Digital Signal Processing Lab (IT-353)

Submitted in partial fulfilment of the requirement of the degree of

BACHELOR IN TECHNOLOGY

COMPUTER SCIENCE & ENGINEERING

By

Anirudh Gautam

ENROLLMENT NO. 01116403220

(5th Sem.)

University School of Information & Communication Technology

GURU GOBIND SINGH INDRAPRASTHA UNIVERSITY

DWARKA, NEW DELHI

2021-22

Under the Supervision of

Dr. C.S. Rai

USICT , GGSIPU

INDEX

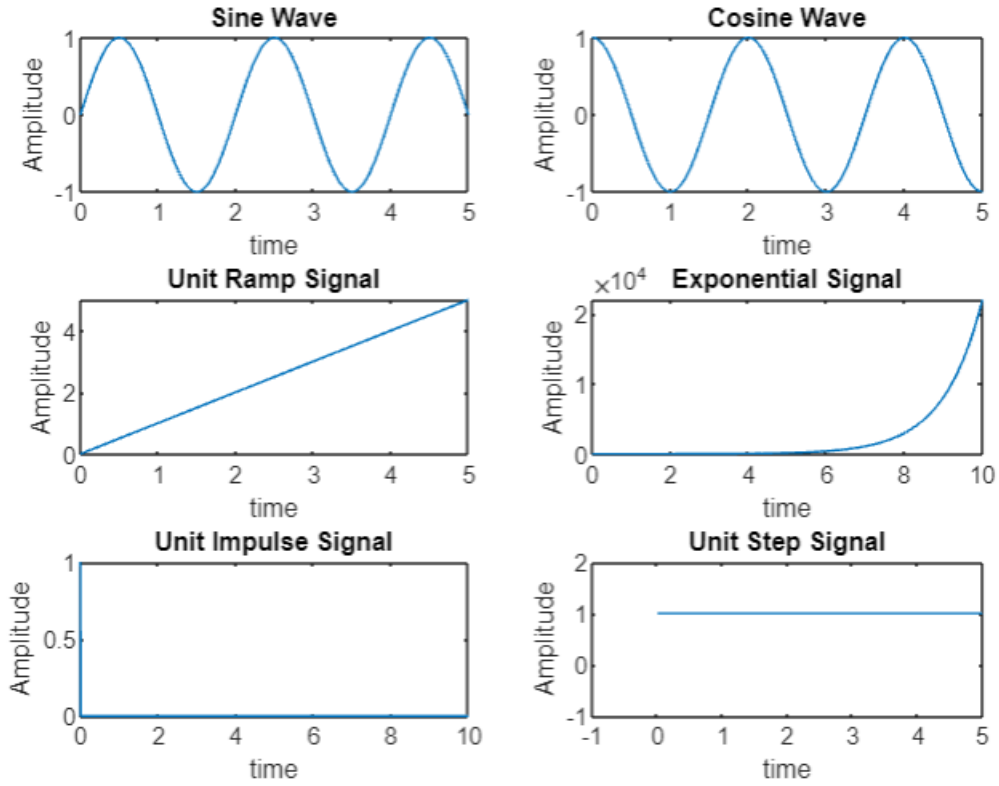
Experiment-1

Aim: To plot various standard functions using MATLAB.

Code:

```
clc;
clear;
t=0:0.01:10;
f = 0.5;
a=sin(2*pi*f*t);
b=cos(2*pi*f*t);
x=t;
y=(t==0);
z=(t>=0);
c=exp(t);
subplot(3,2,1);
plot(t,a);
axis([0 5 -1 1]);
xlabel('time');
ylabel('Amplitude');
title('Sine Wave');
subplot(3,2,2);
plot(t,b);
axis([0 5 -1 1]);
xlabel('time');
ylabel('Amplitude');
title('Cosine Wave');
subplot(3,2,3);
plot(t, x);
axis([0 5 0 5]);
xlabel('time');
ylabel('Amplitude');
title('Unit Ramp Signal')
subplot(3,2,4);
plot(t,c);
xlabel('time');
ylabel('Amplitude')
title('Exponential Signal')
subplot(3,2,5);
plot(t,y);
xlabel('time');
ylabel('Amplitude');
title('Unit Impulse Signal');
subplot(3,2,6);
plot(t,z);
axis([-1 5 -1 2]);
xlabel('time');
ylabel('Amplitude');
title('Unit Step Signal');
```

Output:


Sine Wave
Amplitude
time
Cosine Wave
Amplitude
time
Unit Ramp Signal
Amplitude
time
×10^4
Exponential Signal
Amplitude
time
Unit Impulse Signal
Amplitude
time
Unit Step Signal
Amplitude
time

Experiment-2

Aim: To implement convolution with and without inbuilt MATLAB function.

