUTION:

```
x=input('Enter x[n]:');
h=input('Enter h[n]:');
rev_h = [h(1) h(1,numel(x):-1:2)];
y=zeros(1,numel(x));
z=zeros(1,numel(x));
fori=1:numel(x)
  z=x.*rev_h;
y(i)=sum(z);
temp = rev_h(1,numel(z));
rev_h = [temp rev_h(1,1:numel(y)-1)];
end
disp(y);
subplot(2,2,1)
stem(x);
title('Input x[n]');
subplot(2,2,2)
stem(h);
title('Input h[n]');
subplot(2,2,3);
stem(y);
title('Circular convolution output y[n]')
```

OUTPUT (Circular Convolution)



VIVA / POST-LAB QUESTIONS:

1. **What is the length of linearly convolved signals?**

   L= N1+N2-1

2. **Why linear convolution is important in DSP?**

Convolution is a mathematical way of combining two signals to form a third signal. It is the single most **important** technique in **Digital Signal Processing**. ...

Convolution is important because it relates the three signals of interest: the input signal, the output signal, and the impulse response.

3. **What is zero padding? What are its uses?**

A common tool in frequency analysis of sampled signals is to use **zero-padding** to increase the frequency resolution of the discrete Fourier transform (DFT). By appending artificial zeros to the signal, we obtain a denser frequency grid when applying the DFT.

4. **How to obtain the output sequence of linear convolution through circular convolution?**

If the sequences are not padded with (a sufficient number of) zeros you cannot retrieve the linear convolution. If the sequences are padded with zeros the linear convolution is identical to (a part of) the circular convolution.

This property is used to calculate the linear convolution more efficiently, by calculating the circular convolution, which in turn can be calculated very efficiently in the frequency domain using the FFT algorithm.

5. **Define Sectional Convolution? List the methods used for the sectional convolution.**

The convolution is performed by dividing the long input sequence into different fixed size sections, it is called sectioned convolution.

A long input sequence is segmented to fixed size blocks, prior to FIR filter processing.

Two methods are used to evaluate the discrete convolution −

- **Overlap-save method**
- **Overlap-add method**

6. **Determine the convolution (LC and CC) sum of two sequences x (n) = {3, 2, 1, 2} and h (n) = {1, 2, 1, 2}**

LC: {3,8,8,12,9,4,4} CC: {12,12,12,12}

RESULT:

Thus the MATLAB program to obtain the linear and circular convolutions for the input sequence(s) was executed and their responses were plotted in discrete time domain successfully.

EXP.No: 03	FREQUENCY ANALYSIS USING DFT and IDFT
Date:	

AIM:

To write a MATLAB program to obtain frequency analysis by Discrete Fourier Transform and Inverse Discrete Fourier Transform for the input sequence(s) and plot their responses.

APPARATUS REQUIRED:

Hardware: PC with Windows 8 or 10

Software : MATLAB™ 2021a

THEORY

Basic equation to find the DFT of a sequence is given below.

$$X(k) = \sum_{n=0}^{N-1} x(n) W_N^{nk}$$

$$where \quad W_N^{nk} = e^{-j\frac{2\pi nk}{N}} \text{ [TWIDDLE FACTOR]}$$

Basic equation to find the IDFT of a sequence is given below.

$$x(n) = \frac{1}{N} \sum_{k=0}^{N-1} X(k)\, W^{nk}$$

Where,

$$W = e^{-j2\pi/N}$$

ALGORITHM:

STEP 1: Start the process

STEP 2: Get the Input sequence form the user.

STEP 3: Get the length of the sequence.

STEP 4: Initialize the scale for k and n.

STEP 5: Find the DFT of the input sequence using direct equation of DFT

STEP 6: Compute the Real and Imaginary parts of the spectrum

STEP 7: Plot the Amplitude and Frequency in x & y axis respectively

STEP 8: Find the DFT of the input sequence using direct equation of DFT

STEP 9: Compute the Real and Imaginary parts of the spectrum

STEP 10: Plot the Amplitude and Frequency in x & y axis respectively

STEP 11: Stop the program

PROGRAM:

DFT:

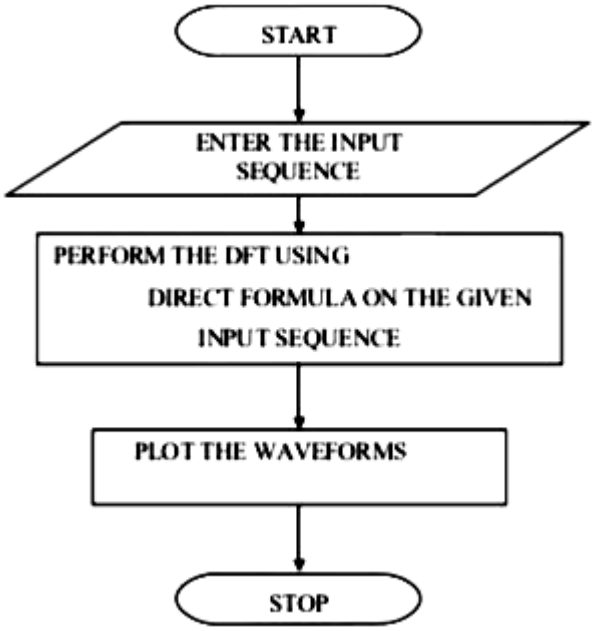
```
xn=input('Enter x : ');
N=input('Number of DFT points : ');
m=length(xn);
xn=[xn zeros(1,N-m)];
xk=zeros(1,N);
for k=1:N
for n=1:N
xk(k)=xk(k)+xn(n)*exp((-1j*2*pi*(k-1)*(n-1))/N);
end
end
disp("DFT: ");
disp(xk);
t=0:N-1;
subplot(2,2,1);
stem(t,xn);
title('x(n)');
xlabel('Time');
ylabel('Amplitude');
subplot(2,2,2);
stem(t,xk);
title('X(k)');
xlabel('Time');
ylabel('Amplitude');
mag=abs(xk);
disp("Magnitude value : ");
disp(mag);
subplot(2,2,3);
stem(t,mag);
title('Magnitude response');
xlabel('Time');
ylabel('Magnitude');
phase=angle(xk);
disp("Phase value : ");
disp(phase);
subplot(2,2,4);
stem(t,phase);
title('Phase response');
xlabel('Time');
ylabel('Phase');
```

IDFT:

```
Xk=input('Enter X[k]: ');
N=length(Xk);
xn=zeros(1,N);
for n=1:N
for k=1:N
xn(n)=xn(n)+Xk(k)*exp((1j*2*pi*(n-1)*(k-1))/N);
end
xn(n)=xn(n)/N;
```

```
end
disp("IDFT value :");
disp(xn);
t=0:N-1;
subplot(2,2,1);
stem(t,Xk);
title('X(k)');
xlabel('Time');
ylabel('Amplitude');
subplot(2,2,2);
stem(t,xn);
title('x(n)');
xlabel('Time');
ylabel('Amplitude');
mag=abs(xn);
disp("Magnitude value :");
disp(mag);
subplot(2,2,3);
stem(t,mag);
title('Magnitude response');
xlabel('Time');
ylabel('Magnitude');
phase=angle(xn);
disp("Phase value :");
disp(phase);
subplot(2,2,4);
stem(t,phase);
title('Phase response');
xlabel('Time');
ylabel('Phase');
```

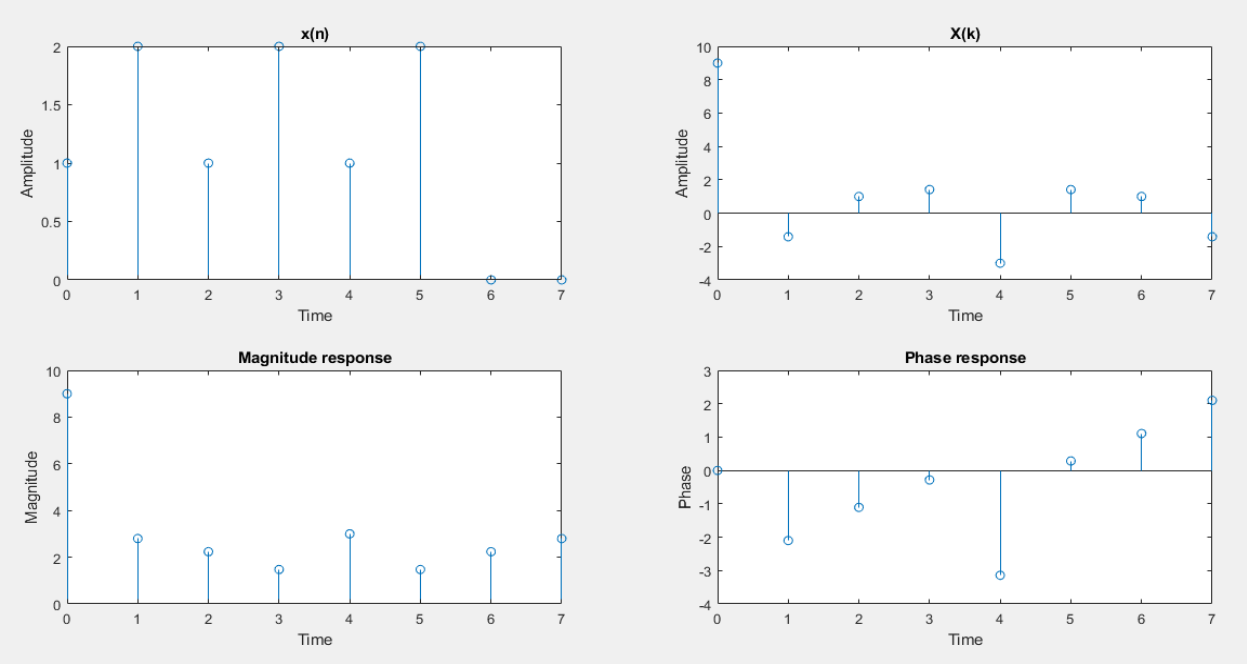
FLOW CHART:



OUTPUT :

DFT:

Enter x : [1 2 1 2 1 2]
Number of DFT points : 8
DFT:
Columns 1 through 6
9.0000 + 0.0000i -1.4142 - 2.4142i 1.0000 - 2.0000i 1.4142 - 0.4142i -3.0000 - 0.0000i 1.4142 + 0.4142i
Columns 7 through 8
1.0000 + 2.0000i -1.4142 + 2.4142i
Magnitude value :
  9.0000 2.7979 2.2361 1.4736 3.0000 1.4736 2.2361 2.7979
Phase value :
  0 -2.1007 -1.1071 -0.2849 -3.1416 0.2849 1.1071 2.1007

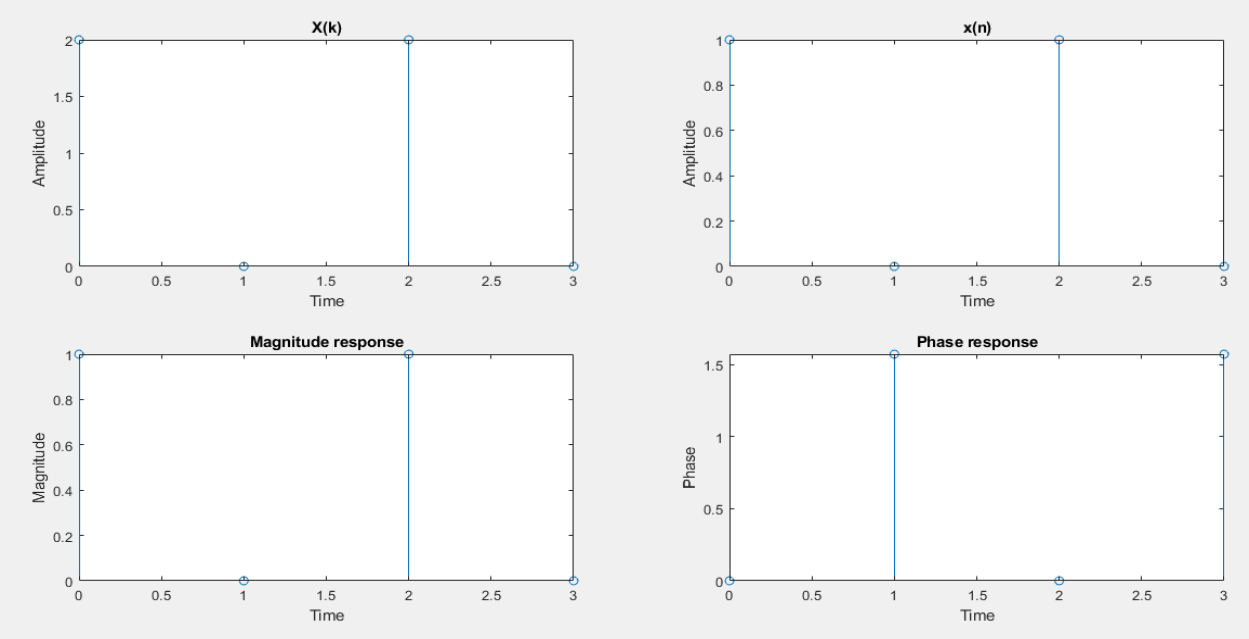


DFT using inbuilt function:

```
>> fft([1 2 1 2 1 2 0 0])

ans =

  Columns 1 through 6

   9.0000 + 0.0000i  -1.4142 - 2.4142i   1.0000 - 2.0000i   1.4142 - 0.4142i  -3.0000 + 0.0000i   1.4142 + 0.4142i

  Columns 7 through 8

   1.0000 + 2.0000i  -1.4142 + 2.4142i
```

