DIGITAL SIGNAL PROCESSING LAB

(Practical file)

IT-353

BACHELOR OF TECHNOLOGY

INFORMATION TECHNOLOGY

University School of Information, Communication and Technology

GURU GOBIND SINGH INDRAPRASTHA UNIVERSITY, DWARKA

Submitted to:

Dr. Mansi Jhamb

Submitted by:

Apoorv Aron

01316401520

B.Tech IT 5TH SEM

Index

EXPERIMENT - 1

AIM:

To plot the waveforms for: Sine, Cos, Exponential, Ramp, Unit Impulse, Unit Step signal in continuous time using MATLAB.

THEORY:

1. **Sine Signal:** Sine signal is in the form of $x(t) = A\sin(\omega_0 \pm \Phi)$.
2. **Cosine Signal :** Cosine signal is in the form of $x(t) = A\cos(\omega_0 \pm \Phi)$.
3. **Exponential Signal:** Exponential signal is in the form of $x(t) = e^{at}$
4. **Unit Step Signal:** Step signal is denoted by u(t) and It is defined as $u(t)=\{0 \text{ if } t<0 \ \& \ 1 \text{ if } t>0 \}$
5. **Ramp Signal:** Ramp signal is denoted by r(t) and It is defined as $r(t)=\{t \text{ if } t\geq 0 \ \& \ 0 \text{ at } t<0\}$
6. **Unit Impulse Signal:** Impulse signal is denoted by $\delta(t)$ and It is defined as $\delta(t)=\{1 \text{ at } t=0 \ \& \ 0 \text{ at } t\neq 0\}$

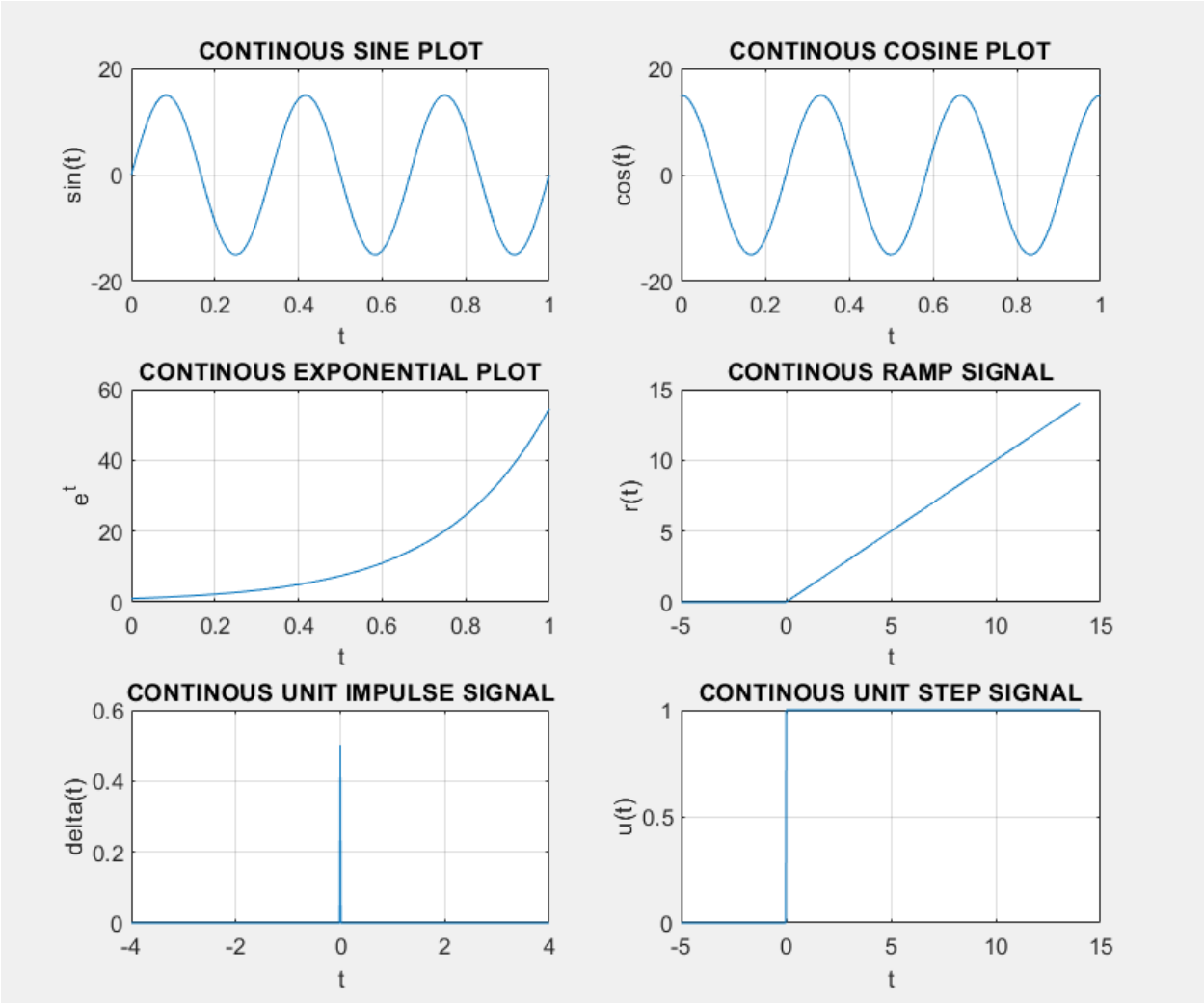
CODE:

```
clc;
clear all;
close all;

% Sine
A=input('Amplitude for sine signal: ');
f=input('Frequency for sine signal: ');
t=0:0.01:1;
Y=A*sin(2*pi*f*t);
subplot(3,2,1);
plot(t,Y);
title('CONTINUOUS SINE PLOT');
ylabel('sin(t)');
xlabel('t');
grid on;

% Cos
A=input('Amplitude for cosine signal: ');
f=input('Frequency for cosine signal: ');
t=0:0.01:1;
Y=A*cos(2*pi*f*t);
```

```
subplot(3,2,2);
plot(t,Y);
title('CONTINUOUS COSINE PLOT');
ylabel('cos(t)');
xlabel('t');
grid on;

% Exponential signal
A=input('Enter Exponent Coefficient: ');
t=0:0.01:1;
Y=exp(A*t);
subplot(3,2,3);
plot(t,Y);
title('CONTINUOUS EXPONENTIAL PLOT');
ylabel('e^t');
xlabel('t');
grid on;

% ramp
N=input('Enter input range for ramp and unit impulse: ');
n=-5:0.01:N-1;
Y=n.*(sign(n)+1)/2;
subplot(3,2,4);
plot(n,Y);
ylabel('r(t)');
xlabel('t');
title('CONTINUOUS RAMP SIGNAL');
grid on;

% Unit impulse signal
n=-4:0.01:4;
Y=(sign(n)+1)/2-(sign(n-0.0001)+1)/2;
subplot(3,2,5);
plot(n,Y);
ylabel('delta(t)');
xlabel('t');
title('CONTINUOUS UNIT IMPULSE SIGNAL');
grid on;

% Unit step signal
N=input('Enter input range for unit step signal: ');
n=-5:0.01:N-1;
Y=(sign(n)+1)/2;
subplot(3,2,6);
plot(n,Y);
ylabel('u(t)');
xlabel('t');
title('CONTINUOUS UNIT STEP SIGNAL');
grid on;
```

Command Window

```
Amplitude for sine signal: 15
Frequency for sine signal: 3
Amplitude for cosine signal: 15
Frequency for cosine signal: 3
Enter Exponent Coefficient: 4
Enter input range for ramp and unit inpulse: 15
Enter input range for unit step signal: 15
fx >>
```



EXPERIMENT - 2

AIM:

To plot the waveforms for: Sine, Cos, Exponential, Ramp, Unit Impulse, Unit Step signal in discrete time using MATLAB .

