Experiment 1
Basic Tools of Digital Signal Processing

Objective:

Objective of this experiment is to introduce students with some basic mathematical operations frequently used in processing of discrete time signal. This experiment will cover plot of periodic and non periodic sequences, determination of period of a sinusoidal sequence, convolution of sequences, moving average filter and *z*-transform and its inverse operation.

1. Plot the sequence, {5, -2, 0, 3, 4, 6}.

```
y=[5 -2 0 3 4 6];
stem(y, 'k')
xlabel('n')
ylabel('x(n)')
grid on
```

The result of above code is shown in fig. 1.1.

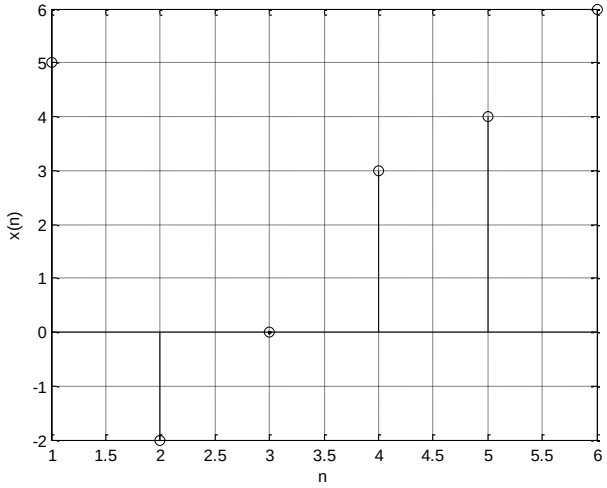
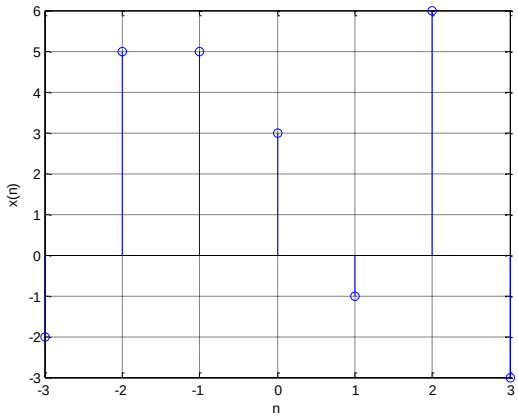


Fig. 1.1 The sequence of section-1

Plot the sequence $x = \left\{ -2, 5, 5, \underset{\uparrow}{3}, -1, 6, -3 \right\}$

```
n=-3:3;
x=[-2 5 5 3 -1 6 -3];
```

stem(n,x)
grid on
xlabel('n')
ylabel('x(n)')



2. Plot of the exponential sequence, $y(n) = ar^n$ for $a = 2$, $r = 0.8$ and 1.2.

n=0:1:20;
subplot(2,1,1)
a=2;
r=0.8;
y=a*r.^n;
stem(n,y, 'k')
xlabel('n')
ylabel('y(n)')
title('y=ar^n; where r<1')
grid on
subplot(2,1,2)
a=2;
r=1.2;
y=a*r.^n;
stem(n,y, 'k')
xlabel('n')
ylabel('y(n)')
title('y=ar^n; where r>1')
grid on

The result of above code is shown in fig. 1.2.

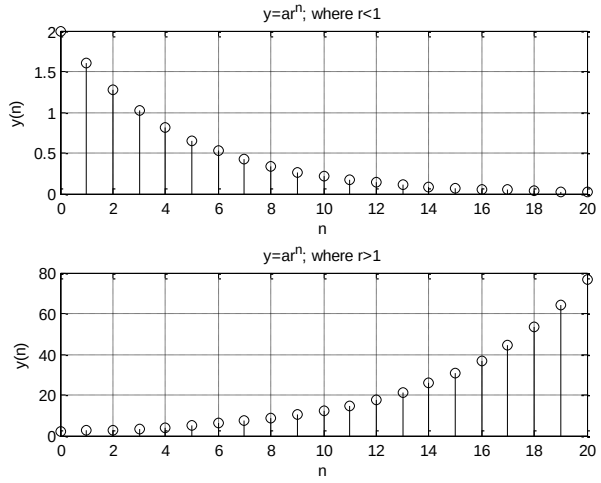


Fig. 1.2 The sequences of section-2

3. Plot the periodic sequence, $x[n]$ = {….5,4,3,2,1, 5,4,3,2,1, 5,4,3,2,1, 5,4,3,2,1……..} for n = -10 to 9.

```
% plot of periodic sequence
n=[-10:9];
x=[5,4,3,2,1];
p=4;
xy=x'*ones(1,p);
% xy indicates, p=4 column of vectors of  [ 5, 4 , 3,  2 , 1]
xy =(xy(:))'; %a long column vector will be converted
stem(n,xy, 'k')
xlabel('n')
ylabel('x(n)')
grid on
```

The result of above code is shown in fig. 1.3.

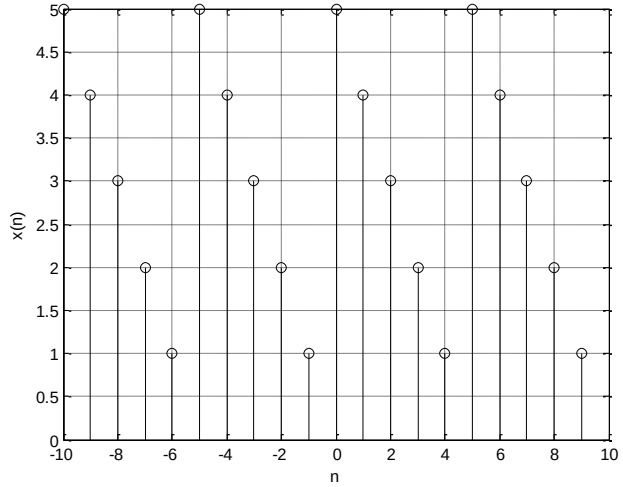


Fig. 1.3 The sequence of section-3

4. Plot the sequence, $x(n)=\left\{0,\ -1,\ 2,\ 5,\ \underset{\uparrow}{7},\ -4,\ 3,\ -5,\ 6,\ -2\right\}$, $y(n) = x(n-2)$, and $z(n) = x(n)+y(n)$.

```
n=-4:5;
x=[0 -1 2  5 7 -4  3 -5 6 -2];
stem(n,x,'r*')
hold on
k=2;
m=n+k;
y=x;
stem(m,y,'bd')
hold on
r=min(min(n),min(m)):max(max(n),max(m));%duration of z
z1=zeros(1,length(r));z2=z1;%initialization
z1(find((r>=min(n))&(r<=max(n))==1))=x;%x with duration of z
z2(find((r>=min(m))&(r<=max(m))==1))=y;%y with duration of z

z=z1+z2;
stem(r,z,'k>')
grid on
xlabel('n')
```

ylabel('x(n),y(n),z(n)')

legend('x(n)','y(n)=x(n-2)', 'z(n)=x(n)+y(n)')

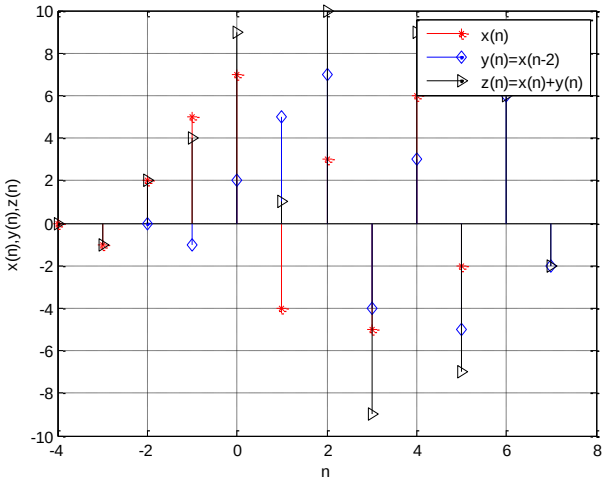


Fig.1.4 Plot of sequence *x(n)*, *x(n-2)* and *x(n)+x(n-2)*

5. For discrete time sequence, a sinusoidal wave is periodic if its frequency is a rational number like, $f = M / K$. Period of the wave is the denominator i.e. K.

%periodic sinusoidal wave

n=0:1:50;

f=1/20; %is a rational number, therefore period is 20

y=sin(2*pi*f*n);

subplot(2,1,1)

stem(n,y, 'k')

xlabel('n')

ylabel('y(n)')

title('Periodic sine wave N=20')

grid on

f=sqrt(2); % is an irrational number, non-periodic sinusoidal wave

y=sin(2*pi*f*n);

subplot(2,1,2)

stem(n,y, 'k')

xlabel('n')

ylabel('y(n)')

title('Non-periodic sine wave')

grid on

The result of above code is shown in fig. 1.5.

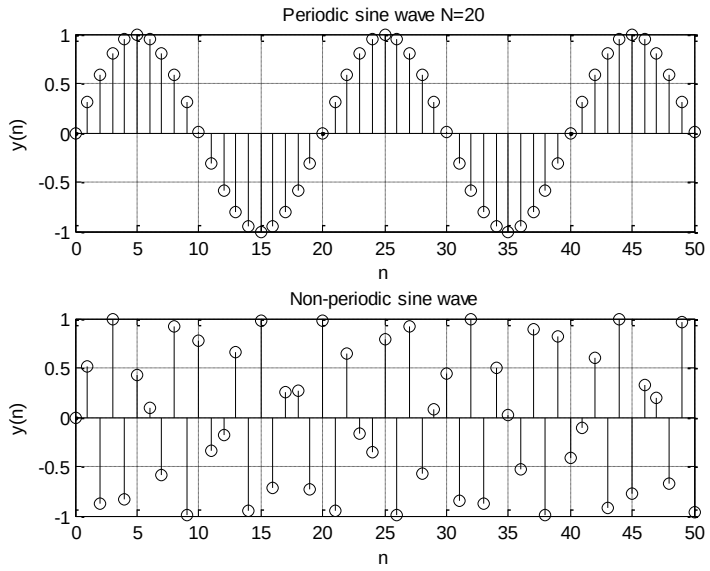


Fig. 1.5 The periodic and non-periodic sinusoidal sequences of step-4

6. The sampling rate of $y(2n)$ is half the rate of $y(n)$ therefore $y(2n)$ is called the under sampled version of $y(n)$. Let us verify the phenomenon.

```
a=0.2;
r=0.8;
N=10;
for n=1:N,
s(n)=n;
y(n)=a*r.^n;
end

M=N/2;
for m=1:M,
m=2*m;
p(m)=m;
z(m)= y(m);
end
stem(s,y,'ks')%Original sequence y(n)
```

hold on
stem(p,z,'k*')%under sampled sequence y(2n)
xlabel('n')
ylabel('y(n) and y(2n)')
grid on
legend('Original sequence', 'Under sampled')
The result of above code is shown in fig. 1.6.

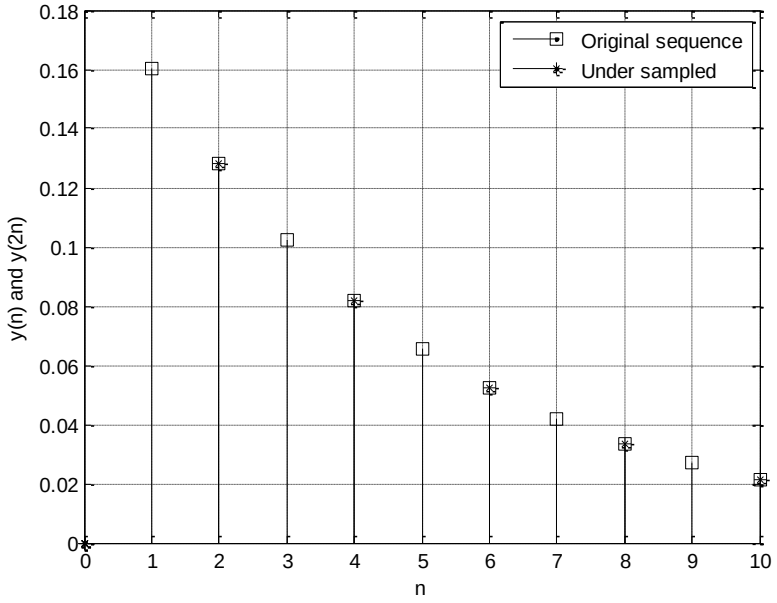


Fig. 1.6 plot of y(n) and y(2n)

7. Determine convolution of two sequences A = {6, 2, 0, 5, 6} and B = {-4, -2, 1, 0, 7} and find the length of resultant sequence.

A=[6 2 0 5 6];
B=[-4 -2 1 0 7];
Y=conv(A,B);
stem(A,'bs')
hold on
stem(B, 'r>')
hold on
stem(Y,'k*')
legend('Sequence A', ' Sequence B', 'Convolution of sequences A*B')
xlabel('n')
ylabel('Amplitude')
length(Y)

grid on

Result of above code is shown in fig. 1.7

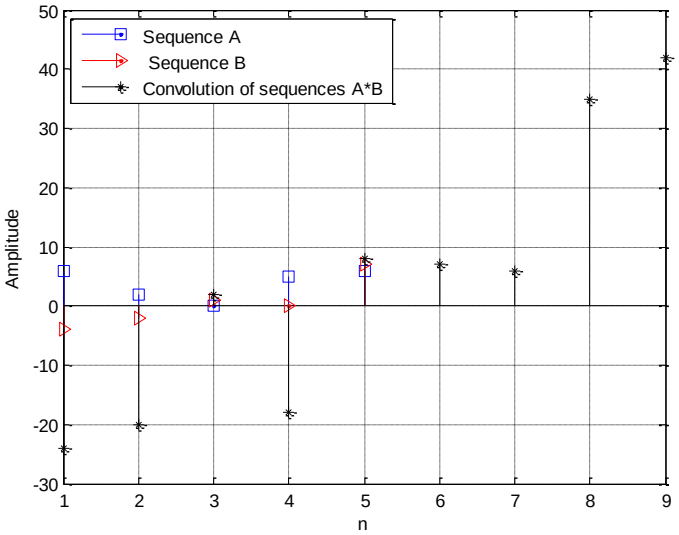


Fig. 1.7 Sequences A and B and their convolution

8. Generate 100 random numbers between 0 and 1. Add them with an exponential sequence as a random noise. Plot both noisy and noiseless signals.

```
a=0.5;
r=0.96;
N=100; %number of samples
for n=1:N,
e(n)=n;
y(n)=a*r.^n;
yn(n)=y(n)+0.04*rand();%noisy sequence
end
stem(e,y,'k*') %values of y(n)
hold on
stem(e,yn,'kd') %plot of noisy sequence
xlabel('n')
ylabel('y(n)')
legend('Original sequence', 'Noisy Sequence')
grid on
```

The result of above code is shown in fig. 1.8.

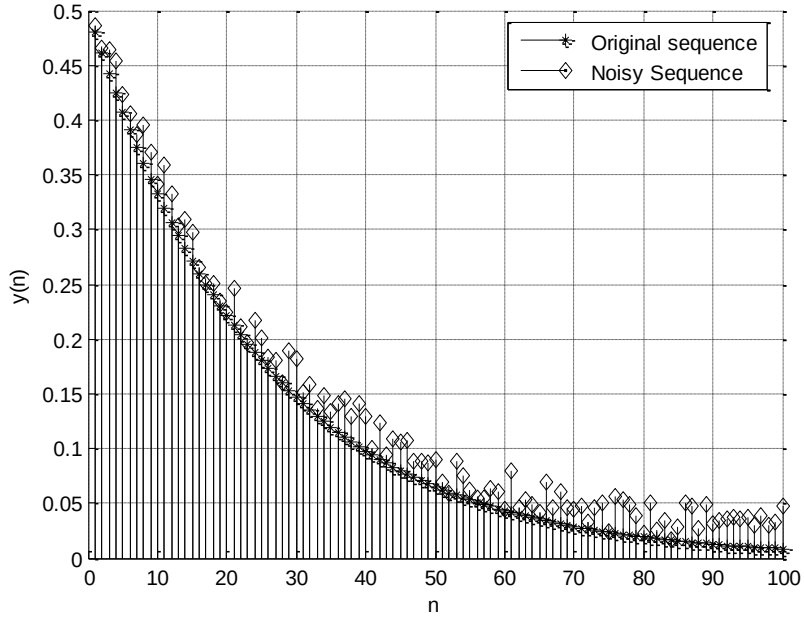


Fig. 1.8 The result of section-7.

9. Generate a unit step sequence for *n* = [-6, 6].

```
n =[-6 -5 -4 -3 -2 -1 0 1 2 3  4 5 6];
x=[0   0  0  0  0  0 1 1 1  1  1  1  1];
stem(n,x, 'k')
xlabel('n')
ylabel('x(n)')
title('Unit step sequence')
grid on
```

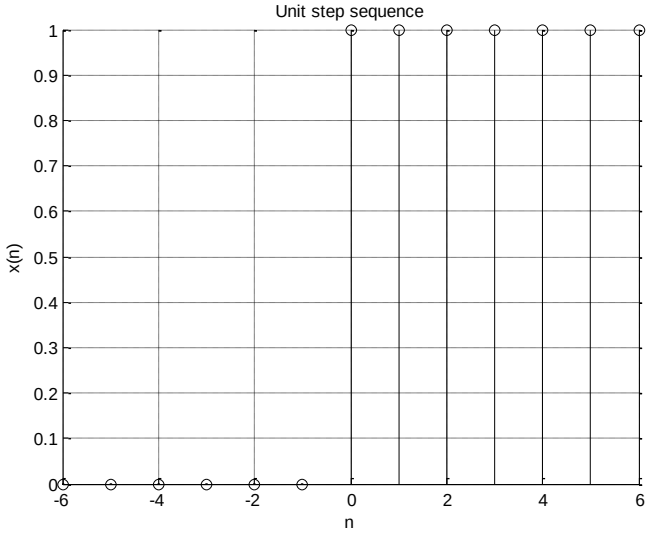


Fig. 1.9 Unit step sequence.

