Digital Signal Processing Lab (IT-353)

Program- B. Tech Course- CSE (5th Semester)

Submitted By: YASH ARYAN

Enrolment Number: 06816403220

University School of Information, Communication and Technology

Guru Gobind Singh Indraprastha University

Dwarka, Delhi

2022-2023

Submitted To:

Prof. C.S. Rai

USICT, GGSIPU

S. No.	Name of Experiment	Page No.	Date of Experiment	Remarks
1	Plot various functions using MATLAB			
2	Find convolution with and without using conv in MATLAB			
3	Find DTFT of given sequence in MATLAB			
4	Find DFT of given sequence in MATLAB			
5	Find DFT with and without using FFT in MATLAB			
6	Implement circular convolution with and without using cconv in MATLAB			
7	Find Z Transform in MATLAB			
8	Design Low Pass Butterworth Filter in MATLAB			

AIM: Plot various functions using MATLAB

Software Used: MATLAB R2022b

Theory:

1. Unit Step Signal

   - Unit step function is denoted by u(t). It is defined as u(t) = {1,t⩾0;0,t<0}
   - It is used as best test signal.
   - Area under unit step function is unity.

2. Unit Ramp Signal

   - Ramp signal is denoted by r(t), and it is defined as r(t) = {t,t⩾;0,t<0}
   - Area under unit ramp is unity.

3. Unit Impulse Signal

   - Impulse function is denoted by δ(t). and it is defined as δ(t) = {1,t=0;0,t≠0}

4. Exponential Signal

   - Exponential signal is in the form of x(t) = e^(αt).
   - The shape of exponential can be defined by α.

5. Sine Signal

   - Sinusoidal signal is in the form of x(t) = Acos(w0±ϕ) or Asin(w0±ϕ)

6. Cosine Signal

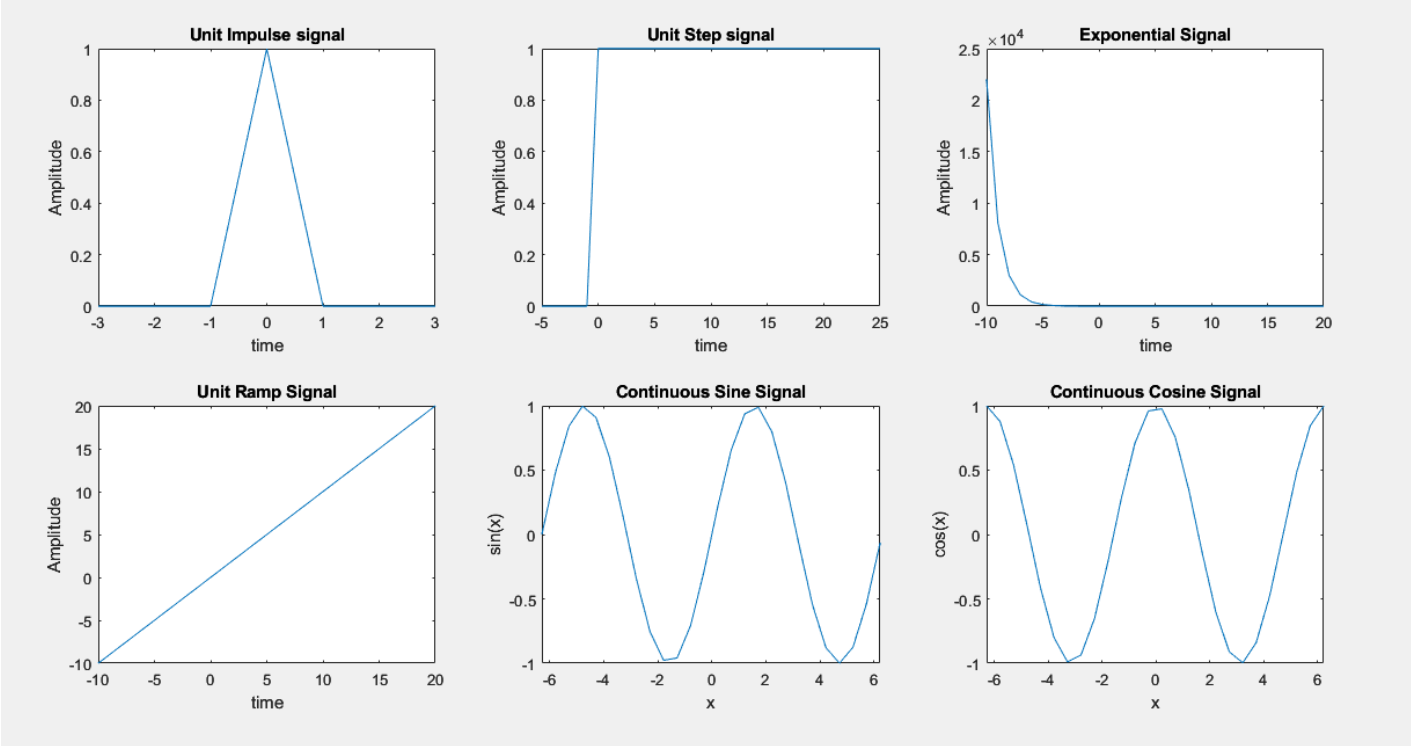
   - It represents a mathematical curve depicting a repetitive oscillation motion similar to a cosine function.
   - Sinusoidal as cos(x)=sine (x+ π/2)