THEORY:

1. **Sine Signal:** Sine signal is in the form of $x(t) = A\sin(\omega_0 \pm \Phi)$.
2. **Cosine Signal :** Cosine signal is in the form of $x(t) = A\cos(\omega_0 \pm \Phi)$.
3. **Exponential Signal:** Exponential signal is in the form of $x(t) = e^{at}$
4. **Unit Step Signal:** Step signal is denoted by u(t) and It is defined as $u(t)=\{0 \text{ if } t<0 \ \& \ 1 \text{ if } t>0\}$
5. **Ramp Signal:** Ramp signal is denoted by r(t) and It is defined as $r(t)=\{t \text{ if } t\geq 0 \ \& \ 0 \text{ at } t<0\}$
6. **Unit Impulse Signal:** Impulse signal is denoted by δ(t) and It is defined as $\delta(t)=\{1 \text{ at } t=0 \ \& \ 0 \text{ at } t\neq 0\}$

CODE:

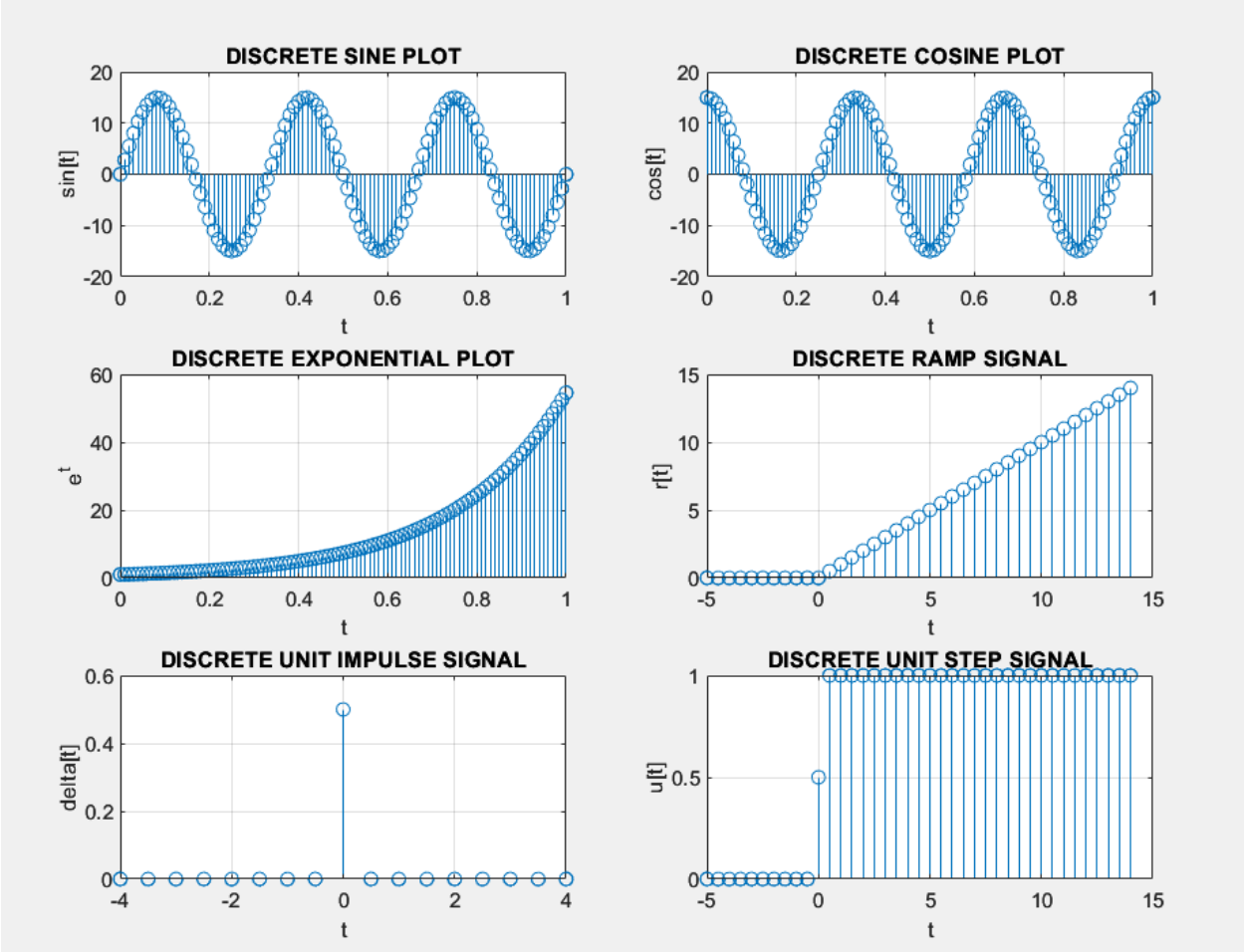
```
clc;
clear all;
close all;

% Sine
A=input('Amplitude for sine signal: ');
f=input('Frequency for sine signal: ');
t=0:0.01:1;
Y=A*sin(2*pi*f*t);
subplot(3,2,1);
stem(t,Y);
title('DISCRETE SINE PLOT');
ylabel('sin[t]');
xlabel('t');
grid on;

% Cos
A=input('Amplitude for cosine signal: ');
f=input('Frequency for cosine signal: ');
t=0:0.01:1;
```

```
Y=A*cos(2*pi*f*t);
subplot(3,2,2);
stem(t,Y);
title('DISCRETE COSINE PLOT');
ylabel('cos[t]');
xlabel('t');
grid on;

% Exponential signal
A=input('Enter Exponent Coefficient: ');
t=0:0.01:1;
Y=exp(A*t);
subplot(3,2,3);
stem(t,Y);
title('DISCRETE EXPONENTIAL PLOT');
ylabel('e^t');
xlabel('t');
grid on;

% ramp
N=input('Enter input range for ramp and unit impulse: ');
n=-5:0.5:N-1;
Y=n.*(sign(n)+1)/2;
subplot(3,2,4);
stem(n,Y);
ylabel('r[t]');
xlabel('t');
title('DISCRETE RAMP SIGNAL');
grid on;

% Unit impulse signal
n=-4:0.5:4;
Y=(sign(n)+1)/2-(sign(n-0.0001)+1)/2;
subplot(3,2,5);
stem(n,Y);
ylabel('delta[t]');
xlabel('t');
title('DISCRETE UNIT IMPULSE SIGNAL');
grid on;

% Unit step signal
N=input('Enter input range for unit step signal: ');
n=-5:0.5:N-1;
Y=(sign(n)+1)/2;
subplot(3,2,6);
stem(n,Y);
ylabel('u[t]');
xlabel('t');
title('DISCRETE UNIT STEP SIGNAL');
```

```
grid on;
```