For impulse sequence above code will be modified like,

```
n =[-6 -5 -4 -3 -2 -1 0 1 2 3  4 5 6];
x=[0   0  0  0  0  0 1 0 0  0  0  0  0];
stem(n,x, 'k')
xlabel('n')
ylabel('x(n)')
title('Impulse sequence')
grid on
```

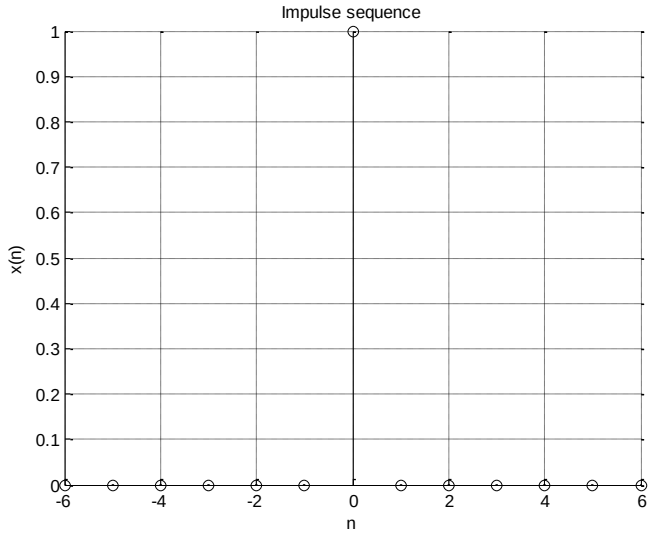


Fig. 1.10 Impulse sequence

10. Computes the two-dimensional convolution of matrices, $A = \begin{bmatrix} 1 & -2 & 0 \\ 3 & 4 & -11 \\ 6 & -5 & 2 \end{bmatrix}$ and $B = \begin{bmatrix} -3 & 2 & -8 \\ 2 & -7 & 0 \\ -1 & 4 & -1 \end{bmatrix}$.

```
A = [1 -2  0; 3  4 -11; 6  -5  2];
B = [-3  2  -8; 2  -7 0; -1  4  -1];
H = conv2(A, B);
mesh(H)
```

Result of above code is shown in fig. 1.11.

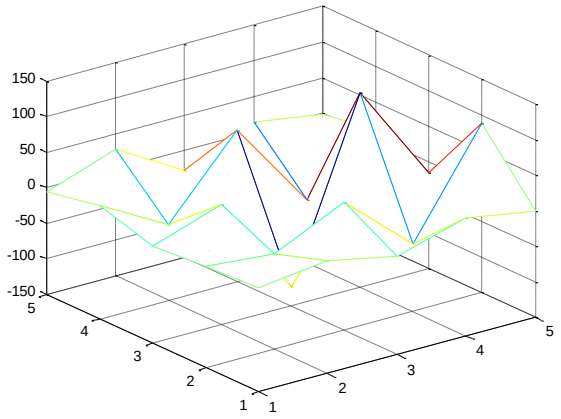


Fig. 1.11 The result of section-8

11. Generate 50 random numbers in the range [1,-1] and add it with the sequence $x(n) = 2n(0.9)^n$. Pass the noisy signal through a moving average filter and observe the signal before and after filtering.

Basic Theory:
Moving average system is defined as,

$$y(n) = \frac{1}{M_1 + M_2 + 1} \sum_{n=-M_1}^{M_2} x(n-k)$$

For example if M_1=1 and M_2=2 then,

$$y(n) = \frac{1}{1+2+1} \sum_{n=-1}^{2} x(n-k)$$

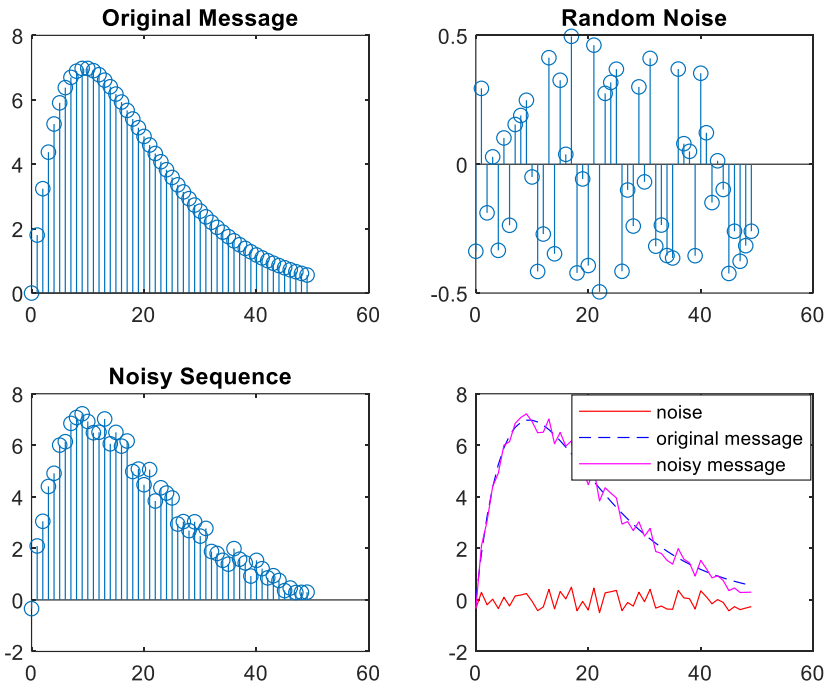
$$= \frac{1}{4}\{x(n+1) + x(n) + x(n-1) + x(n-2)\}$$

Therefore,
y(0)=0.25{x(1)+x(0)+x(-1)+x(-2)}
y(1)=0.25{x(2)+x(1)+x(0)+x(-1)}
y(-1)=0.25{x(0)+x(-1)+x(-2)+x(-3)} etc.

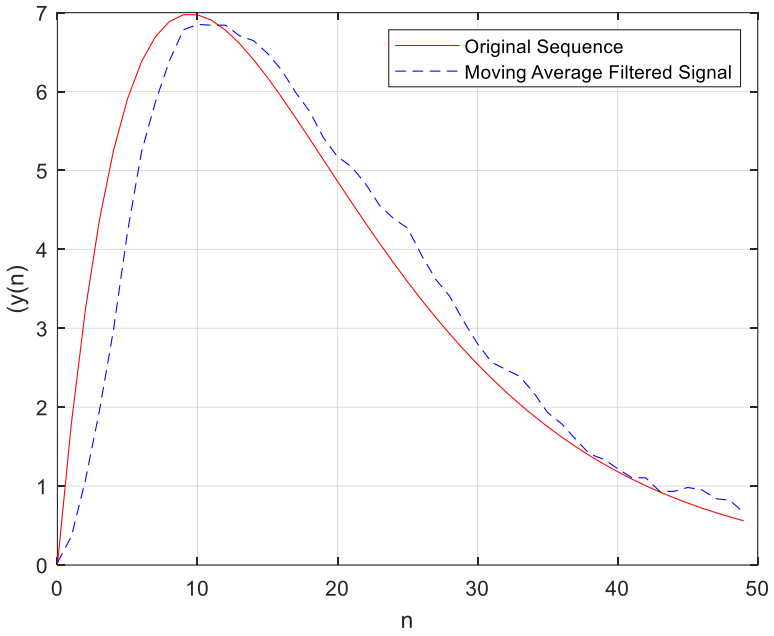
```
N=50;%number of samples
d=rand(N,1)-0.5;%noise with mean 0 and lies between -0.5 to 0.5
m=0:1:N-1;
s=2*m.*(0.9.^m);%original Sequence
x=s+d'; %noisy sequence
subplot(2,2,1)
stem(m,s)
title('Original Message')
subplot(2,2,2)
```

```
stem(m,d)
title('Random Noise')
subplot(2,2,3)
stem(m, x)
title('Noisy Sequence')
subplot(2,2,4)
plot(m, d, 'r-', m, s, 'b--', m, x, 'm-')
legend('noise','original message', 'noisy message')
figure
M=input('Value of M=');% Value of M from key board
b=ones(M, 1)/M;
y=filter(b,1,x);
plot(m,s,'r-',m,y,'b--')
legend('Original Sequence', 'Moving Average Filtered Signal')
grid on
xlabel('n')
ylabel('(y(n)')
```

The result of above code is shown in fig. 1.12 for $M = 4$ which is equivalent to curve smoothing technique of regression.



(a) Original and noisy sequences



(b) Original and filtered signal

Fig. 1.12 Curve smoothing using moving average filter

12. Now we will listen to the speech signal with noise removal.

```
load handel %original music signal
u=y(1:20000);
sound(u);
d=0.5*rand(length(u),1)-0.5;%noise with mean 0 and lies between -0.25 to 0.25
x=u+d; %noisy sequence
sound(x)
M=5;% Value of M
b=ones(M,1)/M;
z=2*filter(b,1,x);
sound(z)
subplot(3,1,1)
plot(u)
subplot(3,1,2)
plot(x)
subplot(3,1,3)
plot(z)
```

13. In this section we will deal with elimination of noise by moving average method. Let us first load a voice/music signal and add some noise with it. Finally we will filter the signal and observe the signals and corresponding spectrograms.

```
load handel %original signal
u=y(1:16000);
[num,den]=ellip(4,3,40,0.75,'high');
noise=filter(num,den,randn(length(u),1));
x=u+noise;
x=x/max(max(x));
M=5;% 5 sample will be averaged
b=ones(M,1)/M;
z=2*filter(b,1,x);
figure(1)
subplot(1,3,1)
```

specgram(u,[],Fs)
title('Original wave')
subplot(1,3,2)
specgram(x,[],Fs)
title('Noisy wave')
subplot(1,3,3)
specgram(z,[],Fs)
title('Filtered wave')
figure(2)
subplot(3,1,1)
plot(u)
title('Original wave')
subplot(3,1,2)
plot(x)
title('Noisy wave')
subplot(3,1,3)
plot(z)
title('Filtered wave')

Result of above code is shown in fig. 1.13.

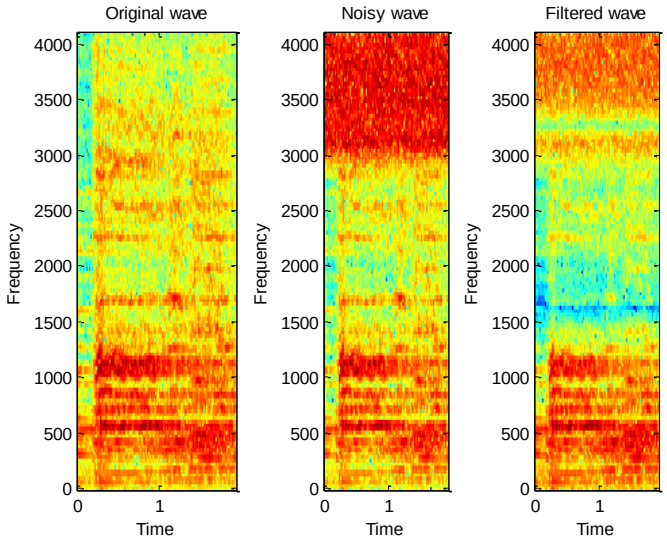


Fig.1.13 (a) Spectrogram of original, noisy and recovered signal

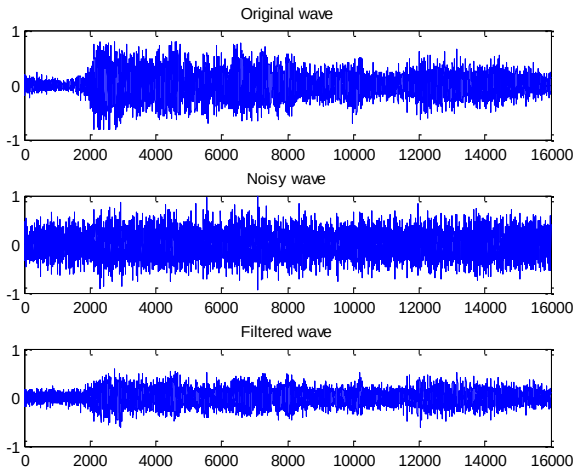


Fig. 1.13 (b) Plot of original, noisy and recovered signal

If your PC contains multimedia service, you can listen to them using,

sound(u,Fs); % original sound
sound(x,Fs); % Noisy sound
sound(z,Fs); % filtered sound

14. Determine *Z*-transform of *x(n) = kn* and *x(n) = cos(kn)*

```
syms k n
ztrans(k*n)
ans =
k*z/(z-1)^2
```

```
syms k n
ztrans(cos(k*n))
ans =
 (z-cos(k))*z/(z^2-2*z*cos(k)+1)
```

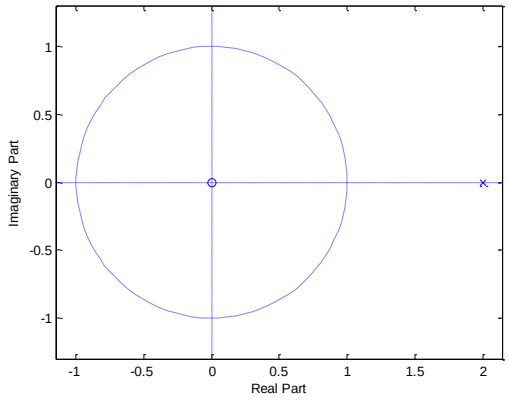
Find ztransform of $2^n U(n)$ and ROC

```
syms n
ztrans(2^n)

ans =

z/(z - 2)
```

```
[z,p,k]=tf2zp([1 0],[1 -2])
zplane(z, p);
```



syms z n
iztrans(z/(z-2))

ans =

2^n

15. Find inverse z-transform of $X(z) = \frac{kz}{(z-1)^2}$

syms k z
iztrans(k*z/(z-1)^2)

ans =
k*n

Example-1
Given, $X_1(z) = 1 + 2z^{-1} + 5z^{-2}$ and $X_1(z) = 1 + 3z^{-1} + 6z^{-2} + 8z^{-3}$. Determine $Y(z) = X_1(z).X_2(z)$
Ans.
$x_1(n) = \left\{1, 2, 5\right\}$ and $x_2(n) = \left\{1, 3, 6, 8\right\}$

x1=[1 2 5];
x2=[1 3 6 8];
y=conv(x1,x2)

y =

1 5 17 35 46 40

$\therefore X_1(z) = 1 + 5z^{-1} + 17z^{-2} + 35z^{-3} + 46z^{-4} + 40z^{-5}$

Find inverse z transform of $X_1(z) = 1 + 3z^{-1} + 6z^{-2} + 8z^{-3}$

x = filter([1 3 6 8], [1] ,[1 0 0 0])

x =

1 3 6 8

Find inverse z transform of

$$X(z)\frac{1 + 2z^{-1} + z^{-2}}{1 - z^{-1} + 0.3561z^{-2}}$$

b=[1 2 1];
a=[1 -1 0.3561];
z=filter(b,a,[1 0 0 0 0 0 0 0 0 0])

z =

1.0000 3.0000 3.6439 2.5756 1.2780 0.3608 -0.0943 -0.2228 -0.1892 -0.1099

16. Plot the transfer function, $F(z) = 1/(z^4+2z^3+5z^2+3z+2)$ in dB scale; where $z = x+jy$

clear
clf
x = -3:0.1:0;
y = -4:0.1:4.0;
[X,Y] = meshgrid(x,y);
p = X + j*Y;
den = [1 2 5 3 2];
s = polyval(den, p); %value of polynomial at p
h = surf(x, y, 20*log10(1./abs(s)));
xlabel('real')
ylabel('imag')
zlabel('magnitude (dB)')

Result of above code is shown in fig. 1.14

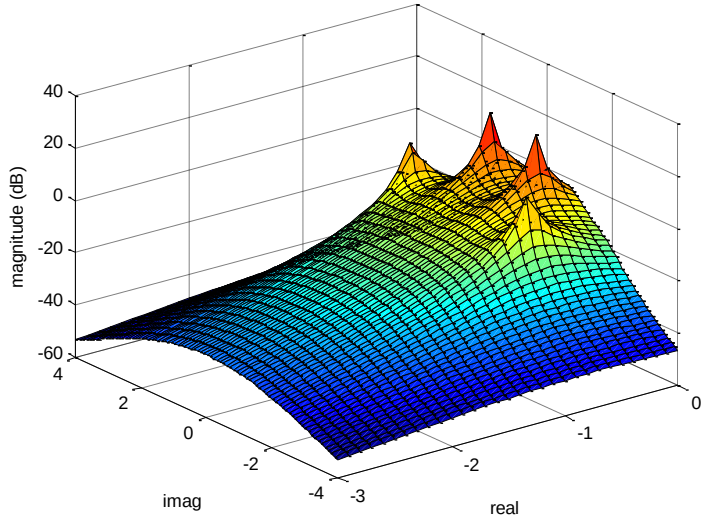


Fig.1.14 Plot of $F(z) = 1/(z^4+2z^3+5z^2+3z+2)$ in dB scale

Example-1

Verify the relation with first 10 terms.

$$(n-2)(0.5)^{n-2}\cos\left\{\frac{\pi}{3}(n-2)\right\}u(n-2) \leftrightarrow \frac{0.25z^{-3}-0.5z^{-4}+0.0625z^{-5}}{1-z^{-1}+0.75z^{-2}-0.25z^{-3}+0.0625z^{-4}}$$

```
n=0:9;
u2=[0 0 1 1 1 1 1 1 1 1];
xn=(n-2).*((0.5).^(n-2)).*cos((pi/3)*(n-2)).*u2; %LHS
b=[0,0,0,0.25,-0.5,0.0625];
a=[1, -1, 0.75,-0.25,0.0625];
z=filter(b,a,[1 0 0 0 0 0 0 0 0 0]); %RHS
```

```
>> xn
xn =
     0     0     0    0.2500   -0.2500   -0.3750   -0.1250    0.0781    0.0938    0.0273
>> z
z =
     0     0     0    0.2500   -0.2500   -0.3750   -0.1250    0.0781    0.0938    0.0273
```