PROGRAM:

```
clc;
clear all;
close all;
t1=-3:1:3;
x1=[0,0,0,1,0,0,0];
subplot(2,3,1);
```

```
plot(t1,x1)
xlabel('time');
ylabel('Amplitude');
title('Unit Impulse signal');

%Unit Step Signal
t2=-5:1:25;
x2=[zeros(1,5),ones(1,26)];
subplot(2,3,2);
plot(t2,x2)
xlabel('time');
ylabel('Amplitude');
title('Unit Step signal');


%Exponential Signal
%Exponential Signal
a=input('Enter the value of a');
t3=-10:1:20;
x3=exp(-1*a*t3);
subplot(2,3,3);
plot(t3,x3)
xlabel('time');
ylabel('Amplitude');
title('Exponential Signal');

%Unit Ramp Signal
t4=-10:1:20;
x4=t4;
subplot(2,3,4);
plot(t4,x4)
xlabel('time');
ylabel('Amplitude');
title('Unit Ramp Signal');

%Sine Signal
x=(-2*pi:0.5:2*pi);
y=sin(x);
subplot(2,3,5)
plot(x,y)
xlabel('x');
ylabel('sin(x)');
title('Continuous Sine Signal');

%Cosine Signal
x=(-2*pi:0.5:2*pi);
z=cos(x);
subplot(2,3,6);
plot(x,z);
xlabel('x');
ylabel('cos(x)');
title('Continuous Cosine Signal');
```

OUTPUTS:

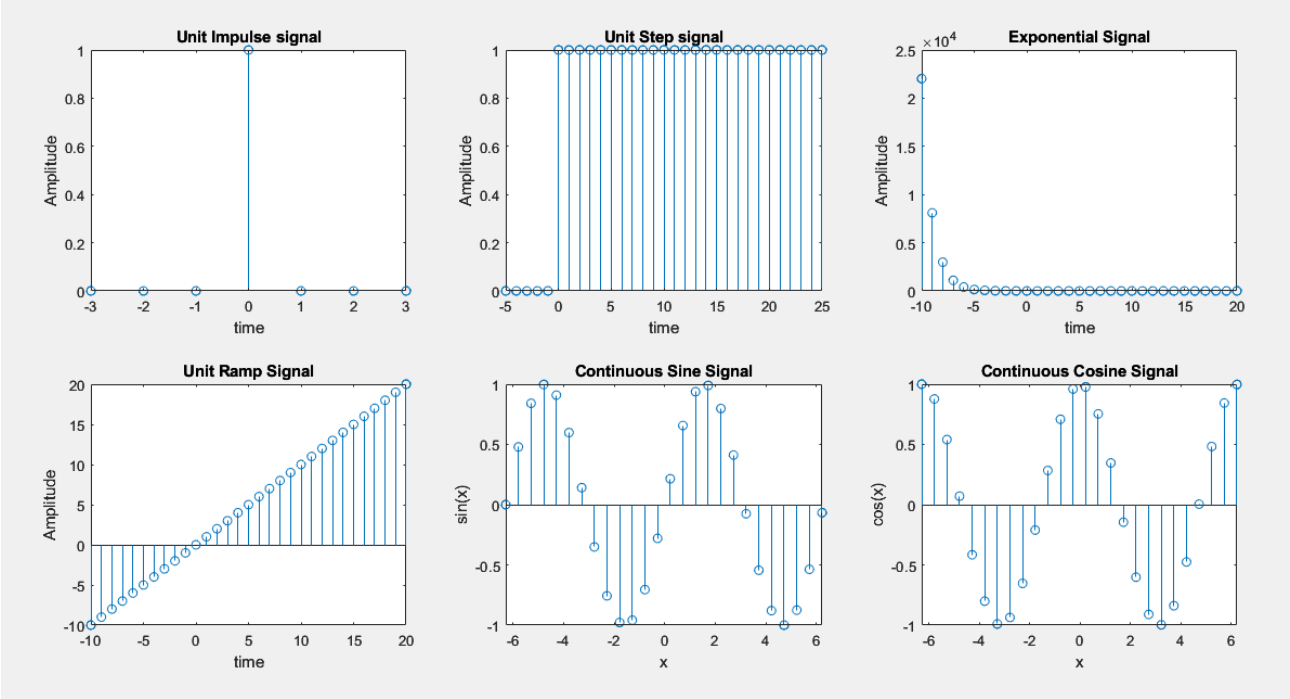

Unit Impulse signal
Amplitude
time
Unit Step signal
Amplitude
time
Exponential Signal
×10^4
Amplitude
time
Unit Ramp Signal
Amplitude
time
Continuous Sine Signal
sin(x)
x
Continuous Cosine Signal
cos(x)
x

PROGRAM :

```
%waveform generation
%DT Signal
%Unit Impulse

clc;
clear all;
close all;
t1=-3:1:3;
x1=[0,0,0,1,0,0,0];
subplot(2,3,1);
stem(t1,x1)
xlabel('time');
ylabel('Amplitude');
title('Unit Impulse signal');


 %Unit Step Signal
t2=-5:1:25;
x2=[zeros(1,5),ones(1,26)];
subplot(2,3,2);
stem(t2,x2)
xlabel('time');
ylabel('Amplitude');
title('Unit Step signal');


 %Exponential Signal
a=input('Enter the value of a');
t3=-10:1:20;
x3=exp(-1*a*t3);
subplot(2,3,3);
stem(t3,x3)
xlabel('time');
ylabel('Amplitude');
title('Exponential Signal');

 %Unit Ramp Signal
t4=-10:1:20;
x4=t4;
subplot(2,3,4);
stem(t4,x4)
xlabel('time');
ylabel('Amplitude');
title('Unit Ramp Signal');


 %Sine Signal
x=(-2*pi:0.5:2*pi);

y=sin(x);
subplot(2,3,5)
stem(x,y)
xlabel('x');
ylabel('sin(x)');
```

```
title('Continuous Sine Signal');


 %Cosine Signal
x=(-2*pi:0.5:2*pi);
z=cos(x);
subplot(2,3,6);
stem(x,z);
xlabel('x');
ylabel('cos(x)');
title('Continuous Cosine Signal');
```

OUTPUT :



AIM: Find convolution by using conv and without using conv in MATLAB

Software Used: MATLAB R2022b

Theory:

Convolution is a formal mathematical operation, just as multiplication, addition, and integration. Addition takes two numbers and produces a third number, while convolution takes two signals and produces a third signal.