Code:

Without inbuilt function

```
clear;
close all;
x = input('enter first sequence');
h = input('enter second sequence');
subplot(3,1,1);
stem(x);
xlabel('x(n)');
ylabel('amplitude');
title('Signal 1');
subplot(3,1,2);
stem(h);
xlabel('y(n)');
ylabel('amplitude');
title('Signal 2');
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);
else
end
end
end
subplot(3,1,3);
stem(Y);
ylabel('Y[n]');
xlabel('------>n');
title('Convulation of Two Signals without conv function');
```

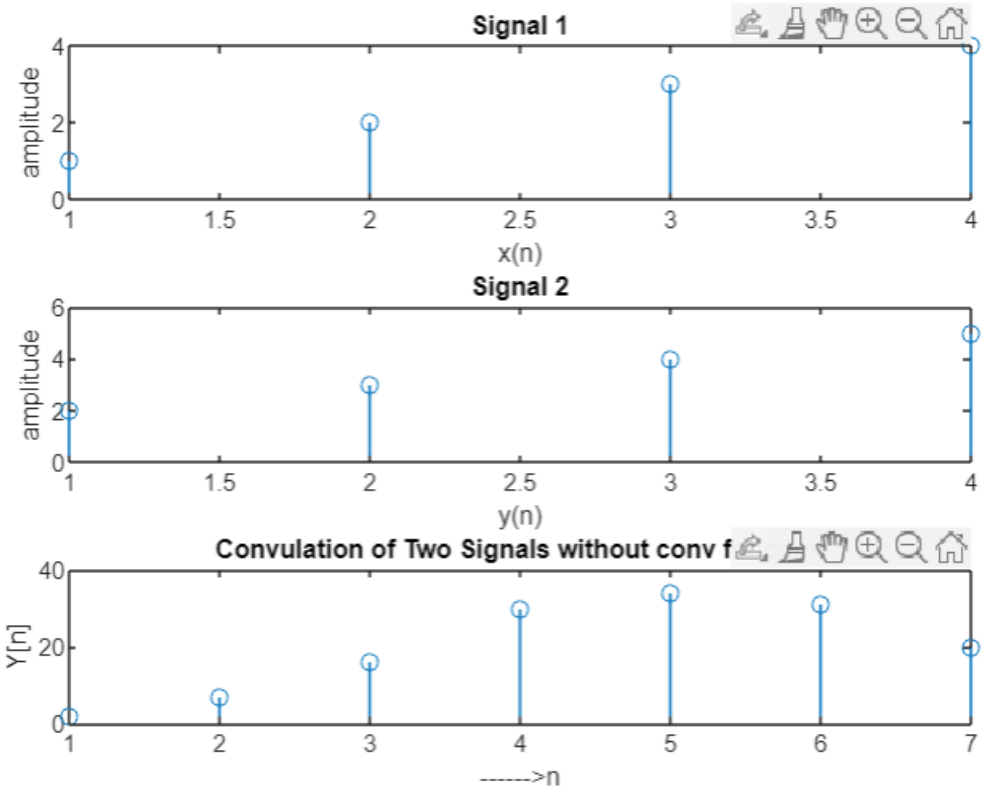
With inbuilt function

```
x=input("enter first sequence: ");
h=input("enter second seqeunce: ");
y=conv(x,h);
subplot(3,1,1);
stem(x);
xlabel('x(n)');
ylabel('amplitude');
subplot(3,1,2);
stem(h);
xlabel('y(n)');
```

```
ylabel('amplitude');
subplot(3,1,3);
stem(y);
xlabel('h(n)');
ylabel('amplitude');
title('the resultant signal is');
```

Output:

```
enter first sequence
[1,2,3,4]
enter second sequence
[2,3,4,5]
>>
```