Command Window

```
Amplitude for sine signal: 15
Frequency for sine signal: 3
Amplitude for cosine signal: 15
Frequency for cosine signal: 3
Enter Exponent Coefficient: 4
Enter input range for ramp and unit inpulse: 15
Enter input range for unit step signal: 15
fx >>
```



EXPERIMENT - 3

AIM:

To perform sampling. Consider an analog signal x(t) = 3 cos2000(pi)t+ 5sin6000(pi)t+10cos12000(pi)t. What is the Nyquist rate for this signal?What is a discrete time signal obtained after sampling at 5000 samples/s?

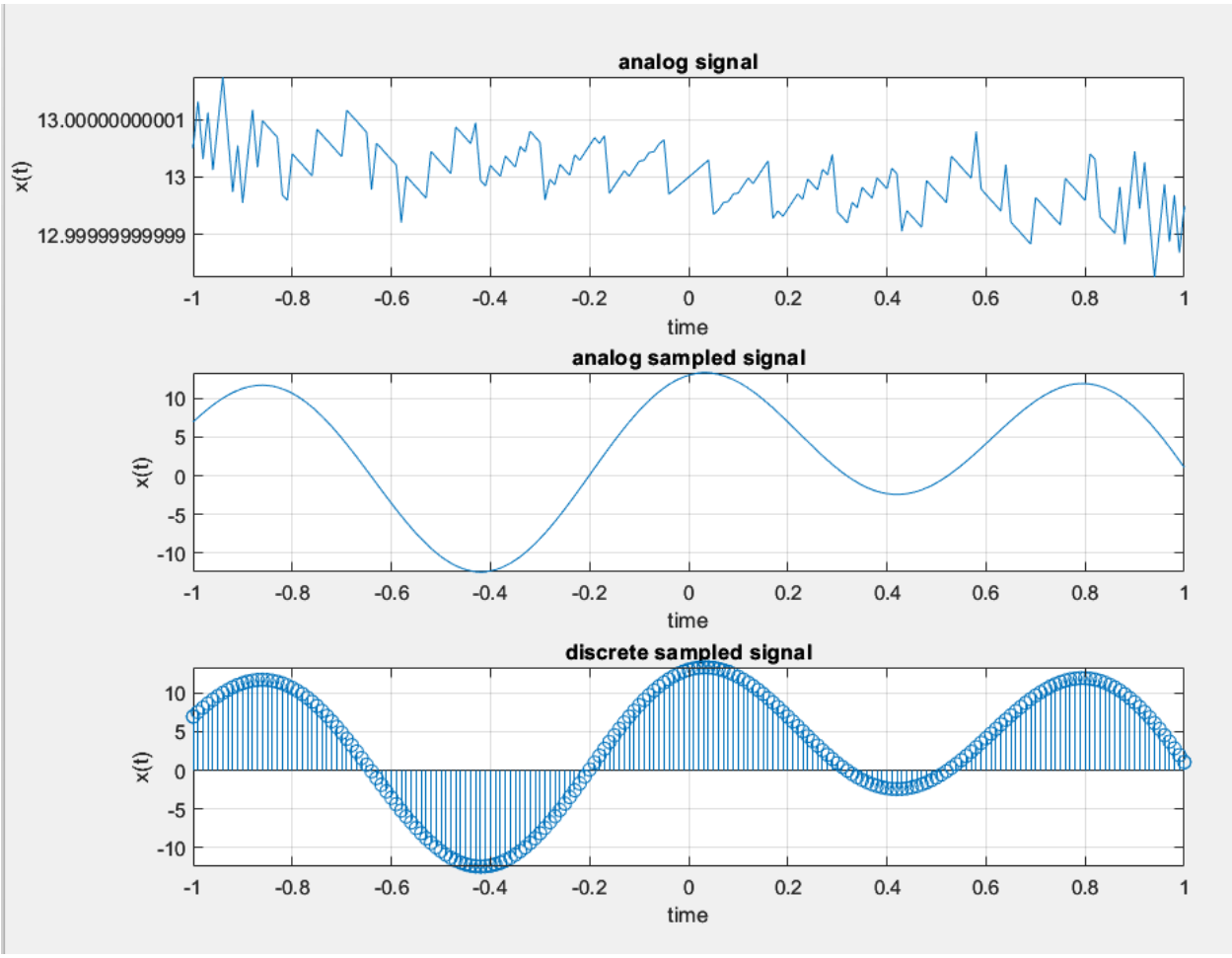
THEORY:

Nyquist rate : the minimum rate at which a signal can be sampled without introducing errors, which is twice the highest frequency present in the signal.

Signal : x(t) = 3 cos2000(pi)t+ 5sin6000(pi)t+10cos12000(pi)t

CODE:

```
clc;
clear all;

t = -1:0.01:1;
x = 3*cos(2000*pi*t) + 5*sin(6000*pi*t) + 10*cos(12000*pi*t);

subplot(3,1,1);
plot(t,x);
xlabel('time');ylabel('x(t)');title('analog signal');
grid;

%after sampling for 5000 sample/s

x = 3*cos(2/5*pi*t) + 5*sin(6/5*pi*t) + 10*cos(12/5*pi*t);
subplot(3,1,2);
plot(t,x);
grid;

xlabel('time');ylabel('x(t)');title('analog sampled signal');

subplot(3,1,3);
stem(t,x);
grid;

xlabel('time');ylabel('x(t)');title('discrete sampled signal')
```


analog signal
13.00000000001
13
12.99999999999
x(t)
-1
-0.8
-0.6
-0.4
-0.2
0
0.2
0.4
0.6
0.8
1
time
analog sampled signal
10
5
0
-5
-10
x(t)
time
discrete sampled signal
10
5
0
-5
-10
x(t)
time

EXPERIMENT - 4

AIM:

To perform operations of folding and the delaying (or advancing) on signal.