Convolution is used in the mathematics of many fields, such as probability and statistics. In linear systems, convolution is used to describe the relationship between threesignals of interest: the input signal, the impulse response, and the output signal.

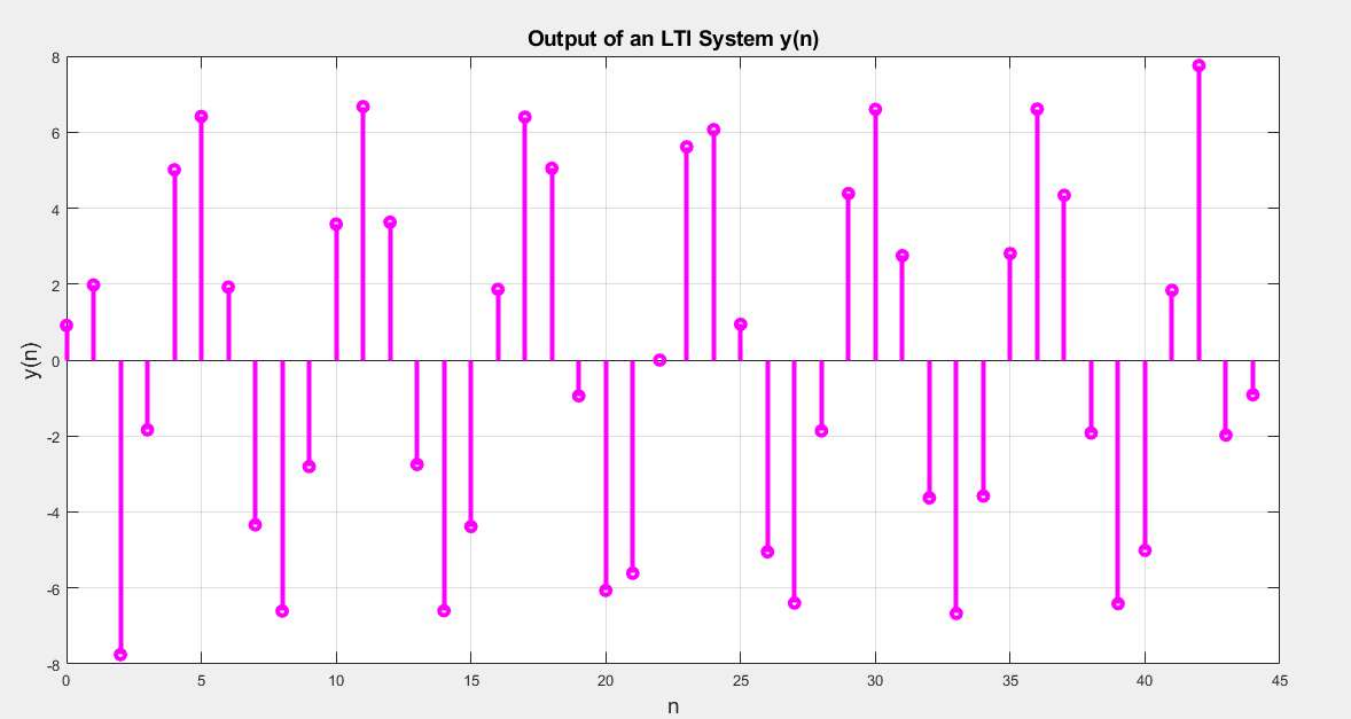
If the input and impulse response of a system are x[n] and h[n] respectively, the convolution is givenby the expression,

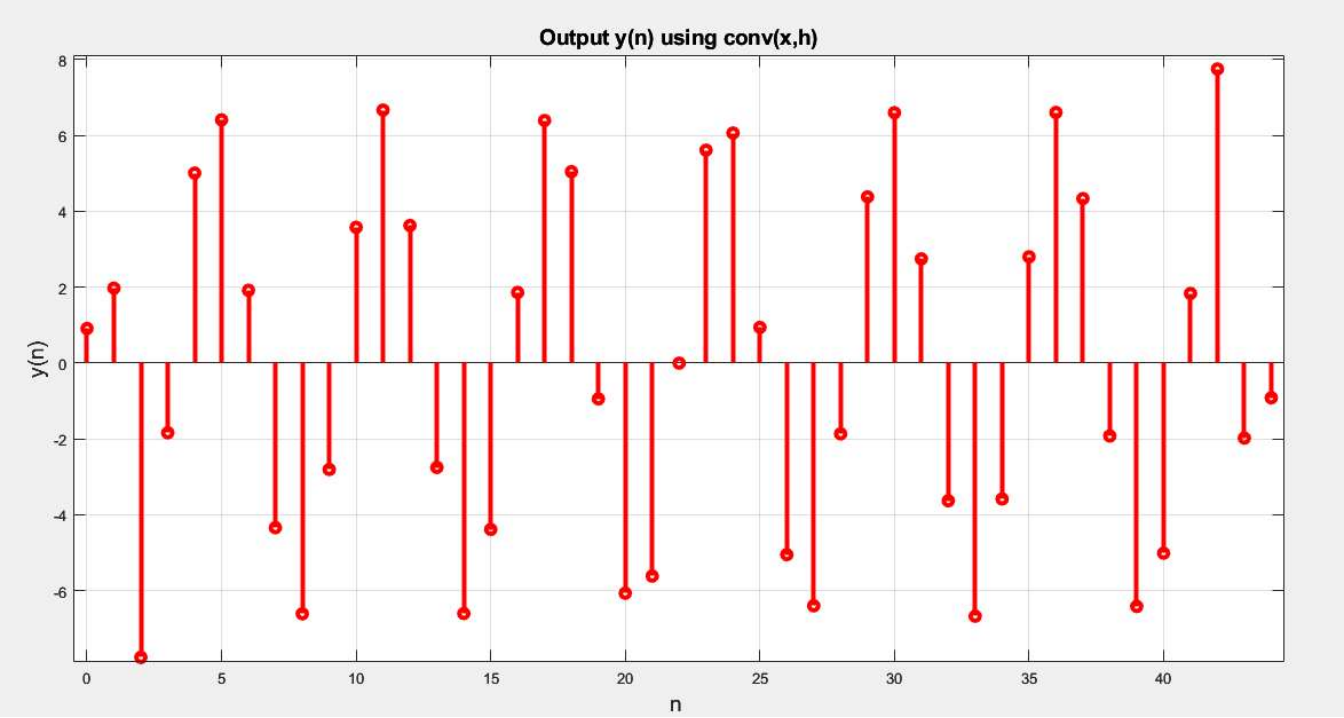
$x[n] * h[n] = \varepsilon\, x[k]\, h[n-k]$

