Digital signal processing Lab
Experiment-2
Designing low pass filters by windowing method

Problem statement:-

- To design FIR filters for various orders and cut-off frequencies.
- To see the performance of the filter in the time domain.
- To see how much the pass band and stop band frequencies get attenuated in different orders of windows.
    - One frequency in the pass band and one frequency in stop band is considered to see this effect.i.e., signal is taken as x[n]=2*cos(0.1*pi*n)+4*cos(0.9*pi*n).
- To see the effect on the input signals contained noise for designed FIR filters.
    - Random noise s=2*transpose(randn(Np+1,1)) is added to the signal x[n] and made the noise signal y[n].
- Calculate the SNR for different orders of windows when noise gets added to the signal.

Window functions:-

Rectangular window	$w[n] = 1 \text{ for } 0 \le n \le N-1$
Triangular Window	$w[n] = 1 - 2[n-(N-1)/2]/(N-1) \text{ for } 0 \le n \le N-1$
Hanning Window	$w[n] = 0.5 - 0.5\cos(2\pi n/[N-1]) \text{ for } 0 \le n \le N-1$
Hamming Window	$w[n] = 0.54 - 0.46\cos(2\pi n/[N-1]) \text{ for } 0 \le n \le N-1$
Blackman Window	$w[n] = 0.42 - 0.5\cos(2\pi n/[N-1]) + 0.08\cos(4\pi n/[N-1]) \text{ for } 0 \le n \le N-1$

Plots:

Rectangular window:

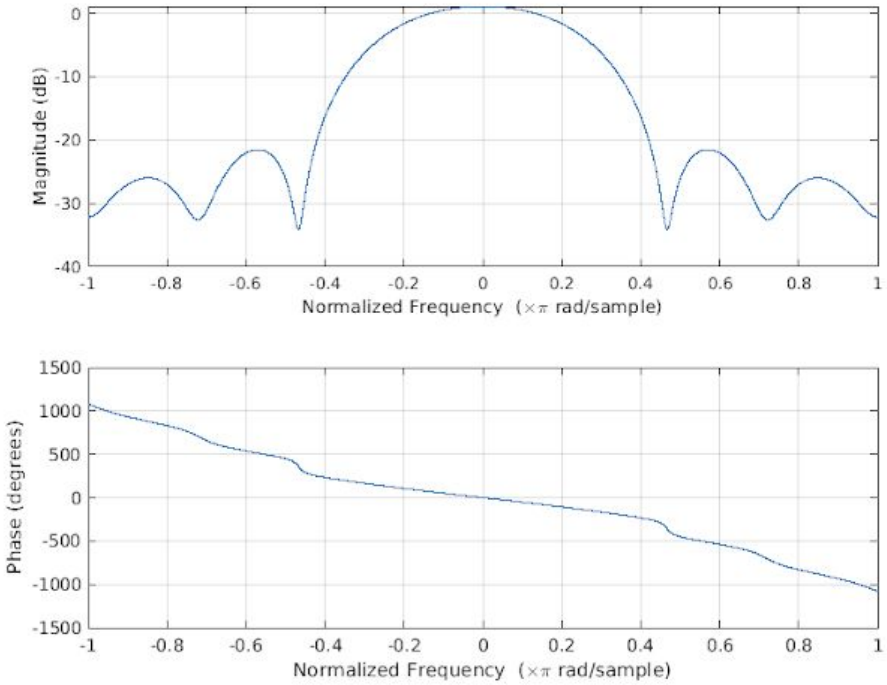

Fig.1_1_a:- frequency response of the FIR filter with filter order N=8

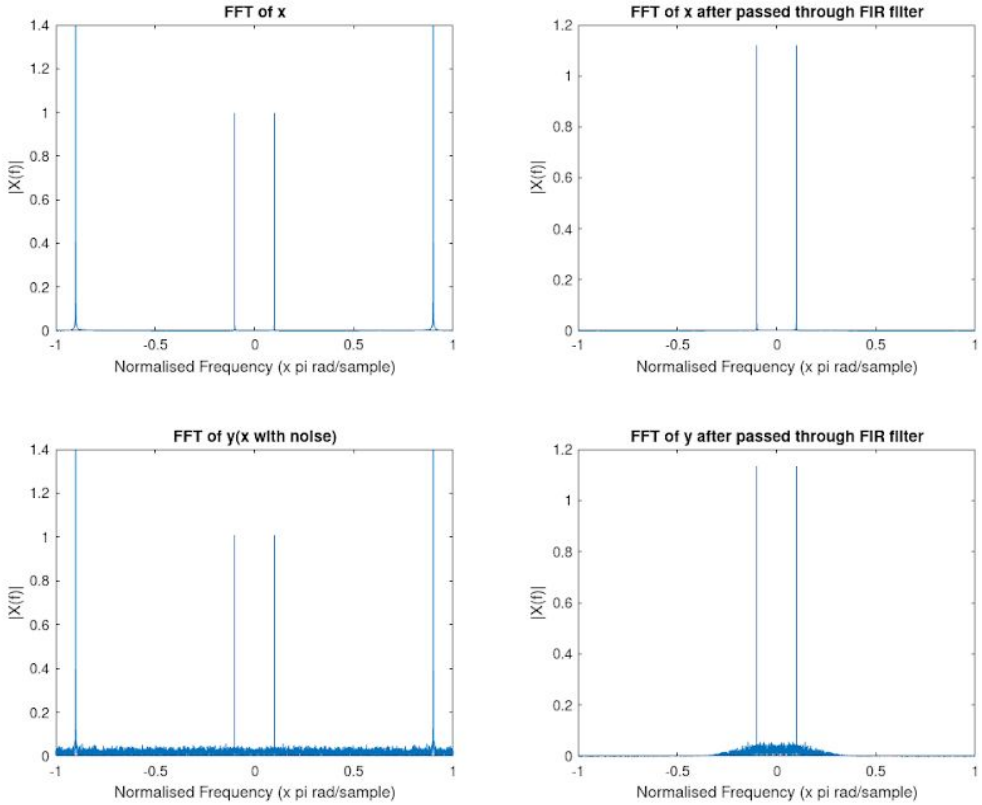

Fig.1_1_b:- FFT of signals when filter order N=8

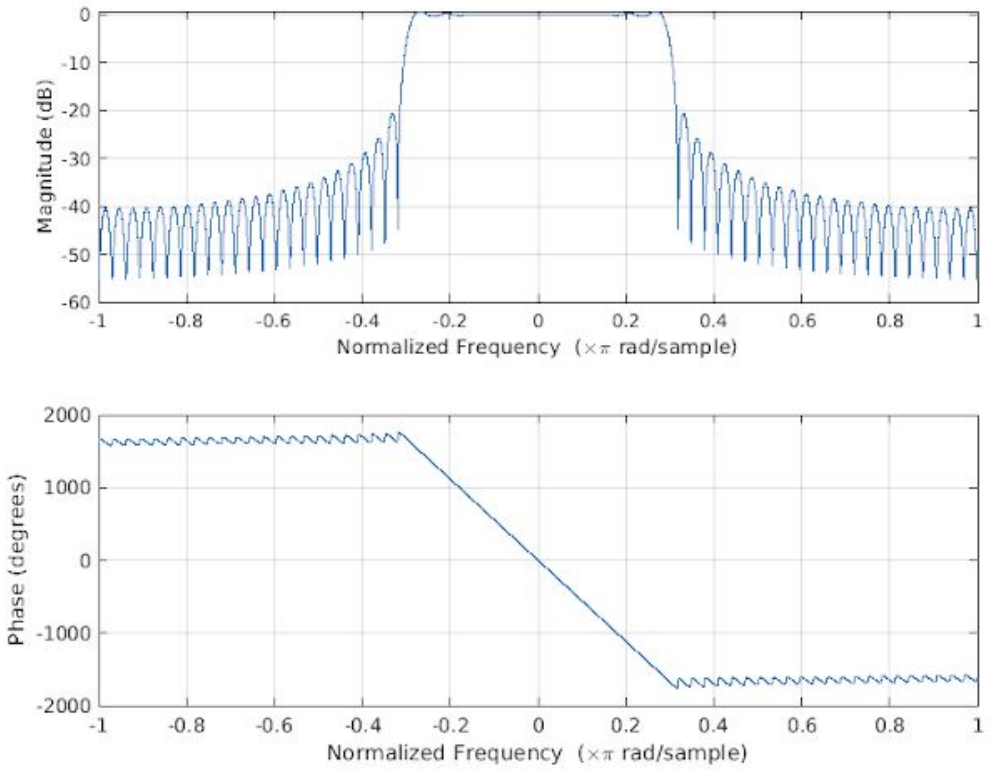

Fig.1_2_a:- frequency response of the FIR filter with filter order N=64

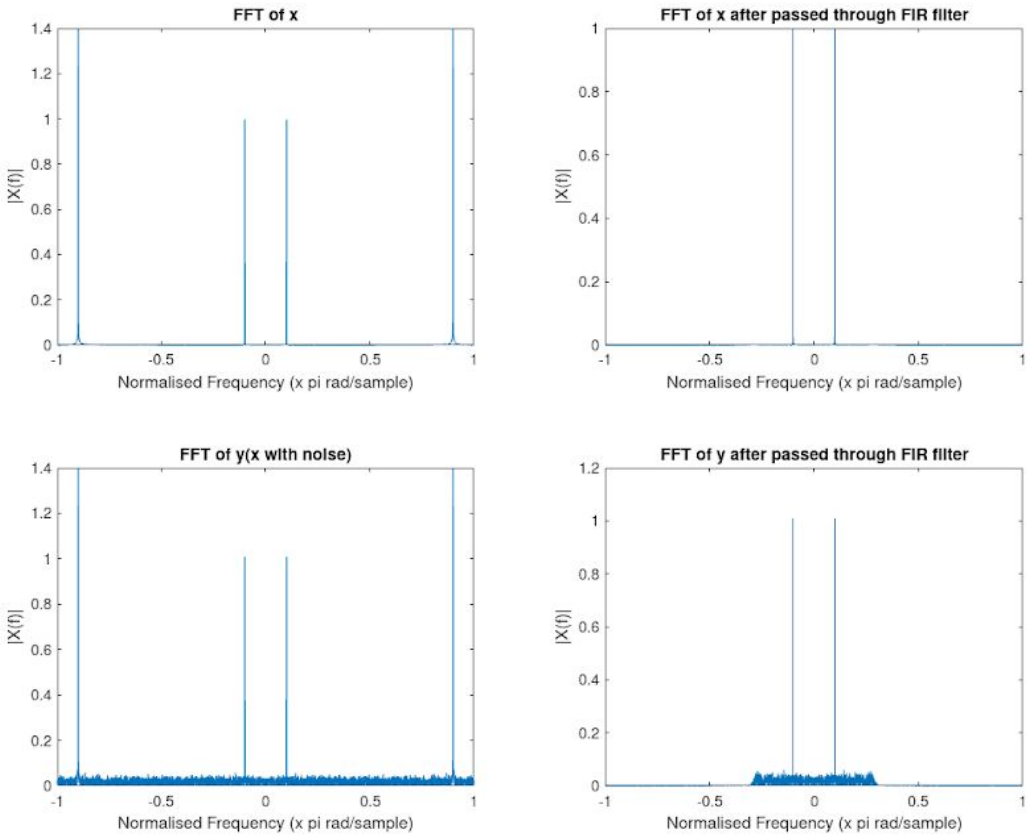

Fig.1_2_b:- FFT of signals when filter order N=64

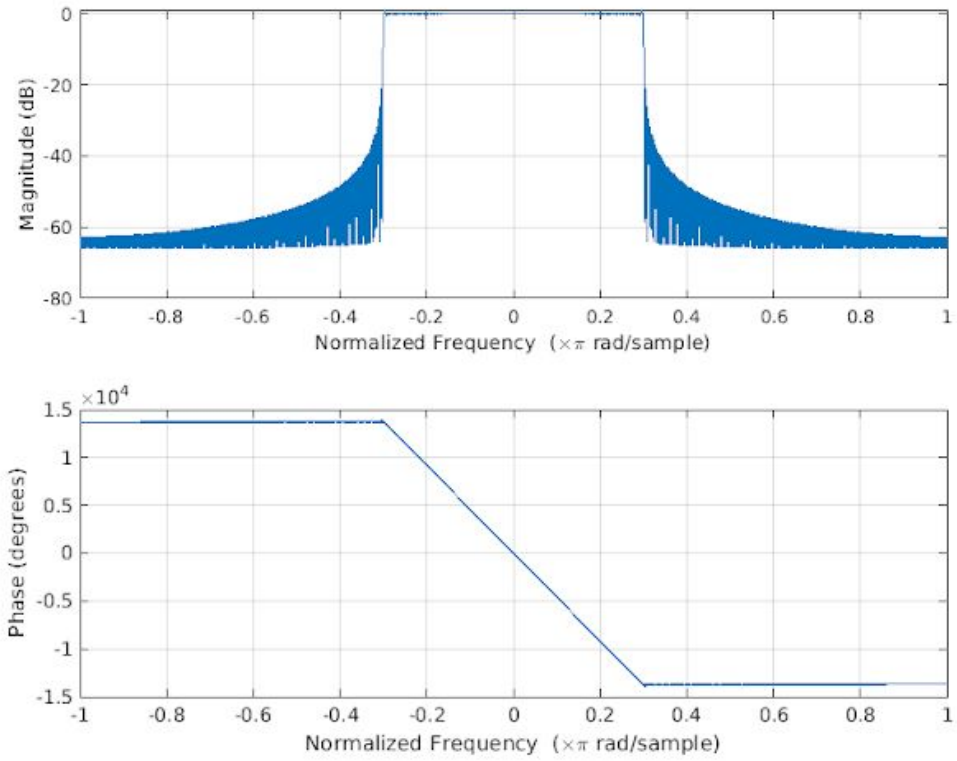

Fig.1_3_a:- frequency response of the FIR filter with filter order N=512

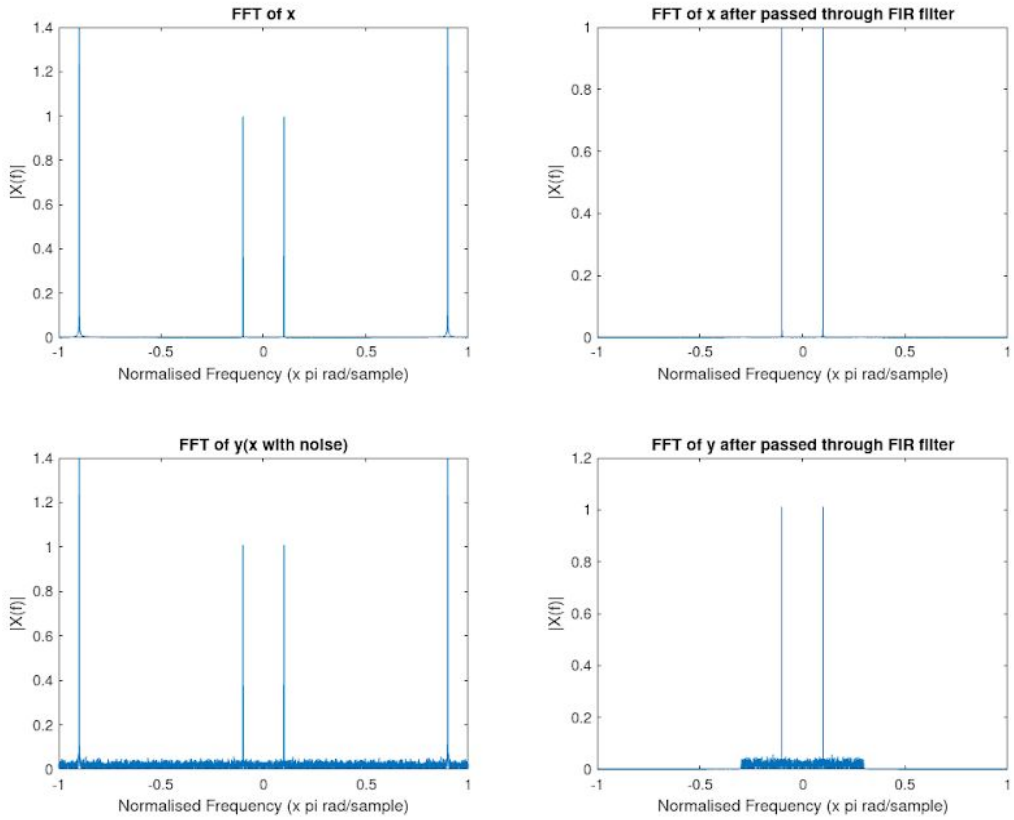

Fig.1_3_b:- FFT of signals when filter order N=512

Triangular Window:

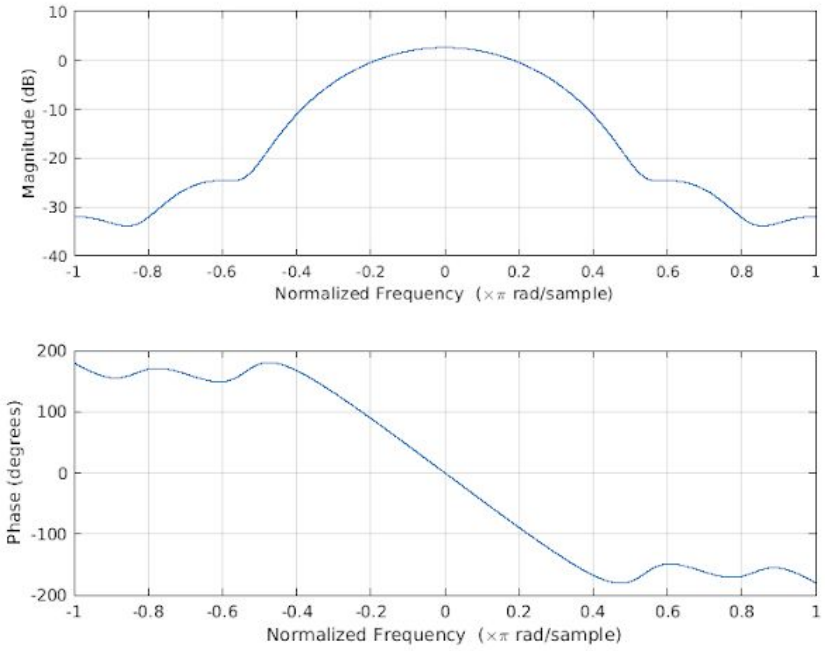

Fig.2_1_a:- frequency response of the FIR filter with filter order N=8

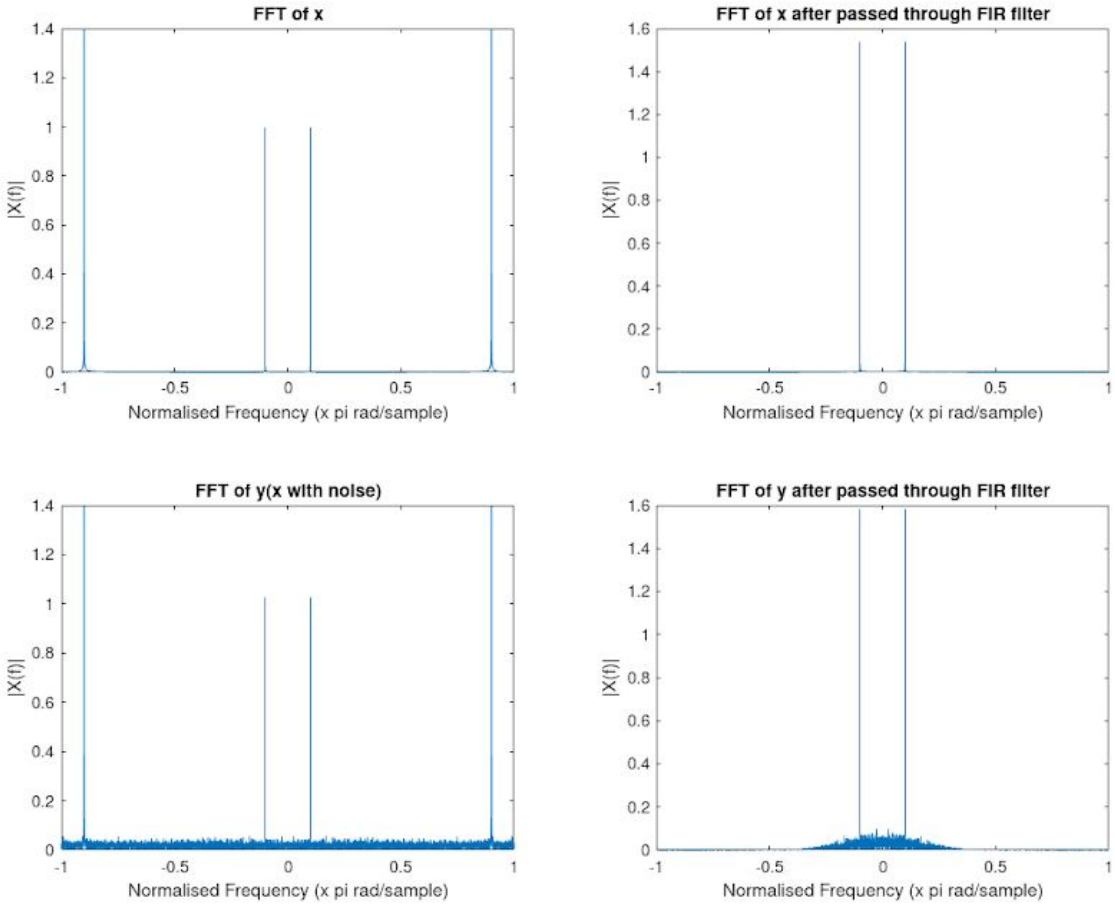

Fig.2_1_b:- FFT of signals when filter order N=8

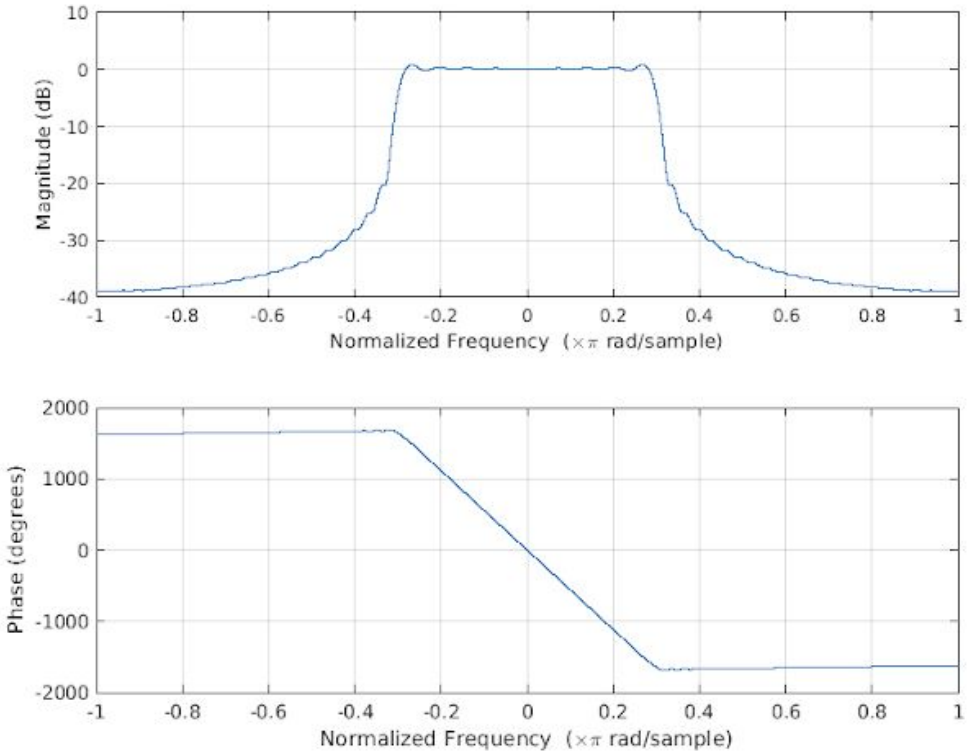

Fig.2_2_a:- frequency response of the FIR filter with filter order N=64