IDFT

Enter X[k]: [2 0 2 0]
IDFT value:
1.0000 + 0.0000i 0.0000 + 0.0000i 1.0000 - 0.0000i 0.0000 + 0.0000i
Magnitude value:
1.0000 0.0000 1.0000 0.0000
Phase value:
0 1.5708 -0.0000 1.5708



IDFT using inbuilt function:

```
>> ifft([2 0 2 0])

ans =

     1     0     1     0
```

VIVA / POST-LAB QUESTIONS:

1. **Define signal. Give Examples for 1-D, 2-D, 3-D signals.**
   A signal is a description of how one parameter varies with another parameter. For instance, voltage changing over time in an electronic circuit, or brightness varying with distance in an image.
   1-D : Speech, Music, ECG
   2-D : Ultrasound scan
   3-D : Animated or 3D cartoon

2. **Define transform. What is the need for transform?**
   Transforms model a signal as a collection of waveforms of a particular form: sinusoids for the Fourier transform, mother wavelets for the wavelet transforms, periodic basis functions for the periodicity transforms.
   Transforms look at signals from a domain other than the natural domain. Transforms are essential for understanding some properties of a signal. The Fourier transform is an important transform to begin with. A Discrete Signal x[n] can be thought of as a vector with countably infinite dimensions.

3. **Differentiate Fourier transform and Discrete Fourier Transform.**

Sr No	Fourier Transform (FT)	Discrete Fourier Transform (DFT)
1	FT x(ω) is the continuous function of x(n).	DFT x(k) is calculated only at discrete values of ω. Thus DFT is discrete in nature.
2	The range of ω is from - ∏ to ∏ or 0 to 2∏.	Sampling is done at N equally spaced points over period 0 to 2∏. Thus DFT is sampled version of FT.
3	FT is given by equation (1)	DFT is given by equation (2)
4	FT equations are applicable to most of infinite sequences.	DFT equations are applicable to causal, finite duration sequences
5	In DSP processors & computers applications of FT are limited because x(ω) is continuous function of ω.	In DSP processors and computers DFT□s are mostly used. **APPLICATION** a) Spectrum Analysis b) Filter Design

4. **How to calculate FT for 1-D signal?**
   FT for 1-D signal is calculated as : $$F(\xi) = \int_{-\infty}^{\infty} f(t)\, e^{-2\pi i \xi t}\, dt$$

5. **How many multiplication terms are required for doing DFT by expressional method and FFT method?**
   DFT expressional method requires $\mathbf{N}^2$ complex multiplications and $\mathbf{N}^2$-**N** complex additions.
   FFT method requires $(\mathbf{N}/2)\log_2\mathbf{N}$ complex multiplications and $\mathbf{N}\log_2\mathbf{N}$ complex additions.

6. **Find 8-point DFT of the sequence x (n) = [1 2 3 4 4 3 2 1]**

DFT : [20 ,-5.828-2.414j ,0 ,-0.172-0.414j ,0 ,-0.172+0.414j ,0 ,-5.828+2.414j]

RESULT

Thus the computation of DFT and IDFT of the sequence was performed and their magnitude and phase spectrum were plotted.

EXP.No: 04	FREQUENCY ANALYSIS USING FFT
Date:	

AIM:

To write a MATLAB program to obtain frequency analysis by Fast Fourier Transform for the input sequence(s) and plots their responses.

APPARATUS REQUIRED:

Hardware: PC with Windows 8 0r 10

Software: MATLAB™ 2021a

THEORY:

Basic equation to find the DFT of a sequence is given below.

$$X(k) = \sum_{n=0}^{N-1} x(n) W_N^{nk}$$

$$where \quad W_N^{nk} = e^{-j\frac{2\pi nk}{N}} \text{ [TWIDDLE FACTOR]}$$

ALGORITHM:

STEP 1 : Start the process

STEP 2 : Get the Input sequence form the user.

STEP 3 : Get the length of the sequence.

STEP 4: Initialize the scale for k and n.

STEP 5: Find the FFT of the input sequence.

STEP 6: Compute the Real and Imaginary parts of the spectrum

STEP 7 : Plot the Amplitude and Frequency in x & y axis respectively

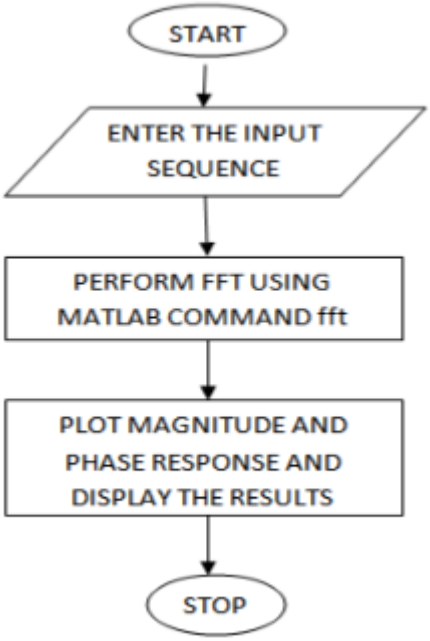
STEP 8: Find the IFFT of the input sequence.

STEP 9: Compute the Real and Imaginary parts of the spectrum

STEP 10 : Plot the Amplitude and Frequency in x & y axis respectively

STEP 11 : Stop the program

FLOW CHART:



PROGRAM for FFT – DIT :

```
xn=input('Enter x[n] : ');
N=input('No of points : ');
n=length(xn);
xn=[xn zeros(1,N-n)]
y=bitrevorder(xn);
M=log2(N)
for m=1:M
   d=2^m;
for l=1:d:N-d+1
for k=0:(d/2)-1
          v=l+k;
          w=exp(-1j*2*pi*k/d);
          b=y(v);
          c=y(v+d/2);
          y(v)=b+(w*c);
          y(v+d/2)=b-(w*c);
end
end
end
disp('X[k] by DIT FFT method : ');
disp(y);
subplot(2,2,1)
stem(xn);
title('x[n]');
xlabel('Time');
ylabel('Amplitude');
subplot(2,2,2)
stem(y);
title('DIT FFT X[k]');
xlabel('Time');
ylabel('Amplitude');
subplot(2,2,3)
```

```
stem(abs(y));
title('Magnitude Response');
xlabel('Time');
ylabel('Magnitude');
subplot(2,2,4)
stem(angle(y));
title('Phase Response');
xlabel('Time');
ylabel('Phase');
```

PROGRAM for FFT - DIF :

```
N=input('Enter the Point    : ');
X=input('Enter the sequence : ');
n=length(X);
x=[X zeros(1,N-n)];
M=log2(N);
for m=1:M
   d=2^(M-m+1);
for l=1:d:(N-d+1)
for k=0:(d/2)-1
          w=exp(-1i*2*pi*k/d);
          z1=x(l+k);
          z2=x(l+k+d/2);
          x(l+k)=z1+z2;
          x(l+k+d/2)=(z1-z2)*w;
end
end
end
y=bitrevorder(x);
disp(y)
subplot(2,2,1)
stem(abs(X));
title('Input Sequence');
subplot(2,2,2)
stem(abs(y));
title('Magnitude Response');
subplot(2,2,3)
stem(angle(y));
title('Phase Response');
subplot(2,2,4)
stem(y);
title('Output Sequence');
```

OUTPUT :

FFT – DIT :

Enter x[n] : [1 2 1 2 1 2]

No of points : 8

xn =1 2 1 2 1 2 0 0

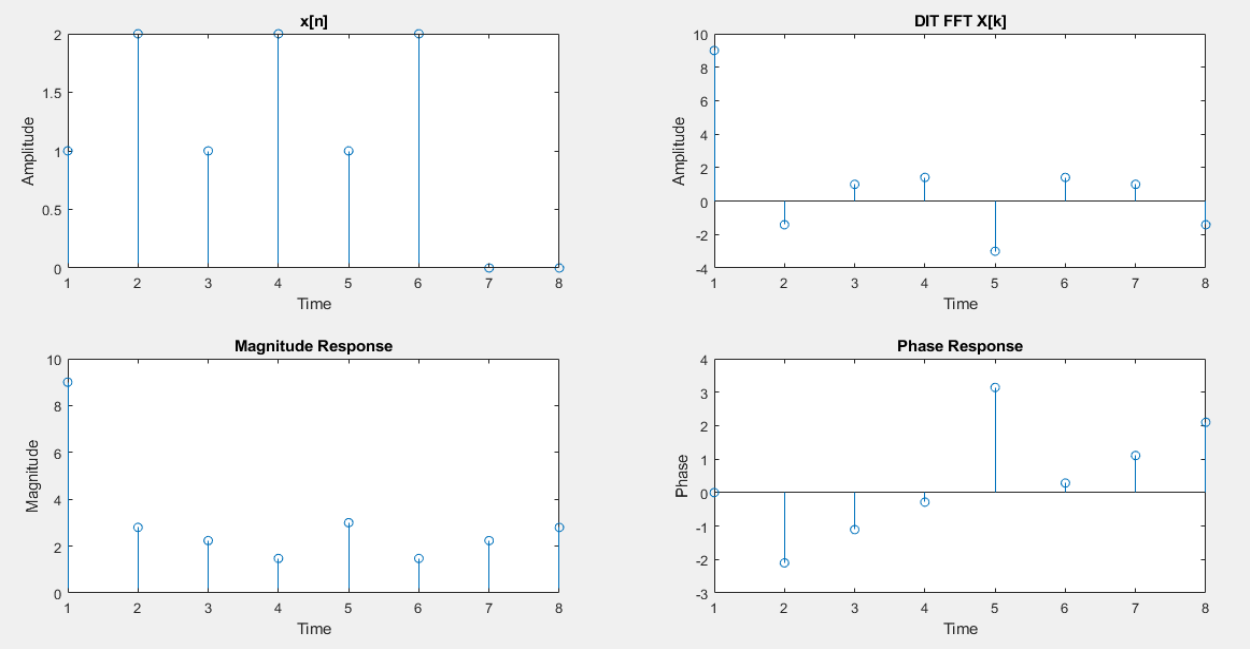
X[k] by DIT FFT method :

Columns 1 through 6

9.0000 + 0.0000i -1.4142 - 2.4142i 1.0000 - 2.0000i 1.4142 - 0.4142i -3.0000 + 0.0000i 1.4142 + 0.4142i