PROGRAM:

```
clear variables
close all
n = -20:20;
x = sin(n);
h = [-1,-2,8,-2,-1];
N = length(x);
M = length(h);
Ny = N + M -1;
y = zeros(1,Ny);
for i = 1:N
      for k = 1:M
       y(i+k-1) = y(i+k-1) + h(k)*x(i);
      end
end
m = 0: Ny-1;
% Make plot
figure
stem(m,y,'linewidth',3,'color','m')
grid;
a = title('Output of an LTI System y(n)');
set(a,'fontsize',14);
a = ylabel('y(n)');
set(a,'Fontsize',14);
a = xlabel('n');
set(a,'Fontsize',14);
% Using matlab built in function (you get the same results)
figure
y2 = conv(x,h);
stem(m,y2,'linewidth',3,'color','r')
grid;
a = title('Output y(n) using conv(x,h)');
set(a,'fontsize',14);
a = ylabel('y(n)');
set(a,'Fontsize',14);
a = xlabel('n ');
set(a,'Fontsize',14);
```

OUTPUTS:


Output of an LTI System y(n)
y(n)
n
8
6
4
2
0
-2
-4
-6
-8
0
5
10
15
20
25
30
35
40
45


Output y(n) using conv(x,h)
y(n)
n
8
6
4
2
0
-2
-4
-6
0
5
10
15
20
25
30
35
40

AIM: Find DTFT of given sequence

Software Used: MATLAB R2022b

Theory:

A discrete-time signal can be represented in the frequency domain using discrete-time Fourier transform. Therefore, the Fourier transform of a discretetime sequence is called the *discrete-time Fourier transform (DTFT).*

Mathematically, if x(n)x(n) is a discrete-time sequence, then its discrete-time Fourier transform is defined as –

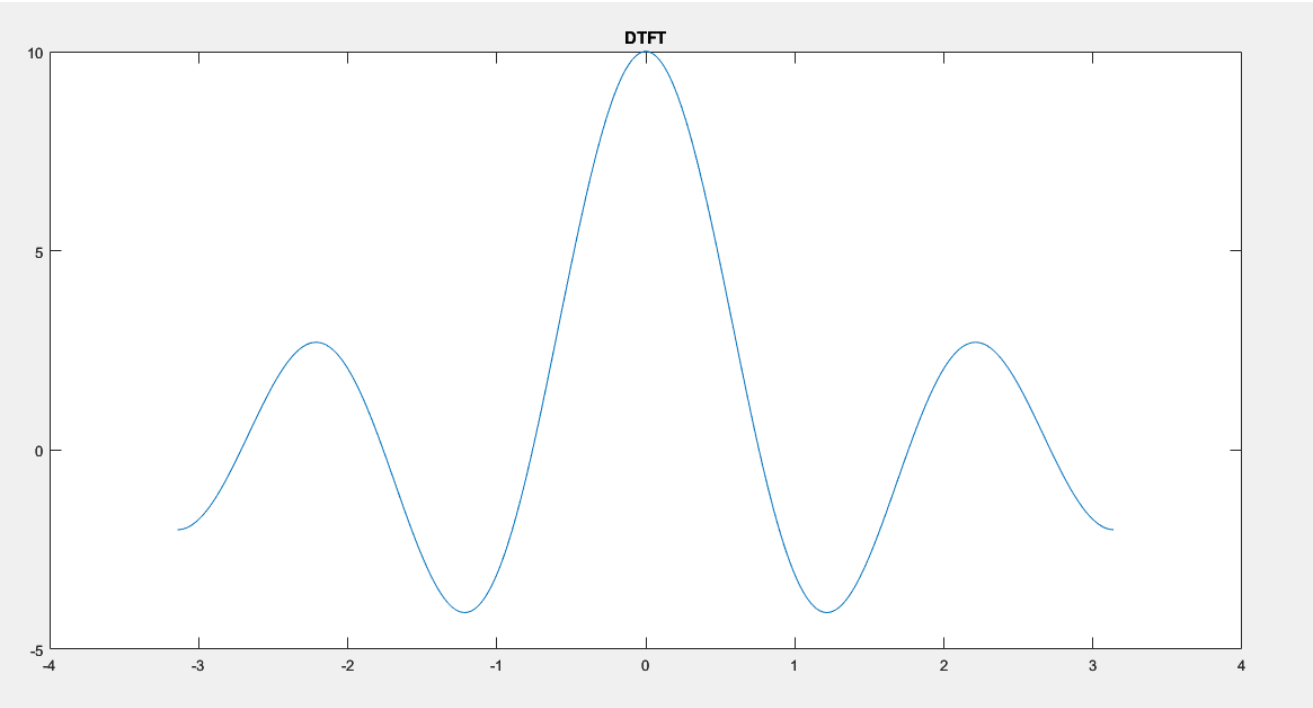
$$F[x(n)] = X(\omega) = \sum_{n=-\infty}^{\infty} x(n)\, e^{-j\omega n}$$

The discrete-time Fourier transform X(ω) of a discrete-time sequence x(n)x(n) represents the frequency content of the sequence x(n)x(n). Therefore, by taking the Fourier transform of the discrete-time sequence, the sequence is decomposed into its frequency components. For this reason, the DTFT X(ω) is also called the **signal spectrum.**