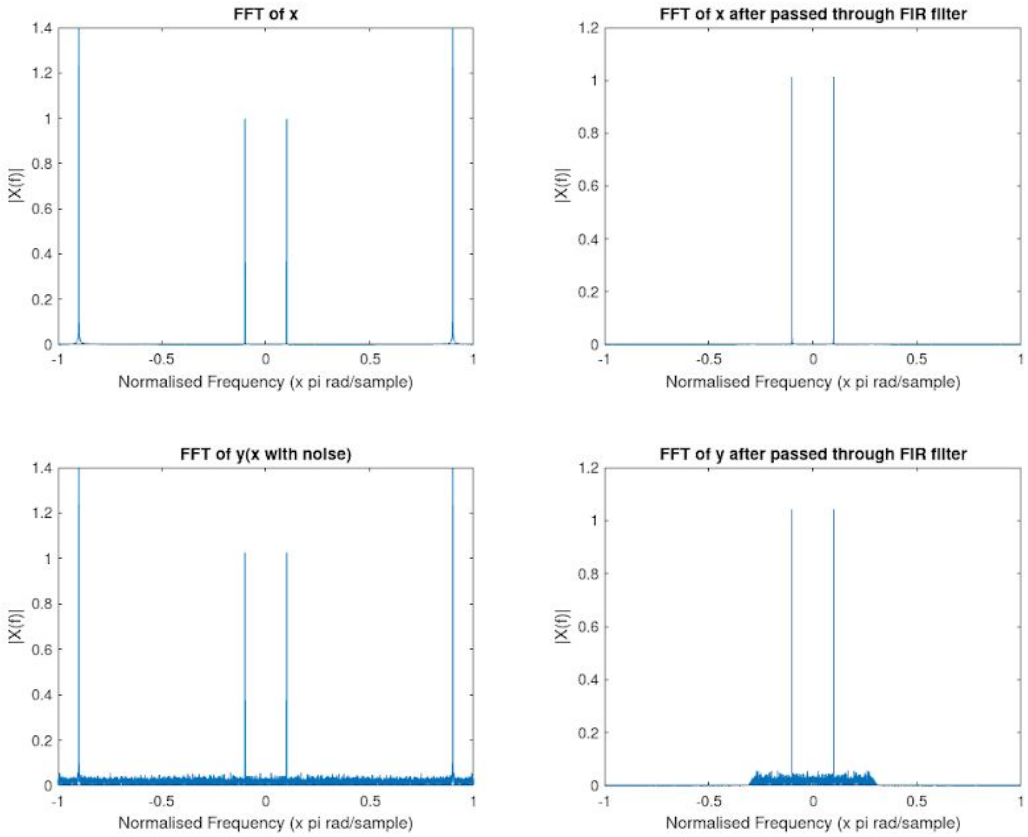

Fig.2_2_b:- FFT of signals when filter order N=64

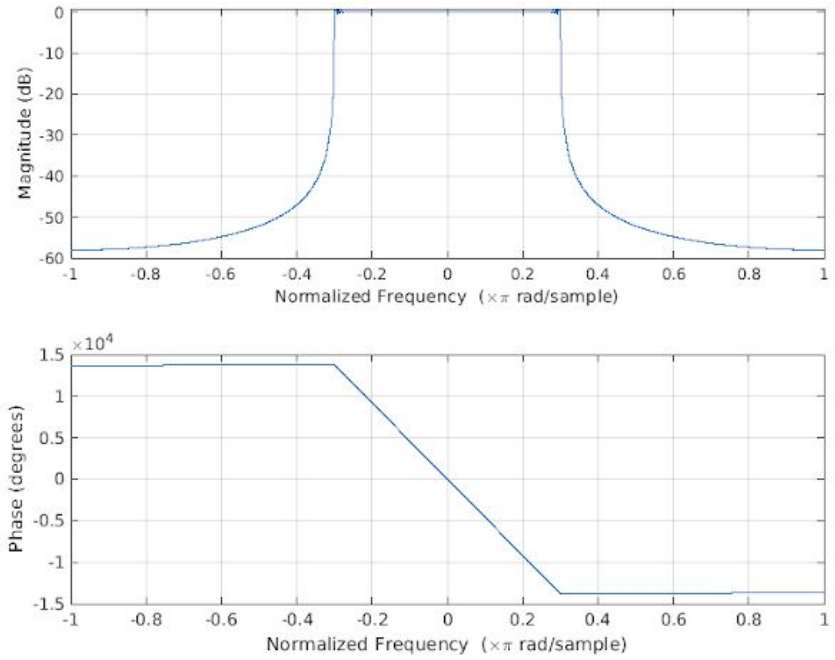

Fig.2_3_a:- frequency response of the FIR filter with filter order N=512

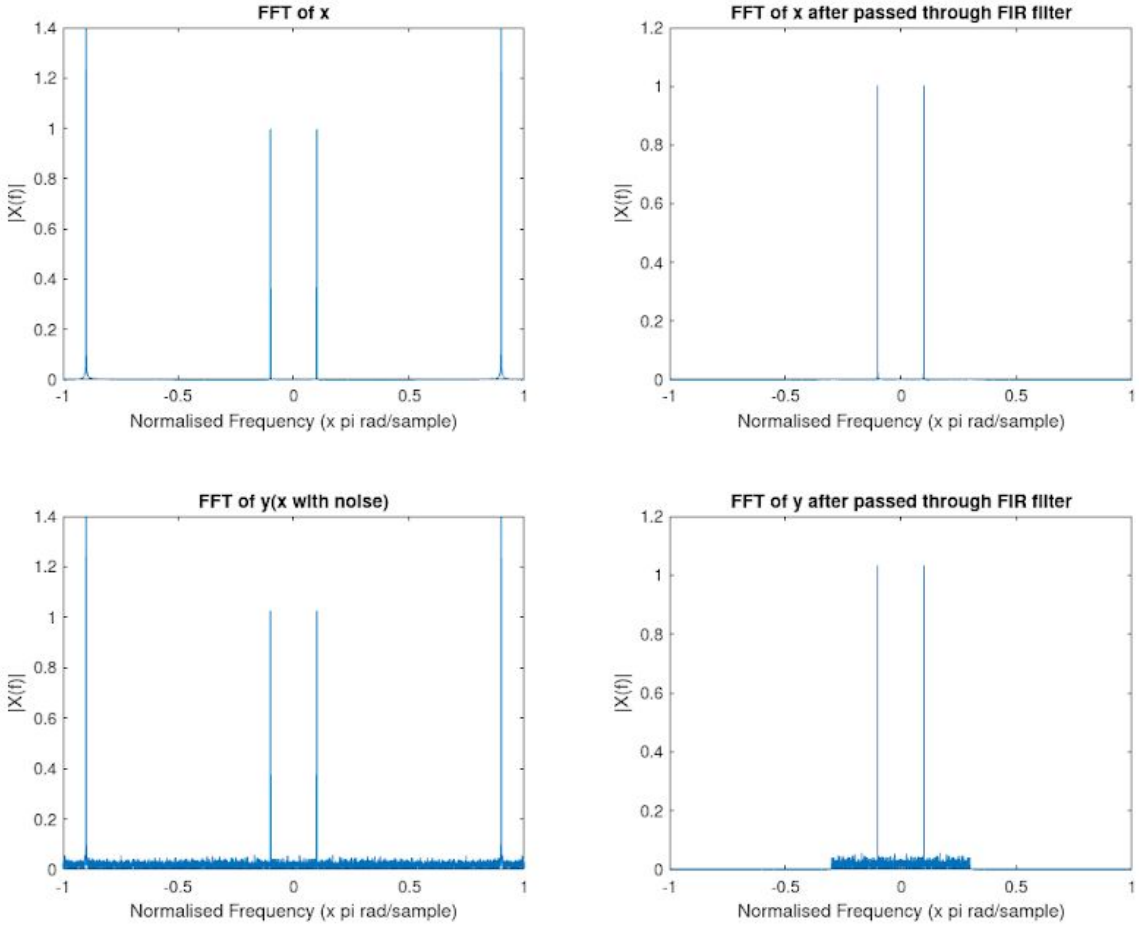

Fig.2_3_b:- FFT of signals when filter order N=512

Hanning Window:

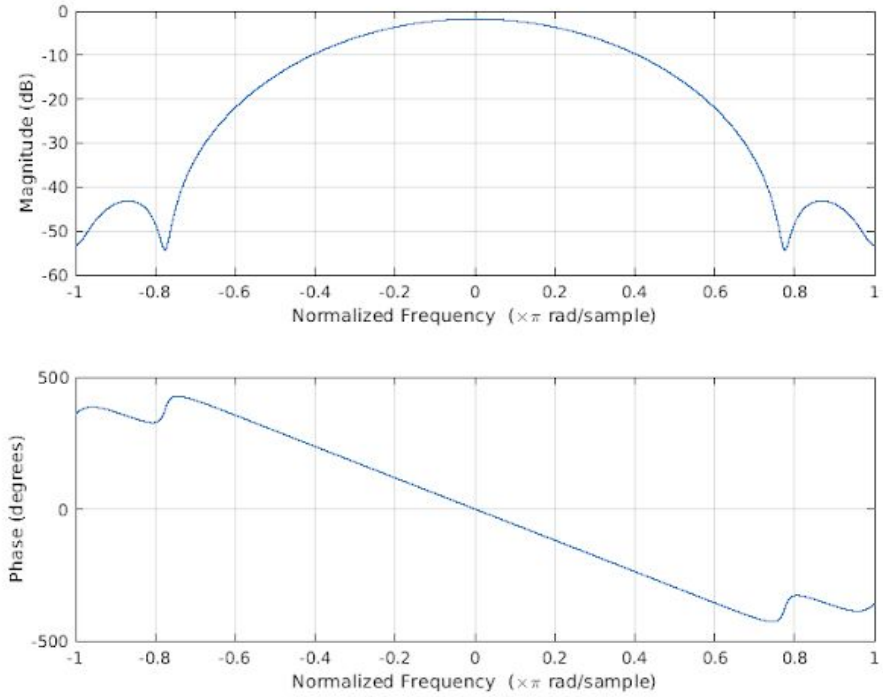

Fig.3_1_a:- frequency response of the FIR filter with filter order N=8

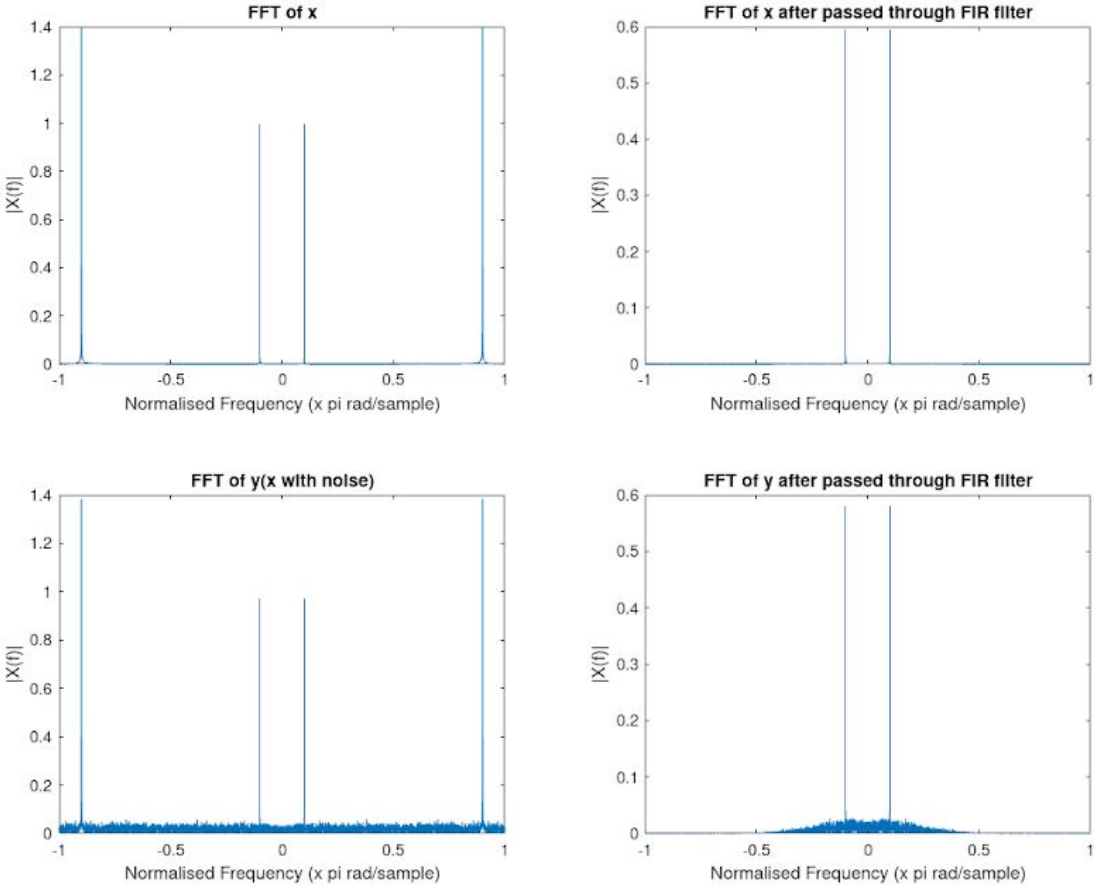

Fig.3_1_b:- FFT of signals when filter order N=8

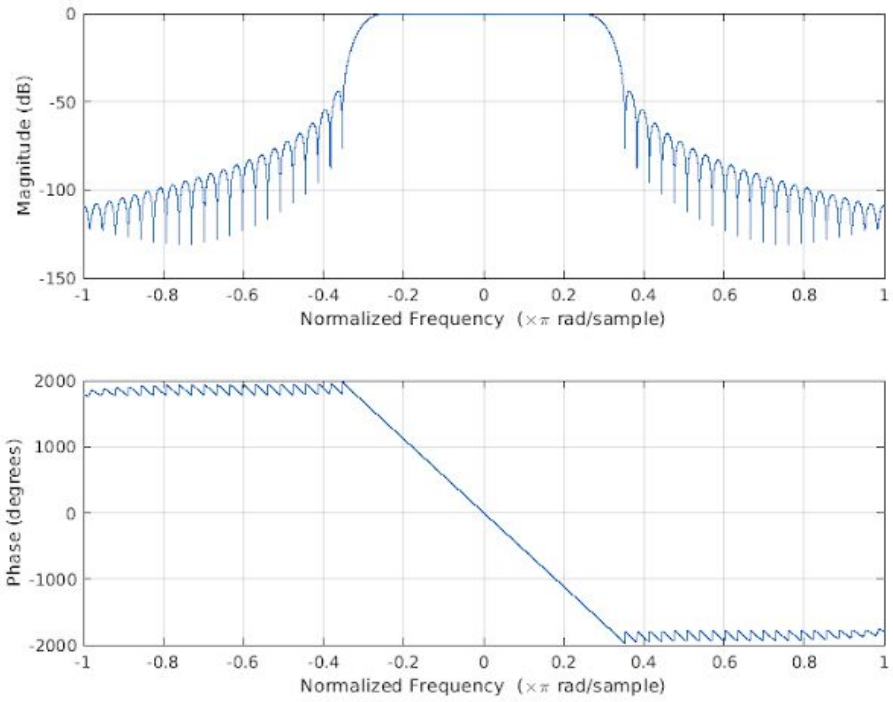

Fig.3_2_a:- frequency response of the FIR filter with filter order N=64

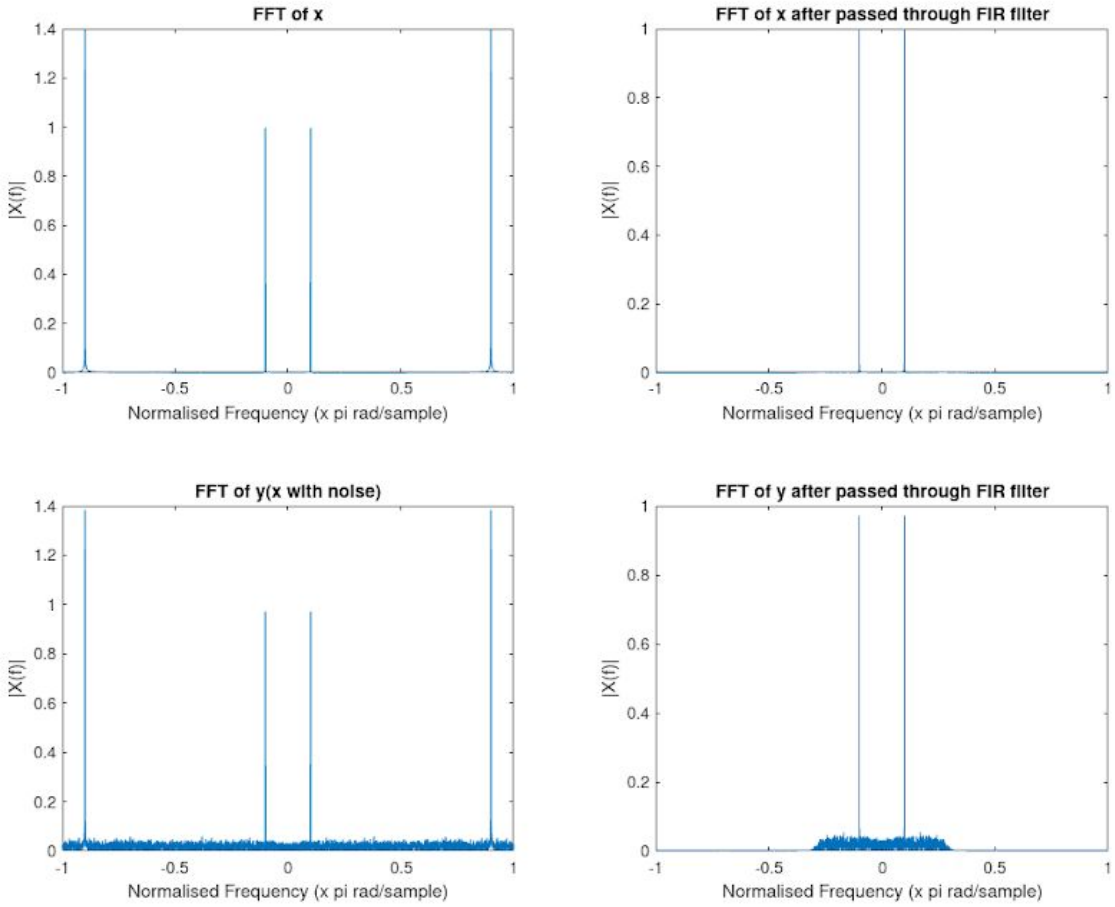

Fig.3_2_b:- FFT of signals when filter order N=64