Columns 7 through 8

1.0 + 2.0000i -1.4142 + 2.4142i

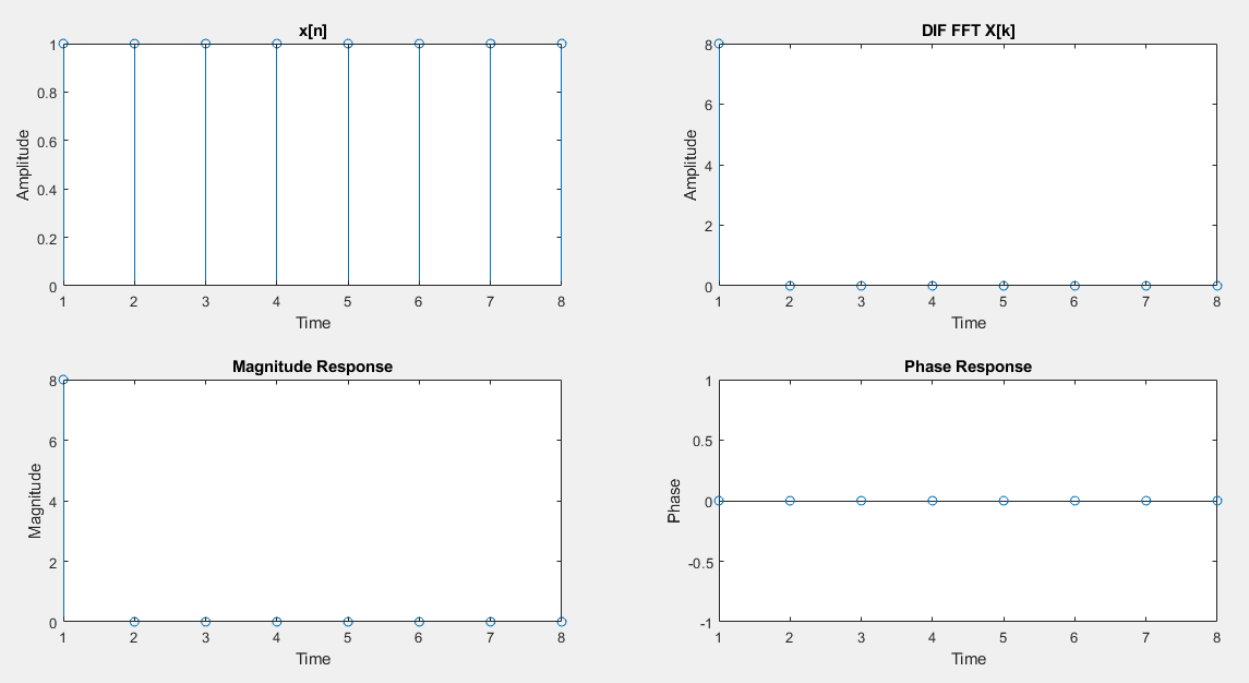


DIT (Using inbuilt function) :

```
>> fft([1 2 1 2 1 2 0 0])

ans =

  Columns 1 through 6

   9.0000 + 0.0000i  -1.4142 - 2.4142i   1.0000 - 2.0000i   1.4142 - 0.4142i  -3.0000 + 0.0000i   1.4142 + 0.4142i

  Columns 7 through 8

   1.0000 + 2.0000i  -1.4142 + 2.4142i
```

FFT – DIF :

Enter x[n] : [1 1 1 1 1 1 1 1]
No of points : 8
x[n] =1 1 1 1 1 1 1 1
X[k] by DIF FFT method :
  8 0 0 0 0 0 0 0



DIF (Using inbuilt function) :

```
>> fft([1 1 1 1 1 1 1 1])

ans =

     8     0     0     0     0     0     0     0
```

VIVA / POST-LAB QUESTIONS:

1. **What is meant by bit reversal in FFT program logics?**
   Bit reversal means mirror imaging the binary numbers. Some signal processors can do this in hardware.

2. **What are the advantages of FFT over DFT?**
   FFT helps in converting the time domain in frequency domain which makes the calculations easier as we always deal with various frequency bands in communication system another very big advantage is that it can convert the discrete data into a continuous data type available at various frequencies.

3. Differentiate DIT-FFT and DIF-FFT algorithms.

Sr No	DIT FFT	DIF FFT
1	DITFFT algorithms are based upon decomposition of the input sequence into smaller and smaller sub sequences.	DIFFFT algorithms are based upon decomposition of the output sequence into smaller and smaller sub sequences.
2	In this input sequence x(n) is splitted into even and odd numbered samples	In this output sequence X(k) is considered to be splitted into even and odd numbered samples
3	Splitting operation is done on time domain sequence.	Splitting operation is done on frequency domain sequence.
4	In DIT FFT input sequence is in bit reversed order while the output sequence is in natural order.	In DIFFFT, input sequence is in natural order. And DFT should be read in bit reversed order.

4. What is radix-2 FFT?

8 point radix-2 DIT-FFT: FFT is an algorithm to convert a time domain signal to DFT efficiently. ... In each algorithm, depending on the sequence needed at the output, the input is regrouped. The groups are decided by the number of samples.

5. Calculate the number of complex multiplications and additions needed in the calculation of a N-point radix-2 FFT when compared to direct DFT

The total number of complex multiplications is (N/2)log2N and the number of complex additions is Nlog2N.

6. Find 8-point DFT of sequence x(n)=[1 2 1 2 3 4 4 3] using FFT algorithm.

DFT : [20 ,-2.707+5.121j ,-1-j ,-1.293-0.879j ,-2 ,-1.293+0.879j ,-1+j ,-2.707-5.121j]

7. What is the need for FFT? Mention the applications of FFT.

FFT is a simpler and faster method of implementing DFT. This is very useful when the value of N is large.

The FFT has lots of applications and is used extensively in audio processing, radar, sonar and software defined radio.

RESULT:

Thus the computation of FFT and IFFT of the sequence was performed and their magnitude and phase spectrum were plotted.

EXP.No: 05	DESIGN OF FIR FILTERS USING MATLAB
Date:	

Aim:

To write a MATLAB program for the design of FIR filters (LPF/HPF/BSF/BPF) for the given cut off frequency using Rectangular, Hamming and Hanning windowing techniques. Also plot the magnitude and log magnitude responses for the same.

Apparatus Required:

Hardware: PC with Windows 8 0r 10

Software: MATLAB™ 2021a

Algorithm:

Step 1 : Start the program.

Step 2 : Get the cutoff frequency and sampling frequency from the user.

Step 3 : Get the filter order value of N.

Step 4 : Get the window function ω (n) for different windowing techniques in Z-domain.

Step 5 : Design a linear phase FIR filter using 'fir1' command for LPF/HPF/BSF/BPF

Step 6 : Compute the complex frequency response of digital transfer function at specified frequency points using 'freqz' command.

Step 7 : Plot the magnitude response and log magnitude response.

Step 8 : Terminate the process.

Program:

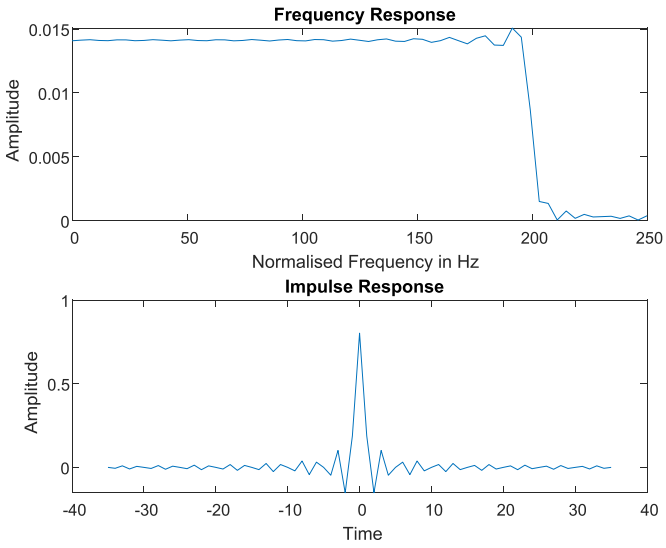
```
n=input('Enter the order of the filter:');
wc=input('Enter the cutoff frequency:');
ws=input('Enter the sampling frequency:');
type=input('Enter the type of filter:');
wc=wc/ws;
win=input('Enter the type_window:');
if(strcmp(win,'rect'))
 b=fir1(n-1,wc,type,rectwin(n));
elseif(strcmp(win,'hamm'))
  b=fir1(n-1,wc,type,hamming(n));
else
    b=fir1(n-1,wc,type,hann(n));
end
t=-(n-1)/2:1:(n-1)/2;
l=length(b);
nf=128;
freq=abs(fft(b,nf))/l;
findex=(ws/2)*linspace(0,1,(nf/2)+1);
subplot(2,1,1);
plot(findex,freq(1:(nf/2)+1));
```

```
title('Frequency Response');
xlabel('Normalised Frequency in Hz');
ylabel('Amplitude');
subplot(2,1,2);
plot(t,b);
title('Impulse Response');
xlabel('Time');
ylabel('Amplitude');
```

RECTANGULAR WINDOW (LPF/HPF/BPF/BSF)

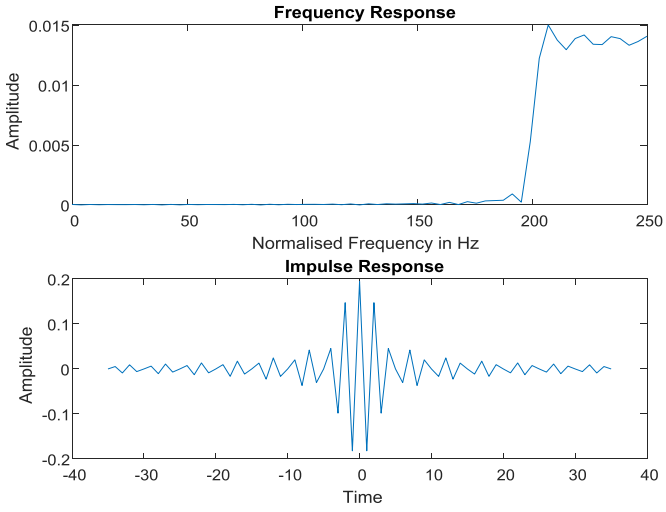
OUTPUT :

LPF:



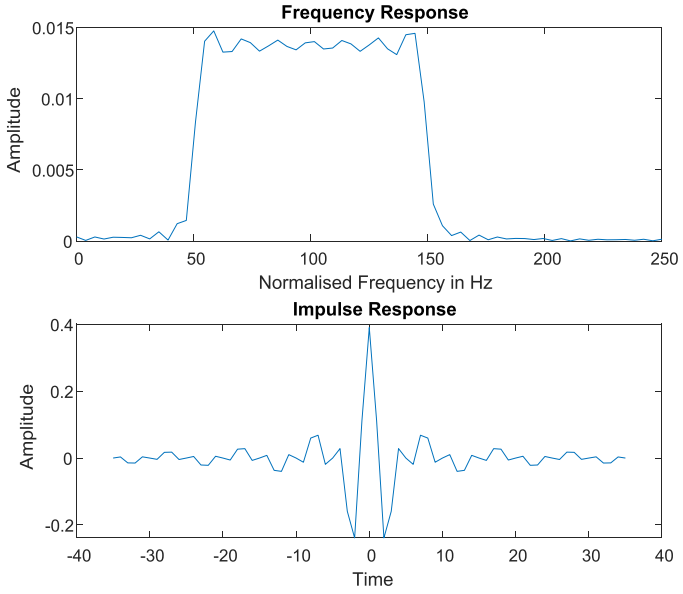

```
>> fir
Enter the order of the filter:71
Enter the cutoff frequency:400
Enter the sampling frequency:500
Enter the type of filter:'low'
Enter the type_window:'rect'
```

HPF:



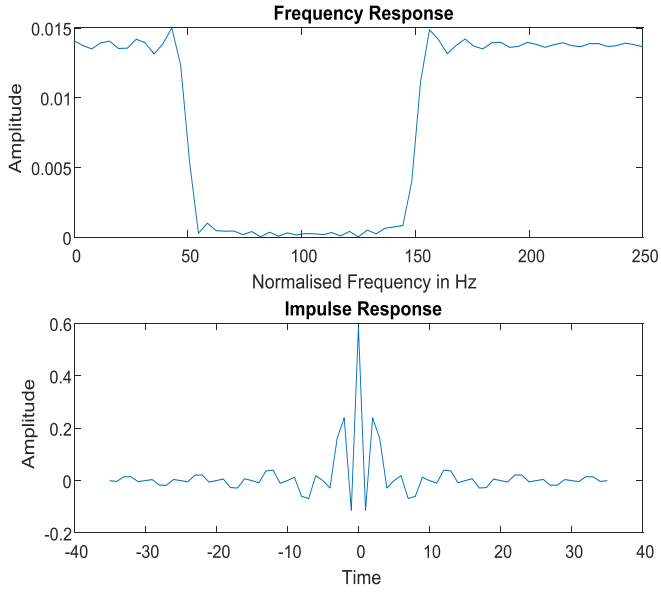

```
>> fir
Enter the order of the filter:71
Enter the cutoff frequency:400
Enter the sampling frequency:500
Enter the type of filter:'high'
Enter the type_window:'rect'
```

BPF:




```
>> fir
Enter the order of the filter:71
Enter the cutoff frequency:[100 300]
Enter the sampling frequency:500
Enter the type of filter:'bandpass'
Enter the type_window:'rect'
```

BSF:

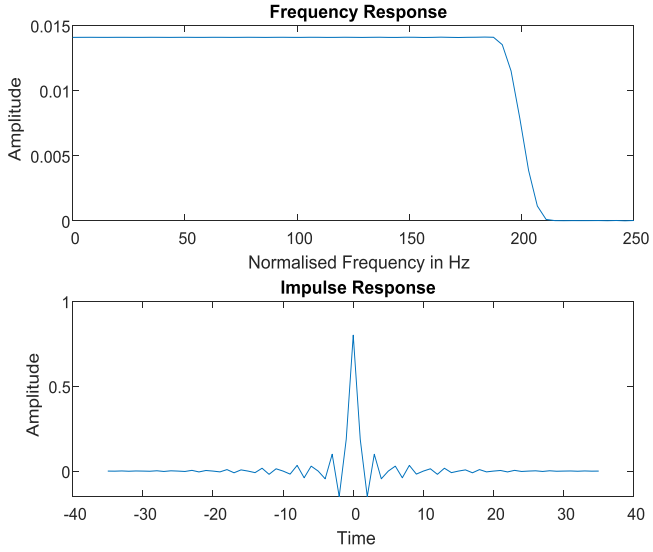



```
>> fir
Enter the order of the filter:71
Enter the cutoff frequency:[100 300]
Enter the sampling frequency:500
Enter the type of filter:'stop'
Enter the type_window:'rect'
```

HAMMING WINDOW (LPF/HPF/BPF/BSF)

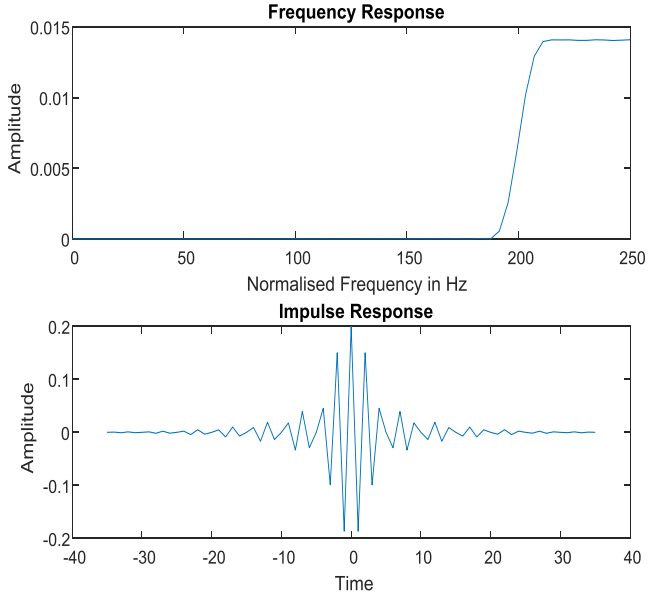
OUTPUT:

LPF:



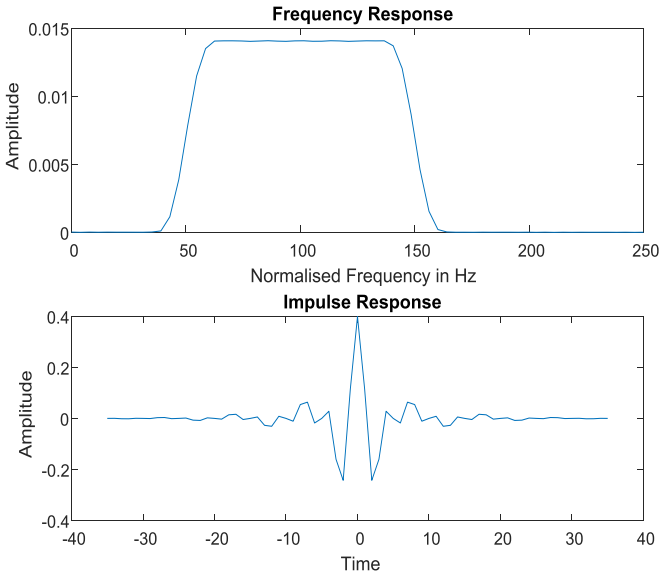

```
>> fir
Enter the order of the filter:71
Enter the cutoff frequency:400
Enter the sampling frequency:500
Enter the type of filter:'low'
Enter the type_window:'hamm'
```

HPF:



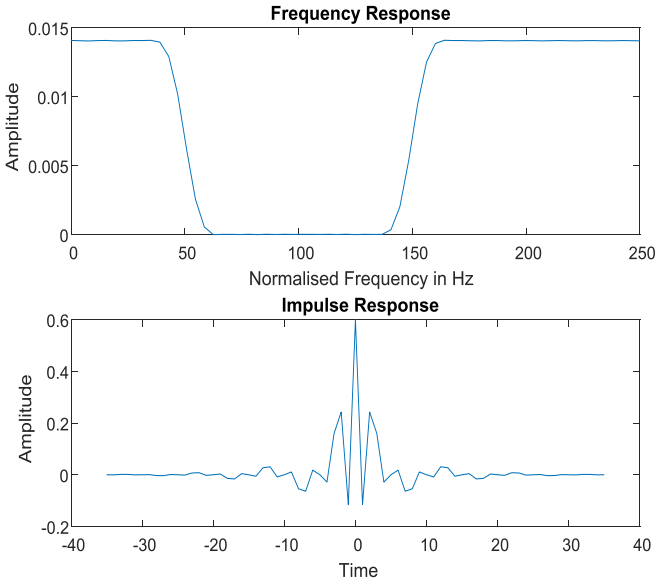
```
>> fir
Enter the order of the filter:71
Enter the cutoff frequency:400
Enter the sampling frequency:500
Enter the type of filter:'high'
Enter the type_window:'hamm'
```

BPF:




```
>> fir
Enter the order of the filter:71
Enter the cutoff frequency:[100 300]
Enter the sampling frequency:500
Enter the type of filter:'bandpass'
Enter the type_window:'hamm'
```

BSF:

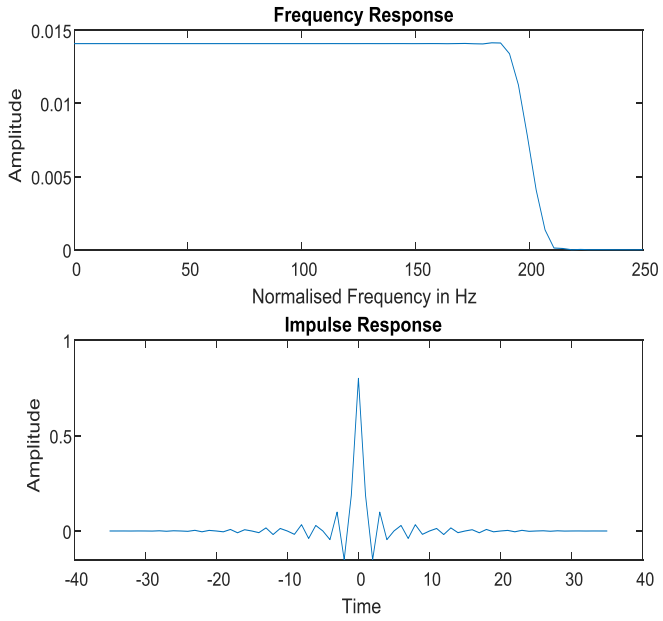



```
>> fir
Enter the order of the filter:71
Enter the cutoff frequency:[100 300]
Enter the sampling frequency:500
Enter the type of filter:'stop'
Enter the type_window:'hamm'
```

HANNING WINDOW (LPF/HPF/BPF/BSF)

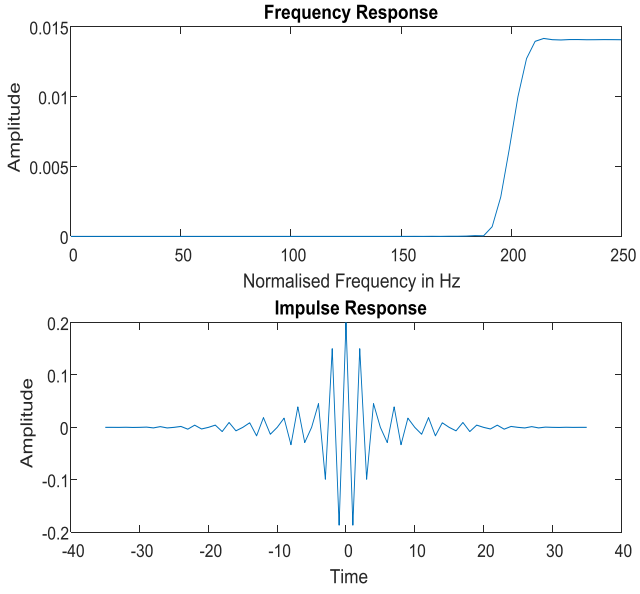
OUTPUT:

LPF:



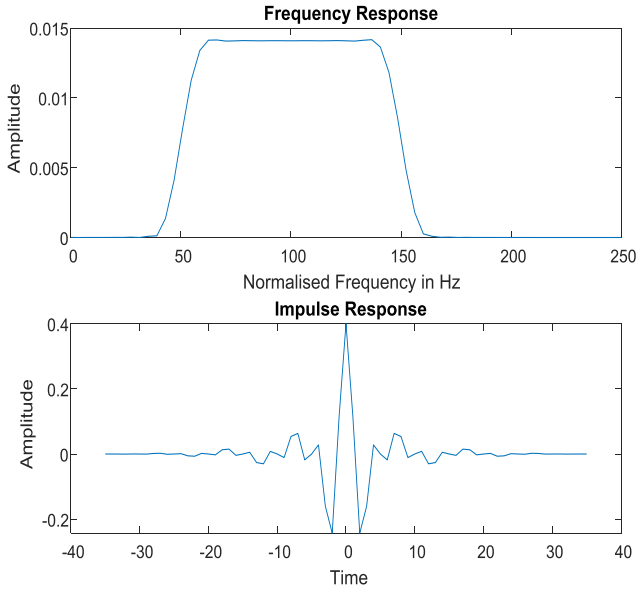

```
>> fir
Enter the order of the filter:71
Enter the cutoff frequency:400
Enter the sampling frequency:500
Enter the type of filter:'low'
Enter the type_window:'hann'
```

HPF:



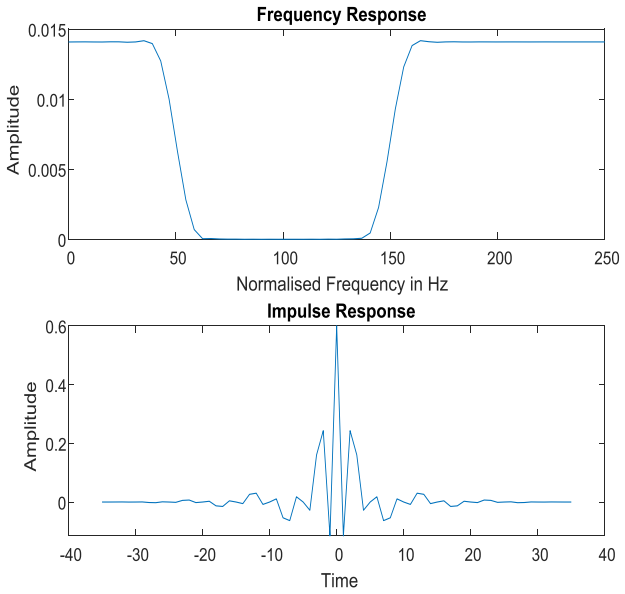

```
>> fir
Enter the order of the filter:71
Enter the cutoff frequency:400
Enter the sampling frequency:500
Enter the type of filter:'high'
Enter the type_window:'hann'
```

BPF:




```
>> fir
Enter the order of the filter:71
Enter the cutoff frequency:[100 300]
Enter the sampling frequency:500
Enter the type of filter:'bandpass'
Enter the type_window:'hann'
```

BSF:




```
>> fir
Enter the order of the filter:71
Enter the cutoff frequency:[100 300]
Enter the sampling frequency:500
Enter the type of filter:'stop'
Enter the type_window:'hann'
```

VIVA / POST-LAB QUESTIONS:

1. What is meant by impulse response?

In **signal** processing, the **impulse response**, or **impulse response** function (IRF), of a dynamic system is its output when presented with a brief input **signal**, called an **impulse**. More generally, an **impulse response** is the reaction of any dynamic system in **response** to some external change.

2. Differentiate between IIR filters and FIR filters.

1. IIR is infinite and used for applications where linear characteristics are not of concern.
2. FIR filters are Finite IR filters which are required for linear-phase characteristics.
3. IIR is better for lower-order tapping, whereas the FIR filter is used for higher-order tapping.
4. FIR filters are preferred over IIR because they are more stable, and feedback is not involved.
5. IIR filters are recursive and used as an alternate, whereas FIR filters have become too long and cause problems in various applications.

3. What are the properties of FIR filter?

a) FIR Filter is always stable.

b) A Realizable filter can always be obtained.

4. What is the necessary and sufficient condition for linear phase characteristics in FIR filter?

The condition for constant phase delay is Phase delay, $\alpha = (N-1)/2$ (i.e., phase delay is constant)

Impulse response, $h(n) = h(N-1-n)$ (i.e., impulse response is symmetric)

5. What is the reason that FIR filter is always stable?

FIR filter is always stable because all its poles are at the origin.