Experiment-3

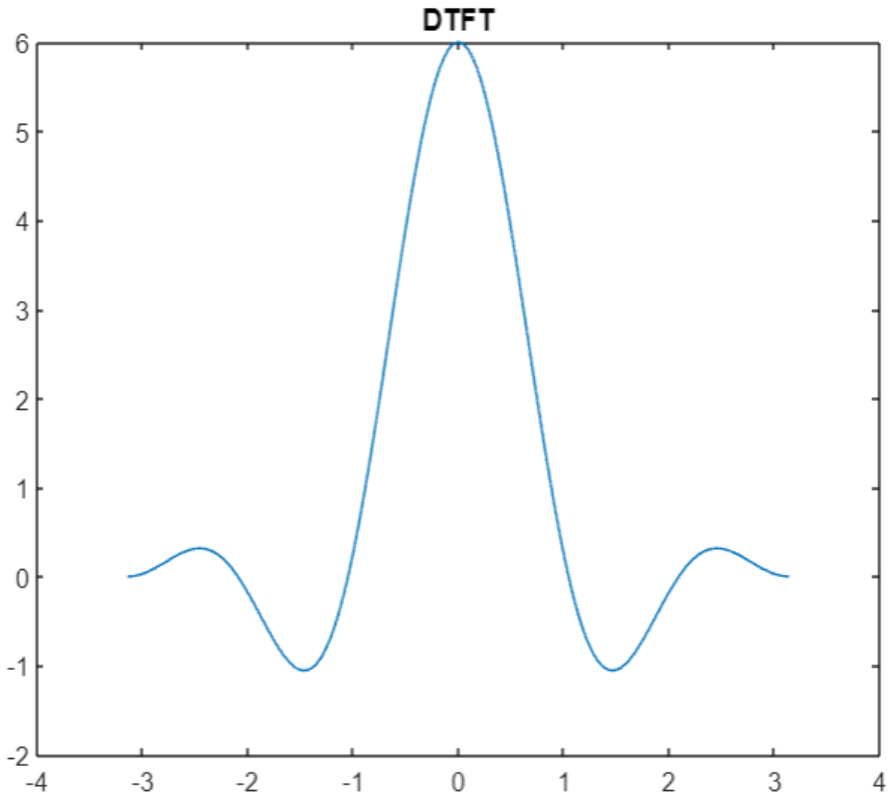
Aim: To find DTFT of a given sequence.

Code:

```
w=-pi:0.01:pi;
n=0:50;
x = input('Enter the sequence: ');
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');
```

Output:

```
>> dtft
Enter the sequence:
[1,2,2,1]
```



Experiment-4

Aim: To implement DFT on a given sequence.

Code:

```
clc;
xn=input('Input sequence: ');
N = input('Enter the number of points: ');
Xk=calcdft(xn,N);
disp('DFT X(k): ');
disp(Xk);
mgXk = abs(Xk);
phaseXk = angle(Xk);
k=0:N-1;
subplot (2,1,1);
stem(k,mgXk);
title ('DFT sequence: ');
xlabel('Frequency');
ylabel('Magnitude');
subplot(2,1,2);
stem(k,phaseXk);
title('Phase of the DFT sequence');
xlabel('Frequency');
ylabel('Phase');

function[Xk] = calcdft(xn,N)
    L=length(xn);
    if(N<L)
        error('N must be greater than or equal to L!!')
    end
    x1=[xn, zeros(1,N-L)];
    for k=0:1:N-1
        for n=0:1:N-1
            p=exp(-i*2*pi*n*k/N);
            W(k+1,n+1)=p;
        end
    end
    disp('Transformation matrix for DFT')
    disp(W);
    Xk=W*(x1.')
end
```