Example-2

The ratio of two polynomial of z, can be expressed in terms of residues R_k, poles, p_k and coefficients of direct terns C_k like,

$$X(z)=\frac{b_0+b_1z^{-1}+b_2z^{-2}+...+b_Mz^{-M}}{a_0+a_1z^{-1}+a_2z^{-2}+...+a_Nz^{-N}}=\sum_{k=1}^{N}\frac{R_k}{1-p_kz^{-1}}+\sum_{\substack{k=0\\ When\quad M\geq N}}^{M-n}C_kz^{-k}$$

Determine residues, poles and coefficients of direct terms of,

$$X(z)=\frac{1+\sqrt{3}z^{-1}}{1-0.5z^{-1}+0.75z^{-2}+0.25z^{-3}}$$

```
b=[1,sqrt(3)];
a=[1, -0.5, 0.75, 0.25];
[R,P,C] = residuez (b,a); R is residues, P is the poles and C is coefficients of direct terms.
R =
   0.6583 - 0.9175i
   0.6583 + 0.9175i
  -0.3166
P =
   0.3815 + 0.8973i
   0.3815 - 0.8973i
  -0.2630
C =
     []
```

Example-2

Determine inverse z-transform of,

$X(z)=\frac{1}{\left(1-0.9z^{-1}\right)^2\left(1+0.9z^{-1}\right)}$ and verify the result.

```
b = 1; a = poly([0.9, 0.9,  -0.9]);
[R,P,C] = residuez (b,a);
R =


   0.2500
   0.2500 + 0.0000i
   0.5000 - 0.0000i
P =
```

-0.9000

0.9000 + 0.0000i

0.9000 - 0.0000i

C =

[]

$$\therefore X(z) = \frac{1}{\left(1-0.9z^{-1}\right)^2\left(1+0.9z^{-1}\right)} = \frac{0.25}{\left(1-0.9z^{-1}\right)} + \frac{0.5}{\left(1-0.9z^{-1}\right)^2} + \frac{0.25}{\left(1+0.9z^{-1}\right)}$$

$$= \frac{0.25}{\left(1-0.9z^{-1}\right)} + \frac{0.5z}{0.9}\frac{\left(0.5z^{-1}\right)}{\left(1-0.9z^{-1}\right)^2} + \frac{0.25}{\left(1+0.9z^{-1}\right)} \leftrightarrow 0.25(0.9)^n u(n) + \frac{5}{9}(n+1)(0.9)^{n+1}u(n+1) + 0.25(-0.9)^n u(n)$$

b = 1; a = poly([0.9, 0.9, -0.9]);

xz=filter(b,a,[1 0 0 0 0 0 0 0 0 0]); %RHS

n=0:9;

u=[1 1 1 1 1 1 1 1 1 1];

xn=0.25*((0.9).^n).*u+(5/9)*(n+1).*((0.9).^(n+1)).*u+0.25*((-0.9).^n).*u;

%srart from n = 0 gives the LHS

xz =

1.0000 0.9000 1.6200 1.4580 1.9683 1.7715 2.1258 1.9132 2.1523 1.9371

xn =

1.0000 0.9000 1.6200 1.4580 1.9683 1.7715 2.1258 1.9132 2.1523 1.9371

17. Determine pole-zero diagram of a system with zeros, *Z* = 1 + 2j, -j and poles *P* = 1, 1-j, 5 + 5j;

% vector for zeros

z = [1+2j ; -j];

% vector for poles

p = [1 ; 1-j ; 5+5j];

zplane(z, p);

title('Pole/Zero Plot ');

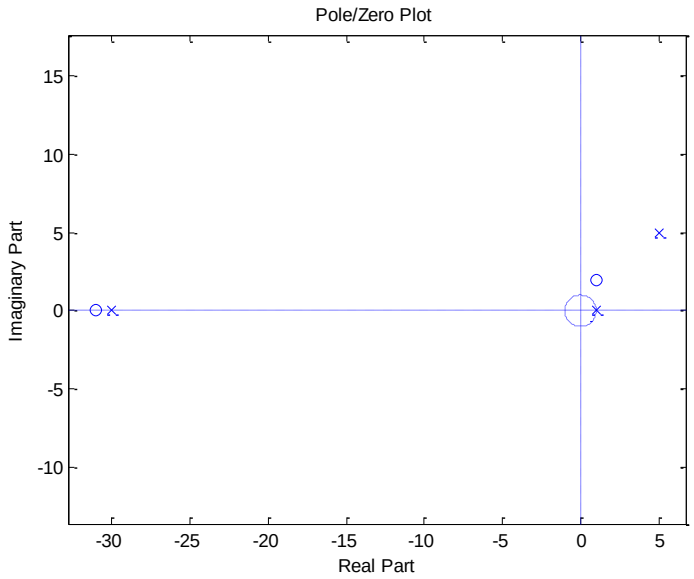


Fig.1.15 Pole zero plot

18. Let us now determine root locus, bode diagram, step response and impulse response of open or closed loop systems in 's' domain of Laplace transform. Such analysis is done in determination of system stability. This part of the experiment is taken based on concept of, A. J. Chipperfield, P. J. Fleming and C. M. Fonseca, 'MATLAB toolboxes an application for control,' Peter Peregrinus Ltd.

(a) Find root locus of the feedback system of fig. 1.16 for $K = 2$.

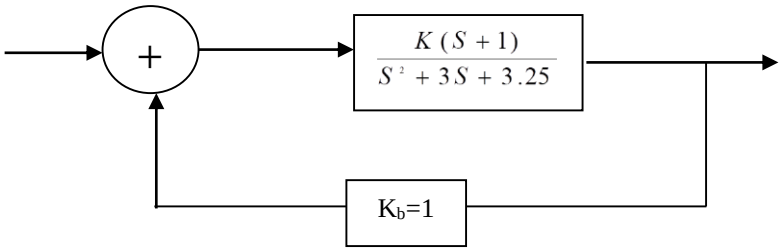


Fig. 1.16 A feed back system in S domain

%Source code

```
K=2;
h1=tf(K, [1.0  3.0   3.25]);
h2=tf([1  1], 1);
```

dcm=feedback(h1*h2,1,1); % h1*h2 is system-1, Kb=1 system-2 and +1 is addition

rlocus(dcm,'k')

The result of the code is shown in fig. 1.17

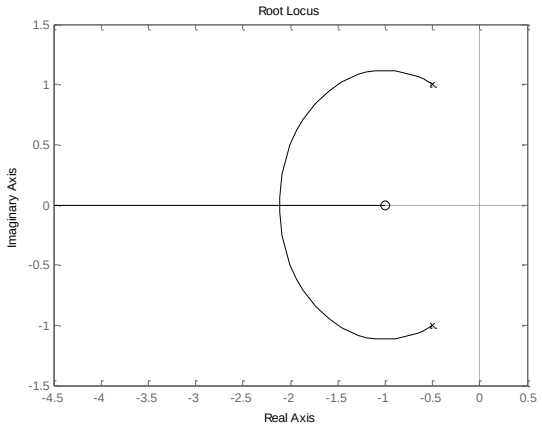


Fig. 1.17 Root locus of fig. 1.16

(b) Find bode diagram of a system of transfer function of, $G(s) = \dfrac{4}{S(S+1)(S+2)}$

num=4;
den=conv([1 0],conv([1,1],[1,2]));
Bode(num,den)

Result of above code is shown in fig. 1.18

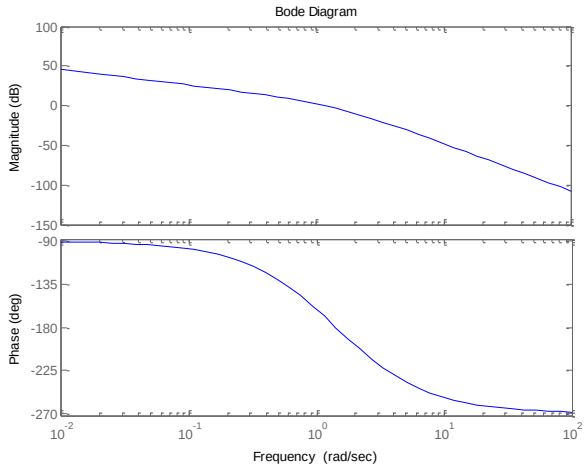


Fig. 1.18 Bode diagram of above transfer function

(c) Determine step response of the following feedback system of fig. 1.19.

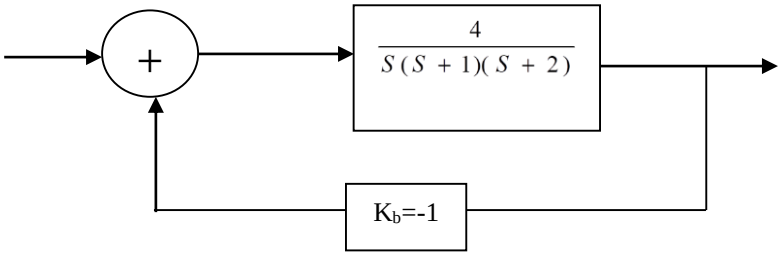


Fig. 1.19 A feedback system

```
num=4;
den=conv([1  0],conv([1,1],[1,2]));
[N, D]=cloop(num, den, -1);
t=0:0.005:25;
step(N, D, t)
```

Result of above code is shown in fig. 1.20.

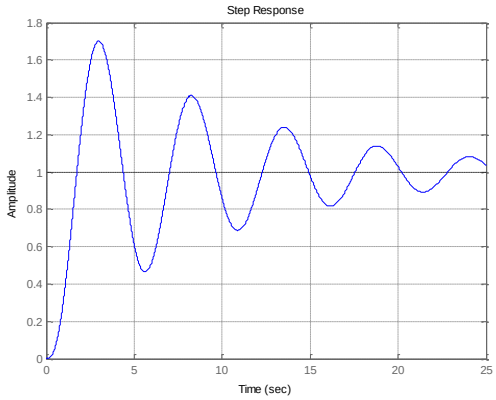


Fig. 1.20 Step response of fig. 1.19

(d) Determine impulse response of the system.

impulse(N, D, t)

Result of above code is shown in fig. 1.21.

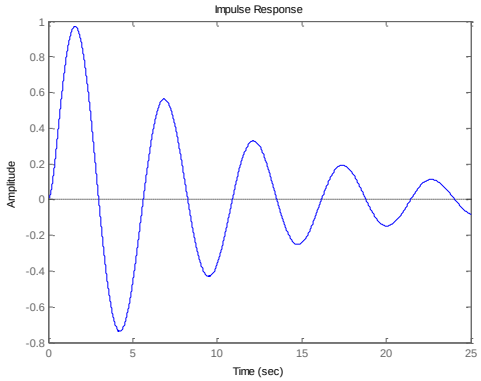


Fig. 1.21 Impulse response of fig. 1.19

Experiment No. 02
Discrete Time Fourier Transform (DTFT)

Basic Theory:

A continuous time signal *f(t)* called analog signal has a finite amplitude at every instant *t*. A discrete time signal *x(n)* called sequence is defined at discrete instants *n* but *x(n)* is undefined at any instant between *n+k* and *n+k+1*. A digital signal is one for which both time and amplitude axis is discrete. Fourier transform is applicable for a discrete time sequence *x(n)* in a different form called Discrete Fourier Transform (DFT).

For a sequence *x(n)* in *n*-domain, its DFT will transform it in *m*-domain resembles to transformation of time to frequency domain. DFT of *x(n)* of infinite length is expressed like,

$X(m\omega) = X(m) = \sum_{n=-\infty}^{\infty} x(n)e^{-jm\omega n} = \sum_{n=-\infty}^{\infty} x(n)e^{-jm(2\pi/N)n}$; Which resembles to continuous time Fourier transform of, $X(\omega) = \int_{-\infty}^{\infty} x(t)e^{-j\omega t}dt$.

A comparison between continuous and discrete time Fourier transform is shown in table-2.1.

Table-2.1

Continuous FT	Discrete FT
X(ω)	X(m)
$\int_{-\infty}^{\infty}$	$\sum_{n=-\infty}^{\infty}$

t	n
ω	mω or m
x(t)	x(n)
T	N

Now the inverse operation i.e. IDFT can be expressed like,

$$x(n) = Lt_{N\to\infty}\frac{1}{N}\sum_{m=-\infty}^{\infty}X(m)e^{jm\omega n} = Lt_{N\to\infty}\frac{1}{N}\sum_{n=-\infty}^{\infty}X(m)e^{jm(2\pi/N)n}$$

For a finite length sequence of length N over 0≤n≤N-1, DFT and IDFT is expressed like,

$$X(m\omega) = X(m) = \sum_{n=0}^{N-1}x(n)e^{-jm\omega n} = \sum_{n=0}^{N-1}x(n)e^{-jm(2\pi/N)n} \text{ and}$$

$$x(n) = \frac{1}{N}\sum_{m=0}^{N-1}X(m)e^{jm\omega n} = \frac{1}{N}\sum_{n=0}^{N-1}X(m)e^{jm(2\pi/N)n}$$

1. Consider a continuous time signal, $x(t) = \sin(2\pi 1000t) + \frac{1}{2}\sin(2\pi 2000t + \frac{3\pi}{4})$.

Determine sampled signal *x(n.Ts) = x(n)*, using Matlab, taking *N* = 8 samples at sampling rate, *Fs* = 8000Hz (samples/sec).

The sampled signal,

$x_s(n) = x(nT_s) = \sin(2\pi 1000nT_s) + \frac{1}{2}\sin(2\pi 2000nT_s + \frac{3\pi}{4})$; where *Ts=1/Fs*=1/8000sec is the sampling period. Let us use Matlab code to determine sampled sequence *x(n)*.

```
n=0:1:7;
Ts=1/8000;
xn=sin(2*pi*1000*n*Ts)+0.5*sin(2*pi*2000*n*Ts+3*pi/4);
xn
```

xn =

0.3536 0.3536 0.6464 1.0607 0.3536 -1.0607 -1.3536 -0.3536

Therefore,

x(0)= *0.3536*

x(1)= *0.3536*

x(2)= *0.6464*

x(3)= *1.0607*

x(4)= *0.3536*

x(5)= *-1.0607*

x(6)= *-1.3536*

x(7)= -0.3536

2. Determine DFT of above sequence using Matlab.

X=fft(xn)

X =

X(0)=0.0000

X(1)=-0.0000 - 4.0000i

X(2)=1.4142 + 1.4142i

X(3)=-0.0000 + 0.0000i

X(4)=-0.0000

X(5)=-0.0000 - 0.0000i

X(6)=1.4142 - 1.4142i

X(7)=-0.0000 + 4.0000i

IDFT of above sequence will retrieve the original sequence *x(n)* like,

Y=ifft(X)

Y =

0.3536 0.3536 0.6464 1.0607 0.3536 -1.0607 -1.3536 -0.3536

3. If $x(n) \leftrightarrow X(m)$; both *x(n)* and *X(m)* are the vector of length *N* then their relation can be expressed in a different way like,

$X = \mathbf{D}_N x$

Where $\boldsymbol{x} = [x(0)\ x(1)\ x(2)\ \ldots\ \ldots\ \ldots\ x(N-1)]^T$,

$X = [X(0)\ X(1)\ X(2)\ \ldots\ \ldots\ \ldots\ X(N-1)]^T$,

$$\mathbf{D_N} = \begin{bmatrix} 1 & 1 & 1 & \cdots & 1 \\ 1 & W_N & W_N^2 & \cdots & W_N^{N-1} \\ 1 & W_N^2 & W_N^4 & \cdots & W_N^{2(N-1)} \\ \vdots & \vdots & \vdots & \cdots & \cdots \\ 1 & W_N^{N-1} & W_N^{2(N-1)} & \vdots & W_N^{(N-1)(N-1)} \end{bmatrix} \text{ and } W_N = e^{-j2\pi/N}$$

Determine the $\mathbf{D}_N$ matrix of dimension of 4×4 using Matlab.

D=dftmtx(4)

D =

1.0000	1.0000	1.0000	1.0000
1.0000	0 - 1.0000i	-1.0000	0 + 1.0000i
1.0000	-1.0000	1.0000	-1.0000
1.0000	0 + 1.0000i	-1.0000	0 - 1.0000i

4. Write Matlab code to determine DFT of *x (n)* = {1, 0, 0, 1} hence show the plot of *X(m)*.

```
x=[1 0 0 1];
y=fft(x);
subplot(2,1,1)
stem(abs(y), 'k')
xlabel('m')
ylabel('X(m)')
title('Absolute value of DFT sequence')
subplot(2,1,2)
stem(angle(y), 'k')
xlabel('m')
ylabel('Angle(X(m))')
title('Angle of DFT sequence')
```

The result of above code is shown in fig. 2.1.