6. Compare the rectangular window and hanning window.

- Hamming window is very similar to the Hanning window but has a higher side lobe and a lower fall off rate and is best used when the dynamic range is about 50 dB.

- Hanning window is most commonly used for unknown signal, provide good compromises between amplitude and frequency accuracy.

7. How phase distortion and delay distortion are introduced.

The phase distortion is introduced when the phase characteristics of a filter is nonlinear within the desired frequency band. The delay distortion is introduced when the delay is not constant within the desired frequency band.

8. For what kind of application, the anti symmetric impulse response and symmetric impulse response are used ?

A digital filter is causal if its impulse response $h(n)=0$ for $n<0$. A digital filter is stable if its impulse response is absolutely summable, i.e, $\infty \sum \square h(n) \square > < \infty$ n=-∞

9. State the conditions for a digital filter to be causal and stable?

a) The width of the transition band depends on the type of window.
b) The width of the transition band can be made narrow by increasing the value of N where N is the length of the window sequence.
c) The attenuation in the stop band is fixed for a given window, except in case of Kaiser Window where it is variable.

10. List desirable characteristics of windowing technique.

(1). Fourier transform of the window function should have a small width of the main lobe.
(2).Fourier transform of the window function should have side lobes that decrease in energy rapidly as / tends to 0. Window function.

RESULT:

Thus the MATLAB program for the design of FIR filters (LPF/HPF/BSF/BPF) using Rectangular, Hamming and Hanning windowing techniques for the given cut off frequency was designed and also magnitude and log magnitude responses for the same were plotted successfully.

EXP.No: 06	DESIGN OF IIR FILTERS USING MATLAB
Date:	

Aim:

i) To write a MATLAB program to design an Analog Butterworth (low pass, high pass) Chebyshev-I (bandpass, bandstop) IIRfilter.

ii) To write a MATLAB program to design Digital Butterworth (low pass, high pass) and Digital Chebyshev (bandpass, bandstop) IIR filterusing bilinear and impulse invariant method.

Apparatus Required:

Hardware: PC with Windows 8 0r 10

Software: MATLAB™ 2021a

Algorithm:

STEP 1: Start the process

STEP 2: Get the pass band ripple and stop band ripple form the user.

STEP 3: Get the pass band frequency and stop band frequency and the sampling frequency.

STEP 4: Calculate the analog pass band edge frequencies.

STEP 5: Calculate the order and 3dB cut off frequency of the analog filter.

Calculate the order of the filter and Assign the value of *w* and then determine the system function using butter and cheby 1 function respectively.

STEP 6: Design Butterworth type filter for Low pass and high pass configurations.

STEP 7: Design Chebyshev type filter for Band pass and band stop configurations.

i) For analog design - use freqs function to find h and w.

ii) For digital design - use freqz function to find h and w

STEP 8: Digitize all the configurations using Impulse Invariance Method and Bilinear Transformation Method.

STEP 9: Find the complex frequency response of the filter configurations.

STEP 10: Calculate the magnitude of the frequency response in decibels (dB).

STEP 11: Plot the magnitude and phase response of both analog filter and digitized filter.

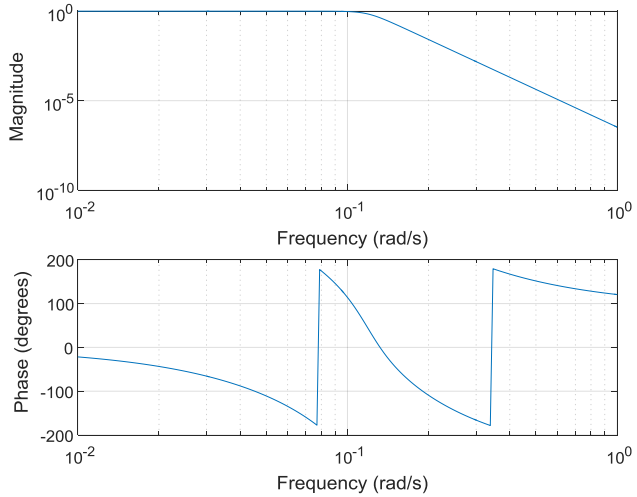
STEP 12: Stop the program.

Program for butterworth and chebyschev:

```
a=input('Enter the type of filter(1-Butterworth/2-Chebyshev):');
wp=input('Enter the passband frequency:');
ws=input('Enter the stopband frequency:');
rp=input('Enter the passband attenuation:');
rs=input('Enter the stopband attenuation:');
nq=input('Enter the nyquist rate:');
type=input('low/high/stop/bandpass:');
wp=wp/nq;
ws=ws/nq;
switch a
case 1
        [n wn] = buttord(wp,ws,rp,rs);
        disp(n)
        [b a]= butter(n,wn,type,'s');
        freqs(b,a);
case 2
        [n wp] = cheb1ord(wp,ws,rp,rs);
        disp(n)
        [b a]= cheby1(n,rp,wp,type,'s');
        freqs(b,a);
end
```

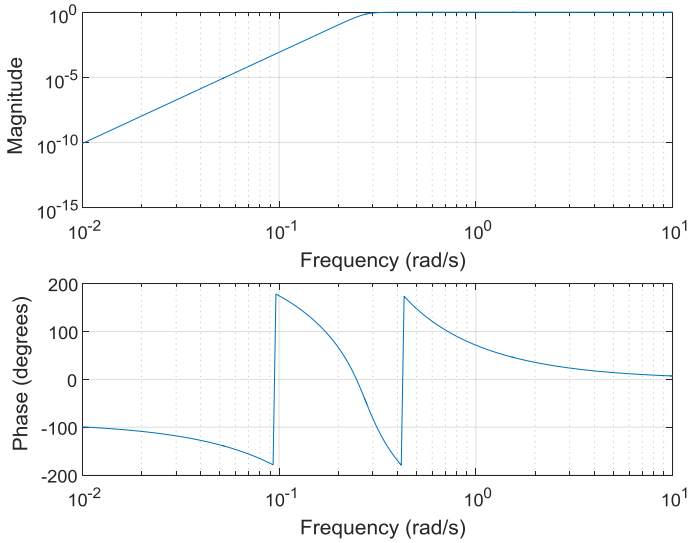
OUTPUT:

Output for the design of Butterworth Low Pass filter:



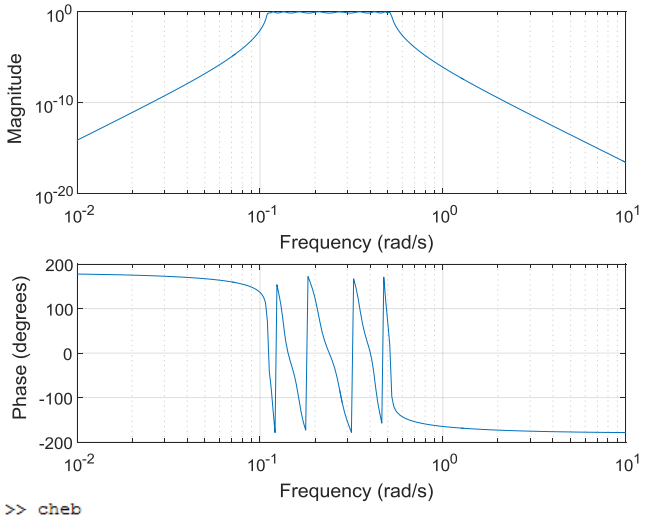

```
>> cheb
Enter the type of filter(1-Butterworth/2-Chebyshev):1
Enter the passband frequency:50
Enter the stopband frequency:160
Enter the passband attenuation:3
Enter the stopband attenuation:65
Enter the nyquist rate:500
low/high/stop/bandpass:'low'
     7
```

Output for the design of Butterworth High pass filter:



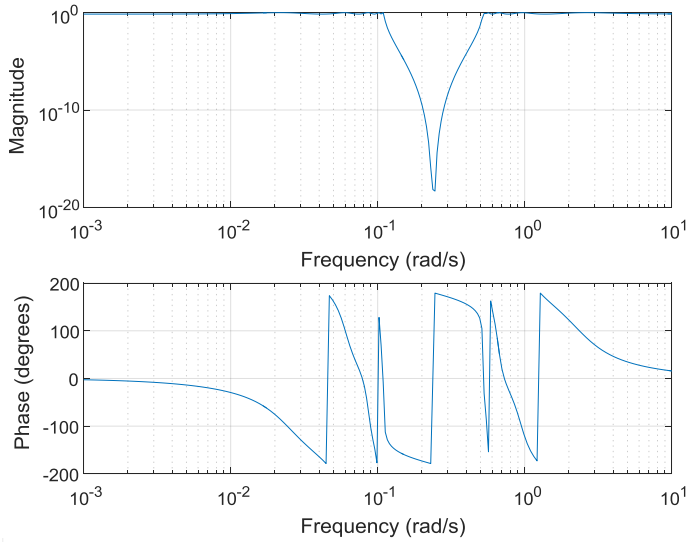

```
>> cheb
Enter the type of filter(1-Butterworth/2-Chebyshev):1
Enter the passband frequency:160
Enter the stopband frequency:50
Enter the passband attenuation:3
Enter the stopband attenuation:65
Enter the nyquist rate:500
low/high/stop/bandpass:'high'
     7
```

Output for the design of Chebyshev-I Band Pass filter:




```
>> cheb
Enter the type of filter(1-Butterworth/2-Chebyshev):2
Enter the passband frequency:[55 260]
Enter the stopband frequency:[45 310]
Enter the passband attenuation:3
Enter the stopband attenuation:60
Enter the nyquist rate:500
low/high/stop/bandpass:'bandpass'
    10
```

Output for the design of Chebyshev-I Band Stop filter:




```
>> cheb
Enter the type of filter(1-Butterworth/2-Chebyshev):2
Enter the passband frequency:[55 260]
Enter the stopband frequency:[45 310]
Enter the passband attenuation:3
Enter the stopband attenuation:60
Enter the nyquist rate:500
low/high/stop/bandpass:'stop'
    10
```

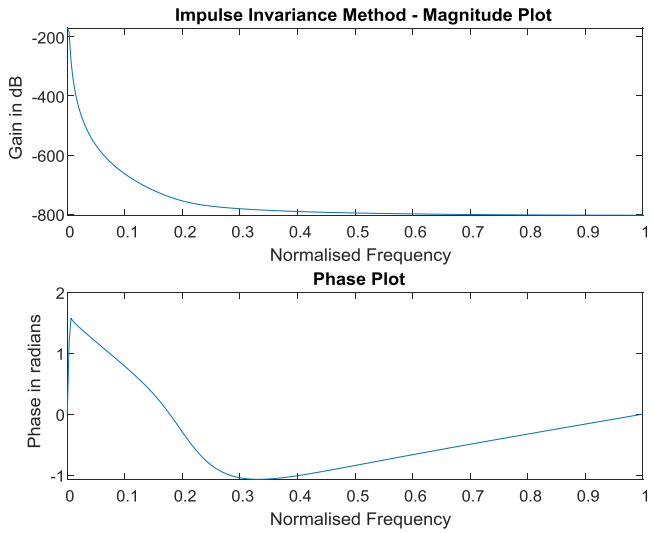
Program for Butterworth and chebyschev using bilinear and IIM:

```
wp=input('Enter the passband frequency:');
ws=input('Enter the stopband frequency:');
rp=input('Enter the passband attenuation:');
rs=input('Enter the stopband attenuation:');
nq=input('Enter the nyquist rate:');
wp=wp/nq;
ws=ws/nq;

type = input('Enter Butterworth or Chebyshev[1/2]:');
ftype = input('Enter the type of filter:[low/high/bandpass/stop]:');


switch type
case 1
      [n wn] = buttord(wp,ws,rp,rs);
      disp(n)
      [b a]= butter(n,wn,ftype,'s');
case 2
      [n wp] = cheb1ord(wp,ws,rp,rs);
      [b a]= cheby1(n,rp,wp,ftype,'s');
end

transform = input('Enter the type of transform:[IIM-1/BLT-2]:');
if transform == 1
   [bz,az] = impinvar(b,a,nq);
   w = 0:0.01:pi;
   [h,om] = freqz(bz,az,w);
```

```
    m = 20*log(abs(h));
    p = angle(h);
    subplot(2,1,1);
    plot(om/pi,m);
    title('Impulse Invariance Method - Magnitude Plot');
    xlabel('Normalised Frequency');
    ylabel('Gain in dB');
    subplot(2,1,2);
    plot(om/pi,p);
    title('Phase Plot');
    xlabel('Normalised Frequency');
    ylabel('Phase in radians');
else
    [bz,az] = bilinear(b,a,nq);
    w = 0:0.01:pi;
    [h,om] = freqz(bz,az,w);
    m = 20*log(abs(h));
    p = angle(h);
    subplot(2,1,1);
    plot(om/pi,m);
    title('Bilinear Transformation - Magnitude Plot');
    xlabel('Normalised Frequency');
    ylabel('Gain in dB');
    subplot(2,1,2);
    plot(om/pi,p);
    title('Phase Plot')
    xlabel('Normalised Frequency');
    ylabel('Phase in radians');
end
```