PROGRAM:

```
w=-pi:0.01:pi;
n=0:50;
x=[1 2 3 4];
for i=1:length(w);
   X(i)=0;
   for k=1:length(x);
       X(i)=X(i)+x(k).*exp(-j.*w(i).*n(k));
    end
 end
plot(w,X);
title('DTFT');
```

OUTPUTS:


DTFT
10
5
0
-5
-4
-3
-2
-1
0
1
2
3
4

AIM: Find DFT of given sequence

Software Used: MATLAB R2022b

Theory:

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

Where TWIDDLE FACTOR , $W_N^{nk} = e^{-j2n\pi k/N}$

Here k = 0,1,2,3…… N-1

Basic IDFT equation : $x(n) = (1/N) \sum_{k=0}^{N-1} X(k) W_N^{-nk}$

Where TWIDDLE FACTOR, $W_N^{-nk} = e^{j2n\pi k/N}$

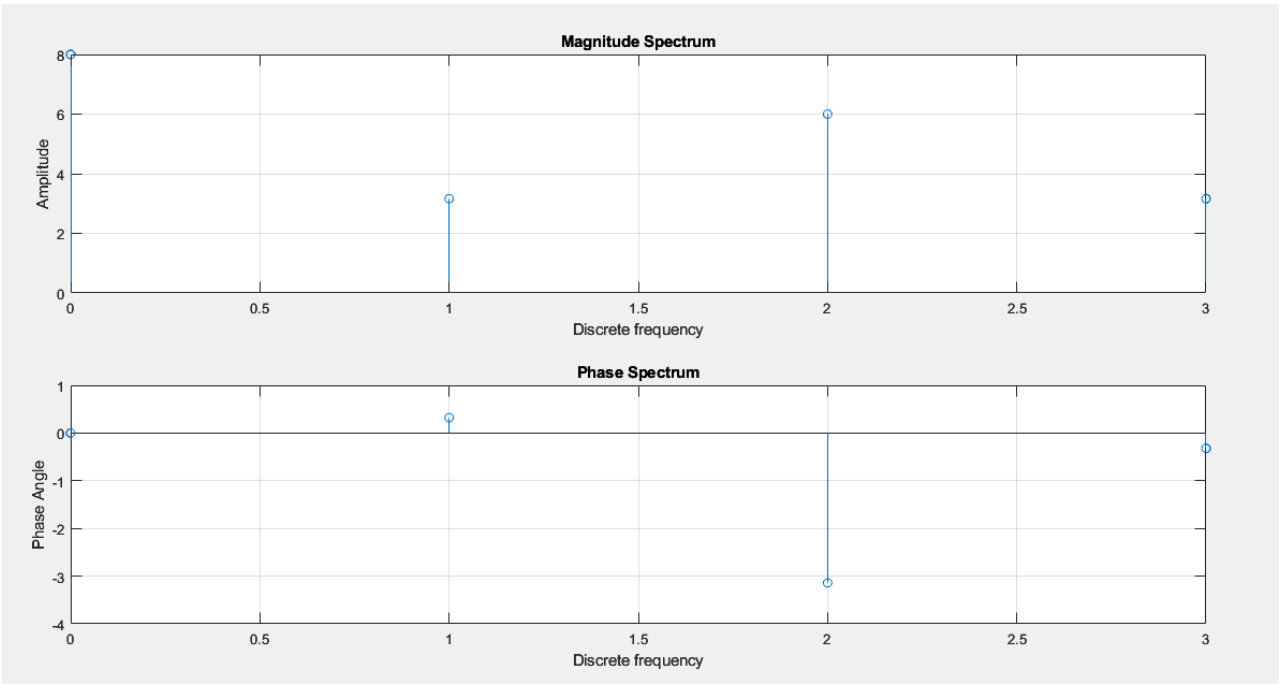
Here n = 0,1,2,3…… N-1

PROGRAM:

```
clc;
close all;
clear all;
x=input('Enter the sequence x= ');
N=input('Enter the length of the DFT N= ');
len=length(x);
if N>len
    x=[x zeros(1,N-len)];
elseif N<len
    x=x(1:N);
end
i=sqrt(-1);
w=exp(-i*2*pi/N);
n=0:(N-1);
k=0:(N-1);
nk=n'*k;
W=w.^nk;
X=x*W;
disp(X);
subplot(211);
stem(k,abs(X));
title('Magnitude Spectrum');
xlabel('Discrete frequency');
ylabel('Amplitude');
grid on;
subplot(212);
stem(k,angle(X));
title('Phase Spectrum');
xlabel('Discrete frequency');
ylabel('Phase Angle');
grid on;
```

OUTPUTS:

```
Command Window
  Enter the sequence x= [2 3 -1 4]
  Enter the length of the DFT N= 4
     8.0000 + 0.0000i   3.0000 + 1.0000i  -6.0000 - 0.0000i   3.0000 - 1.0000i

fx >>
```



AIM: Find DFT with and without FFT

Software Used: MATLAB R2022b

Theory:

A FFT rapidly computes such transformations by factorizing the DFT matrix into a product of sparse (mostly zero) factors.[2] As a result, it manages to reduce the complexity of computing the DFT from $O(N^2)$, which arises if one simply applies the definition of DFT, to O(N logN) where N is the data size. The difference in speed can be enormous, especially for long data sets where *N* may be in the thousands or millions. In the presence of round-off error, many FFT algorithms are much more accurate than evaluating the DFT definition directly or indirectly. There are many different FFT algorithms based on a wide range of published theories, from simple complex-number arithmetic to group theory and number theory.