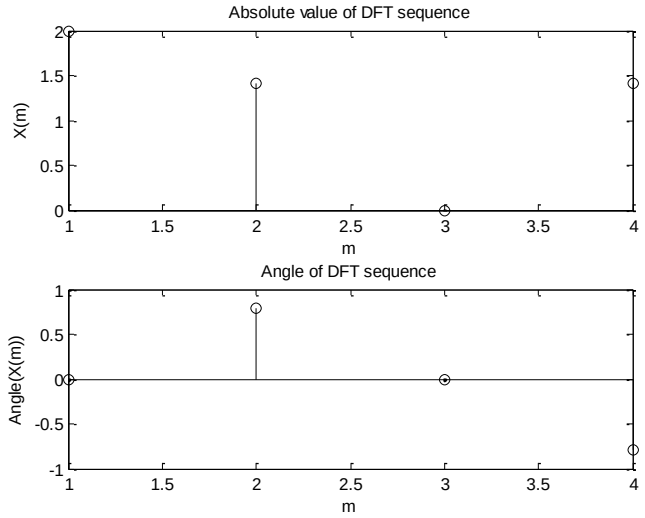
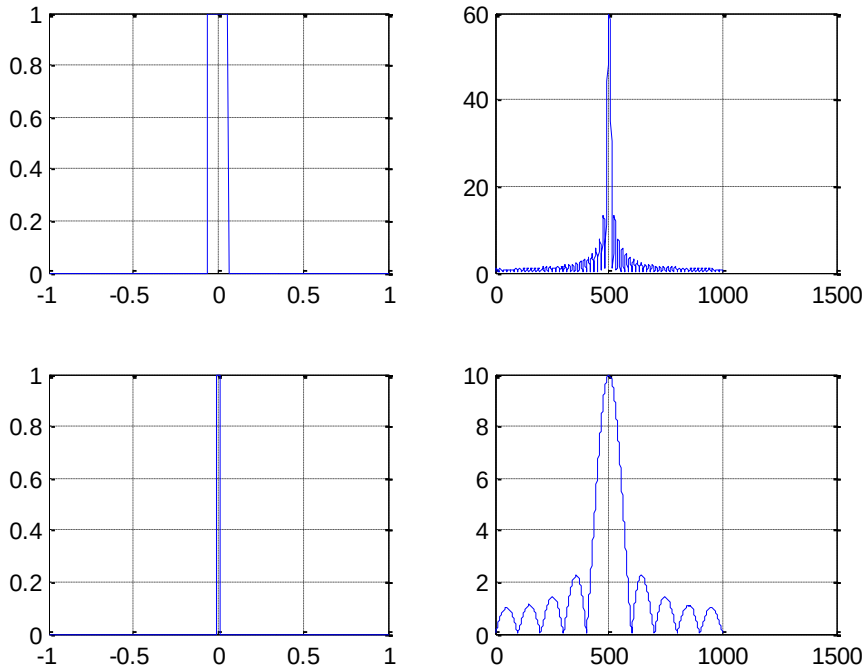


Fig. 2.1 Plot of $|X(m)|$ and $\angle X(m)$

DTFT on a rectangular pulse

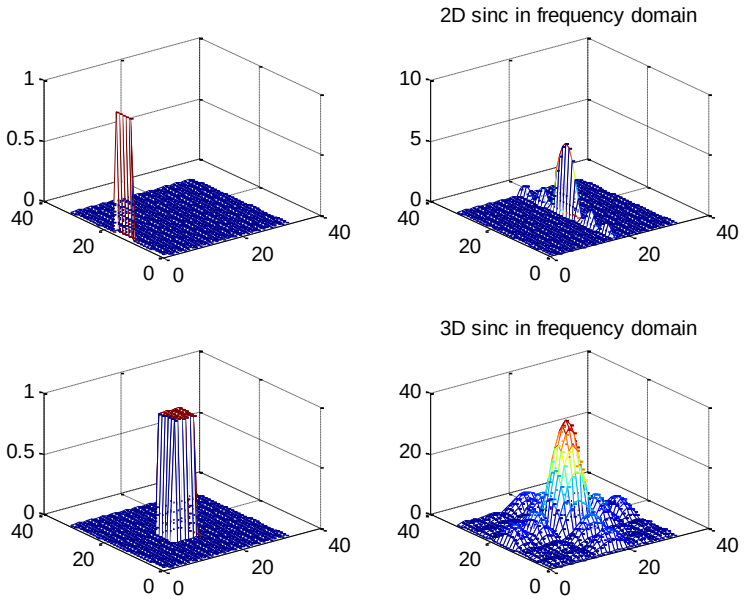
```
fs = 500;
t = -1:1/fs:1;
x = rectpuls(t,0.12);
subplot(2,2,1)
plot(t, x)
grid on
y = fft(x);
y = fftshift(y);
subplot(2,2,2)
plot( abs(y))
grid on
x = rectpuls(t,0.02);
subplot(2,2,3)
plot(t, x)
grid on
y = fft(x);
y = fftshift(y);
subplot(2,2,4)
plot( abs(y))
grid on
```



DTFT on a 3D rectangular pulse

```
x=zeros(32);
x(12:17)=ones(6,1);
subplot(2,2,1)
title('2D rectangular pulse')
mesh(x)
x=fft(x);
x=fftshift(x);%Dc avlue is at the corner of the array
%let us move it at the middle
subplot(2,2,2)
mesh(abs(x))
title('2D sinc in frequency domain')
```

```
x=zeros(32);
x(12:17,12:17)=ones(6);
subplot(2,2,3)
title('3D rectangular pulse')
mesh(x)
x=fft2(x);
x=fftshift(x);%Dc avlue is at the corner of the array
%let us move it at the middle
subplot(2,2,4)
mesh(abs(x))
title('3D sinc in frequency domain')
```



Experiment No. 03

Design and Simulation of Different Types of Infinite Impulse Response (IIR) Filters

Objective:

Infinite Impulse Response (IIIR) filter is actually the analog equivalent filter in digital domain. Here infinite number of filter coefficients has to be taken to simulate the transfer function of the filter; the situation is resolved introducing feed back with the filter. In this experiment Butterworth, Chebyshev and elliptic filters are designed and response curves are determined based on Matlab.

1. Design a 10th order band pass (BP) Butterworth filter with a passband from 100 to 200 Hz and plot both impulse response and frequency response.

```
Fs=1000;% Fs=1000 i.e. sampling frequency
n=10; %order of the filter
Wn=[100 200]/500;  %normalized by Fs/2
% Normalized frequency Wn can vary between 0 and 1
%Normalization is done by Fs/2, here Wn = 0.2  to  0.4
[b,a]=butter(n,Wn);
figure(1)
[y,t]=impz(b,a,101);  % 101 samples of impulse response
plot(t,y)
title('Impulse Response')
grid on
figure(2)
freqz(b,a,512, Fs)
title('Frequency Response')
```

The results of above code are shown in fig. 3.1.

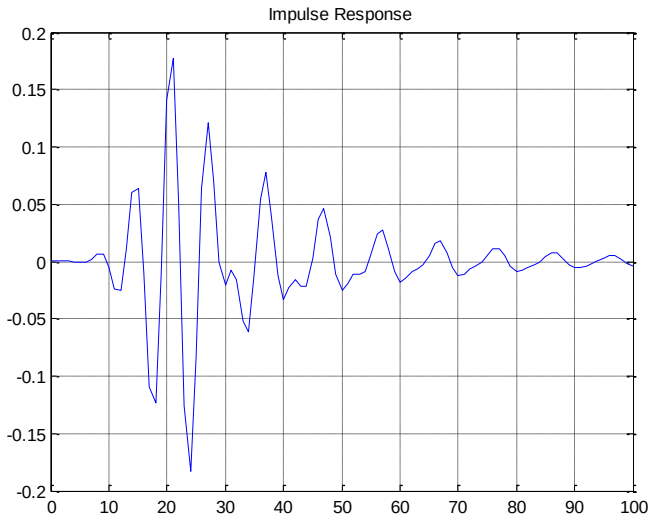


Fig. 3.1 (a) Impulse response of a 10th order BP Butterworth filter

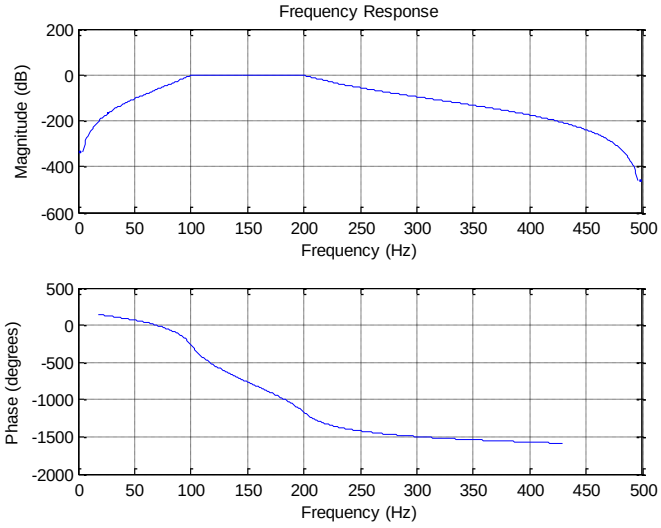


Fig. 3.1(b) Frequency response of a 10th order BP Butterworth filter

Let us now determine the step response of Butterworth digital filters.

Fs=1000;% Fs=1000 i.e. sampling frequency

n=10; %order of the filter

Wn=[100 200]/500; %normalized by Fs/2

[b,a]=butter(n,Wn);

stepz(b,a)

grid on

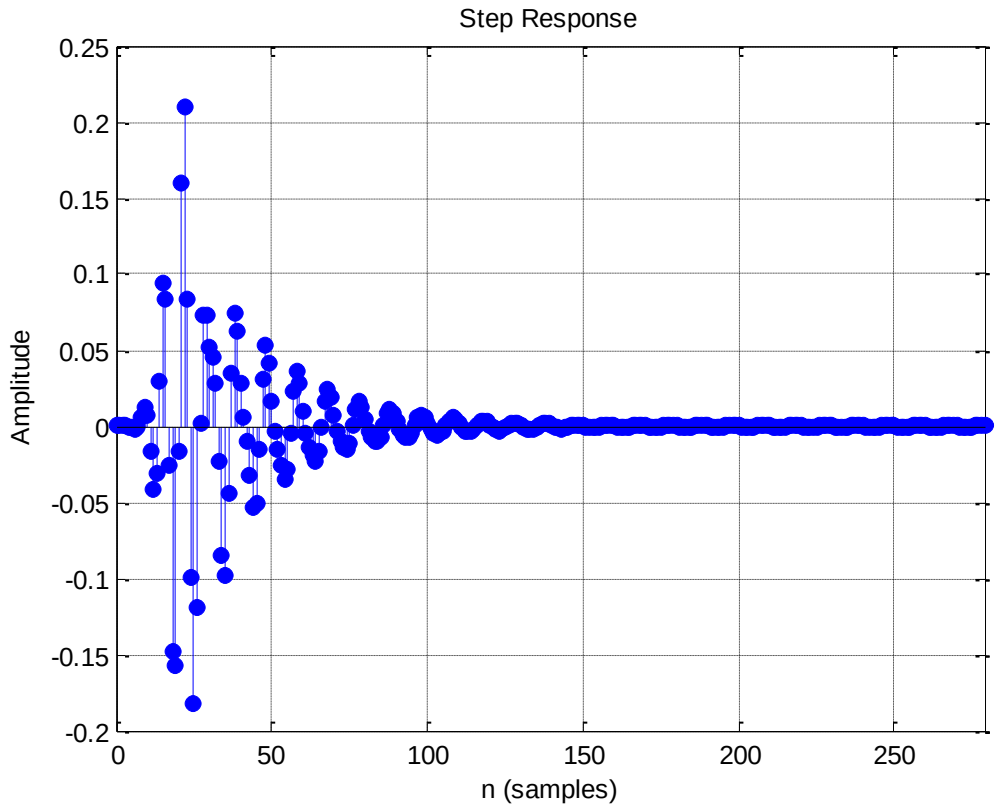


Fig. 3.1 (c)Step response of above filter

2. Design a 10th order BP Chebyshev filter of type-I with a passband from 100 to 200 Hz and plot the impulse response and frequency response. Consider 25dB ripple in passband.

Fs=1000;
n=10;
Rp=25;
Wn=[100 200]/(Fs/2);
[b,a]=cheby1(n,Rp,Wn); % For stop band [b,a]=cheby1(n,Rp,Wn,'stop');
[y,t]=impz(b,a,101);
figure(1)
plot(t,y)

grid on
title('Impulse Response')
figure(2)
freqz(b, a, 512, Fs)
title('Frequency Response')

The results of above code are shown in fig. 3.2.

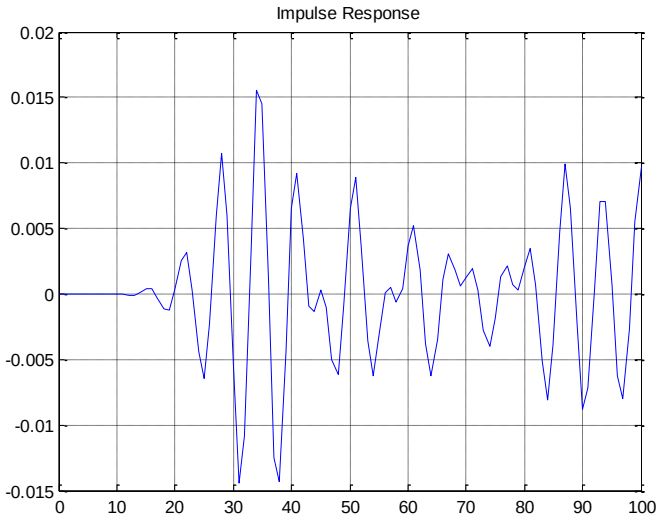


Fig. 3.2 (a) Impulse response of a 10th order BP Chebyshev filter

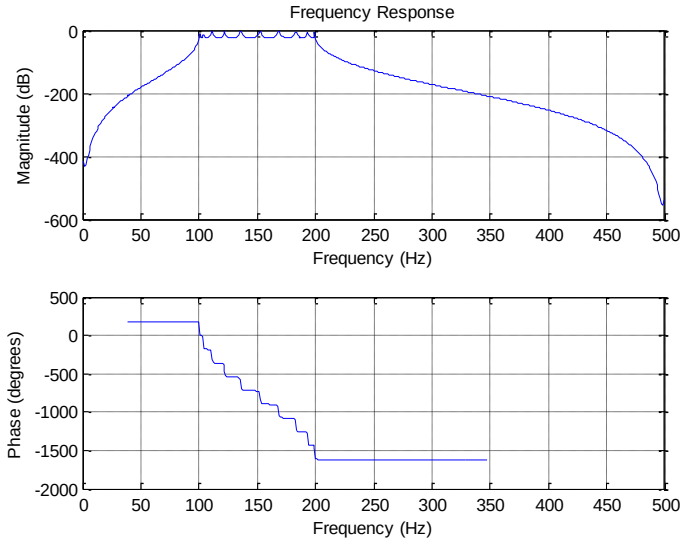


Fig. 3.2 (b) Frequency response of a 10th order BP Chebyshev filter

3. Design a 10th order BP elliptic filter with passband in the range of 100 to 200 Hz. Plot the frequency response and impulse response of the filter. Consider 5dB ripple in passband and 20dB attenuation in stopband.

Fs=1000; %Sampling Frequency

```
n=10;
Rp=5;
Rs=20; % Rs dB down from peak value in pass-band
Wn=[100 200]/(Fs/2);
[b,a]=ellip(n,Rp,Rs,Wn);
figure(1);
freqz(b,a,512,Fs); %512 samples
title('Frequency Response')
figure(2);
[y,t]=impz(b, a, 500);  % here 500 means number of samples
plot(t,y)
grid on
title('Impulse response of Elliptic filter')
```

The results of above code are shown in fig. 3.3.

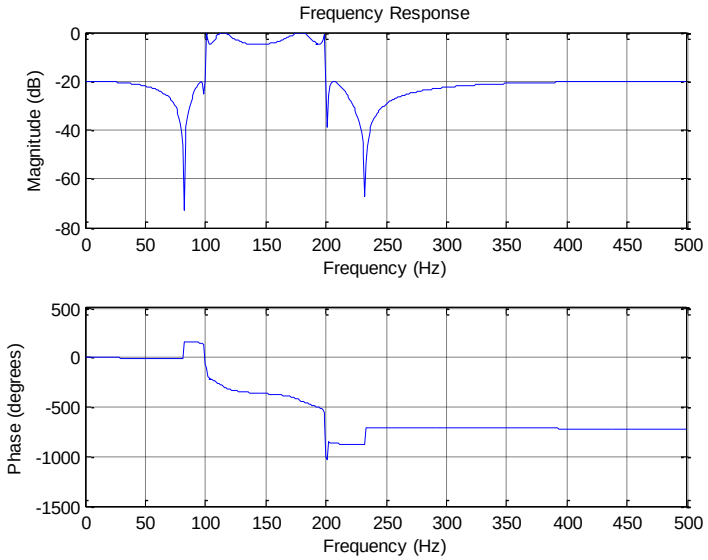


Fig. 3.3(a) Frequency response of a 10th order BP elliptic filter

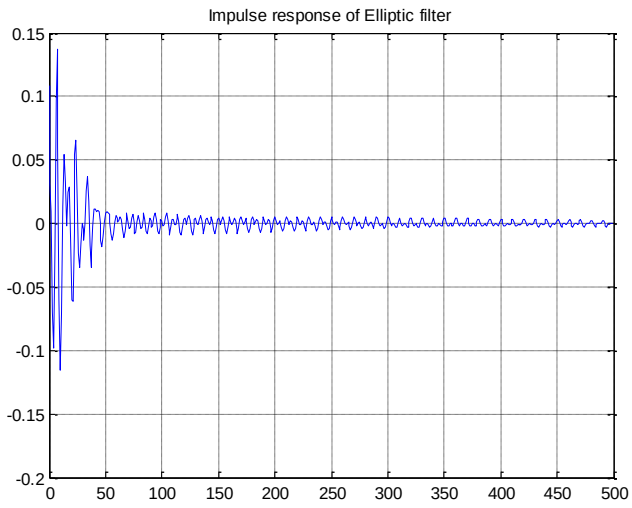


Fig. 3.3(b) Impulse response of a 10th order BP elliptic filter

4. A noisy sinusoidal signal is filtered with Butterworth filter finally filtered signal is compared with original signal

```
t=0:0.01:1;
Fs=1000;
y=sin(8*pi*t); % original signal
yn=y+0.5*rand(1,length(t)); % signal with noise
subplot(2,2,1);
plot(t,y,'k')
title('Original signal')
subplot(2,2,2);
plot(t,yn,'r')
title('Noisy signal')
[b,a]=butter(2, 200/Fs);
z=filter(b,a,yn);
subplot(2,2,3);
plot(t,z,'b')
title('Filtered signal')
subplot(2,2,4)
plot(t, y, 'm', t, z, 'k');
title('Comparison of original and filtered signal')
```

The results of above code are shown in fig. 3.4.