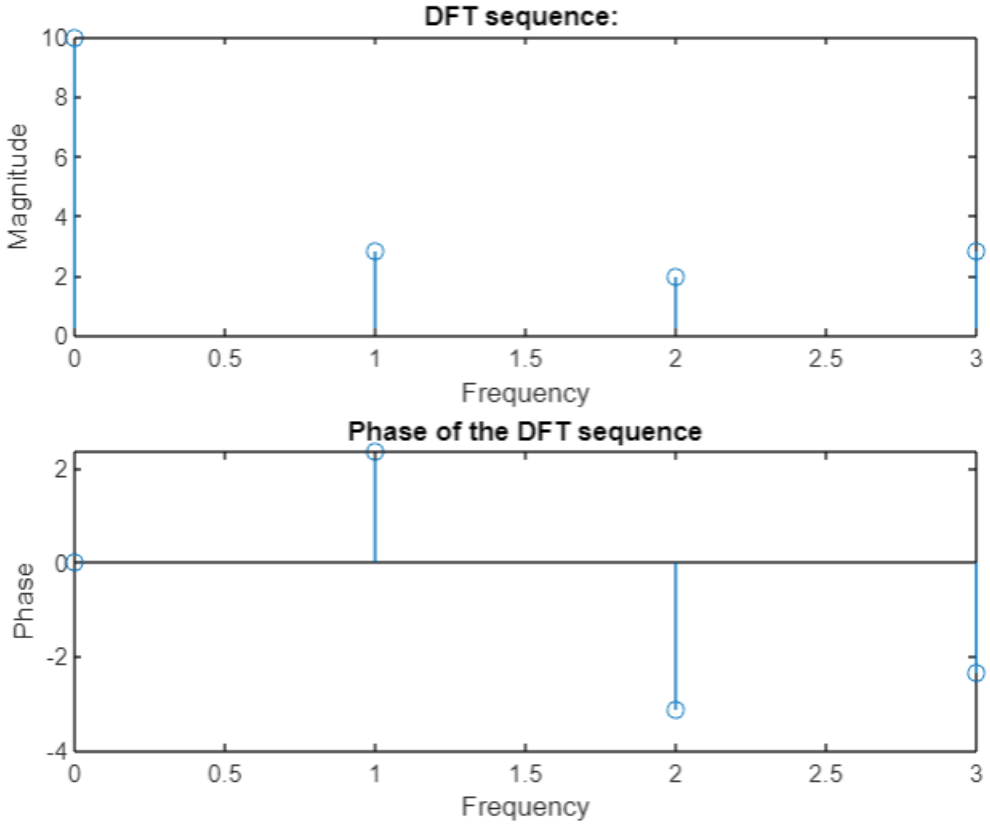
Output:

```
Input sequence:
[1,2,3,4]
Enter the number of points:
4
```

```
Transformation matrix for DFT
   1.0000 + 0.0000i   1.0000 + 0.0000i   1.0000 + 0.0000i   1.0000 + 0.0000i
   1.0000 + 0.0000i   0.0000 - 1.0000i  -1.0000 - 0.0000i  -0.0000 + 1.0000i
   1.0000 + 0.0000i  -1.0000 - 0.0000i   1.0000 + 0.0000i  -1.0000 - 0.0000i
   1.0000 + 0.0000i  -0.0000 + 1.0000i  -1.0000 - 0.0000i   0.0000 - 1.0000i
```

```
Xk =

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

```
DFT X(k):
  10.0000 + 0.0000i
  -2.0000 + 2.0000i
  -2.0000 - 0.0000i
  -2.0000 - 2.0000i
```



Experiment-5

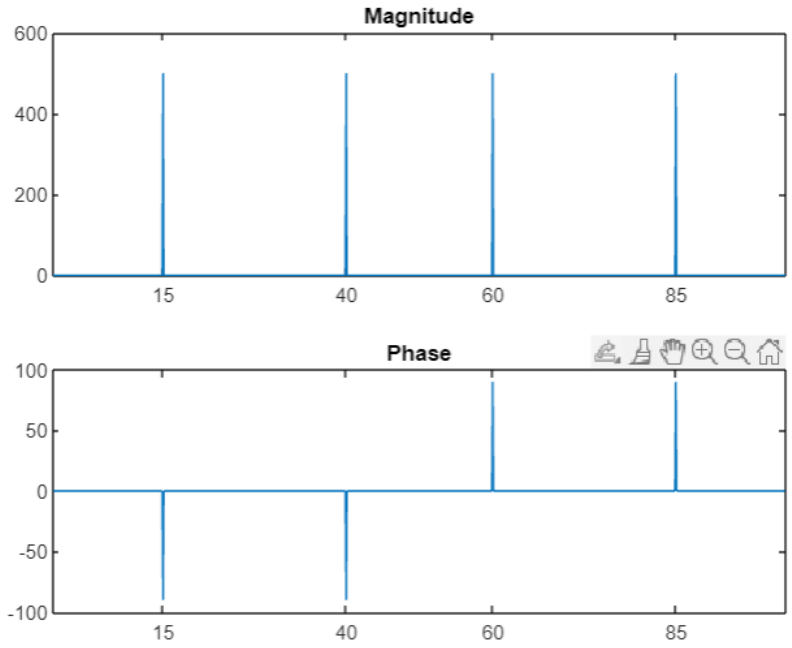
Aim: To find the DFT of a given sequence with and without FFT.

Code:

With fft function:

```
clc;
clear;
t = 0:1/100:10-1/100;                    % Time vector
x = sin(2*pi*15*t) + sin(2*pi*40*t);     % Signal
y = fft(x);                              % Compute DFT of x
m = abs(y);                              % Magnitude
y(m<1e-6) = 0;
p = unwrap(angle(y));
f = (0:length(y)-1)*100/length(y);       % Frequency vector

subplot(2,1,1)
plot(f,m)
title('Magnitude')
ax = gca;
ax.XTick = [15 40 60 85];

subplot(2,1,2)
plot(f,p*180/pi)
title('Phase')
ax = gca;
ax.XTick = [15 40 60 85];
```

Output:



Without fft function:

```
clc;
xn=input('Input sequence: ');
N = input('Enter the number of points: ');
Xk=calcdft(xn,N);
disp('DFT X(k): ');
disp(Xk);
mgXk = abs(Xk);
phaseXk = angle(Xk);
k=0:N-1;
subplot (2,1,1);
stem(k,mgXk);
title ('DFT sequence: ');
xlabel('Frequency');
ylabel('Magnitude');
subplot(2,1,2);
stem(k,phaseXk);
title('Phase of the DFT sequence');
xlabel('Frequency');
ylabel('Phase');

function[Xk] = calcdft(xn,N)
    L=length(xn);
    if(N<L)
        error('N must be greater than or equal to L!!')
    end
    x1=[xn, zeros(1,N-L)];
    for k=0:1:N-1
        for n=0:1:N-1
            p=exp(-i*2*pi*n*k/N);
            W(k+1,n+1)=p;
        end
    end
    disp('Transformation matrix for DFT')
    disp(W);
    Xk=W*(x1.')
end
```

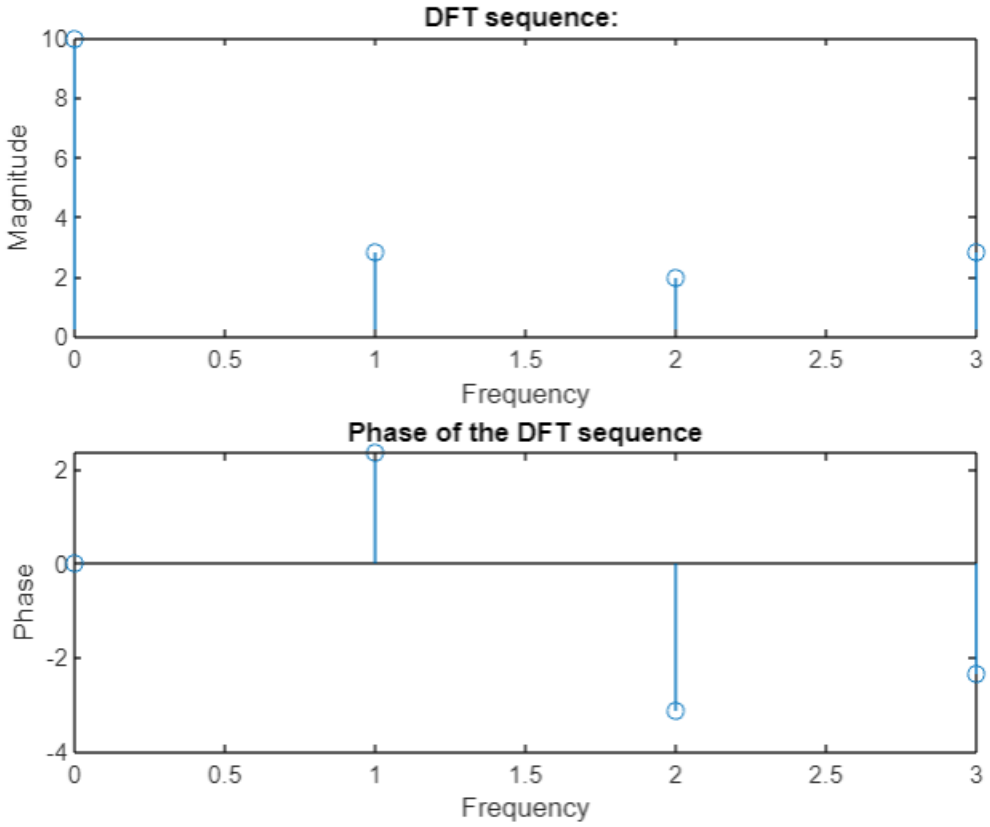
Output:

```
Input sequence:
[1,2,3,4]
Enter the number of points:
4
```

```
Transformation matrix for DFT
   1.0000 + 0.0000i   1.0000 + 0.0000i   1.0000 + 0.0000i   1.0000 + 0.0000i
   1.0000 + 0.0000i   0.0000 - 1.0000i  -1.0000 - 0.0000i  -0.0000 + 1.0000i
   1.0000 + 0.0000i  -1.0000 - 0.0000i   1.0000 + 0.0000i  -1.0000 - 0.0000i
   1.0000 + 0.0000i  -0.0000 + 1.0000i  -1.0000 - 0.0000i   0.0000 - 1.0000i
```

```
Xk =

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

```
DFT X(k):
  10.0000 + 0.0000i
  -2.0000 + 2.0000i
  -2.0000 - 0.0000i
  -2.0000 - 2.0000i
```



Experiment-6

Aim: To Implement circular convolution with and without inbuilt functions.

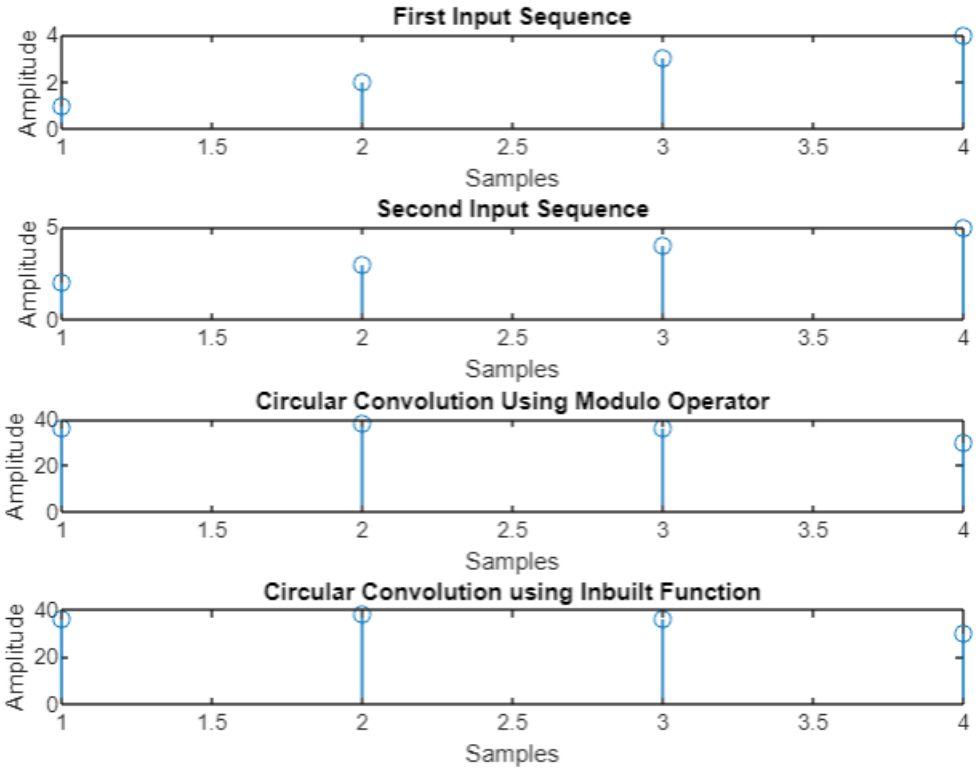
Code:

```
clc;
close all;
clear;
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 Using Modulo Operator');
xlabel('Samples');
ylabel('Amplitude');
```

```
%In built function

y=cconv(x1,x2,N);
subplot(4,1,4)
stem(y);
title('Circular Convolution using Inbuilt Function');
xlabel('Samples');
ylabel('Amplitude');
```

Output:

```
Enter the first sequence :
[1,2,3,4]
Enter the second sequence:
[2,3,4,5]
>>
```



Experiment-7

Aim: To find Z-transform of the given sequence.