THEORY:

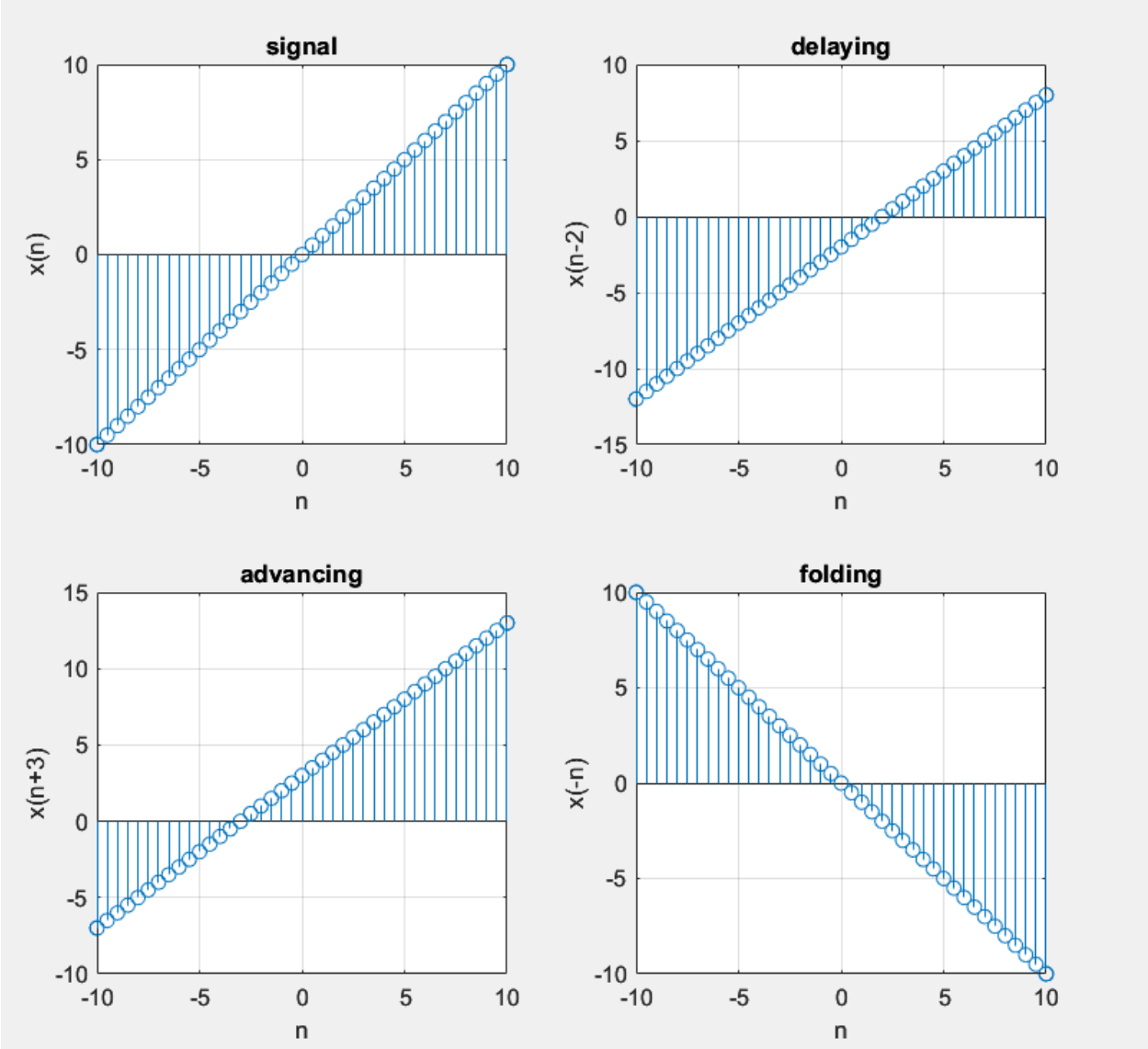
Folding is a technique that involves reflecting the signal about a certain point in time.

Delaying a signal involves shifting the signal in time by a certain amount.

Advancing a signal is similar to delaying a signal, but involves shifting the signal in the opposite direction in time.

CODE:

```
clc;
clear all;
close all;
n= -10:0.5:10;
x= n; % x(n)
x1=n-2;% x(n-3)
x2=n+3;% x(n+2)
x3=-n; % x(-n)
subplot(2,2,1);
stem(n,x);
xlabel('n');ylabel('x(n)');title('signal');grid;
subplot(2,2,2);
stem(n,x1);
xlabel('n');ylabel('x(n-2)');title('delaying');grid;
subplot(2,2,3);
stem(n,x2);
xlabel('n');ylabel('x(n+3)');title('advancing');grid;
subplot(2,2,4);
stem(n,x3);
xlabel('n');ylabel('x(-n)');title('folding');grid;
```


signal
x(n)
n
delaying
x(n-2)
n
advancing
x(n+3)
n
folding
x(-n)
n

EXPERIMENT - 5

AIM:

To perform Convolution.

THEORY:

Convolution: Convolution is the process by which one may compute the overlap of two graphs. In fact, convolution is also interpreted as the area shared by the two graphs over time. Metaphorically, it is a blend between the two functions as one passes over the other.

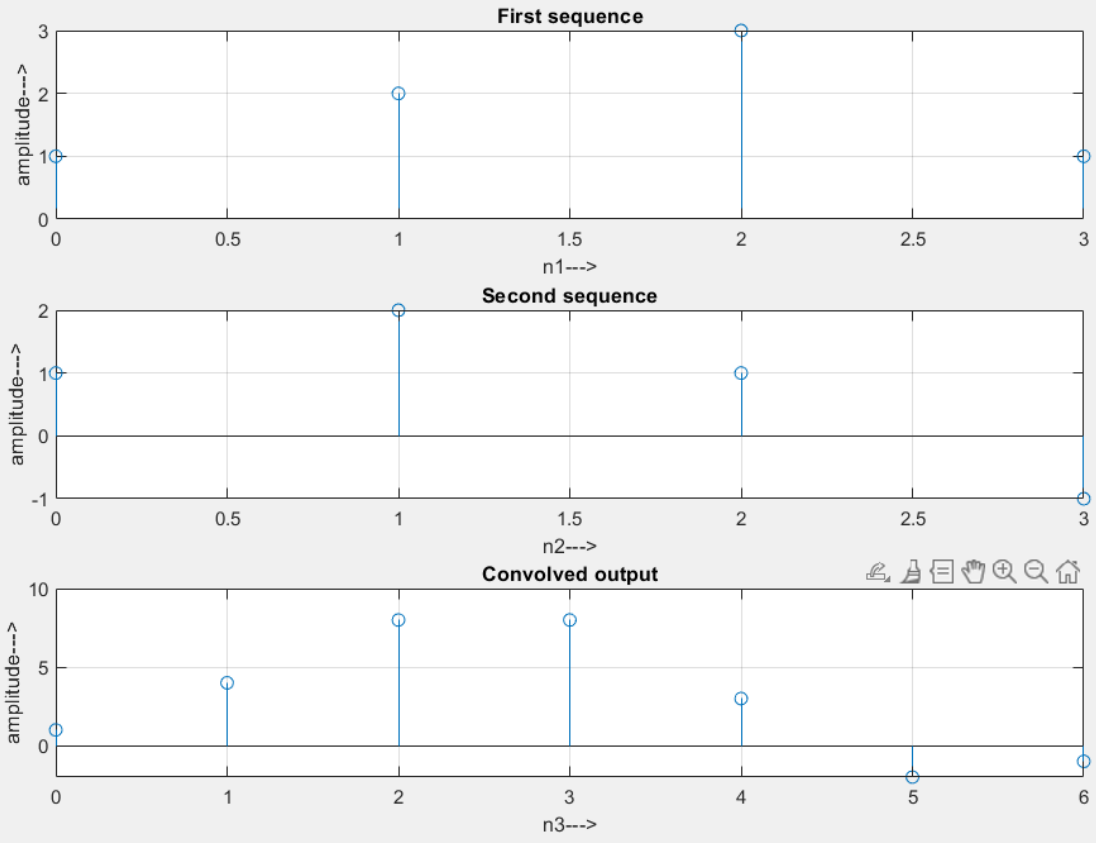
CODE:

```
clc;
close all;
x1=input('Enter the first sequence x1(n) = ');
x2=input('Enter the second sequence x2(n) = ');
L=length(x1);
M=length(x2);
N=L+M-1;
yn=conv(x1,x2);
disp('The values of y(n) are= ');
disp(yn);
n1=0:L-1;
subplot(311);
stem(n1,x1);
grid on;
xlabel('n1--->');
ylabel('amplitude--->');
title('First sequence');
n2=0:M-1;
subplot(312);
stem(n2,x2);
grid on;
xlabel('n2--->');
ylabel('amplitude--->');title('Second sequence');
n3=0:N-1;
subplot(313);
stem(n3,yn);
grid on;
```

```
xlabel('n3--->');
ylabel('amplitude--->');
title('Convolved output');
```