$$\sum_{n=0}^{N-1} a_n e^{-2\pi i n k/N} = \sum_{n=0}^{N/2-1} a_{2n} e^{-2\pi i (2n) k/N}$$

$$+ \sum_{n=0}^{N/2-1} a_{2n+1} e^{-2\pi i (2n+1) k/N}$$

$$= \sum_{n=0}^{N/2-1} a_n^{\text{even}} e^{-2\pi i n k/(N/2)}$$

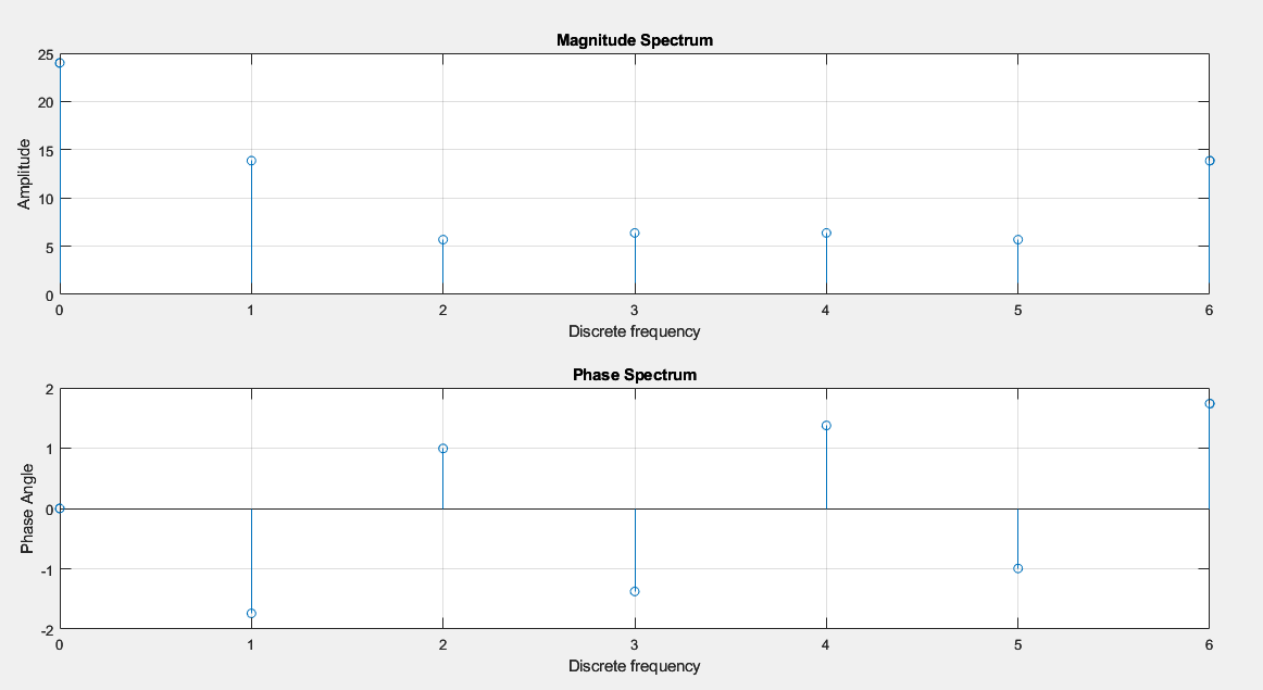
$$+ e^{-2\pi i k/N} \sum_{n=0}^{N/2-1} a_n^{\text{odd}} e^{-2\pi i n k/(N/2)},$$

PROGRAM:

```
% DFT program without function
clc;
close all;
clear all;
x=input('Enter the sequence x= ');
N=input('Enter the length of the DFT N= ');
len=length(x);
if N>len
    x=[x zeros(1,N-len)];
elseif N<len
    x=x(1:N);
end
i=sqrt(-1);
w=exp(-i*2*pi/N);
n=0:(N-1);
k=0:(N-1);
nk=n'*k;
W=w.^nk;
X=x*W;
disp(X);
subplot(211);
stem(k,abs(X));
title('Magnitude Spectrum');
xlabel('Discrete frequency');
ylabel('Amplitude');
grid on;
subplot(212);
stem(k,angle(X));
title('Phase Spectrum');
xlabel('Discrete frequency');
ylabel('Phase Angle');
grid on;
```

OUTPUTS:

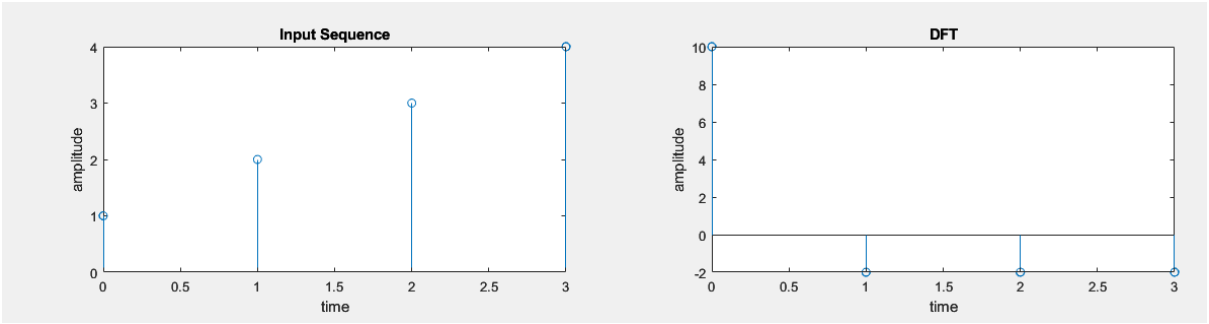
```
Command Window
Enter the sequence x= [4 5 6 9]
Enter the length of the DFT N= 7
  Columns 1 through 6

  24.0000 + 0.0000i  -2.3264 -13.6637i   3.0930 + 4.7651i   1.2334 - 6.2528i   1.2334 + 6.2528i   3.0930 - 4.7651i

  Column 7

  -2.3264 +13.6637i

fx >>
```



```
% DFT program with FFT


clear all;
clc;
xn=input('Enter the input sequence')
N=length(xn)
n=0:1:N-1;
subplot(2,2,1)
stem(n,xn)
xlabel('time')
ylabel('amplitude')
title('Input Sequence')
xk=fft(xn,N)
disp(xk)
k=0:1:N-1;
subplot(2,2,2)
stem(k,xk)
xlabel('time')
ylabel('amplitude')
title('DFT')
```