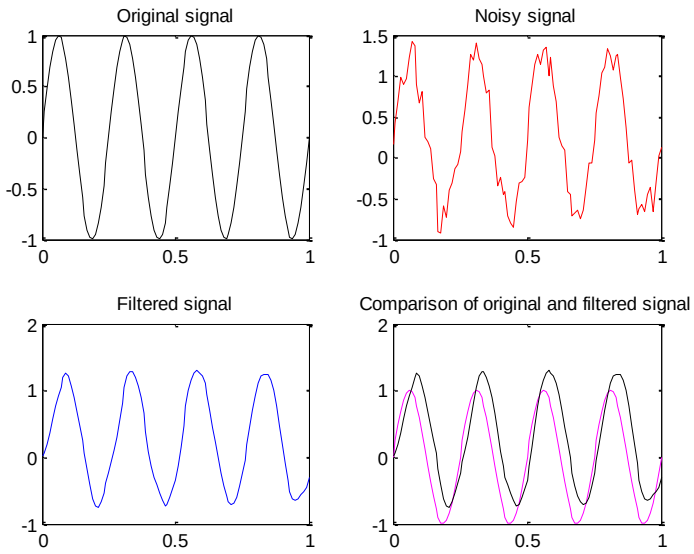


Fig.3.4 Performance of the filter of section-4

5. Objective of this section is to eliminate cos10πt term from the signal, *x(t)* = 1+cos2πt+cos10πt; using LP Butterworth filter. Here the highest frequency f_h = 5Hz, therefore the sampling frequency, $Fs \geq 2\times5 = 10$Hz. Let us take the sampling frequency, F_s = 50Hz and the corresponding sampling period $T = 1/F_s$ sec. We will consider a Butterworth filter of cutoff frequency f_c = 2Hz to filter the signal component of 5Hz.

```
n = 0:1:200;
Fs=50;
T=1/Fs;
x=1+cos(2*pi*n*T)+cos(10*pi*n*T); % original signal
subplot(2,1,1)
stem(n,x)
title('Original signal')
fc=2; %pass band edge freqency
[b,a]=butter(10, fc/(Fs/2));
z=filter(b,a,x);
subplot(2,1,2)
stem(n,z)
title('Filtered signal')
```

The results of above code are shown in fig.3.5.

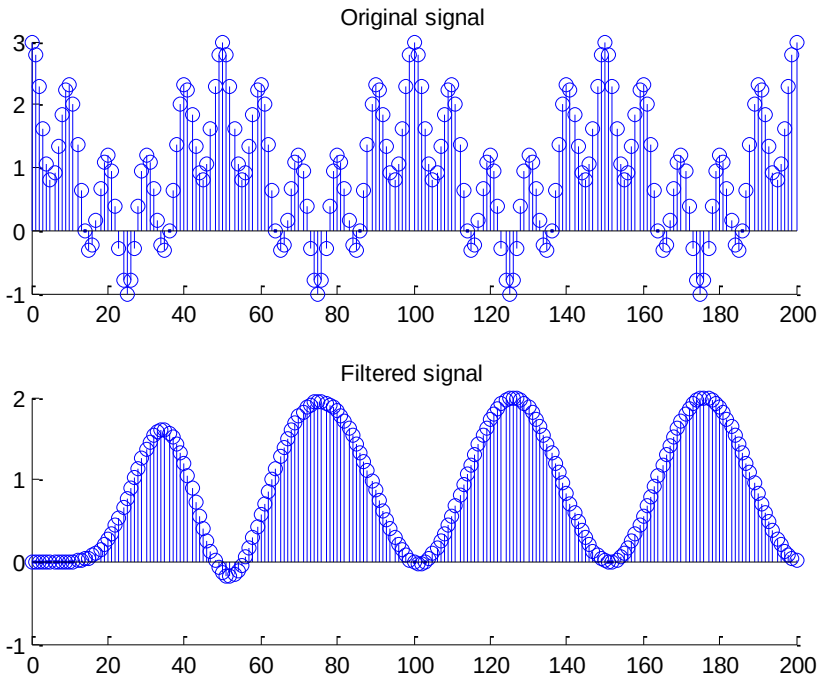


Fig.3.5 Filtering of 5Hz sinusoidal signal

Next we will consider a Butterworth filter of cutoff frequency f_c = 3 to 8 Hz to filter the DC and signal component of 2Hz taking F_s = 100Hz.

```
n = 0:1:200;
Fs=100;
T=1/Fs;
x=1+cos(2*pi*n*T)+cos(10*pi*n*T); % original signal
subplot(2,1,1)
stem(n,x)
title('Original signal')
fc=[3  12]; %pass band edge freqency
[b,a]=butter(10, fc/(Fs/2));
z=filter(b,a,x);
subplot(2,1,2)
stem(n,z)
title('Filtered signal')
```

The results of above code are shown in fig.3.6.

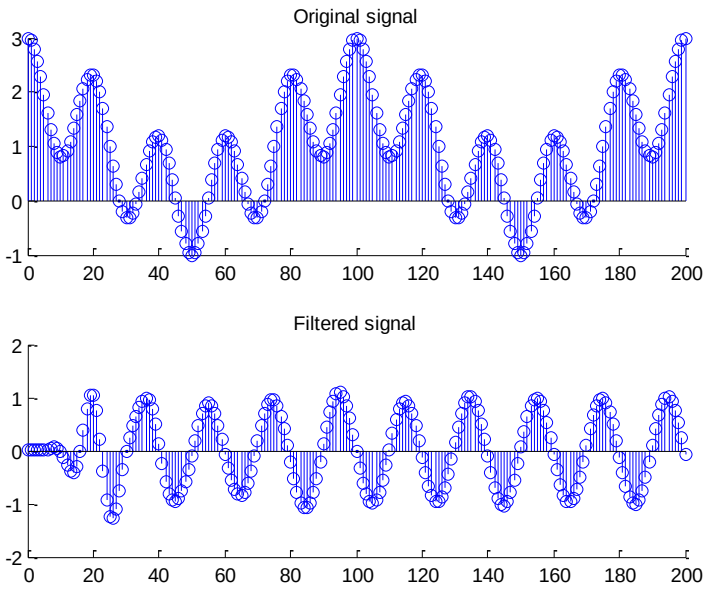


Fig.3.6 Filtering of DC and 2Hz sinusoidal signal

6. A bandpass IIR filer has the transfer function of, $H(z) = \frac{z^2 - 1}{z^2 + 0.877}$; determine poles and zeros of the filter and frequency response taking sampling frequency *Fs* = 500Hz.

[P,Z]=pzmap([1 0 -1],[1 0 0.877])

P =

0 + 0.9365i

0 - 0.9365i

Z =

-1

1

pzmap ([1 0 -1],[1 0 0.877])

The result of above code is shown in fig. 3.7(a).

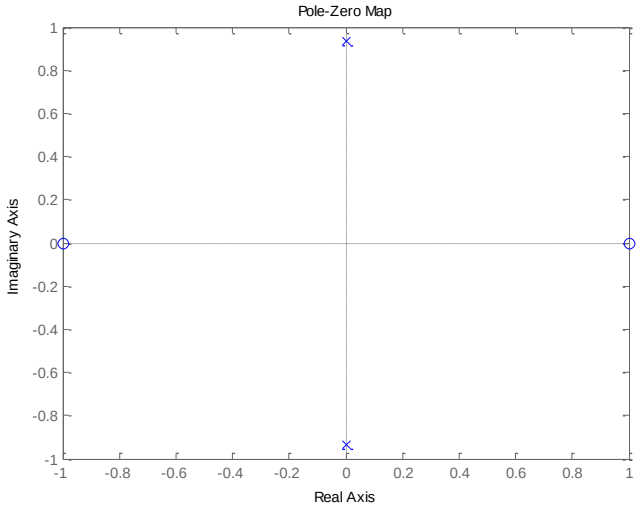


Fig. 3.7(a) Pole-zero diagram of the filter

The transfer function of the filter, $H(z) = \dfrac{z^2 - 1}{z^2 + 0.877} = \dfrac{1 - z^{-2}}{1 + 0.877 z^{-2}}$.

b=[1, 0, 1]; %numerator

a=[1, 0, 0.877];% denominator

Fs=500;

freqz(b, a, 512, Fs); %512 samples

The result of above code is shown in fig. 3.7(b).

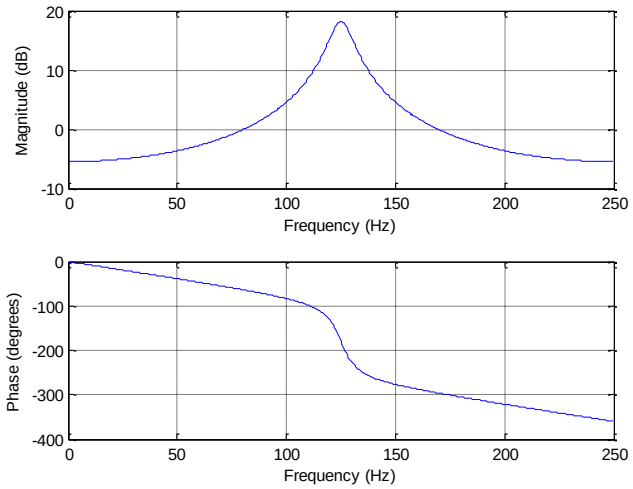


Fig. 3.7(b) Frequency response of the filter

Observe the frequency response of, $H(z)=\frac{1-z^{-2}}{1-0.877z^{-2}}$ and comment on the result.

b=[1, 0, 1]; %numerator

a=[1, 0, -0.877];% denominator

Fs=500;

freqz(b, a, 512, Fs); %512 samples

The frequency response of the filter is shown in fig. 3.7(c).

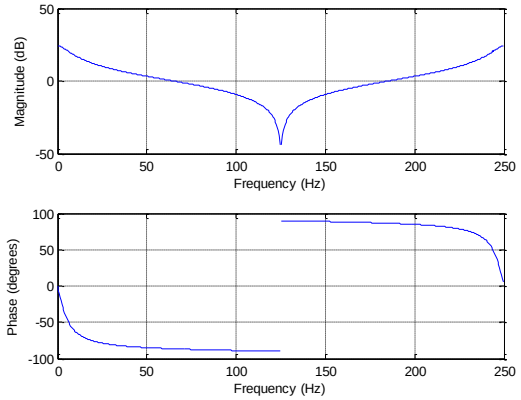


Fig. 3.7(c) Frequency response of the filter

It is a notch filter visualized from the frequency response curve.

7. In this section we will consider about bilinear transformation where the *s*-domain transfer function $H_a(s)$ of an analog filter is converted to its equivalent transfer function $H(z)$ in z-domain of a digital filter. The transformation is expressed as,

$$H(z)=H_a(s)\big|_{s=2F_s\frac{1-z^{-1}}{1+z^{-1}}}$$; Where $Fs = 1/T$ is the sampling frequency

Convert the transfer function, $H_a(s)=\frac{s+1}{s^2+5s+6}$ to its equivalent $H(z)$ using bilinear transformation taking $T = 1$.

Ans. $H(z)=H_a(s)\big|_{s=2F_s\frac{1-z^{-1}}{1+z^{-1}}}=H(z)=H_a\left(\frac{2}{T}\frac{1-z^{-1}}{1+z^{-1}}\right)=\frac{0.15+0.1z^{-1}-0.05z^{-2}}{1+0.2z^{-1}}$

Let us implement the result using Matlab code.

c=[1,1];

d=[1,5,6];

T=1; Fs=1/T;

[b,a]=bilinear(c,d,Fs)

The result of the code:

b =

0.1500 0.1000 -0.0500

a =

1.0000 0.2000 -0.0000

Let us now observe the frequency response of the filter,

freqz(b, a, 512, Fs); %512 samples

The result of above code is shown in fig. 3.7.

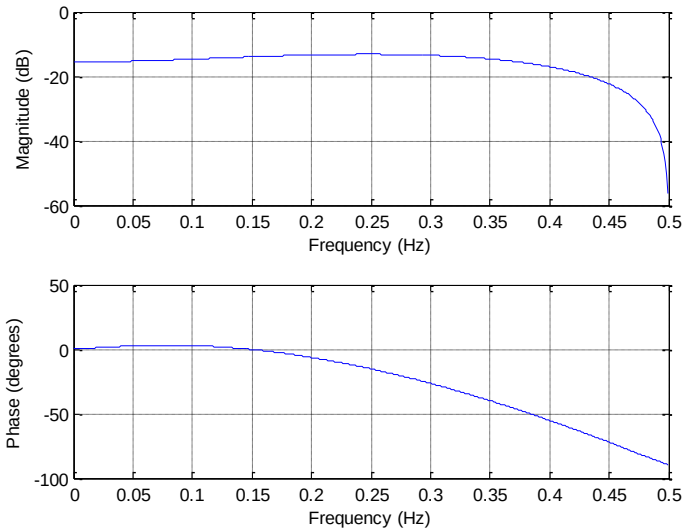


Fig.3.7 Frequency response of the filter

Consider the two pole analog filter,

$$H(s) = \frac{\omega_c^2}{s^2 + \sqrt{2}\omega_c s + \omega_c^2}$$; ω_c = 2 and T = 0.2;

Putting, $s = \frac{z-1}{z+1}.\frac{2}{T}$

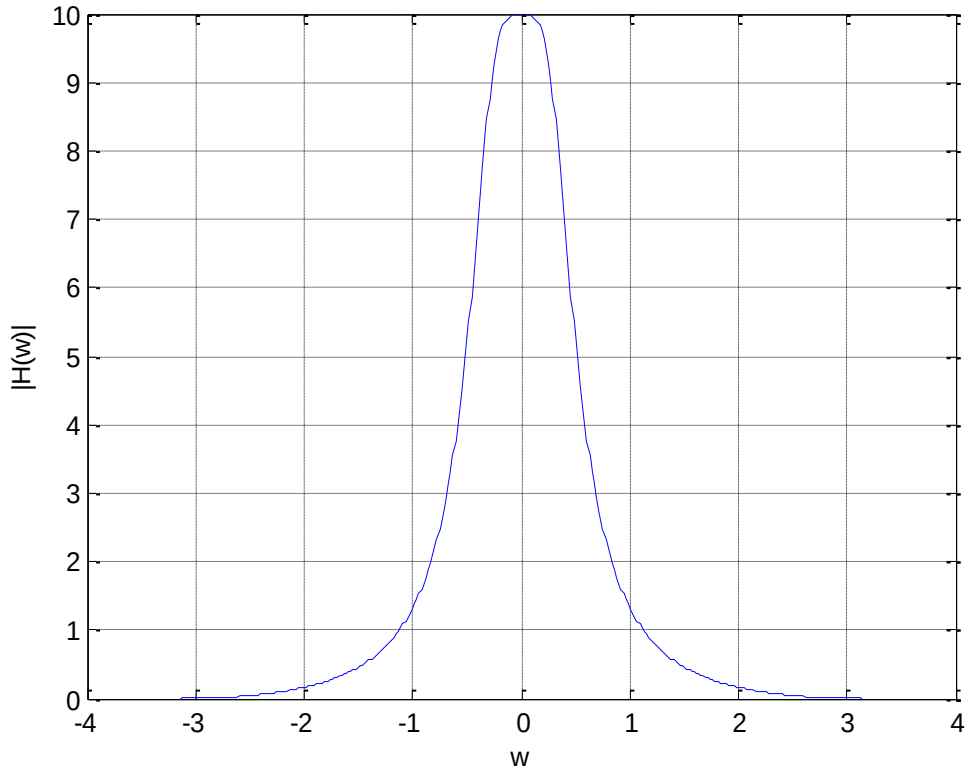
$$H(z) = \frac{0.0302\left(z^2 + 2z + 1\right)}{z^2 - 1.451z + 0.5724}$$

Let us plot the magnitude response if the filter.

w=-pi:2*pi/300:pi;

z=exp(j*w);

```
H=0.302*(z.^2+2*z+1)./(z.^2-1.4514*z+0.5724);
plot(w,abs(H))
xlabel('w')
ylabel('|H(w)|')
grid on
```



Questions:

1. Mention three applications of digital filter.
2. Is it possible to achieve linear phase response from a digital filter?
3. Which parameters are essential to design a digital filter?
4. If IIR filters are characterized by, $y(n)=\sum_{k=0}^{N} a_k x(n-k)+\sum_{k=1}^{M} b_k y(n-k)$; show that the transfer function of the filter, $H(z)=\dfrac{\sum_{k=0}^{N} a_k z^{-k}}{1+\sum_{k=1}^{M} b_k z^{-k}}$.
5. Distinguish between analog and digital filter.

Experiment No. 04

Design and Implementation of Finite Impulse Response (FIR) Filter

Objective:

FIR filters can provide linear phase response, sharp cutoff and stability at the expense of process time and memory storage. Such filter requires more coefficients than IIR filter. The length of the ideal impulse response $h_D(n)$ of a FIR filter is infinite. Sharp truncation of elements of $h_D(n)$ beyond certain limit will generate ripples can be explained based on Gibb's phenomenon. In this experiment FIR filter is designed based on window method where ideal impulse response of the filter $h_D(n)$ is multiplied by a smooth window function $w(n)$ of finite duration to alleviate excess ripples.

1. Plot Hamming, Hann and Kaiser window functions used in design of a FIR filter.

```
beta=5.2;
N=20;
n=1:1:20;
y=hamming(N);
y1=hann(N);
y2=kaiser(N,beta);
plot(n,y,'k^-',n,y1,'kd:',n,y2,'k*:')
xlabel('n')
ylabel('h(n)')
legend('Hamming','Hann','Kaiser')
grid on
```

The result of above code is shown in fig. 4.1.