Command Window

```
Enter the first sequence x1(n) = [1,2,3,1]
Enter the second sequence x2(n) = [1,2,1,-1]
The values of y(n) are=
     1     4     8     8     3    -2    -1

>>
```



EXPERIMENT - 6

AIM:

To perform Cross-Correlation and Auto Correlation.

THEORY:

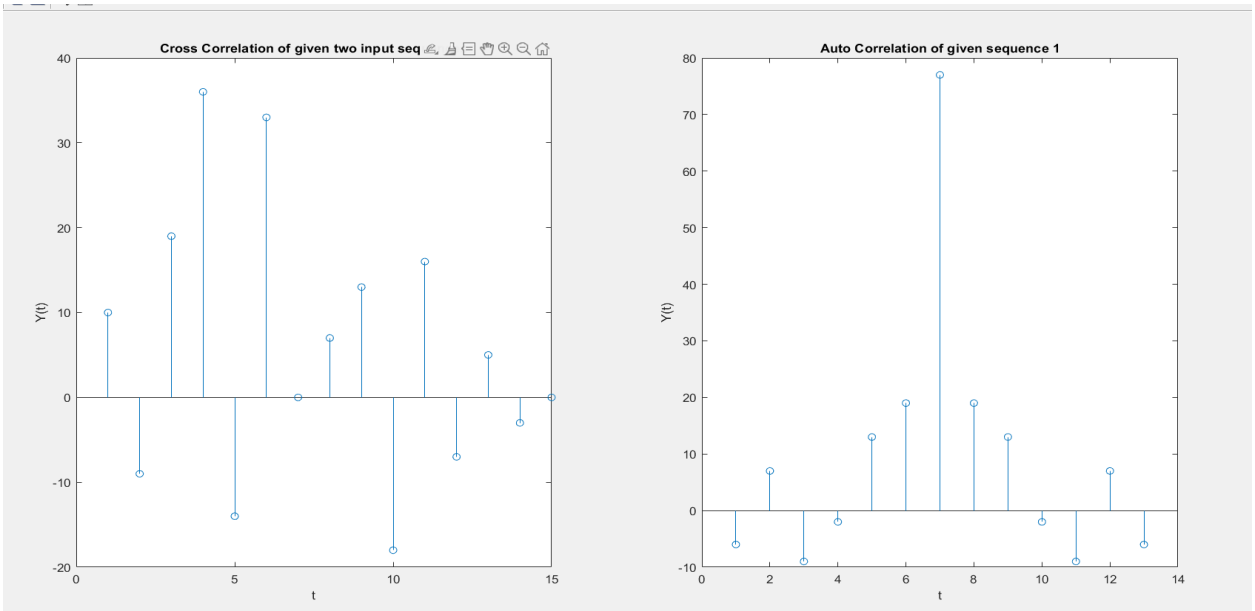
Cross-correlation is a measure of the similarity between two signals x[n] and y[n] as a function of the time lag between them.

Autocorrelation is similar to cross-correlation, but involves comparing a signal to a copy of itself rather than to another signal.

CODE:

```
clc;
close all;
a = input('Enter 1st sequence x1(n) = ');
b = input('Enter 2nd sequence x2(n) = ');
disp('Cross Correlation of given two input sequences');
y = xcorr(a,b);
disp(y);
subplot(1,2,1);
stem(y);
xlabel('t');
ylabel('Y(t)');
title('Cross Correlation of given two input sequences');
disp('Auto Correlation of given sequence 1');
y = xcorr(a,a);
disp(y);
subplot(1,2,2);
stem(y);
xlabel('t');
ylabel('Y(t)');
title('Auto Correlation of given sequence 1');
```

```
Enter 1st sequence x1(n) = [2,-1,3,7,1,2,-3]
Enter 2nd sequence x2(n) = [1,-1,2,-2,4,1,-2,5]
Cross Correlation of given two input sequences
  Columns 1 through 14

   10.0000   -9.0000   19.0000   36.0000  -14.0000   33.0000   -0.0000    7.0000   13.0000  -18.0000   16.0000   -7.0000    5.0000   -3.0000

  Column 15

   -0.0000

Auto Correlation of given sequence 1
   -6.0000    7.0000   -9.0000   -2.0000   13.0000   19.0000   77.0000   19.0000   13.0000   -2.0000   -9.0000    7.0000   -6.0000
```



EXPERIMENT - 7

AIM:

To compute DFT.

THEORY:

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

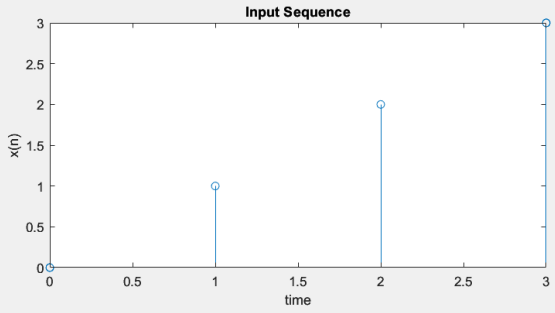
$$where \quad W_N^{nk} = e^{-j\frac{2\pi nk}{N}} \text{ [TWIDDLE FACTOR]}$$

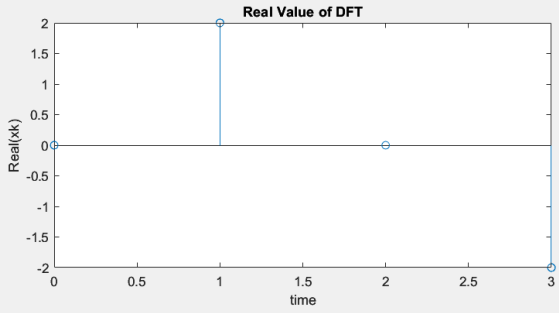
CODE:

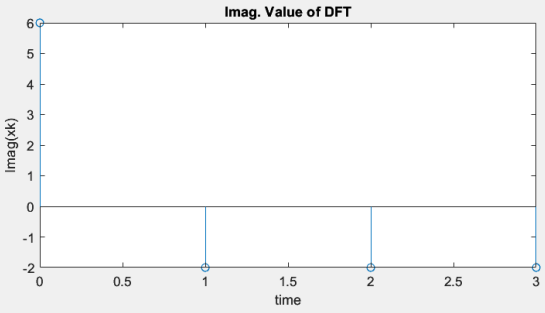
```
clc;
clear all;
close all;
xn = input('Enter the input sequence: ');
N = length(xn);
n=0:1:N-1;
subplot(2,2,1);
stem(n,xn);
xlabel('time');
ylabel('x(n)');
title('Input Sequence');
xk = fft(xn,N);
disp(xk);
k=0:1:N-1;
subplot(2,2,2);
stem(k,imag(xk));
xlabel('time');
ylabel('Real(xk)');
title('Real Value of DFT');
subplot(2,2,3);
stem(k,real(xk));
xlabel('time');
ylabel('Imag(xk)');
title('Imag. Value of DFT');
```

Command Window

```
Enter the input sequence: [0,1,2,3]
   6.0000 + 0.0000i  -2.0000 + 2.0000i  -2.0000 + 0.0000i  -2.0000 - 2.0000i

fx >>
```







EXPERIMENT - 8

AIM:

To perform Circular Convolution.

THEORY:

Circular convolution is a technique used to perform convolution of two periodic signals. It is defined as:

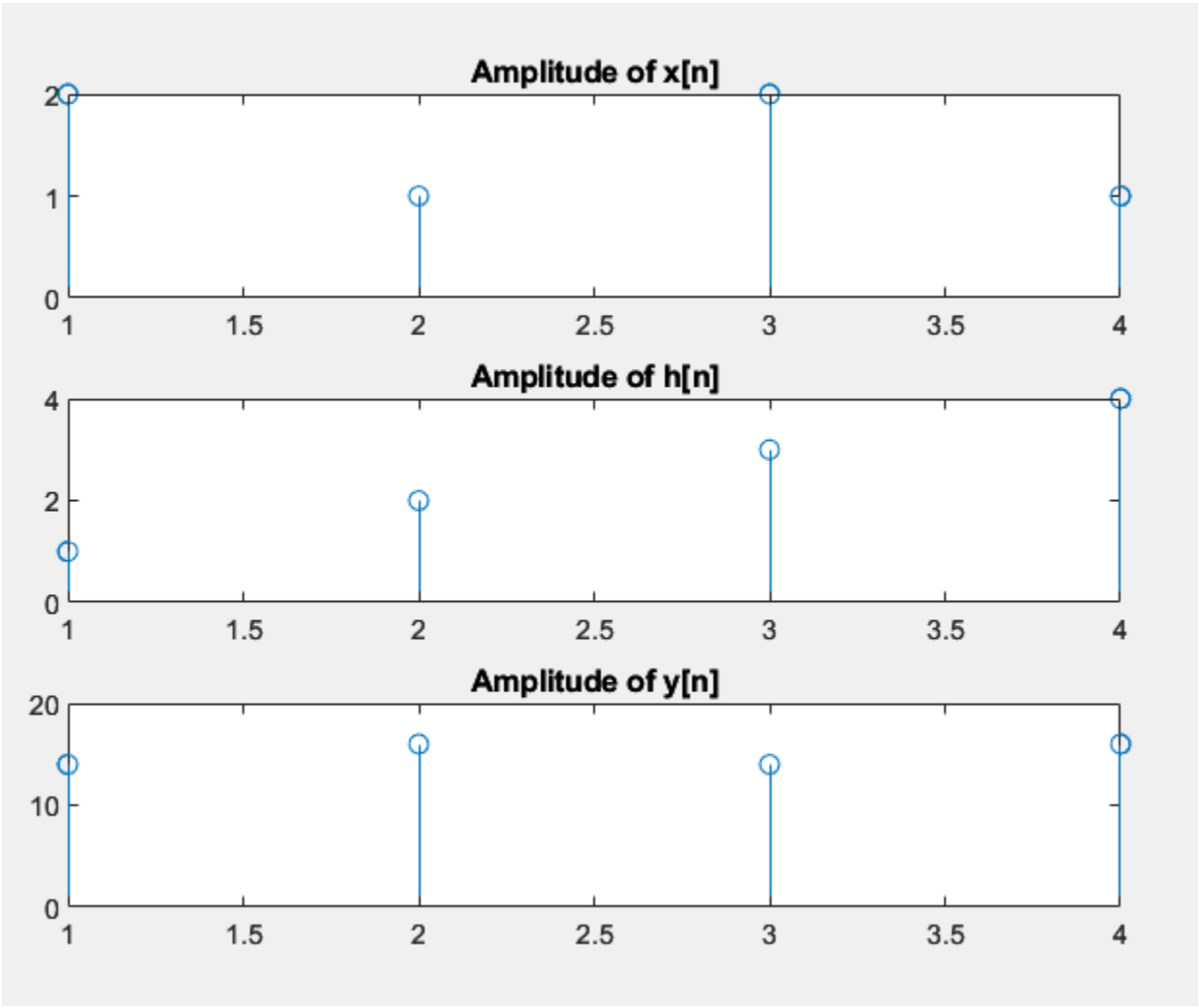
$$y[n] = \sum x[k]h[n-k]$$

where the sum is taken over all time indices k such that k is equivalent to n modulo the period of the signals.

CODE:

```
clc;
clear all;
close all;
x = input('Enter the first sequence: ');
h = input('Enter the second sequence: ');
n = length(x);
y = cconv(x, h, n);
amplitude_x = abs(x);
amplitude_h = abs(h);
amplitude_y = abs(y);
figure;
subplot(3,1,1);
stem(amplitude_x);
title('Amplitude of x[n]');
subplot(3,1,2);
stem(amplitude_h);
title('Amplitude of h[n]');
subplot(3,1,3);
stem(amplitude_y);
title('Amplitude of y[n]');
disp('Circular Convolution');
disp(amplitude_y);
```

```
Command Window
Enter the first sequence: [2,1,2,1]
Enter the second sequence: [1,2,3,4]
Circular Convolution
    14    16    14    16
```



EXPERIMENT - 9

AIM:

To perform z transform.

THEORY:

The z-transform is a mathematical tool used to represent discrete-time signals and systems in the frequency domain. It is defined as:

$$X(z) = \sum x[n]z^{-n}$$

where x[n] is the discrete-time signal, z is a complex variable, and the sum is taken over all time indices n.