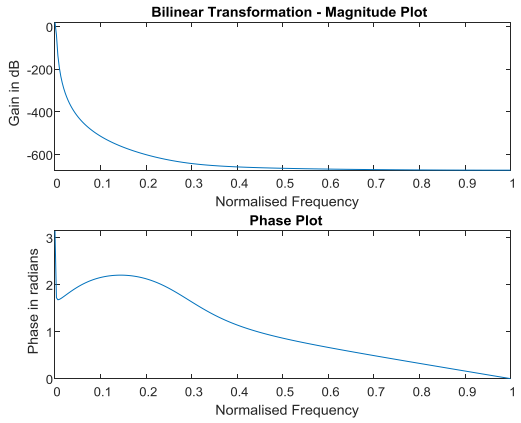
OUTPUT:

Butterworth Low pass filter using IIM:



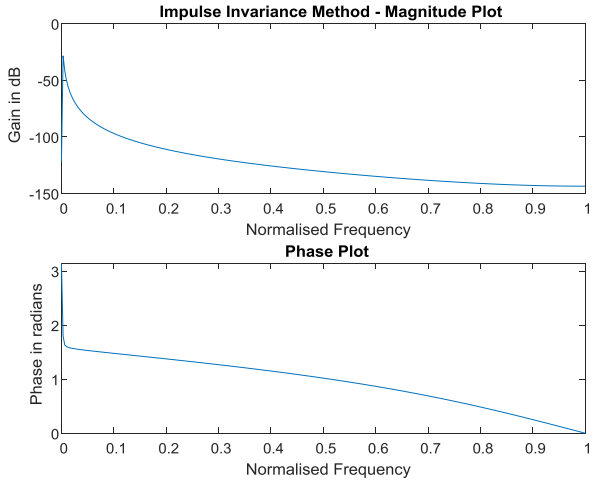
```
>> iir2
Enter the passband frequency:50
Enter the stopband frequency:160
Enter the passband attenuation:3
Enter the stopband attenuation:65
Enter the nyquist rate:500
Enter Butterworth or Chebyshev[1/2]:1
Enter the type of filter:[low/high/bandpass/stop]:'low'
     7

Enter the type of transform:[IIM-1/BLT-2]:1
```

Butterworth Lowpass filter using BLT:



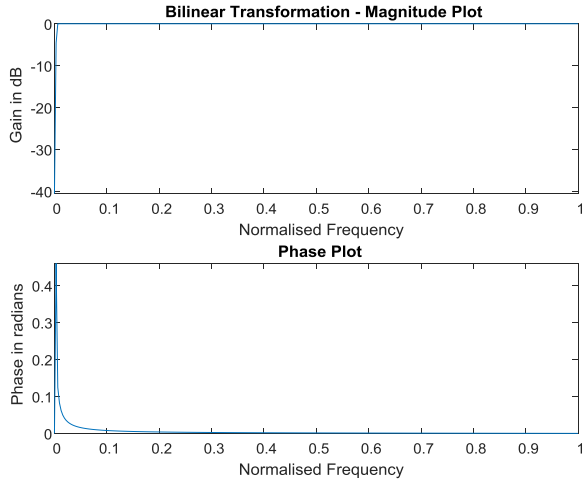

```
>> iir2
Enter the passband frequency:50
Enter the stopband frequency:160
Enter the passband attenuation:3
Enter the stopband attenuation:65
Enter the nyquist rate:500
Enter Butterworth or Chebyshev[1/2]:1
Enter the type of filter:[low/high/bandpass/stop]:'low'
     7

Enter the type of transform:[IIM-1/BLT-2]:2
```

Butterworth High-pass filter using IIM



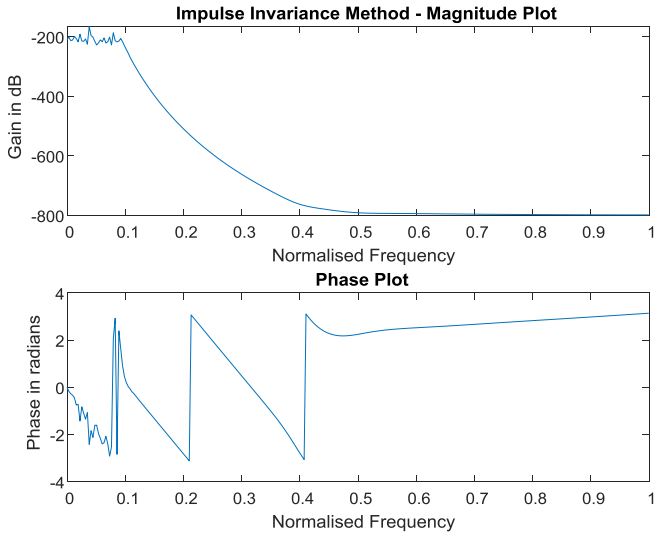

```
>> iir2
Enter the passband frequency:160
Enter the stopband frequency:50
Enter the passband attenuation:3
Enter the stopband attenuation:65
Enter the nyquist rate:500
Enter Butterworth or Chebyshev[1/2]:1
Enter the type of filter:[low/high/bandpass/stop]:'high'
     7

Enter the type of transform:[IIM-1/BLT-2]:1
```

Butterworth High-pass filter using BLT



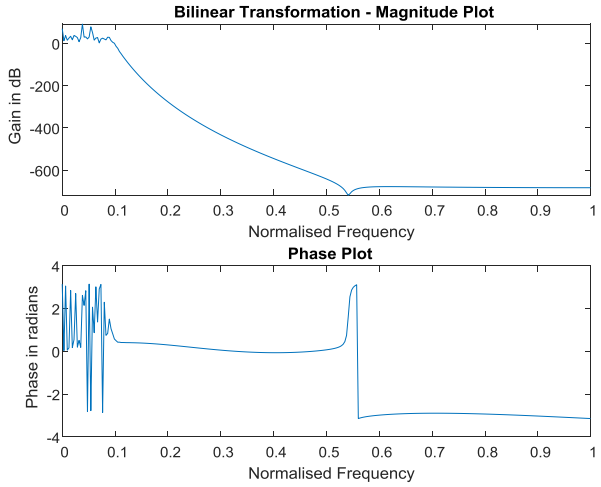

```
>> iir2
Enter the passband frequency:160
Enter the stopband frequency:50
Enter the passband attenuation:3
Enter the stopband attenuation:65
Enter the nyquist rate:500
Enter Butterworth or Chebyshev[1/2]:1
Enter the type of filter:[low/high/bandpass/stop]:'high'
     7

Enter the type of transform:[IIM-1/BLT-2]:2
```

Chebyshev Bandpass using IIM



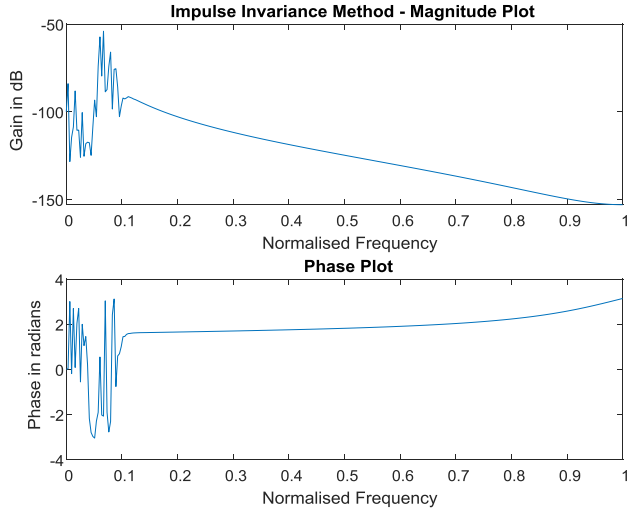

```
>> iir2
Enter the passband frequency:[55 260]
Enter the stopband frequency:[45 310]
Enter the passband attenuation:3
Enter the stopband attenuation:60
Enter the nyquist rate:500
Enter Butterworth or Chebyshev[1/2]:2
Enter the type of filter:[low/high/bandpass/stop]:'bandpass'
Enter the type of transform:[IIM-1/BLT-2]:1
```

Chebyshev Bandpass using BLT



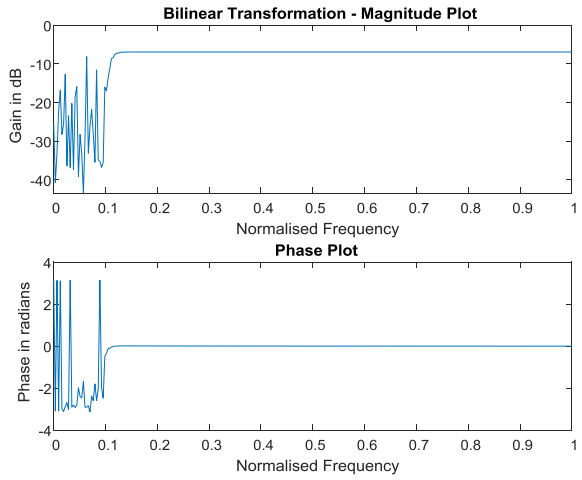

```
>> iir2
Enter the passband frequency:[55 260]
Enter the stopband frequency:[45 310]
Enter the passband attenuation:3
Enter the stopband attenuation:60
Enter the nyquist rate:500
Enter Butterworth or Chebyshev[1/2]:2
Enter the type of filter:[low/high/bandpass/stop]:'bandpass'
Enter the type of transform:[IIM-1/BLT-2]:2
```

Chebyshev Bandstop using IIM




```
>> iir2
Enter the passband frequency:[55 260]
Enter the stopband frequency:[45 310]
Enter the passband attenuation:3
Enter the stopband attenuation:60
Enter the nyquist rate:500
Enter Butterworth or Chebyshev[1/2]:2
Enter the type of filter:[low/high/bandpass/stop]:'stop'
Enter the type of transform:[IIM-1/BLT-2]:1
```

Chebyshev Band stop using BLT




```
>> iir2
Enter the passband frequency:[55 260]
Enter the stopband frequency:[45 310]
Enter the passband attenuation:3
Enter the stopband attenuation:60
Enter the nyquist rate:500
Enter Butterworth or Chebyshev[1/2]:2
Enter the type of filter:[low/high/bandpass/stop]:'stop'
Enter the type of transform:[IIM-1/BLT-2]:2
```

VIVA / POST-LAB QUESTIONS:

1. **Compare analog and digital filters.**

   The main difference between the two methods is that a digital filter circuit has to sample the analogue signal and convert it into a set of binary numbers. In contrast, analogue filters do not have to do this type of conversion and the signal remains in its pure analogue form throughout the filtering process.

2. **What is the procedure to design a digital Butterworth filter?**

   Step 1: Use the given specifications and find the order of the N and formulate H(S)
   Step 2: Find the ILT of H(S) to get h(t)
   Step 3: t=nT substitute, where T is the sampling period
   Step 4: Find the Z-transform to get h(z)

3. **What is the difference between Butterworth, Chebyshev I and Chebyshev II filters?**

   Butterworth filter:

   The Butterworth filter provides the best Taylor Series approximation to the ideal lowpass filter response at analog frequencies $\Omega = 0$ and $\Omega = \infty$; for any order **N**, the magnitude squared response has 2**N**-1 zero derivatives at these locations

(**maximally flat** at $\Omega = 0$ and $\Omega = \infty$). Response is monotonic overall, decreasing smoothly from $\Omega = 0$ to $\Omega = \infty$. $|H(j\Omega)| = \sqrt{1/2}$ at $\Omega = 1$.

Chebyshev I:

The Chebyshev Type I filter minimizes the absolute difference between the ideal and actual frequency response over the entire passband by incorporating an equal ripple of Rp dB in the passband. Stopband response is maximally flat. The transition from passband to stopband is more rapid than for the Butterworth filter. $|H(j\Omega)| = 10^{-Rp/20}$ at $\Omega = 1$.

Chebyshev II:

The Chebyshev Type II filter minimizes the absolute difference between the ideal and actual frequency response over the entire stopband by incorporating an equal ripple of Rs dB in the stopband. Passband response is maximally flat.The stopband does not approach zero as quickly as the type I filter (and does not approach zero at all for even-valued filter order **n**). The absence of ripple in the passband, however, is often an important advantage. $|H(j\Omega)| = 10^{-Rs/20}$ at $\Omega = 1$.