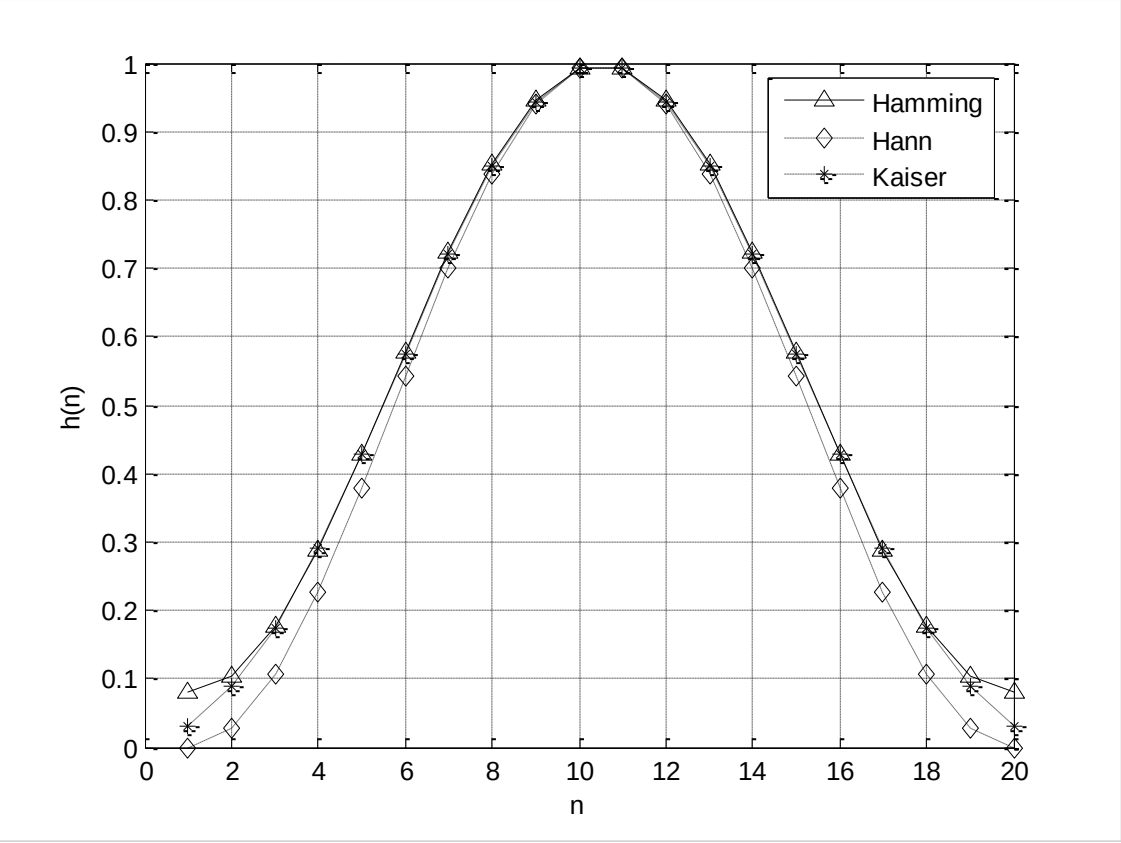


Fig. 4.1 Window functions

Let us observe the impact of window function in frequency response of a LP FIR filter.

```
N=25; %Range of sample
n=-N:1:N;
fc=0.1930;%cutoff frequency of LP filter
hd=sinc(2*fc*n);%impulse response of ideal LP filter
subplot(2,2,1)
stem(n,hd)
xlabel('n')
ylabel('hD')
title('Impulse Response of Ideal LP FIR Filter')
grid on

f=n*fc/N;%Normalized frequency
subplot(2,2,2)
HD=fft(hd);
```

```
HD=fftshift(HD);
plot(f,abs(HD)/max(abs(HD)))
xlabel('f')
ylabel('HD(f)')
title('Frequency Response of Ideal LP FIR Filter')
grid on
subplot(2,2,3)
wn=hann(2*N+1);%hanning Window
stem(n,wn)
xlabel('n')
ylabel('Wn')
title('Hann Window')
grid on
subplot(2,2,4)
h=hd'.*wn;%Practical Impulse Response
stem(n,h)
xlabel('n')
ylabel('h')
title('Impulse Response of Practical LP FIR Filter')
grid on

figure
f=n*fc/N;
subplot(2,1,1)
HD=fft(hd);
HD=fftshift(HD);
plot(f,abs(HD)/max(abs(HD)))
xlabel('f')
ylabel('HD(f)')
title('Frequency Response of Ideal LP FIR Filter')
grid on
f=n*fc/N;
subplot(2,1,2)
H=fft(h);
H=fftshift(H);
plot(f,abs(H)/max(abs(H)))
```

xlabel('f')

ylabel('H(f)')

title('Frequency Response of Practical LP FIR Filter')

grid on

A comparison is made between ideal and practical FIR LP filter in fig.4.1 and 4.2.

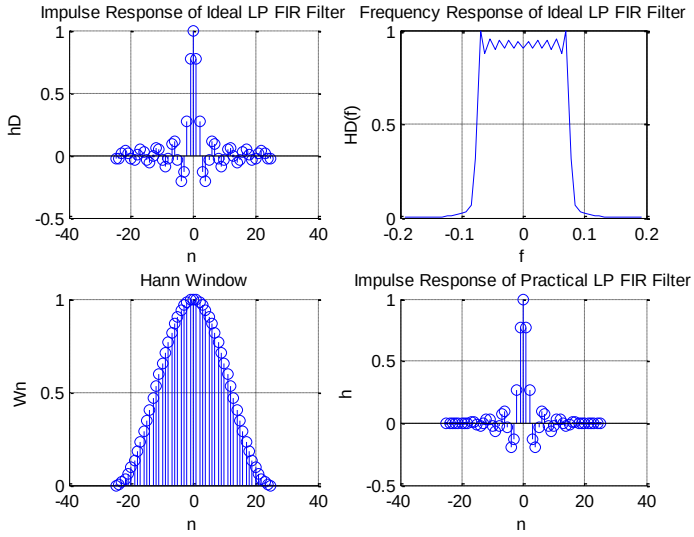


Fig.4.1 Comparison of ideal and practical impulse response of LP FIR filter

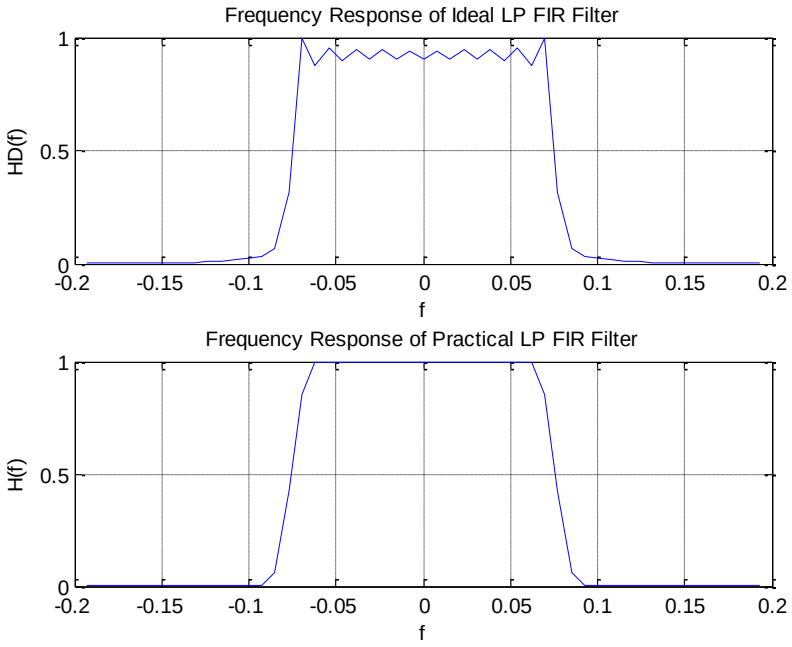


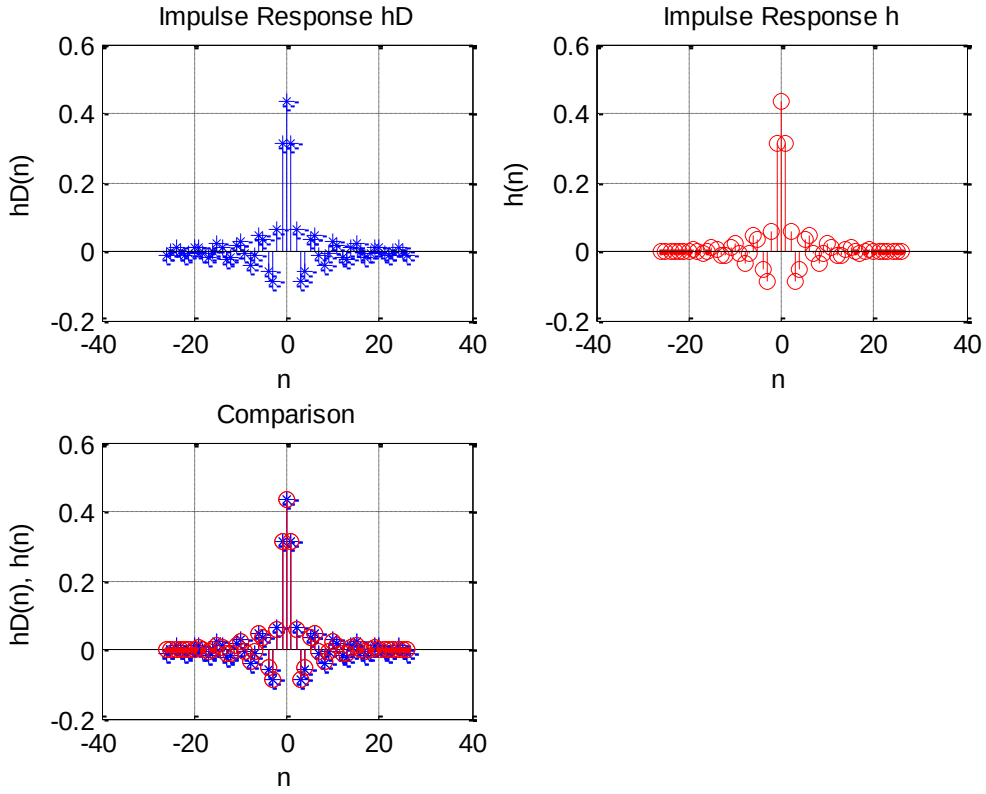
Fig.4.2 Comparison of ideal and practical frequency response of LP FIR filter

Compare the performance of ideal and practical LP filter using hamming window function for the following specifications.
Fs = 8KHz
Passband edge frequency 1.5 KHz
Transition width 0.5KHs
Stop band attenuation >50dB

```
%Matlab Code
Fs=8;
Tr_w=0.5;%Transition width
fc=1.5;%Passband edge frequency
delf=Tr_w/Fs;
N=3.3/delf;
fcp=(fc+Tr_w/2)/Fs;
%cutoff frequency or the midpoint of transition band


n=-int8(N/2):1:int8(N/2);
hD=double(2*fcp*sinc(double(n)*2*fcp));
Wn=0.54+0.46*cos(2*pi*double(n)/double(int8(N)));
h=Wn.*hD;
subplot(2,2,1)
stem(n,hD,'b*')
title('Impulse Response hD')
xlabel('n')
ylabel('hD(n)')
grid on
subplot(2,2,2)
stem(n,h,'r')
grid on
xlabel('n')
ylabel('h(n)')
title('Impulse Response h')
subplot(2,2,3)
stem(n,hD,'b*')
hold on
```

stem(n,h,'r')
title('Comparison')
grid on
xlabel('n')
ylabel('hD(n), h(n)')



2. Design a bandpass FIR filter with upper and lower cutoff frequency of 275 and 125Hz (use both Hamming and Kaiser Window function). Consider sampling frequency of 2 KHz.

Fs=2000; %Sampling frequency
Fn=Fs/2;
N=100;%number of co-efficient
beta=5.65;%beta parameter of kaiser window
fc1=125/Fn; % normalized cutoff frequency of lower side
fc2=275/Fn; % normalized cutoff frequency of upper side
Fc=[fc1 fc2];
hn=fir1(N-1,Fc,kaiser(N,beta));%FIR co-efficients

```
[H,f]=freqz(hn,1,512,Fs);%frequency response of FIR
mag=20*log10(abs(H));% conversion in dB
subplot(2,1,1)
plot(f,mag), grid on
xlabel('Frequency in Hz')
ylabel('Magnetude in dB')
title('Using Kaiser Window')
hn=fir1(N-1,Fc,hamming(N));%FIR co-efficients
[H,f]=freqz(hn,1,512,Fs);%frequency response of FIR
mag=20*log10(abs(H));%convertion in dB
subplot(2,1,2)
plot(f,mag), grid on
xlabel('Frequency in Hz')
ylabel('Magnetude in dB')
title('Using hamming Window')
```

The result of code is shown in fig. 4.2.

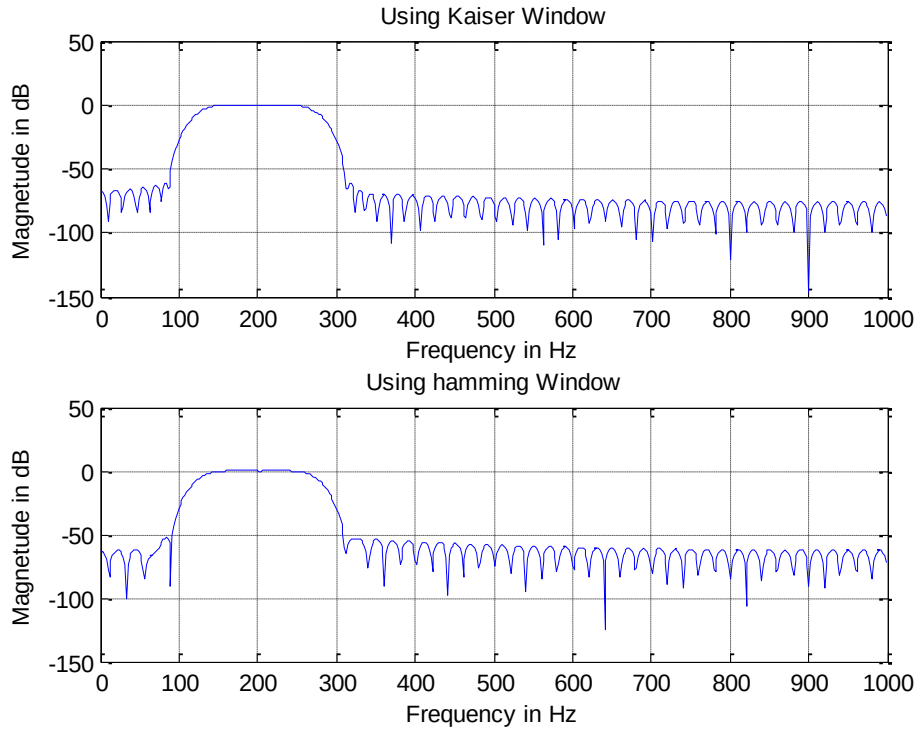
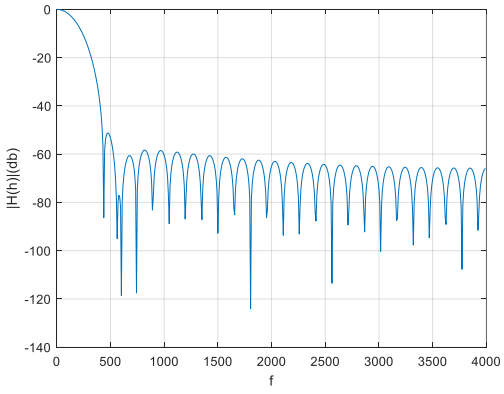
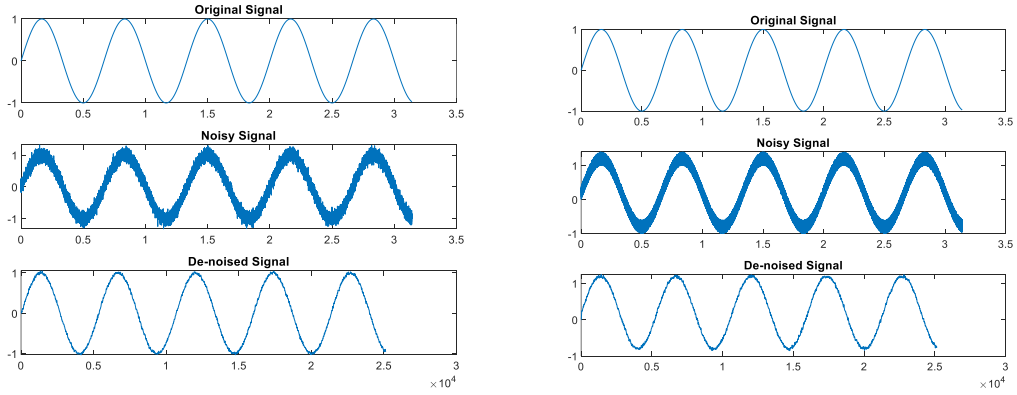


Fig. 4.2 Frequency response of a FIR BP filter

Next we will reduce the BW of the filter to eliminate random noise from a sinusoidal wave.

```
fc = 0.21875/10; %let us reduce the BW of the filter
N= 53; % divide the previous fc by 10
Fs = 8000;
wn = hamming(N);
hn = fir1(N-1,2*fc, wn);
% fc is multiplied by 2 since F_N = Fs/2
[H,f] = freqz(hn,1 , 512, Fs);
mag = 20*log10(abs(H));
plot(f, mag);
grid on
xlabel('f')
ylabel('|H(h)|(db)')
%Generate sinusoidal wave
t=0:1/Fs:pi;
x=sin(3*pi*t);  %Original signal
%xn=x+0.4*rand(1,length(t)); %Noisy signal with random noise
xn= awgn(x, 20); % awgn Noisy signal
out = filter(hn,1,xn); %filtered signal
figure
subplot(3,1,1)
plot(t, x)
title ('Original Signal')
subplot(3,1,2)
plot(t, xn)
title ('Noisy Signal')
subplot(3,1,3)
plot(out)
title ('De-noised Signal')
```





(a) awgn (b) random noise

Fig.4.3 De-noising of signal using narrow LP filter

In this section we will deal with design of an FIR filter using frequency sampling technique. In frequency sampling method the FIR filter is rwpresented by desired frequency response instead of impulse response. The co-efficient of an FIR filter is evaluated as,

$h(n) = IDFT\{H(k)\}$; k = 0, 1, 2, 3, … N-1

$$= \frac{1}{N}\sum_{k=0}^{N-1} H(k)e^{j\left(\frac{2\pi}{N}\right)nk} \text{ ; where } H(k) = |H(k)|e^{-j\left(\frac{2\pi\alpha k}{N}\right)} \text{ and } \alpha = (N-1)/2 \qquad (1)$$

Here *H(k)* is the desired frequency response (Normalized form) of the filter of *N* samples taken at intervals of kF_s/N.