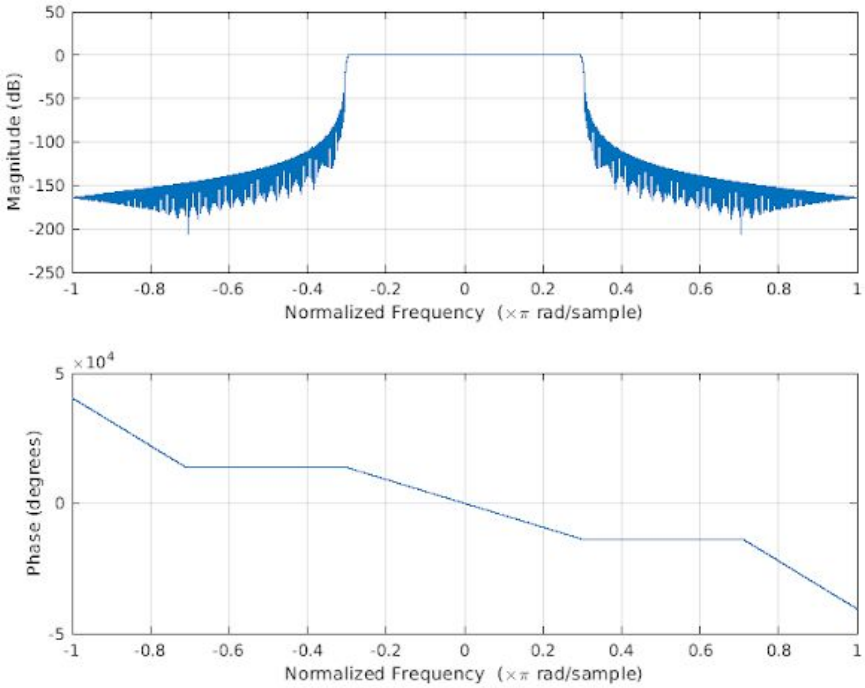

Fig.3_3_a:- frequency response of the FIR filter with filter order N=512

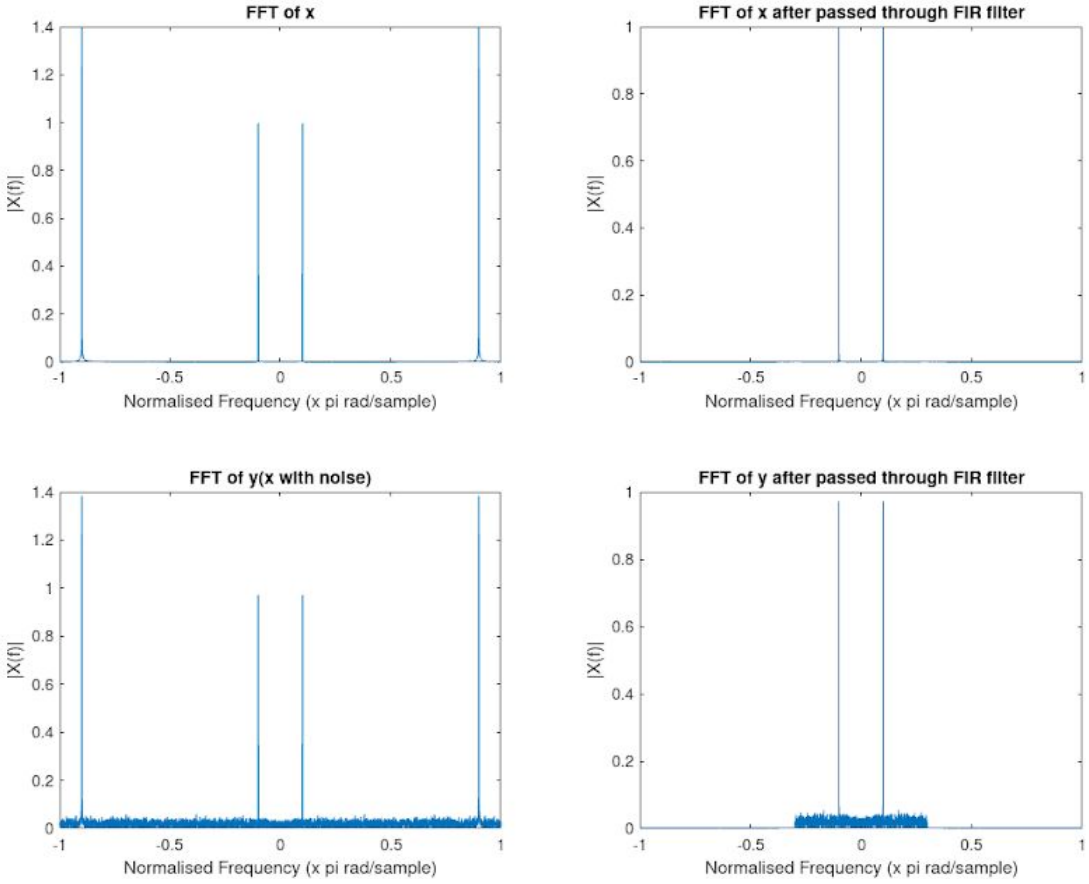

Fig.3_3_b:- FFT of signals when filter order N=512

Hamming Window:-

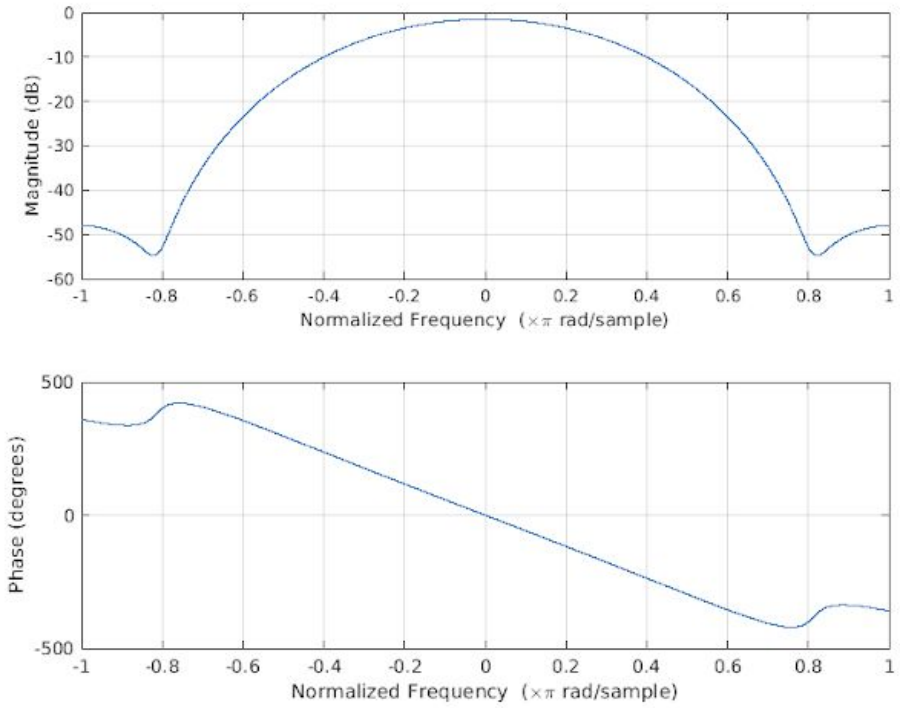

Fig.4_1_a:- frequency response of the FIR filter with filter order N=8

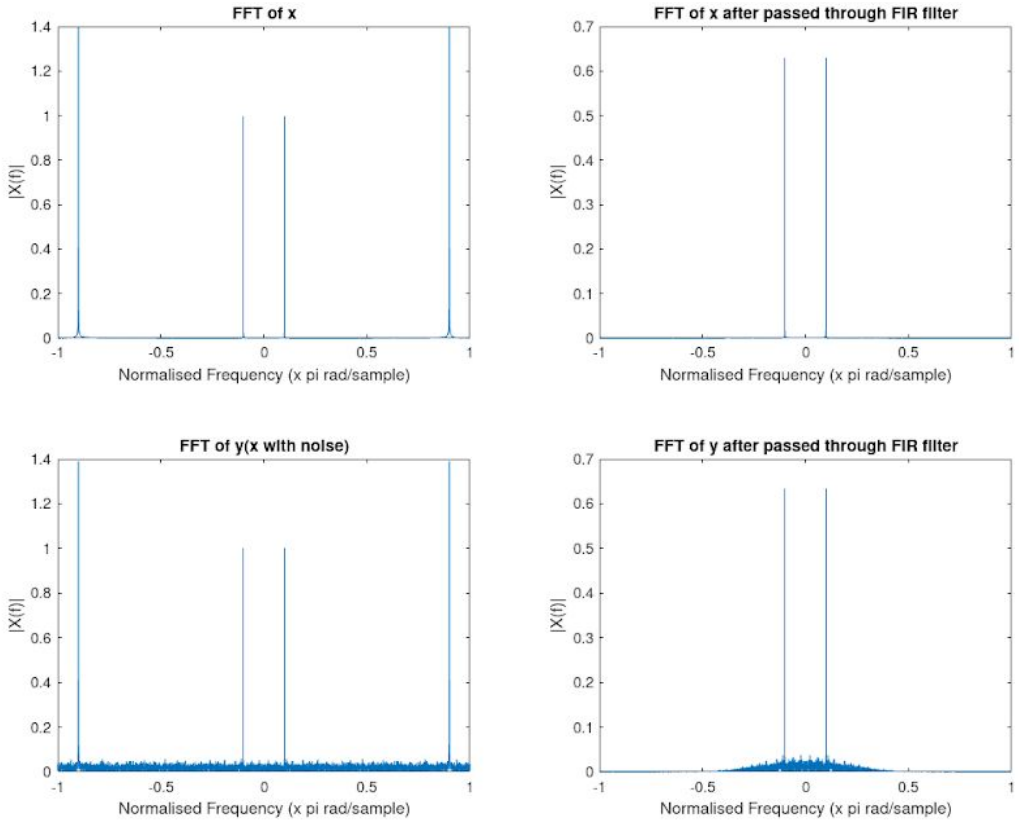

Fig.4_1_b:- FFT of signals when filter order N=8