4. Why FIR filter mostly preferred rather than IIR filter?

- They can have exactly linear phase.
- They are always stable.
- The design methods are generally linear.
- They can be realized efficiently in hardware.

5. State the transfer function of IIR filter.

$$H(z) = \frac{B(z)}{A(z)} = \frac{\sum_{k=0}^{N} b[k] \cdot z^{-k}}{1 + \sum_{k=1}^{N} a[k] \cdot z^{-k}}$$

6. How can u design a digital filter from analog filter?

Analog filters are the reference for designing a digital filter theoretically. If you design a discrete filter its much more easier to carry out the analysis and have control better control over your design. The general procedure to design a digital filter is: discritize your analog input which obeys Nyquist rate and then pass it to a filter. And revert back to analog mode if required. Refer digital filter design text for more details.

7. Differentiate Butterworth and Chebyshev filter.

Chebyshev and Butterworth filters are designed for totally different applications.

Butterworth filters are used in applications where maximum pass band flatness is required. Chebyshev filters are optimized to give a steeper roll off.So, they will pass through the filter without no attenuation.

8. **Why impulse invariant method is not preferred in the design of IIR filters other than low pass filter?**

    - BLT includes imaginary axis too.
    - Impulse invariance method undergoes many to one mapping where as BLT undergoes one to one mapping.
    - Impulse invariance method is bandlimited warpping where as BLT is not bandlimited warpping.

9. **What is pre-warping and warping effect?**

    Effect compresses the magnitude and phase response. This effect is called warping effect.Pre-warp our desired digital frequency and substitute that into the transformed digital filter function.

Warping effect is one of the main reasons that the application of interval arithmetic to the enclosure of dynamical systems is difficult. A new method for reducing the wrapping effect is proposed, based on an interval ellipsoid arithmetic.

10. **List the various structures of an IIR filters.**

- Direct form
- Transposed direct form
- Lattice-ladder form
- Parallel realization
- Cascade realization
- Bi-quad coupled realization
- State space realization

Result

i) Thus the MATLAB program to design an Analog Butterworth (low pass, high pass) Chebyshev-I (bandpass, bandstop)IIR filterhas been completed and its magnitude response is plotted successfully.

ii) Thus the MATLAB program to design Digital Butterworth(low pass, high pass)and Digital Chebyshev (bandpass, bandstop)IIR filterusing bilinear and impulse invariant method has been completed and its magnitude response is plotted successfully.

EXP.No: 07	AUTO CORRELATION AND CROSS CORRELATION
Date:	

AIM:

To write a MATLAB program to obtain the auto and cross correlation for the input sequence(s) and plot their response in discrete time domain.

APPARATUS REQUIRED:

Hardware : PC with Windows 8 (or) 10

Software : MATLAB™ 2021a

ALGORITHM:

Step 1: Start the program.

Step 2: Get the sequence x(n).

Step 3: Get the range of the sequence x(n).

Step 4: Get the sequence h(n).

Step 5: Get the range of the sequence h(n)..

Step 6: Find the correlation between the sequences x(n) and h(n)

Step 7: Plot the correlated sequence y(n).

Step 8: Terminate the process.

Auto correlation function is a measure of similarity between a signal & its time delayed version. It is represented with R(k). The auto correlation function of x(n) is given by

$$R11(k)=R(k)=\sum_{n=-\infty}^{\infty} x_1(n)x_2(n-k)$$

PROGRAM:

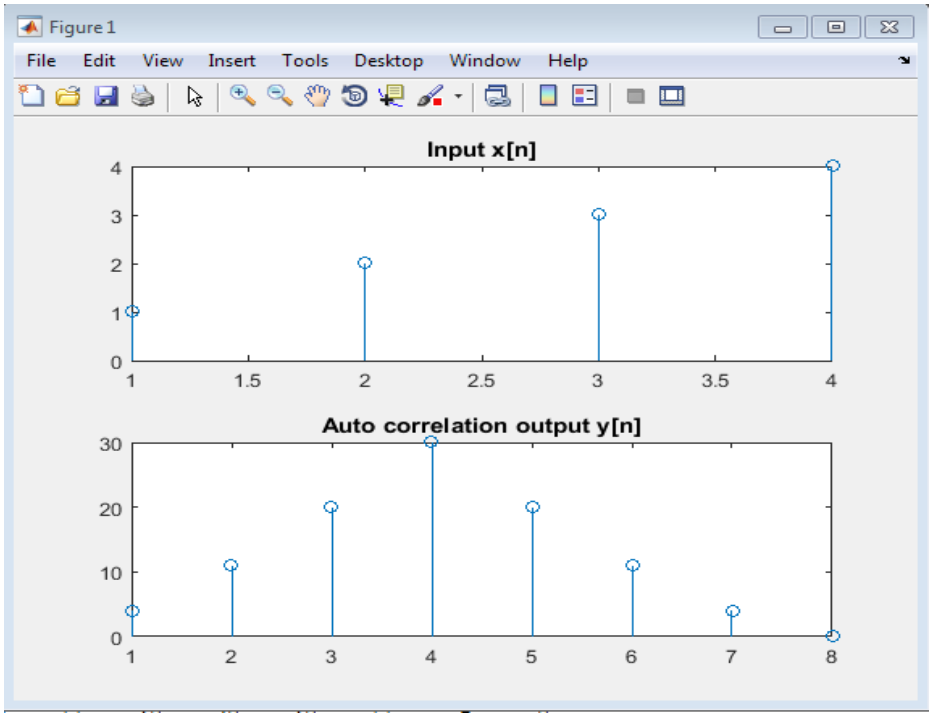
AUTO CORRELATION:

```
x=input('Enter x[n]:')
m=length(x);
rev_x=flip(x)
X=[x,zeros(1,n)];
rev_X=[rev_x,zeros(1,n)];
fori=1:n+m
Y(i)=0;
for j=1:m
if(i-j+1>0)
Y(i)=Y(i)+X(j)*rev_X(i-j+1);
end
```

```
end
end
Y
subplot(2,1,1)
stem(x);
title('Input x[n]');
subplot(2,1,2);
stem(Y);
title('Auto correlation output y[n]')
```

OUTPUT (Auto Correlation)

```
Enter x[n]:[1 2 3 4]

x =

     1     2     3     4


rev_x =

     4     3     2     1


Y =

     4    11    20    30    20    11     4

>> xcorr(x)

ans =

    4.0000   11.0000   20.0000   30.0000   20.0000   11.0000    4.0000
```

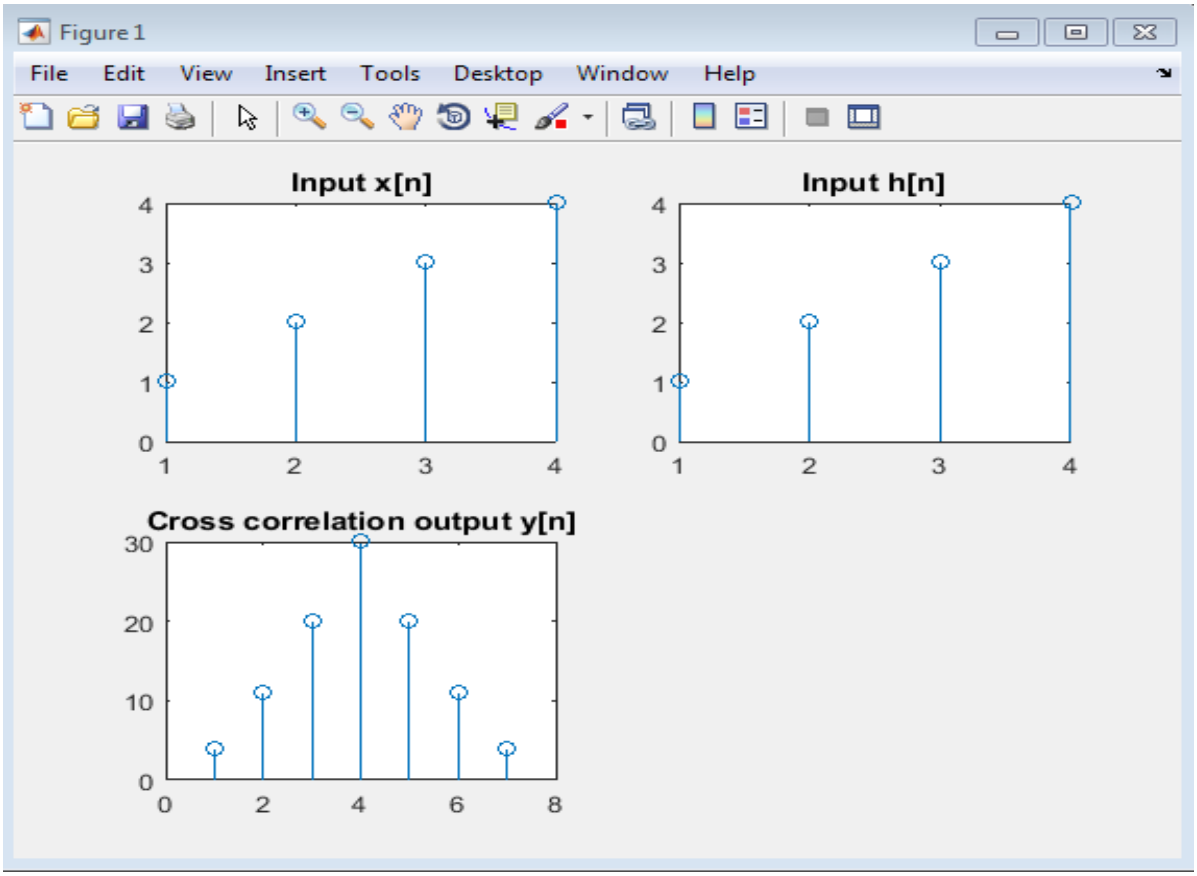


CROSS CORRELATION:

```
x=input('Enter x[n]:')
h=input('Enter h[n]:')
m=length(x);
n=length(h);
rev_h=flip(h)
X=[x,zeros(1,n)];
H=[rev_h,zeros(1,m)];
fori=1:n+m-1
Y(i)=0;
for j=1:m
if(i-j+1>0)
Y(i)=Y(i)+X(j)*H(i-j+1);
end
end
end
Y
subplot(2,2,1)
stem(x);
title('Input x[n]');
subplot(2,2,2);
stem(h);
title('Input h[n]')
subplot(2,2,3);
stem(Y);
title('Cross correlation output y[n]')
```

OUTPUT (Cross Correlation)

```
>> cross
Enter x[n]:[1 2 3 4]

x =

     1     2     3     4

Enter h[n]:[1 2 3 4]

h =

     1     2     3     4


rev_h =

     4     3     2     1


Y =

     4    11    20    30    20    11     4

>> xcorr(x,h)

ans =

    4.0000   11.0000   20.0000   30.0000   20.0000   11.0000    4.0000
```



VIVA / POST-LAB QUESTIONS:

1. **Write mathematical formula to find cross correlation and auto correlation?**

   Auto correlation; $R_{xx}[t_1, t_2]= E_{xx}[X_1,X_1']$
   Cross correlation: $R_{xy}[t_1,t_2]=E_{xy}[X_1,Y_1']$

2. **Define auto correlation and cross correlation.**

Cross correlation refers to the **correlation** between the entries of two independent random variables/vectors (X & Y) while **auto-correlation** refers to the **correlation** between the entries of a variable(X) itself.

3. **Difference between Auto correlation and convolution?**

Convolution are linear operations on the signal or signal modifiers, whereas **correlation is** a measure **of** similarity **between** two signals. As you rightly mentioned, the basic **difference between convolution** and **correlation is** that the**convolution** process rotates the matrix by 180 degrees.

4. **Difference between Auto correlation and cross correlation?**

An **autocorrelation** is the **correlation** of one variable **with a** form of itself that is "ahead or behind" in "time". **Autocorrelation**, also known as serial **correlation**, while the **cross-correlation** is the correlation between two variables. Informally, it is the similarity **between** observations as a function of the time lag **between** them.

5. **What is the length of the resultant sequence of auto correlation?**

2N-1 is the length of resultant sequence of autocorrelation

6. **List few applications of correlation.**

Correlation has many uses other than serving as a similarity measure as seen in model selections. Correlation is a **statistical** measures that indicates the extent to which two or more variables are fluctuate together

7. **Give the properties of auto correlation.**

1) Autocorrelation function is bounded, $-1 \leq\leq \rho x \rho x(l) \leq\leq 1$ and has white noise, $x(n) \sim\sim WN(\mu x \mu x, \sigma 2x \sigma x2) : \rho x(l) \rho x(l) = \delta(l)\delta(l)$

2) Helps in assigning meaning to estimated values from signals.