Let us consider a low pass FIR filter of, passband: 0 – 5 KHz, Sampling frequency *Fs* =18 KHz and the number of samples, *N* = 9.

Now kFs/N = k×18/9 = 2 KHz. For the pass band of 0 – 5 KHz, *k* = 0, 1 and 2. For stop band *k* = 4, 5, 6, 7 and 8.

$$\therefore H(k) = \begin{cases} 1; & k = 0, 1, 2 \\ 0; & k = 4, 5, 6, 7, 8 \end{cases}$$

Now we can evaluate *h(n)* using the equation (1). In Matlab syntax of FIR filter based on frequency sampling technique is, *hn = fir2(N-1, F, H);* where **H** is the vector of desired magnitude response at the corresponding frequency points of the vector **F**. The frequency points of the vector **F** is in normalized form in the range of 0 to 1.

```
Fs=18;
N=9;
fts=[0 1 2  3  4  5 6 7]/7;
Hk=[1 1 1 0 0 0 0 0];
b=fir2(N-1, fts, Hk);
% b = fir2(N,F,A) designs an N'th order FIR digital filter with the frequency response
% specified by vectors F and A, and returns the filter coefficients in length N+1 vector b.
[h, f]=freqz(b,1,512,Fs);
```

plot(f*(Fs/N),abs(h));
xlabel('Frequency in KHz')
ylabel('Magnitude')
grid on
The result of the code is shown in fig.4.3.

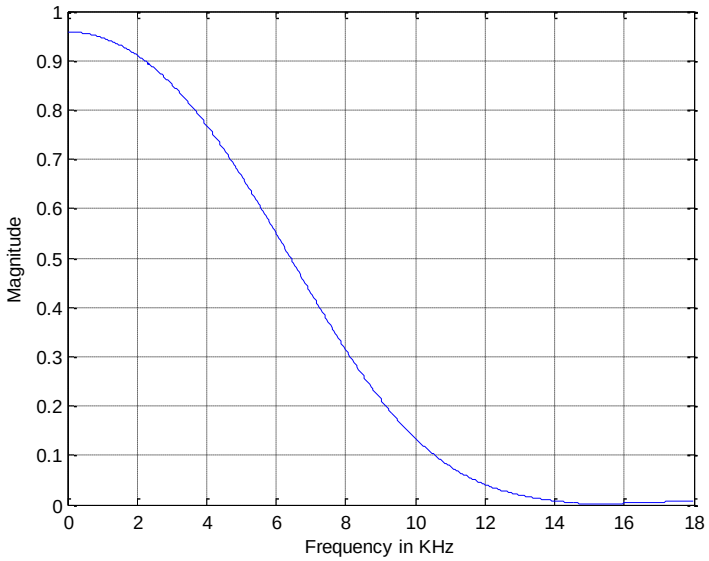


Fig. 4.3. Frequency response of the low pass FIR filter

The impulse response of an FIR filter is symmetric hence *h(n)* of FIR filter based on frequency sampling technique can be simplified,

$$h(n)=\frac{1}{N}\left[\sum_{k=1}^{\frac{N-1}{2}}2\left|H(k)\right|\cos\left\{\frac{2\pi k}{N}\left(n-\alpha\right)\right\}+H(0)\right]\text{; when } N \text{ is odd}$$

$$h(n)=\frac{1}{N}\left[\sum_{k=1}^{\frac{N}{2}-1}2\left|H(k)\right|\cos\left\{\frac{2\pi k}{N}\left(n-\alpha\right)\right\}+H(0)\right]\text{; when } N \text{ is even}$$

Here *H(k)* = 0 or 1 depending on the position of stopband or passband. For above example the impulse response of the filter with positive symmetry will be,

$$h(n)=\frac{1}{9}\left[\sum_{k=1}^{2}2\left|H(k)\right|\cos\left\{\frac{2\pi k}{9}\left(n-4\right)\right\}+H(0)\right]$$

$$=\frac{1}{9}\left[\sum_{k=1}^{2}2\cos\left\{\frac{2\pi k}{9}\left(n-4\right)\right\}+1\right]$$

The Matlab code to determine h(n) is given below.

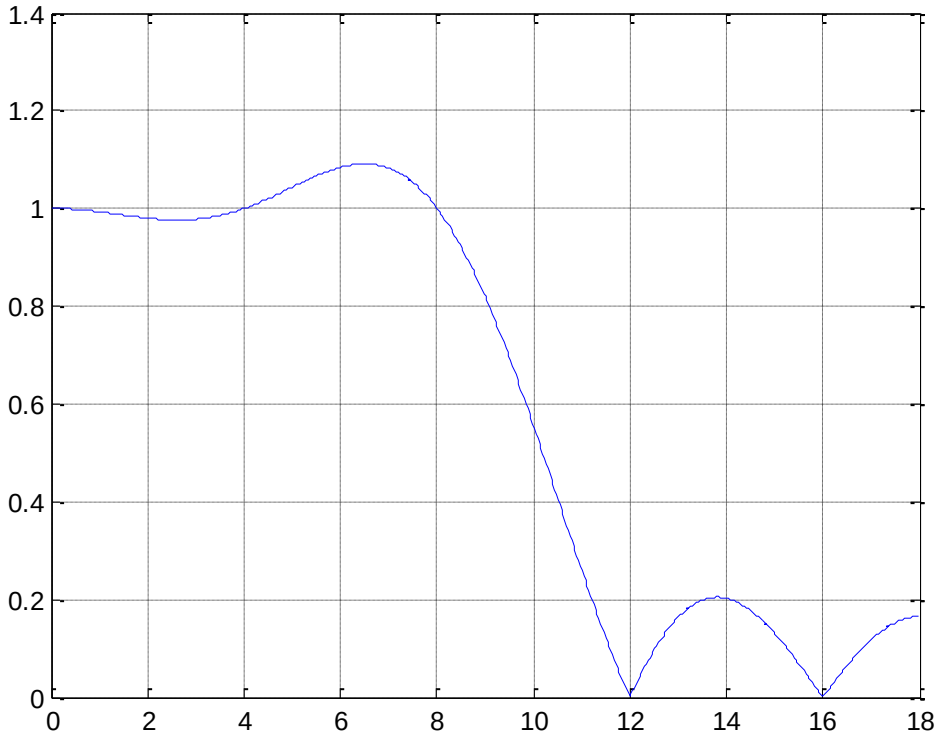
```
Fs = 18;  N = 9;
alpha = (N-1)/2;
for  j = 1:N, % the loop for n
n = j-1; %since n= 0 to 8 hence j is decreased by 1
s = 1; % is the value of H(0)
h = 0; % initialization of h(n)

for k = 1:2,
s = s+2*cos(2*pi*k*(n-alpha)/N);
end

h = (1/N)*s; %value of h(n)
h_s(j) = h;
end
h_s

[h, f] = freqz(h_s,1,512,Fs);
plot(f*(Fs/N),abs(h));
grid on
```

```
h_s =  0.0725   -0.1111   -0.0591    0.3199    0.5556    0.3199   -0.0591   -0.1111    0.0725
```



3. An FIR filter of length 50 with three different constant magnitude levels: 0.5 in frequency range 0 to 0.27, 1 in the frequency range 0.32 to 0.51 and 0.72 in the frequency range 0.54 to 1. Draw the frequency response of the filter in normalized scale.

```
fts=[0 0.27 0.32 0.51 0.54 1];
Hk=[ 0.5 0.5 1.0 1.0 0.72 0.72];
b=fir2(50, fts, Hk);
[h, omega]=freqz(b,1,512);
plot(omega/pi,abs(h));
xlabel('\omega/ \pi')
ylabel(' Magnitude')
grid on
```

The result of the code is shown in fig.4.4.

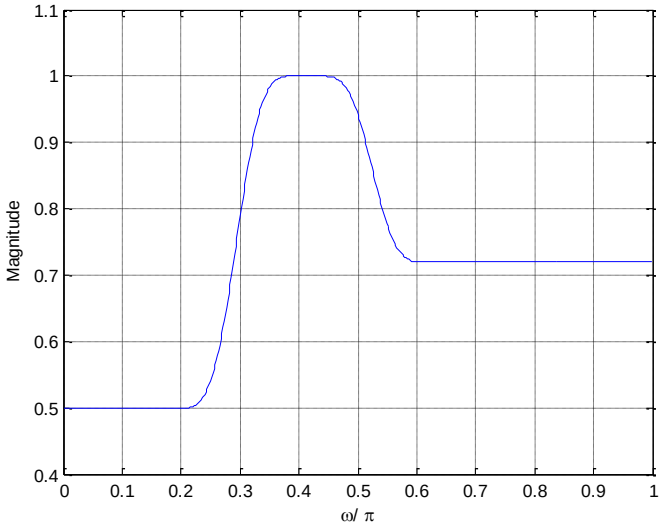


Fig.4.4 Frequency response of the band pass FIR filter in normalized scale

Design a LP and HP FIR filter using frequency sampling technique hence filter a speech signal by both of the filters and comment on the results. Give the length of the LP or HP filter is 8 and Fs=18KHz.

```
Fs=18;%sampling Frequency
N=8;%length of filter
fts=[0 1 2  3  4  5 6 7]/7;%sampled frequency
nval=[1  1 0.707 0 0 0 0 0];%amplitude H(k) of LP filter
bLP=fir2(N, fts, nval);

load nearspeech
M=length(v);%number of iteration
for i=1:M,
x(i)=v(i);
end

xn = x + 0.05*(rand(size(x))-0.5);
y = filter(bLP,1,xn);
t = (0:length(y)-1)/Fs;
subplot(2,2,1)
plot(t,x)
title('Original Signal')
grid on
```

```
subplot(2,2,2)
plot(t,xn)
title('Noisy Signal')
grid on

subplot(2,2,3)
plot(t,y)
title('Lowpass Filtered Signal')
xlabel('Time (s)')
grid on

%%%%%%%%%%%%%%%%%%%%%%%%%%

fts=[0 1 2  3  4  5 6 7]/7;
nval=[0 0 0 0 0.707 1 1 1];%amplitude H(k) of HP filter
bHP=fir2(N, fts, nval);
outlo = filter(bHP,1,xn);
subplot(2,2,4)
plot(t,outlo)
title('Highpass Filtered Signal')
xlabel('Time (s)')
grid on
```

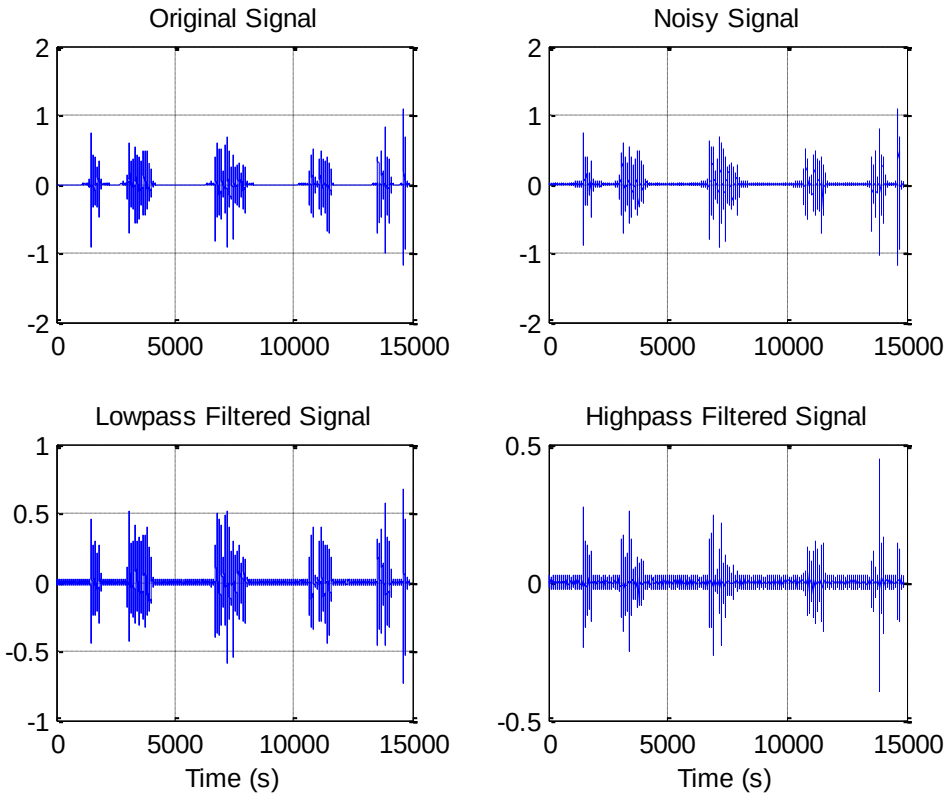


Fig.4.5 Output of LP and Hp filter

Here LP provides better result compared to HP since the output of HP is mainly noise shown in fig.4.5. The LP is not fully successful since noise and signal are of same frequency hence we need adaptive filter to de-nose the signal.

Experiment No. 05

Design and Implementation of Adaptive Filter

Basic Theory:

Extraction of the original signal from a noisy environment is achieved by FIR or IIR filter but their coefficients remain fixed during entire operation. In adaptive filter adaptive algorithms are used to adjust the coefficients of the digital filter so that the difference between the noisy input signal, X_k and the desired signal, d of fig. 5.1 called the error signal ε attains at minimum value. The common algorithms found widespread applications are the least mean square (LMS), the recursive least squares (RLS), and the Kalman filter algorithms. In terms of computational complexity and storage requirements, the LMS algorithm is the most efficient. Objective of this experiment is to observe the profile of mean square error, convergence of error toward its minimum value and elimination of noise from a noisy input signal.

1. If mean square error of a linear combiner of fig. 5.1 (The fig.5.1 is taken from: Bernard Widrow and Samuel D. Stearns, Adaptive Signal Processing, Pearson Education) is, $\xi = \frac{1}{2}\left(W_1^2 + W_1^2\right) + W_1 W_2 \cos(2\pi / N) + 2W_2 \sin(2\pi / N) + 2$; Then show the profile of mean square error using mesh and contour plot on W_1-W_2 plane (use N = 3).

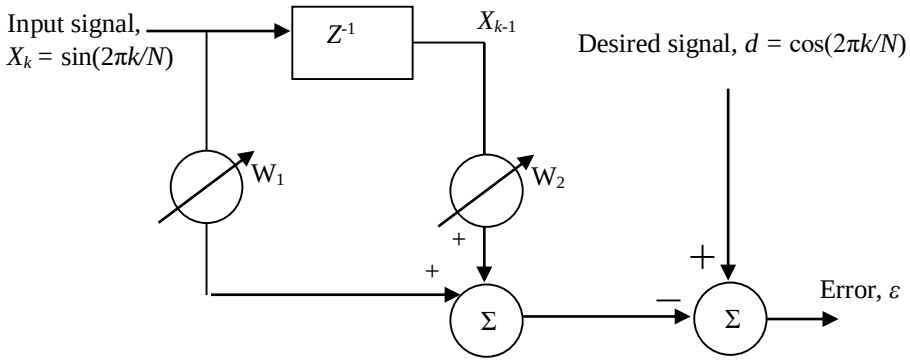


Fig. 5.1 Adaptive linear combiner

```
N=3;% here N=3
[W1,W2]=meshgrid(-8:0.5:8);
% grid lines lies between -8 to 8 with intervals of 0.5
z=(0.5)*(W1.^2+W2.^2)+W1.*W2*cos(2*pi/N)+2*W2*sin(2*pi/N)+2;
figure(1)
mesh(z); %mesh plot of z
title('mesh plot')
figure(2)
```

contour(z); %contour plot of z

title('contour plot')

y=min(z);

emin=min(y) %minimum value of error

The result of above code is shown in fig. 5.2 where the weighting factors W_1 and W_2 are along x and y axis and the mean square error is along z axis. Here the bottom of the bowl provides the minimum value of error, ε_{min} = 0.0359. A vertical line from the point of minima intersects the W_1 - W_2 plane at a point provides the optimum weighting vector.

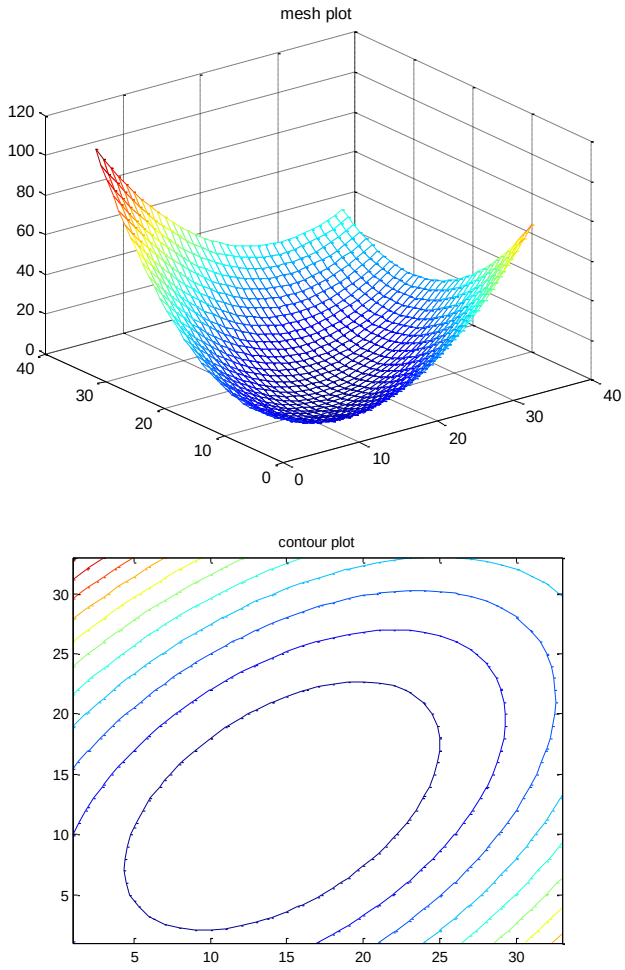


Fig. 5.2 Profile of mean square error

2. In this section we will determine: (a) the profile of weighting factors W_1 and W_2 against the number of iteration steps k (b) values of weighting factors W_1 and W_2 on W_1 -W_2 plane (c) the profile of mean square error against the number of steps to observe the converge rate (number of steps required to attain minimum error) based on LMS algorithm.