OUTPUTS:

```
Enter the input sequence[1 2 3 4]

xn =

     1     2     3     4


N =

     4


xk =

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

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



AIM: Implement circular convolution with and without cconv in MATLAB

Software Used: MATLAB R2022b

Theory:

Circular convolution is defined for periodic sequences, whereas convolution is defined for aperiodic sequences. The circular convolution of two N-point periodic sequences $x(n)$ and $y(n)$ is the N-point sequence $a(m) = x(n)^* y(n)$, defined by

$$a(m) = x(m)*y(m) = \sum_{n=0}^{N-1} x(n)y(m-n), \qquad m = 0, 1, 2, \ldots, N-1$$

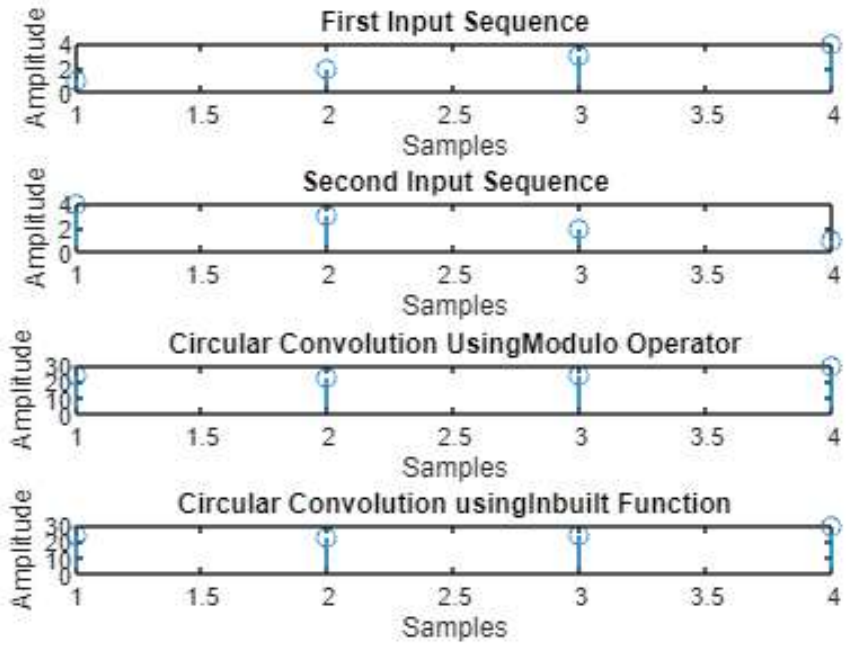
PROGRAM:

```
clc;
close all;
clear all;
x1=input('Enter the first sequence :');
x2=input('Enter the second sequence: ');
N1=length(x1);
N2=length(x2);
N=max(N1,N2);
 if(N2>N1)
x4=[x1,zeros(1,N-N1)];
x5=x2;
elseif(N2==N1)
        x4=x1;
        x5=x2;
else
x4=x1;
x5=[x2,zeros(1,N-N2)];
end
 x3=zeros(1,N);
for m=0:N-1
x3(m+1)=0;
for n=0:N-1
        j=mod(m-n,N);
        x3(m+1)=x3(m+1)+x4(n+1).*x5(j+1);
end
end

subplot(4,1,1)
stem(x1);
title('First Input Sequence');
xlabel('Samples');
ylabel('Amplitude');
subplot(4,1,2)
stem(x2);
title('Second Input Sequence');
xlabel('Samples');
ylabel('Amplitude');
subplot(4,1,3)
stem(x3);
title('Circular Convolution UsingModulo Operator');
xlabel('Samples');
ylabel('Amplitude');
```

```
%In built function
y=cconv(x1,x2,N);
subplot(4,1,4)
stem(y);
title('Circular Convolution usingInbuilt Function');
xlabel('Samples');
ylabel('Amplitude');
```

OUTPUTS:

```
Enter the first sequence :
[1 2 3 4]
Enter the second sequence:
[4 3 2 1]
>>
```



AIM: Find Z Transform of given sequence

Software Used: MATLAB R2022b

Theory:

Mathematically, if $x(n)$ is a discrete-time signal or sequence, then its *bilateral or two-sided Z-transform* is defined as −

$$Z[x(n)] = X(z) = \sum_{n=-\infty}^{\infty} x(n)\, z^{-n}$$

Where, z is a complex variable and it is given by,

$$z = r\, e^{j\omega}$$

Where, r is the radius of a circle.

Also, the *unilateral or one-sided z-transform* is defined as −

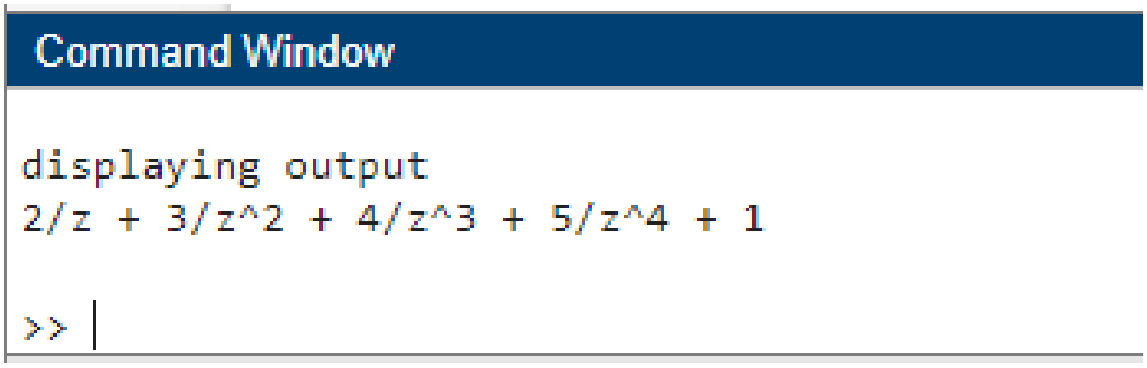
$$Z[x(n)] = X(z) = \sum_{n=0}^{\infty} x(n)\, z^{-n}$$

The unilateral or one-sided z-transform is very useful because we mostly deal with causal sequences. Also, it is mainly suited for solving difference equations with initial conditions.

PROGRAM:

```
clc
clear all
close all
x=[1 2 3 4 5]
l=length(x);
X=0;
z=sym('z');
for i=0:l-1
    X=X+x(i+1)*z^(-i);
end
disp('displaying output');
disp(X);
z=ztrans(X);
```

OUTPUT:



AIM: Design Low Pass Butterworth Filter using MATLAB

Software Used: MATLAB R2022b

Theory:

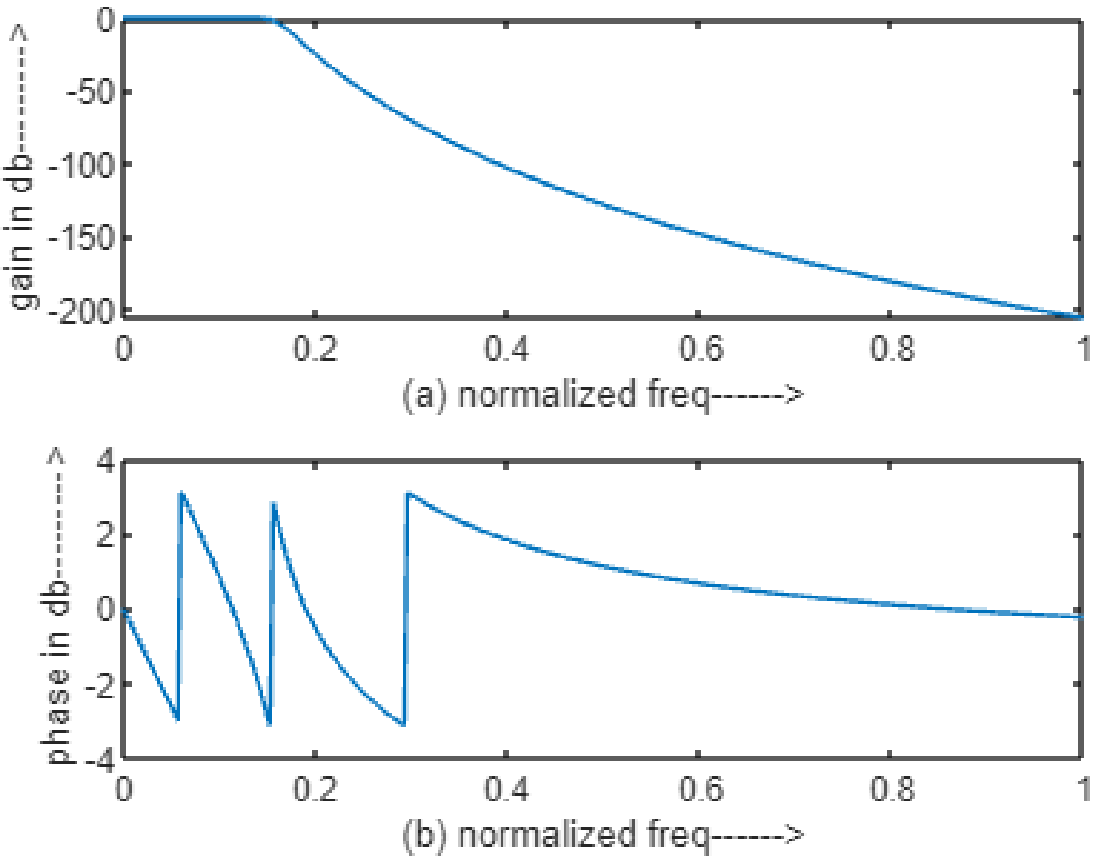
The Butterworth filter is a filter approximation technique that is also known as the maximally flat filter technique. This filter gives a very flat frequency response in the Pass Band, which ensures that there are no ripples present. Hence, as the name suggests, the maximal flat response you get with the Butterworth Filter cannot be matched by the Chebyshev, Inverse Chebyshev, or Bessel Filters. Though it may not be as effective as the other filters in giving a filter response with a steep transition band, it has a better tolerance to overshoot or ringing compared to the Chebyshev filters. It also allows for minimum attenuation in the Stop Band.

Type of transformation	Transformation(Replace s by the following)
Low Pass	$\frac{s}{\omega_c}$
High Pass	$\frac{\omega_c}{s}$
Band Pass	$\frac{s^2+\omega_l\omega_h}{s(\omega_h-\omega_l)}$
Band Stop	$\frac{s(\omega_h-\omega_l)}{s^2+\omega_l\omega_h}$

PROGRAM:

```
%Butterworth Low Pass Filter
clc;
close all;
clear all;
format long;
rp=input('enter the passband ripple:(default:0.15)');
rs=input('enter the stopband ripple:(default:60)');
wp=input('enter the passband frequency:(default:1500)');
ws=input('enter the stopband frequency:(default:3000)');
fs=input('enter the sampling frequency:(default:7000)');
w1=2*wp/fs;
w2=2*ws/fs;
[n,wn]=buttord(w1,w2,rp,rs,'s');
[z,p,k]= butter(n,wn);
[b,a]=butter(n,wn,'s');
w=0:0.01:pi;
[h,om]=freqs(b,a,w);
m=20*log10(abs(h));
an=angle(h);
subplot(2,1,1);
plot(om/pi,m);
ylabel('gain in db--------->');
xlabel('(a) normalized freq------>');
subplot(2,1,2);
plot(om/pi,an);
ylabel('phase in db--------->');
xlabel('(b) normalized freq------>');
```

OUTPUTS:


gain in db------->
0
-50
-100
-150
-200
0
0.2
0.4
0.6
0.8
1
(a) normalized freq------>
phase in db------->
4
2
0
-2
-4
0
0.2
0.4
0.6
0.8
1
(b) normalized freq------>