3) For the autocovariance function $\gamma\gamma$ of a stationary time series {Xtt} we have

$\gamma(0)\gamma(0) \geq\geq 0$

$|\gamma(h)\gamma(h)| \leq\leq \gamma(0)\gamma(0)$

$\gamma(h)\gamma(h) = \gamma(-h)\gamma(-h)$

$\gamma\gamma$ is positive semi definite

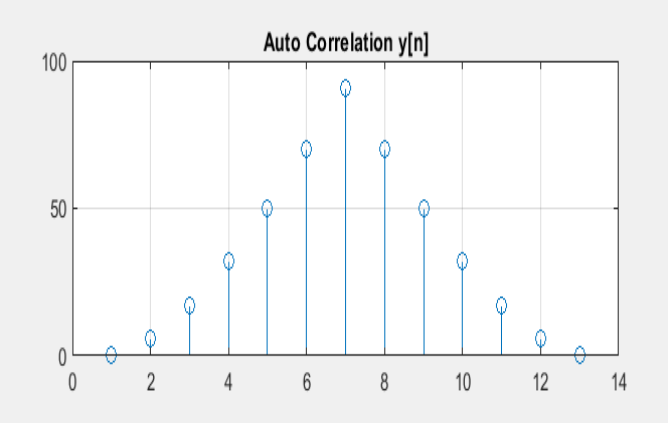
4) All auto-correlation functions are even functions

5) A time shift of a signal does not make any change of its auto-correlation.

8. **Find the auto correlation function of ramp sequence for 0≤n≤6.**

Auto Correlation :

{0 6 17 32 50 70 91 70 50 32 17 6 0}



RESULT:

Thus the MATLAB program to obtain the auto and cross correlation for the input sequence(s) was executed and their responses were plotted in discrete time domain successfully.

EX. NO : **STABILITY ANALYSIS OF LTI SYSTEMS**

DATE :

Aim

To write a MATLAB program to analyze the stability of given LTI system. Also find the magnitude, phase response, impulse and step response of the given system.

Apparatus Required

HARD WARE: IBM PC (or) Compatible PC

SOFTWARE: MATLAB 6.0 (or) Higher version

Theory

Stability analysis is carried out as a part of the design of discrete-time system. A useful stability criterion for LTI systems is that all bounded inputs produce bounded outputs. This is the so-called BIBO (Bounded Input, Bounded Output) condition. An LTI system is said to be BIBO stable if and only if it satisfies the criterion

$$\sum_{k=0}^{\infty}\left|h(k)\right| < \infty$$

where h(k) is the impulse response of the system. In other words, if the impulse response function h(k) of an LTI system is absolutely integrable, then the system is stable, i.e. the ROC of its transfer function H(z) includes the unit circle.

For a causal system,

$$H(z) = \sum_{k=0}^{\infty} h(n) z^{-n}$$

$$H(z) = \sum_{k=0}^{\infty}\left|h(n)z^{-n}\right| \le \sum_{k=0}^{\infty}\left|h(n)\right|\left|z^{-n}\right|$$

When evaluated on the unit circle (i.e., |z| = 1)

$$\left|H(z)\right| \le \sum_{k=0}^{\infty}\left|h(n)\right| < \infty$$

For a stable system the ROC of the system function includes the unit circle. But for a causal system the ROC is the exterior of a circle including infinity. A causal LTI system with a rational transfer function H(z) is stable if and only if all poles of H(z) are inside the unit circle of the z-plane, i.e., the magnitudes of all poles are smaller than 1.

The testing of stability is done by finding the pole positions using z transform. For the output to be bounded all the poles must lie inside the unit circle. When a pole lies outside the unit circle then the system is said to be unstable. A system having pole on the unit circle is regarded as marginally stable.

Algorithm

Step 1: Start the program.

Step 2: Get the difference equation of a causal LTI system

Step 3: Find the System function and also the Poles and Zeros on z plane

Step 4: Supply the poles and zero values to program

Step 5: Plot the Zeros and Poles on Z plane

Step 6: Find the Magnitude and Phase values of the LTI system

Step 7: Plot the Magnitude and Phase response characteristics

Step 8: Find the impulse and step response of the given system and plot the same

Step 9: Terminate the process.

Program

```
%Find the Coefficients of the given LTI system
clc; clear all; close all;
b=[1];
a=[1,-1,0.9];

 %Plot the Zeros and Poles on Z plane
zplane(b,a)
grid;
figure;

% Find the Magnitude and Phase response of LTI system
w=[0:1:500]*pi/500;
H=freqz(b,a,w);
m=abs(H);
p=angle(H);
subplot(2,1,1);
plot(w/pi,m);
xlabel('Frequency in pi units');
ylabel('Magnitude');
title('Magnitude Response');
grid;
subplot(2,1,2);
plot(w/pi,p);
xlabel('Frequency in pi units');
ylabel('Phase in pi units');
```

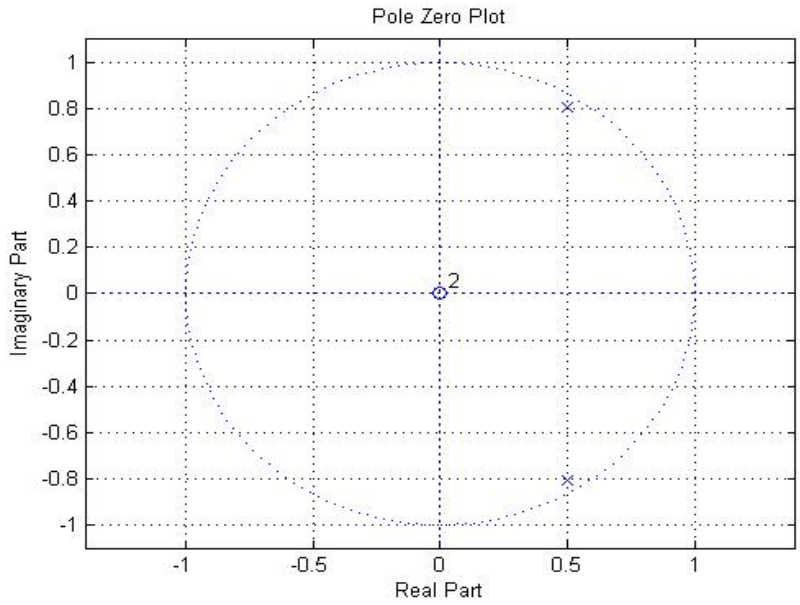
```
title('Phase Response');
grid;
figure;

 % Impulse response
x=impseq(0,-20,120);
n=-20:120;
h=filter(b,a,x);
subplot(2,1,1);stem(n,h);
title('Impulse Response');
xlabel('n');
ylabel('h(n)');

 % Step response
x=stepseq(0,-20,120);
n=-20:120;
s=filter(b,a,x);
subplot(2,1,2);stem(n,s);
title('Step Response');
xlabel('n');
ylabel('s(n)');

% Step sequence
function [x,n] = stepseq(n0,n1,n2)
n=[n1:n2];      % Actual computation
x=[(n-n0)>=0];

% Impulse sequence
function [x n]=impseq(n0,n1,n2)
n = [n1:n2];
x = [(n-n0)==0];
```

Input:

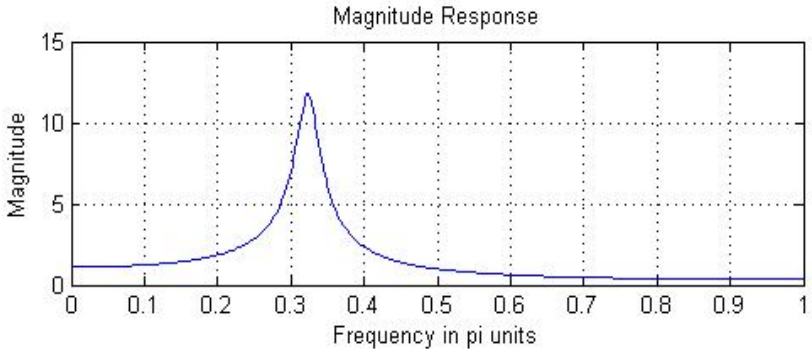
$$y(n)-y(n-1)+0.9y(n-2)=x(n)$$

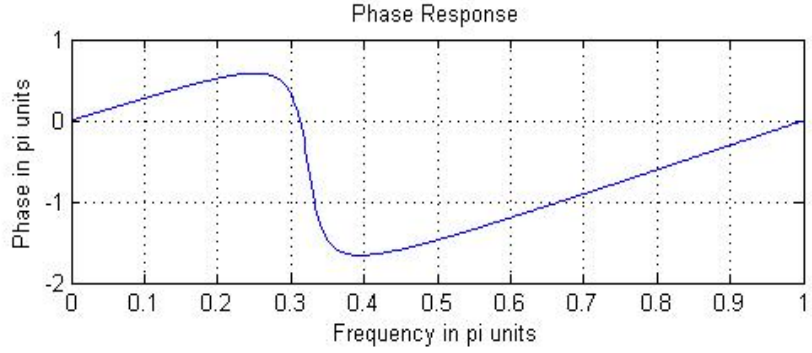
Output:

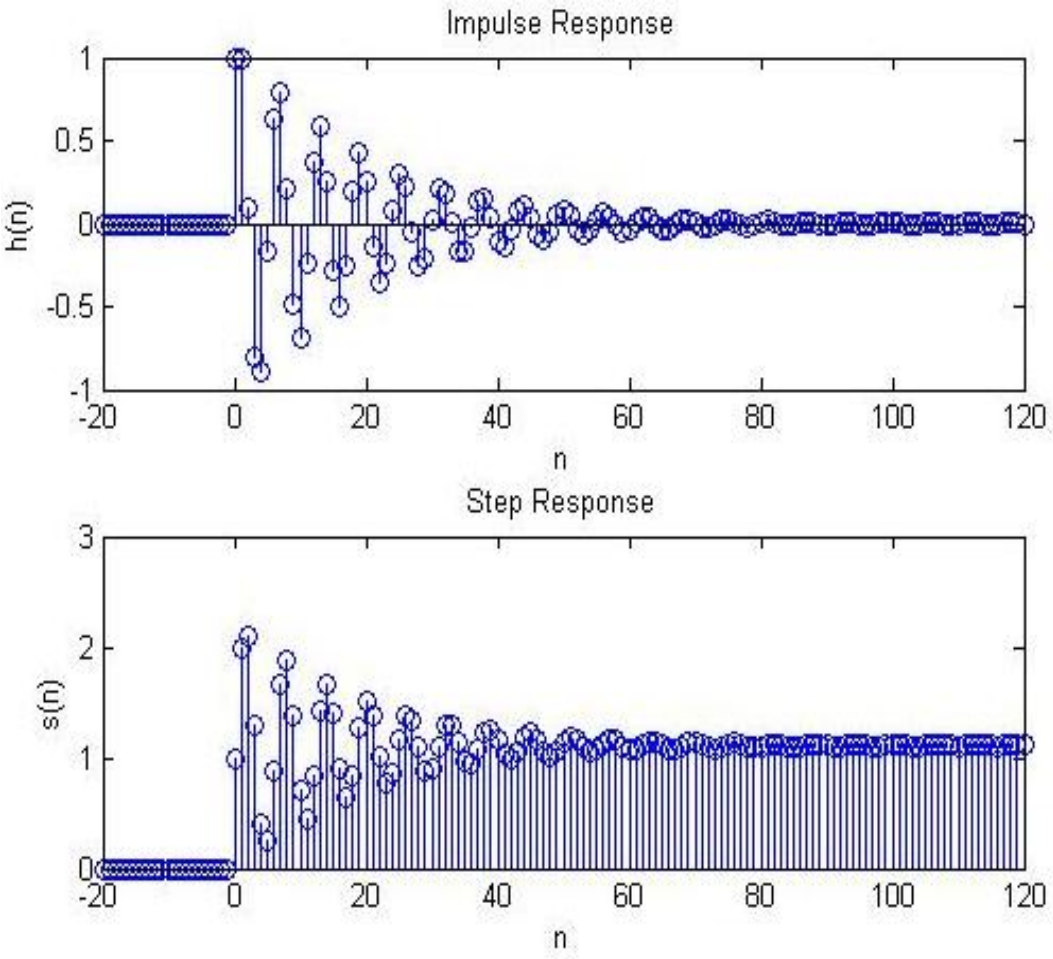
Pole-Zero Plot of the given LTI system



Magnitude & Phase Response of the LTI System






Impulse Response
h(n)
n
1
0.5
0
-0.5
-1
-20
0
20
40
60
80
100
120
Step Response
s(n)
n
3
2
1
0
-20
0
20
40
60
80
100
120

Result:

Thus the MATLAB program to obtain the stability of given LTI system and also to find the magnitude, phase response, impulse and step response of the given system.