The mean square error of the adaptive linear combiner of the fig.5.1 is,

$$\xi = \frac{1}{2}\left(W_1^2 + W_1^2\right) + W_1 W_2 \cos(2\pi / N) + 2W_2 \sin(2\pi / N) + 2 .$$

```
clear all
close all
M=200;%number of iteration
mu=0.5;%LMS parameter
N=16;
w1(1)=0; %initial weight W1
w2(1)=0;  %initial weight W2

for k=2:M;
d(k)=2*cos(2*pi*k/N);  %desired signal
x1(k)=sin(2*pi*k/N); %input signal
x2(k)=sin(2*pi*(k-1)/N);%delayed input signal
E(k)=d(k)-transpose([x1(k);x2(k)])*[w1(k-1);w2(k-1)]; %error
W = [w1(k-1);w2(k-1)]+ mu*E(k)*[x1(k);x2(k)]; %updated weight
w1(k)=W(1);
w2(k)=W(2);
error(k)=(0.5)*(W(1)^2+W(2)^2)+W(1)*W(2)*cos(2*pi/N)+ 2*W(2)*sin(2*pi/N)+2;
end

k=1:1:length(w1);
subplot(2,2,1)
plot(k,w1,'b')
xlabel('k')
ylabel('w1')
title('Profile of w1 ')
subplot(2,2,2)
plot(k,w2,'r')
xlabel('k')
ylabel('w2')
title('Profile of  w2')
subplot(2,2,3)
plot(w1,w2,'m')
xlabel('w1')
ylabel('w2')
title('w1 and w2')
subplot(2,2,4)
plot(k,error,'r')
```

```
xlabel('k')
ylabel('error')
title('Profile of error')

W  %final weight
error=(0.5)*(W(1)^2+W(2)^2)+W(1)*W(2)*cos(2*pi/N)+2*W(2)*sin(2*pi/N)+2
%final error
```

W =

4.7080

-5.1457

error =

0.0015

The result of above code is shown in fig. 5.3.

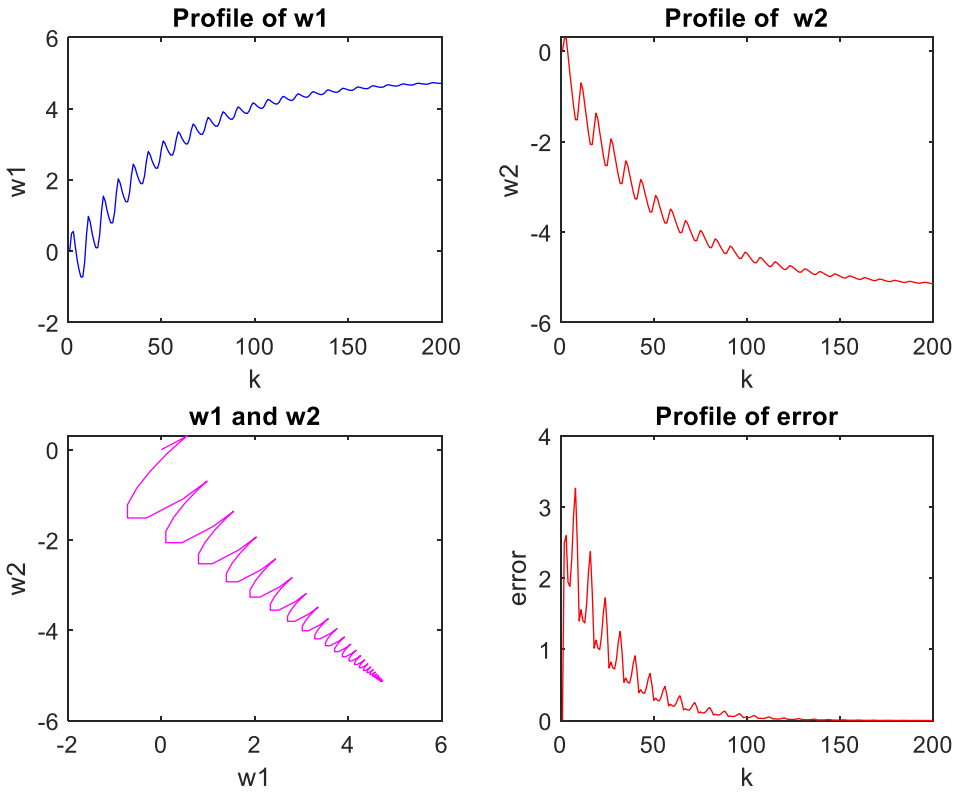
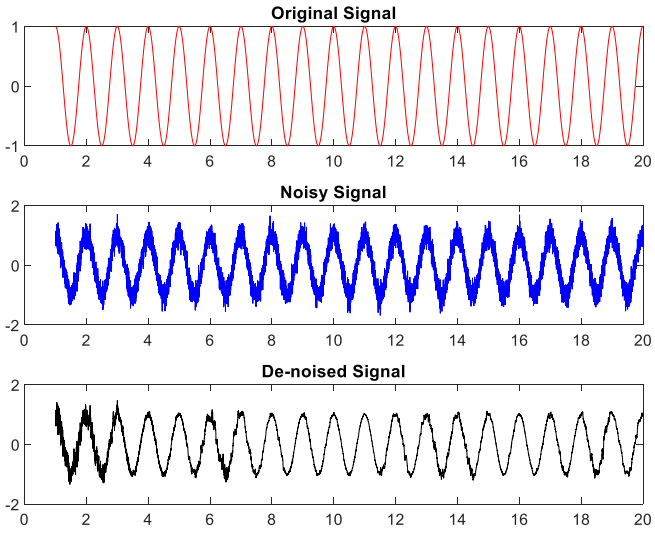


Fig. 5.3 Profile of weighting factors and mean square error

3. Apply LMS adaptive filter of order 32 to filer a noisy sinusoidal signal.

```
clear all
```

```
close all
t=1:0.002:20;
s=cos(2*pi*t);%Original Signal
noise = 0.2*randn(1,length(t)); % random noise
sn=s+noise;% noisy signal
%LMS Algorithm
lmsfilt = dsp.LMSFilter('Length',16,'Method','Normalized LMS','StepSize',0.02);
[y,e,w] = lmsfilt(noise',sn'); %filtering the signal
subplot(3,1,1); plot(t,s,'r') ;title('Original Signal')
subplot(3,1,2)
plot(t,sn,'b')
title('Noisy Signal')
subplot(3,1,3)
plot(t,e,'k')
title('De-noised Signal')
```



4. Do the similar job for a speech signal.

```
clear all
close all

load handel %original signal
s = y(1:4000); %taking 4000 samples of sound wave
s=s';
noise = 0.12*randn(1,length(s)); % random noise
sn=s+noise;% noisy signal
```

```
%LMS Algorithm
lmsfilt = dsp.LMSFilter('Length', 16, 'Method', 'Normalized LMS', 'StepSize', 0.02);
[y,e,w] = lmsfilt(noise', sn'); %filtering the signal with LMS of step size mu = 0.02

t=1:1:length(s);
t=t/500;
subplot(3,1,1)
plot(t,s,'r')
title('Original Signal')
subplot(3,1,2)
plot(t,sn,'b')
title('Noisy Signal')
subplot(3,1,3)
plot(t,e,'k')
title('De-noised Signal')
```

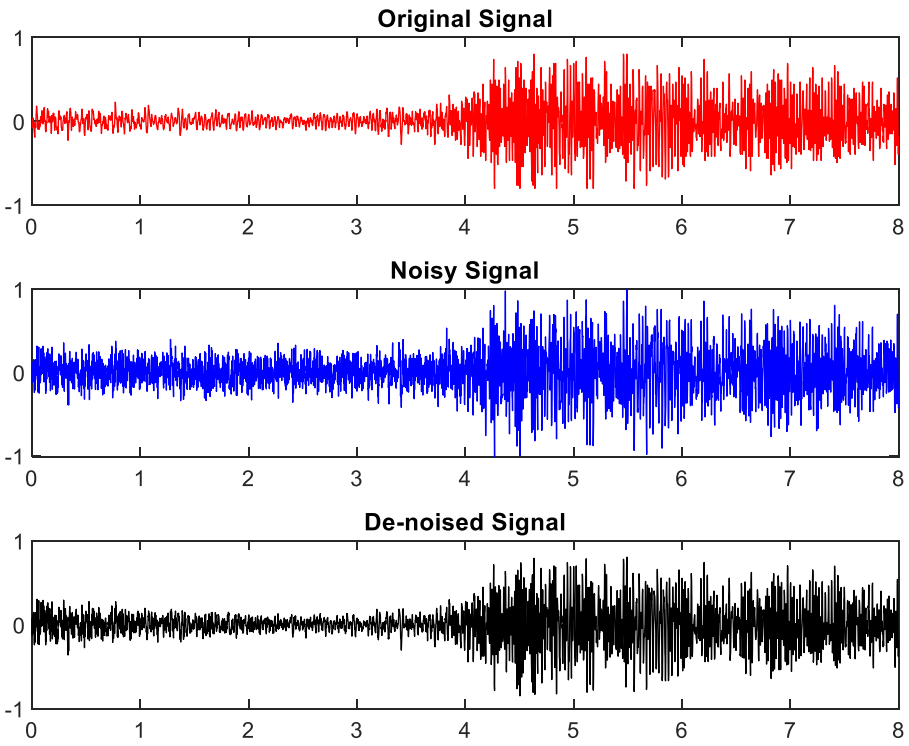


Fig. 5.4(a) Original and noisy input signal

Verify the sound quality using: sound(s), sound(sn) and sound(e) taking 20,000 samples in above code.

Experiment 6
Performance evaluation of LMS, RLS, FDAF filters in noise cancellation

Different types of adaptive algorithm like: least-mean-square (LMS), recursive least squares (RLS), frequency-domain adaptive filter (FDAF), Kalman filter are widely used in noise cancellation of a signal. This experiment makes a comparison among LMS, RLS and FDAF in recovery of a cosine signal from a noisy environment. The code of this experiment is based on Matlab 7.x or above.

```
%LMS Filter
t=1:1:1024;
t=(t/1024)*2*pi;
x=cos(2*pi*t);%Original Signal
N=31; Wn=0.5;
b  = fir1(N,Wn);     % FIR filter using window
n  = 0.2*randn(1,length(t)); % noise signal
d  = filter(b,1,x)+n; % Desired signal
mu = 0.009;          % LMS step size.
ha = adaptfilt.lms(32,mu);
[y1,e1] = filter(ha,x,d);
subplot(3,1,1); plot(1:length(t),[d;y1;e1]);
title('LMS');
legend('Desired Signal','Output Signal','Error Signal');
xlabel('Time Index'); ylabel('Signal Value');

%RLS Filter
t=1:1:1024;
t=(t/1024)*2*pi;
x=cos(2*pi*t);
N=31; Wn=0.5;
b  = fir1(N,Wn);     % FIR system to be identified
n  = 0.1*randn(1,length(t)); % Observation noise signal
d  = filter(b,1,x)+n; % Desired signal
lam = 0.99;          % RLS forgetting factor
ha = adaptfilt.rls(32,lam);
[y2,e2] = filter(ha,x,d);
subplot(3,1,2); plot(1:length(t),[d;y2;e2]);
title('RLS');
legend('Desired Signal','Output Signal','Error Signal');
```

```
xlabel('Time Index'); ylabel('Signal Value');

%FDAF
t=1:1:1024;
t=(t/1024)*2*pi;
x=cos(2*pi*t);
N=31; Wn=0.5;
b  = fir1(N,Wn);     % FIR system to be identified
n  = 0.1*randn(1,length(t)); % Observation noise signal
d  = filter(b,1,x)+n; % Desired signal
del = 1;
lam = 0.99;    % Averaging factor
mu = 0.009;% LMS step size.
ha = adaptfilt.fdaf(32,mu,1,del,lam);
[y3,e3] = filter(ha,x,d);
subplot(3,1,3); plot(1:length(t),[d;y3;e3]);
title('FDAF');
legend('Desired Signal','Output Signal','Error Signal');
xlabel('Time Index'); ylabel('Signal Value');

figure(2)
plot(1:length(t),[e1;e2;e3]);
xlabel('Time'); ylabel('Error Signal');
title('Comparison of Error')
legend('LMS','RLS','FDAF');
figure(3)
plot(t,x,'b',t,y1,'c',t,y2,'r',t,y3,'m');
legend('Original','LMS','RLS','FDAF');
xlabel('Time'); ylabel('Output Signal');
title('Comparison of Output Signals')
```

Performance of above filters is shown in fig.6.1 to 6.3.

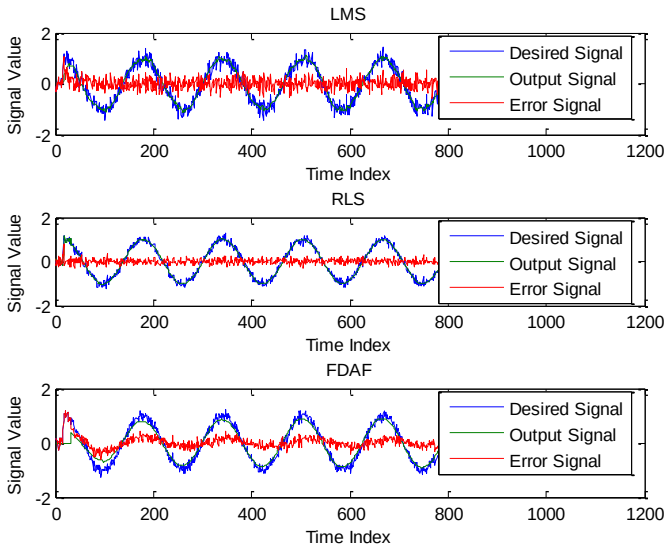


Fig.6.1 Output, desired and error signals

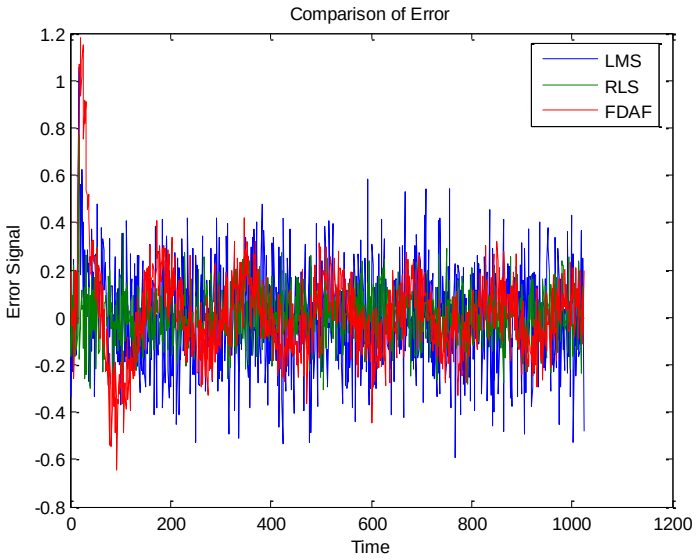


Fig.6.2 Comparison of error signals

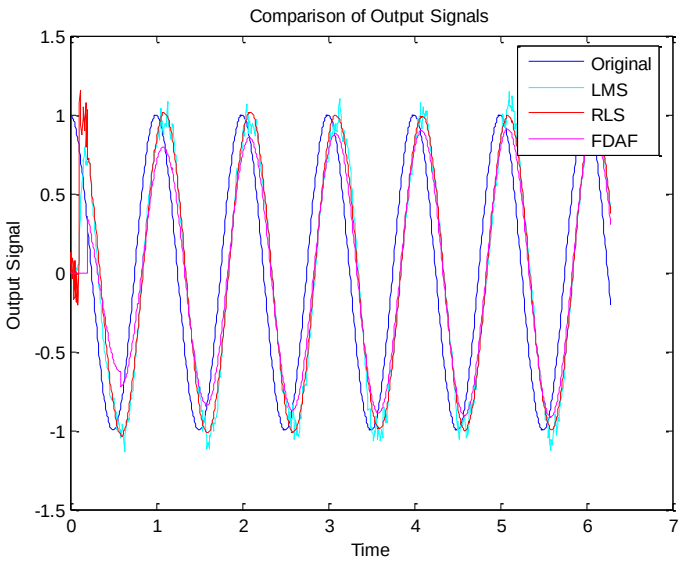


Fig.6.3 Comparison of output signals

Questions

1. Run the following code and comment on the results**.**

```
load nearspeech
n = 1:length(v);
t = n/fs;
%LMS
N=31; Wn=0.5;
b  = fir1(N,Wn);     % FIR filter using window
d1  = filter(b,1,v);
d=awgn(d1,50);  % Desired signal
mu = 0.009;           % LMS step size.
ha = adaptfilt.lms(32,mu);
[y1,e1] = filter(ha,v,d);
plot(d,'b')
hold on
plot(y1,'g')
hold on
plot(e1,'r')
legend('Desired Signal','Output Signal','Error Signal');
```