Code:

```
%ztransform of finite duration sequence
clc;
close all;
clear;
syms 'z';
disp('If you input a finite duration sequence x(n), we will give you its z-
transform');
nf=input('Please input the initial value of n = ');
nl=input('Please input the final value of n = ');
x= input('Please input the sequence x(n)= ');
syms 'm';
syms 'y';
f(y,m)=(y*(z^(-m)));
disp('Z-transform of the input sequence is displayed below');
k=1;
for n=nf:1:nl
    answer(k)=(f((x(k)),n));
   k=k+1;
end
disp(sum(answer));
```

Output:

```
Please input the initial value of n =
0
Please input the final value of n =
4
Please input the sequence x(n)=
[1,0,3,-1,2]
```

```
Z-transform of the input sequence is displayed below
3/z^2 - 1/z^3 + 2/z^4 + 1

>>
```

Experiment-8

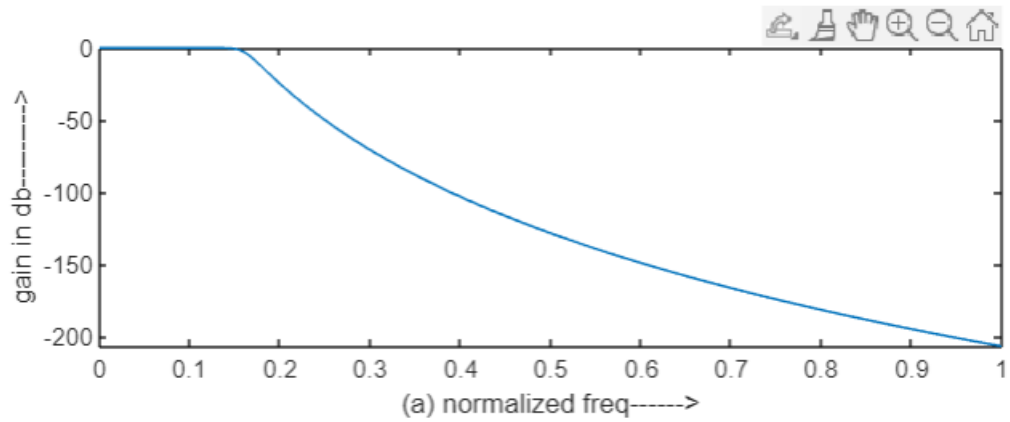
Aim: To design low pass Butterworth filter.

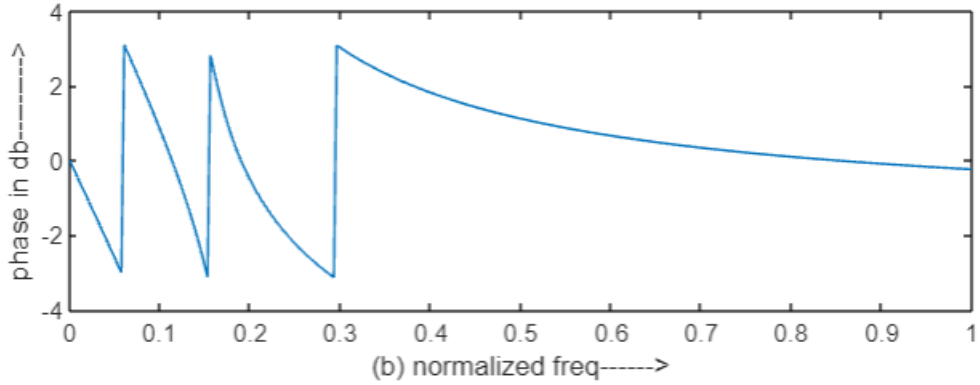
Code:

```
%Butterworth Low Pass Filter
clc;
close all;
clear;
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------>');
```

Output:

```
enter the passband ripple:(default:0.15)
0.15
enter the stopband ripple:(default:60)
60
enter the passband frequency:(default:1500)
1500
enter the stopband frequency:(default:3000)
3000
enter the sampling frequency:(default:7000)
7000
>>
```


0
-50
-100
-150
-200
gain in db------->
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1
(a) normalized freq------>


4
2
0
-2
-4
phase in db------->
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1
(b) normalized freq------>