CODE:

```
clc;
close all;
z=sym('z');
x = input('Enter the sequence = ');
l=length(x);
X=0;
for i=0:l-1
 X=X+x(i+1)*z^(-i);
end
disp('z-transform: ')
disp(X)
```

```
Enter the sequence = [1,2,5,7,0,1]
z-transform:
2/z + 5/z^2 + 7/z^3 + 1/z^5 + 1

fx >>
```

EXPERIMENT - 10

AIM:

To perform linear convolution using DFT.

THEORY:

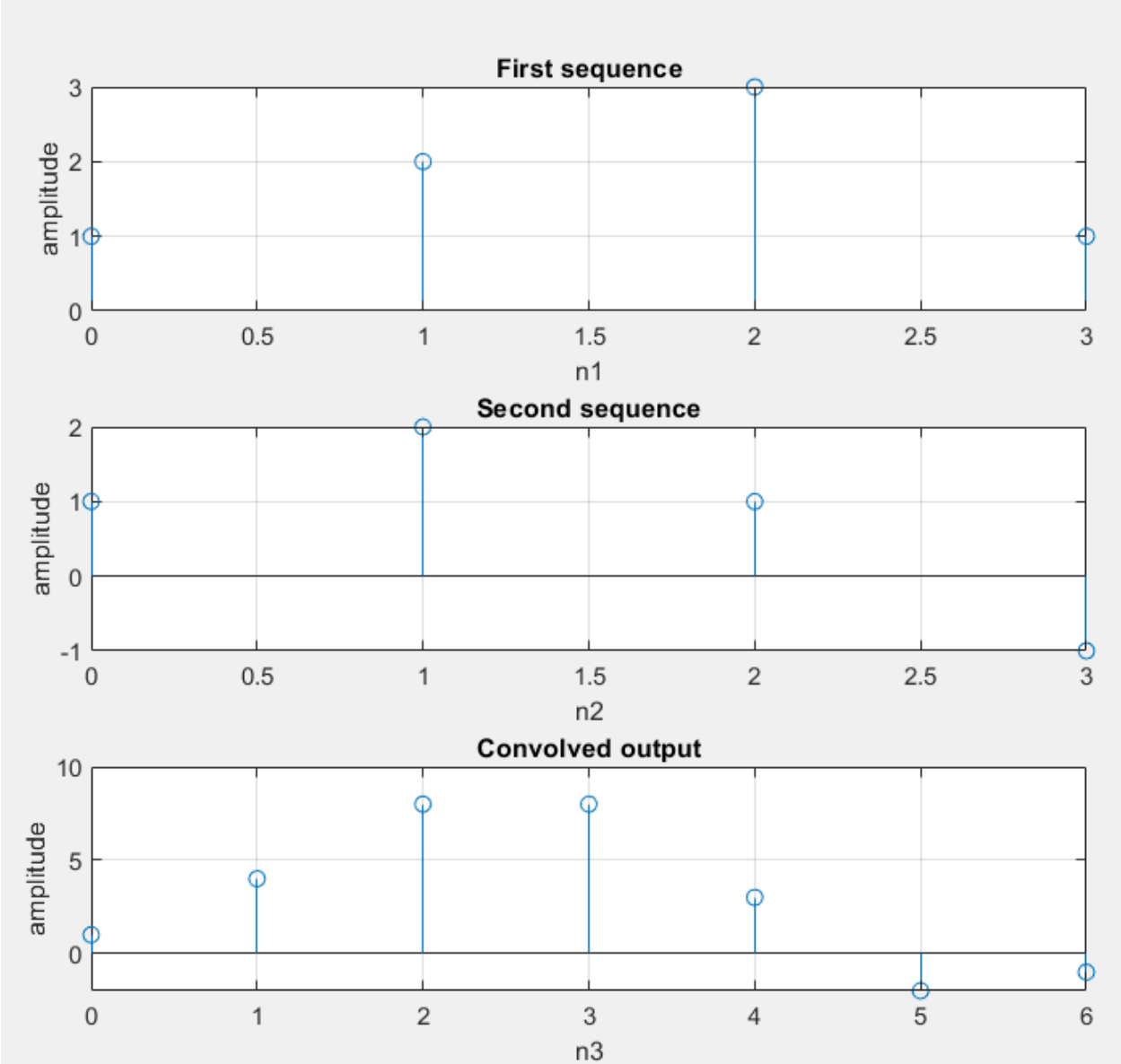
Linear convolution can be computed using the discrete Fourier transform (DFT) by taking the DFTs of the two signals, multiplying them element-wise, and then computing the inverse DFT (IDFT) of the product. This method is known as the fast convolution algorithm.

CODE:

```
clc;
clear all;
close all;

x1=input ('Enter the first input sequence x1[n]:');
x2=input('Enter the second input sequence x2[n]:');

Lx1=length(x1);
Lx2=length(x2);
N=Lx1+Lx2-1;
X1=fft (x1,N);
X2=fft (x2,N);
Y=X1.*X2;
y=ifft (Y, N);

disp ('Linear Convolution of x1 [n] & x2[n] is ')
disp (y)

% displaying First sequence (x1)
n1=0:Lx1-1;
subplot(3,1,1);
stem(n1,x1);
grid;
xlabel('n1');
ylabel('amplitude');
title('First sequence');
```

```
% displaying Second sequence (x2)
n2=0:Lx2-1;
subplot(3,1,2);
stem(n2,x2);
grid;
xlabel('n2');
ylabel('amplitude');
title('Second sequence');

% displaying Convolved output (yn)
n3=0:N-1;
subplot(3,1,3);
stem(n3,y);
grid;
xlabel('n3');
ylabel('amplitude');
title('Convolved output');

%Verification
z=conv (x1, x2) ;
disp ('Linear Convolution of x1 [n] and x2 [n] using builtin function is z [n]
= ');
disp (z);
```

```
Command Window
Enter the first input sequence x1[n]:[1,2,3,1]
Enter the second input sequence x2[n]:[1,2,1,-1]
Linear Convolution of x1 [n] & x2[n] is
    1.0000    4.0000    8.0000    8.0000    3.0000   -2.0000   -1.0000

Linear Convolution of x1 [n] and x2 [n] using builtin function is z [n] =
     1     4     8     8     3    -2    -1

fx >>
```



EXPERIMENT - 11

AIM:

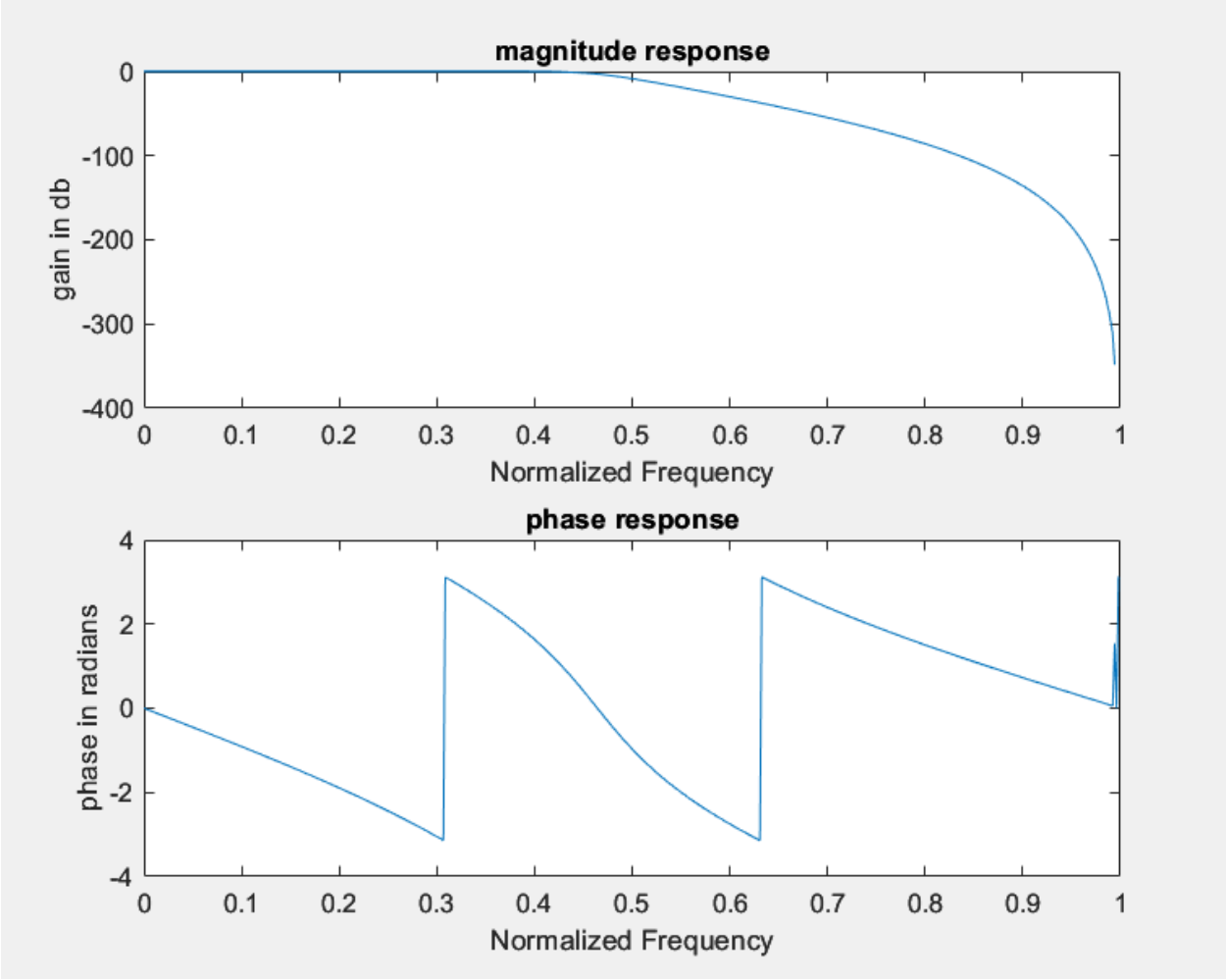
Design ButterWorth low pass Filter.

THEORY:

A Butterworth lowpass filter is a type of linear filter that is designed to pass low-frequency signals and reject high-frequency signals. It is characterized by a flat frequency response in the passband and a monotonic roll-off in the stopband.

CODE:

```
clc;
clear all;
close all;
alphas = 30;
alphap = 0.5;
fpass = 1000;
fstop = 1500;
fsam = 5000;
wp = 2*fpass/fsam;
ws = 2*fstop/fsam;
[n,wn] = buttord(wp,ws,alphap,alphas);
[b,a] = butter(n,wn);
[h,w] = freqz(b,a);
subplot(2,1,1);
plot(w/pi,20*log10(abs(h)));
xlabel('Normalized Frequency');
ylabel('gain in db');
title('magnitude response');
subplot(2,1,2);
plot(w/pi,angle(h));
xlabel('Normalized Frequency');
ylabel('phase in radians');
title('phase response');
```



1. "clc" clears the command window, this command clear all the previous output and command that are present in the command window.
2. "clear all" clears all variables in the workspace, it is used to free up memory and to prevent any unwanted interactions between variables.
3. "close all" closes all open figures or plots.
4. "alphas" and "alphap" are the stopband and passband attenuation levels in dB, respectively.
5. "fpass" and "fstop" are the cutoff frequencies for the passband and stopband, respectively.
6. "fsam" is the sampling frequency.
7. "wp" and "ws" are the normalized passband and stopband frequencies, respectively.

8. "buttord" is a MATLAB function that finds the minimum order of a Butterworth filter that meets a set of filter design specifications. It takes the normalized passband and stopband frequencies, and the passband and stopband attenuation levels as inputs and returns the order of the filter "n" and the cutoff frequency "wn" in radians per sample.
9. "butter" is a MATLAB function that designs a Butterworth filter. It takes the order of the filter "n" and the cutoff frequency "wn" as inputs and returns the filter coefficients "b" and "a" for the numerator and denominator of the transfer function, respectively.
10. "freqz" is a MATLAB function that computes the frequency response of a filter. It takes the filter coefficients "b" and "a" as inputs and returns the frequency response "h" and the frequency vector "w".
11. "subplot" is a MATLAB function that creates axes in tiled positions. It is used to create multiple plots in a single figure. In this code, it is used to create two plots: one for the magnitude response and one for the phase response.
12. "plot" is a MATLAB function that generates a 2D line plot of a given set of data points. It is used to plot the frequency response "h" against the frequency vector "w" in the first subplot, and to plot the phase response of the filter against the frequency vector "w" in the second subplot.
13. "xlabel" and "ylabel" are MATLAB functions that add labels to the x-axis and y-axis of a plot, respectively.
14. "title" is a MATLAB function that adds a title to a plot.

Passband and stopband attenuation refer to the amount of reduction in signal amplitude that occurs at certain frequencies when the signal is passed through a filter.

The passband of a filter is the range of frequencies that the filter is designed to pass through with minimal loss of signal amplitude. The passband attenuation is the maximum loss of signal amplitude that occurs within the passband. It is typically measured in decibels (dB) and is denoted by the parameter "alphap".

The stopband of a filter is the range of frequencies that the filter is designed to stop or reject. The stopband attenuation is the minimum loss of signal amplitude that occurs within the stopband. It is also measured in dB and is denoted by the parameter "alphas".

For example, if a filter has a passband attenuation of 0.5 dB, it means that the signal amplitude within the passband will be reduced by no more than 0.5 dB. Similarly, if a filter has a stopband attenuation of 30 dB, it means that the signal amplitude within the stopband will be reduced by at least 30 dB.

The passband and stopband attenuation are important design specifications for filters because they determine the filter's ability to selectively pass or reject certain frequencies. The higher the passband attenuation and the lower the stopband attenuation, the more selective the filter is and the better it is at separating different frequency components of the signal