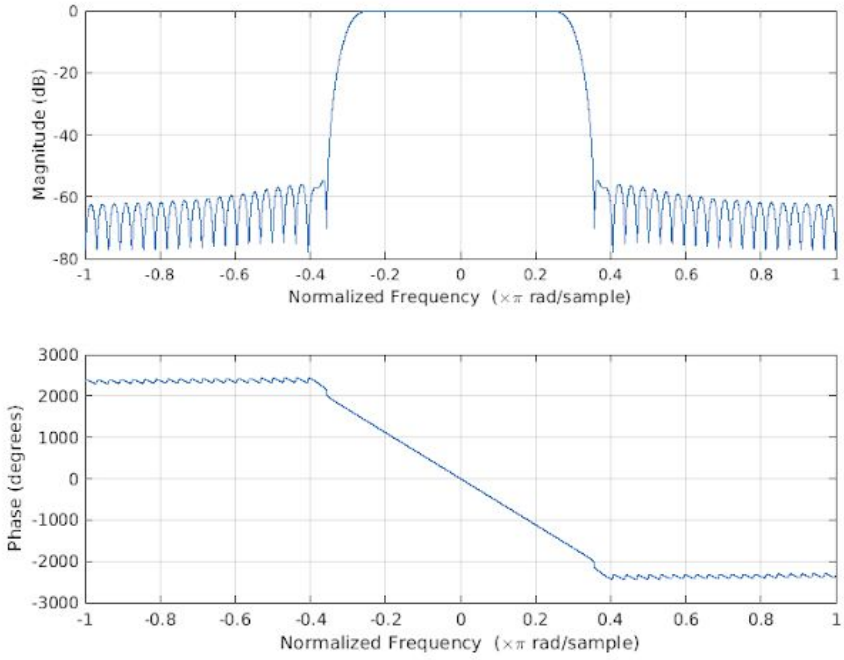

Fig.4_2_a:- frequency response of the FIR filter with filter order N=64

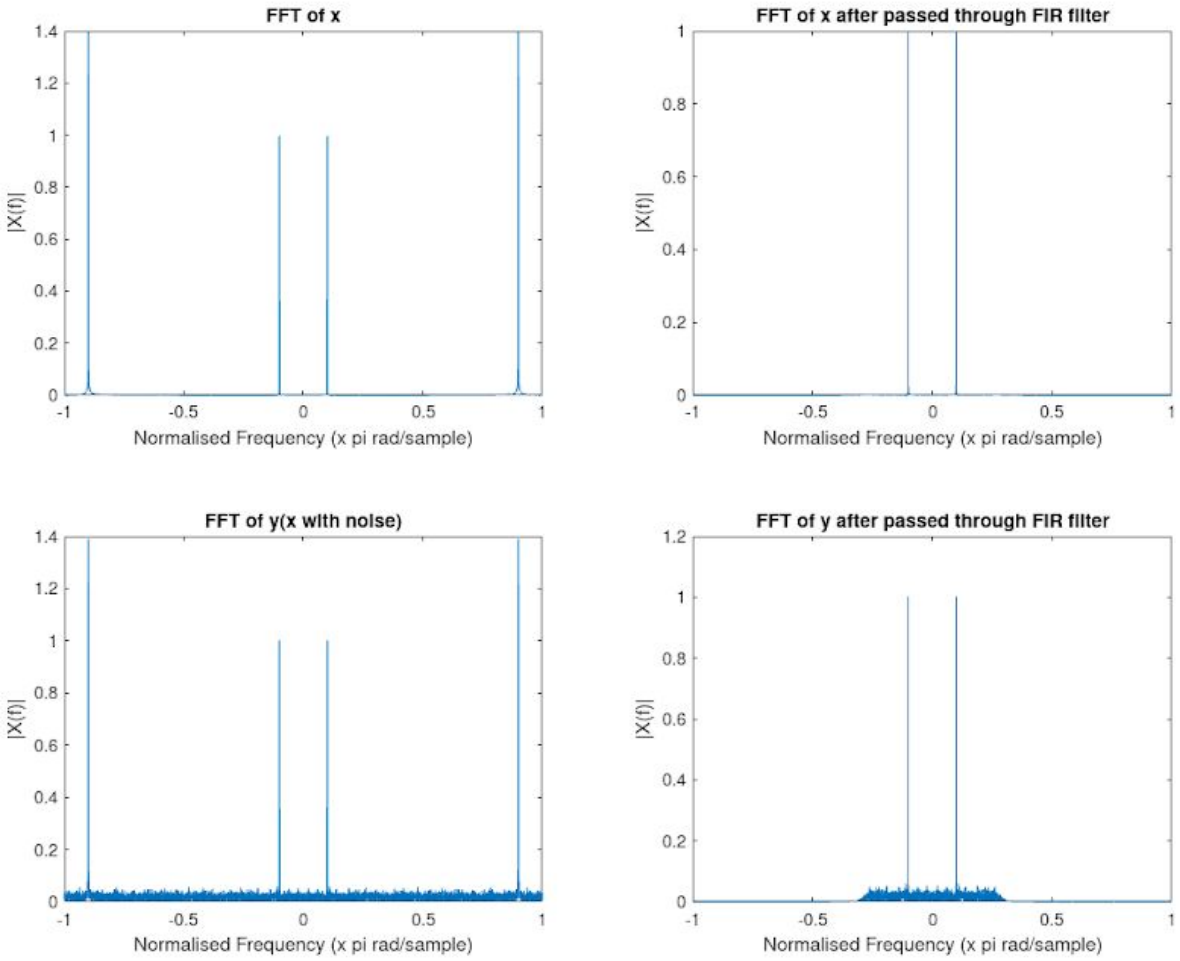

Fig.4_2_b:- FFT of signals when filter order N=64

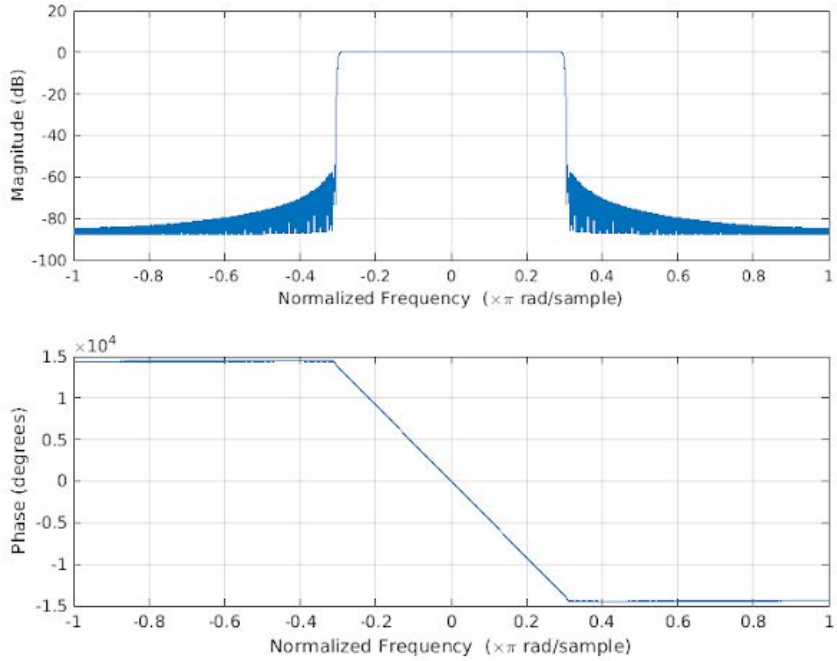

Fig.4_3_a:- frequency response of the FIR filter with filter order N=512

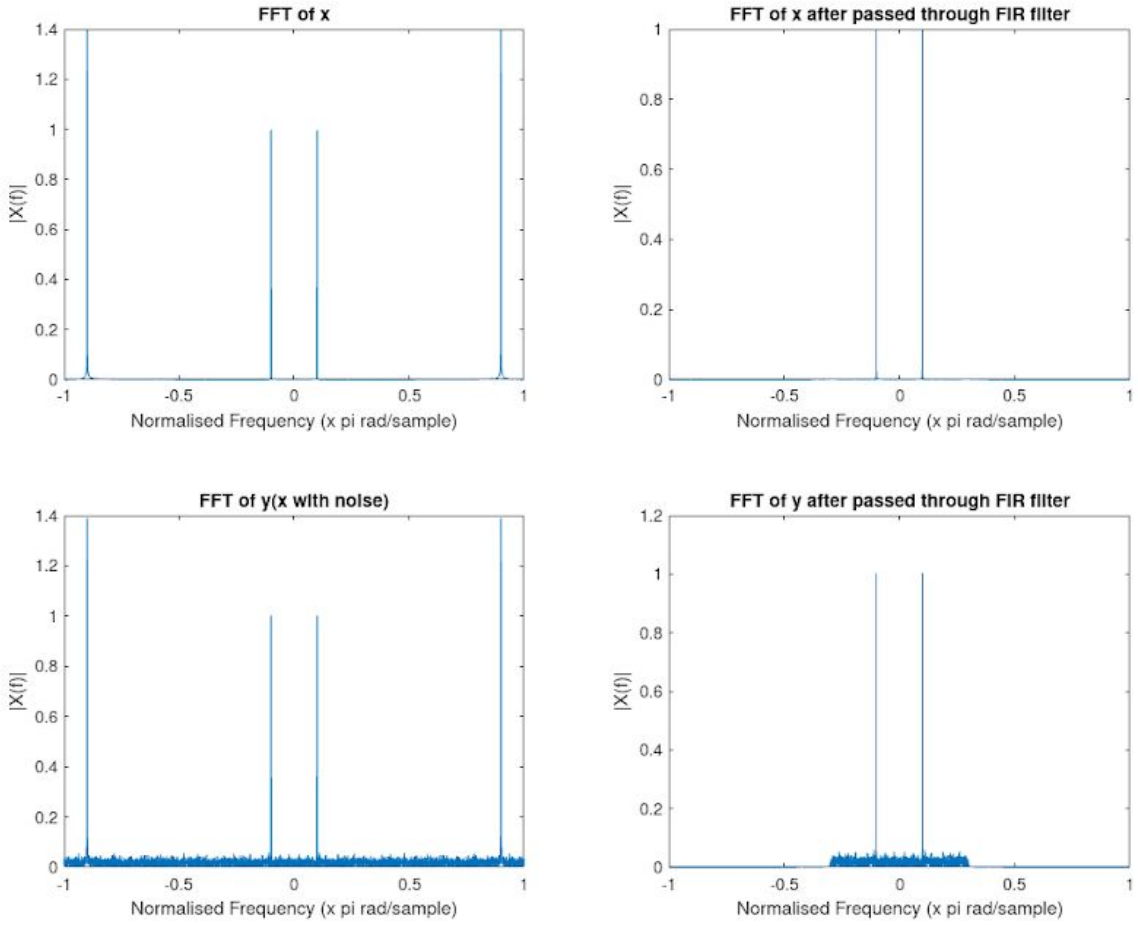

Fig.4_3_b:- FFT of signals when filter order N=512

Blackmnan Window:

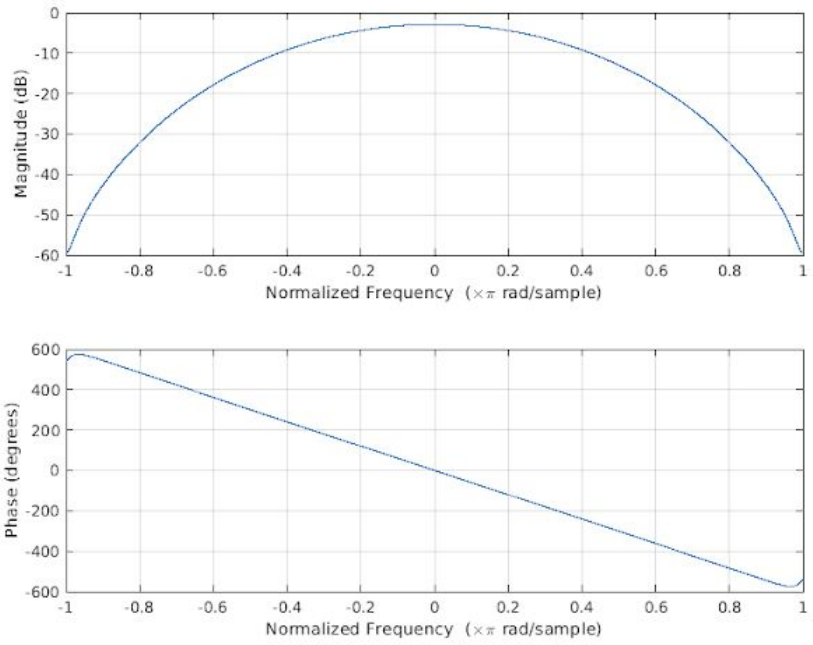

Fig.5_1_a:- frequency response of the FIR filter with filter order N=8

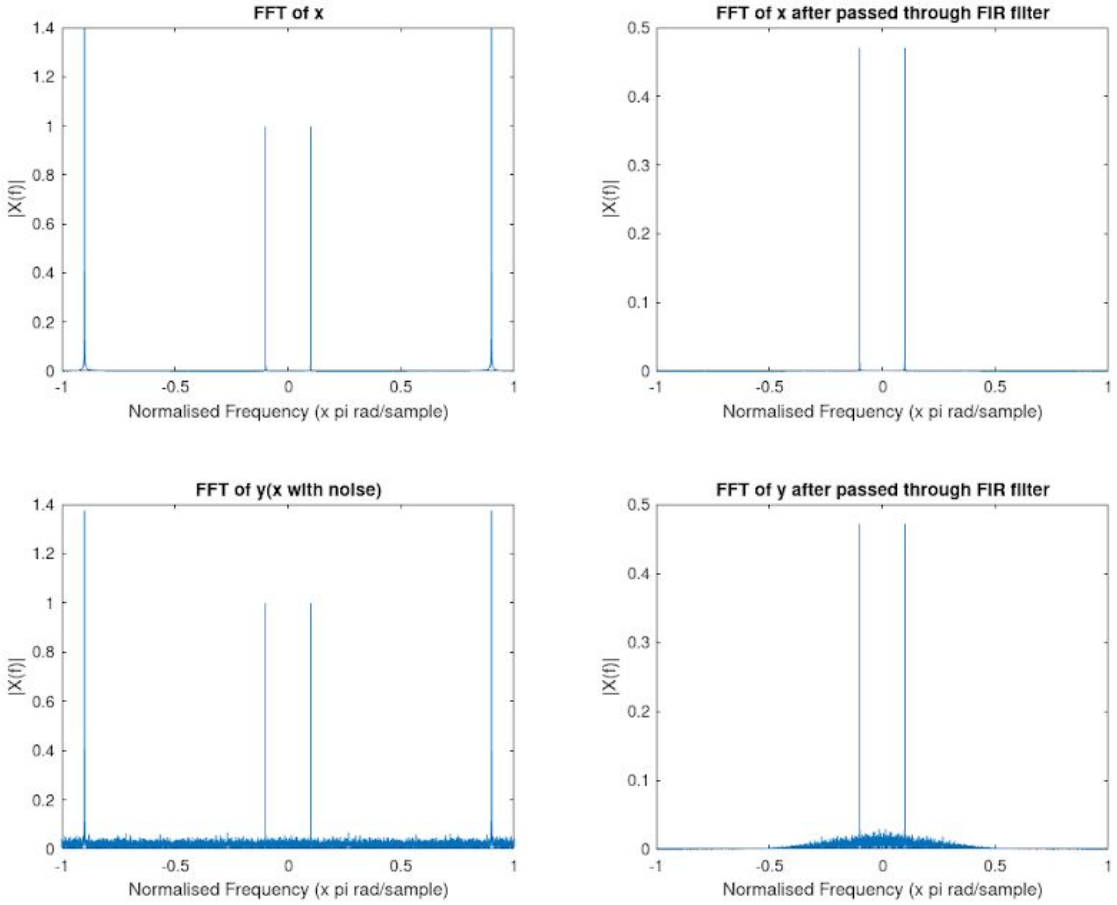

Fig.5_1_b:- FFT of signals when filter order N=8

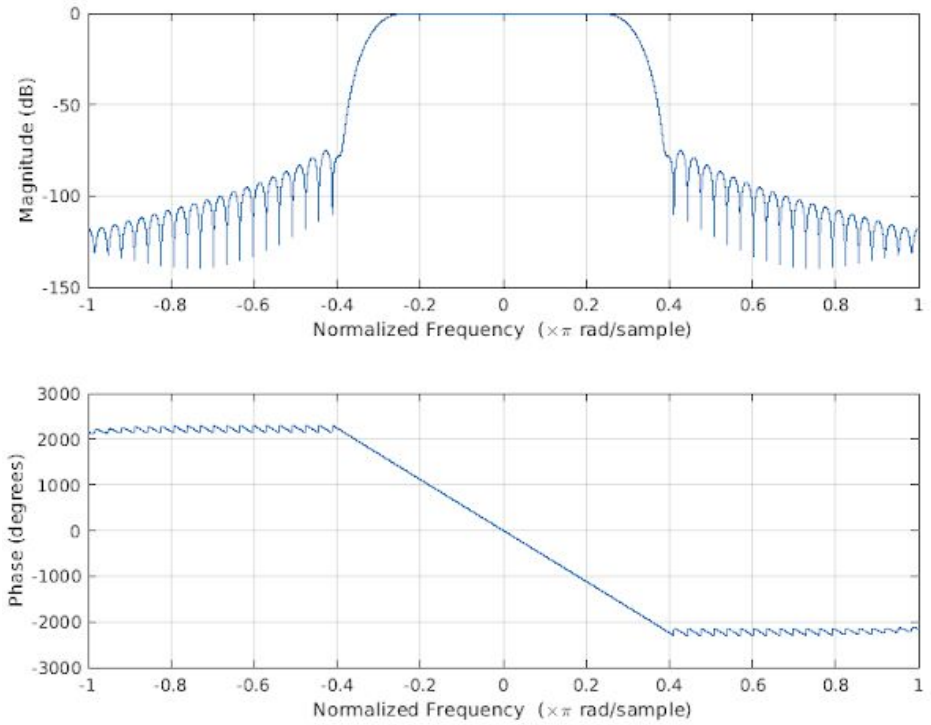

Fig.5_2_a:- frequency response of the FIR filter with filter order N=64

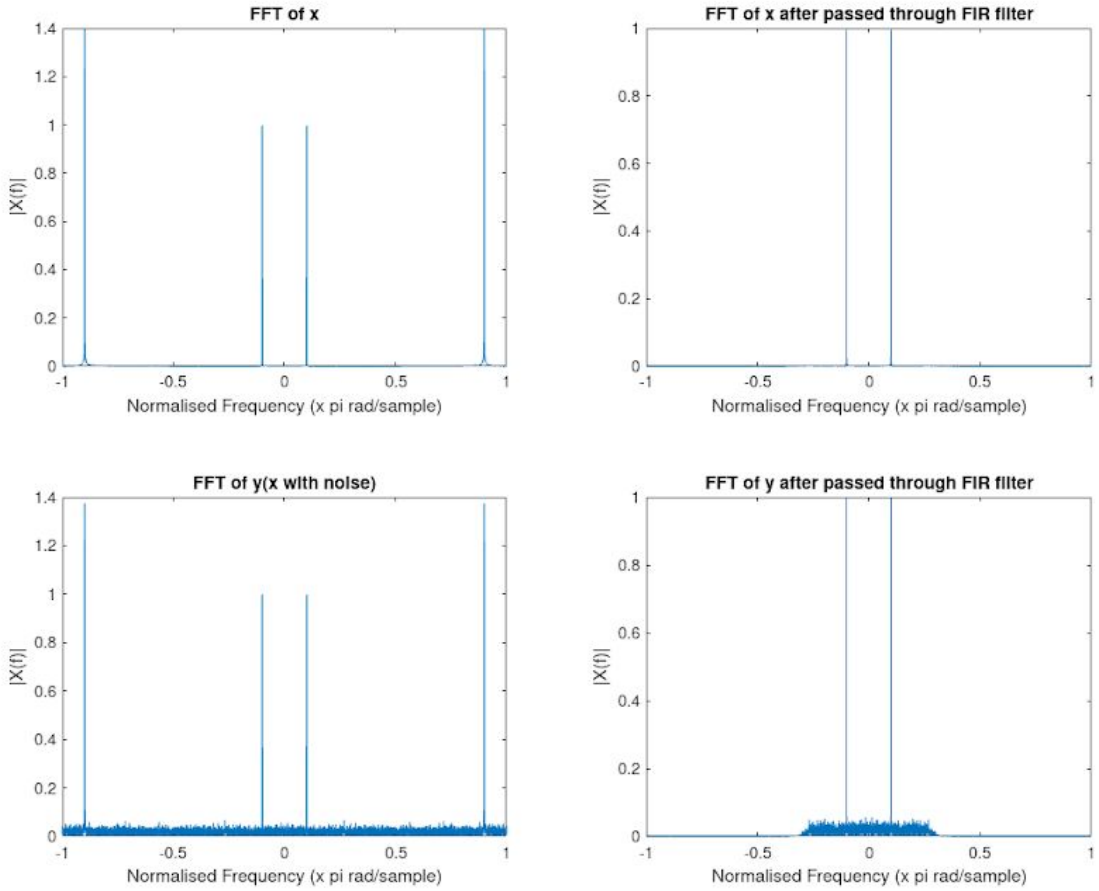

Fig.5_2_b:- FFT of signals when filter order N=64

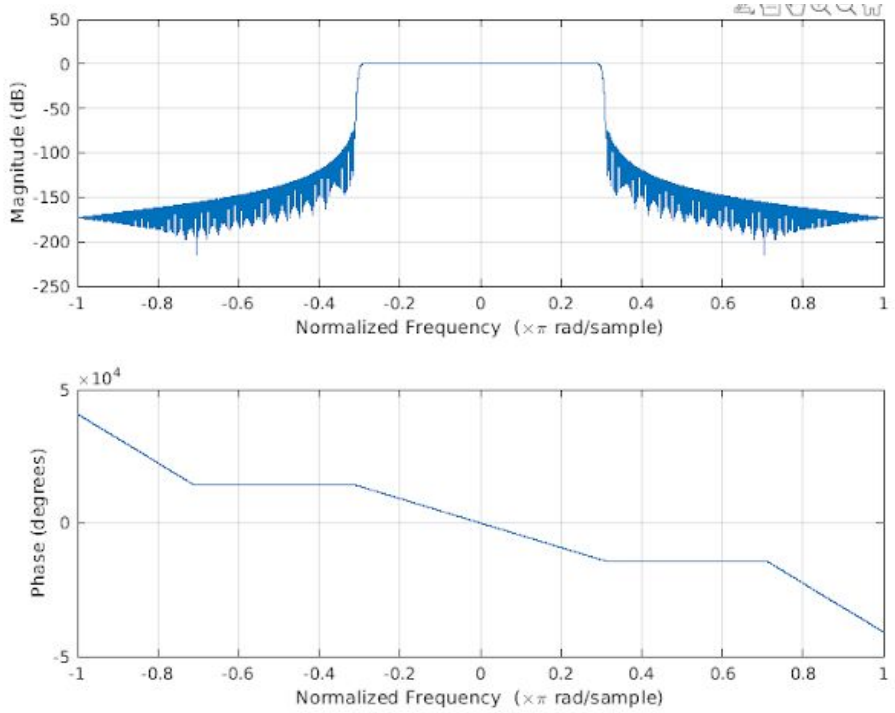

Fig.5_3_a:- frequency response of the FIR filter with filter order N=512

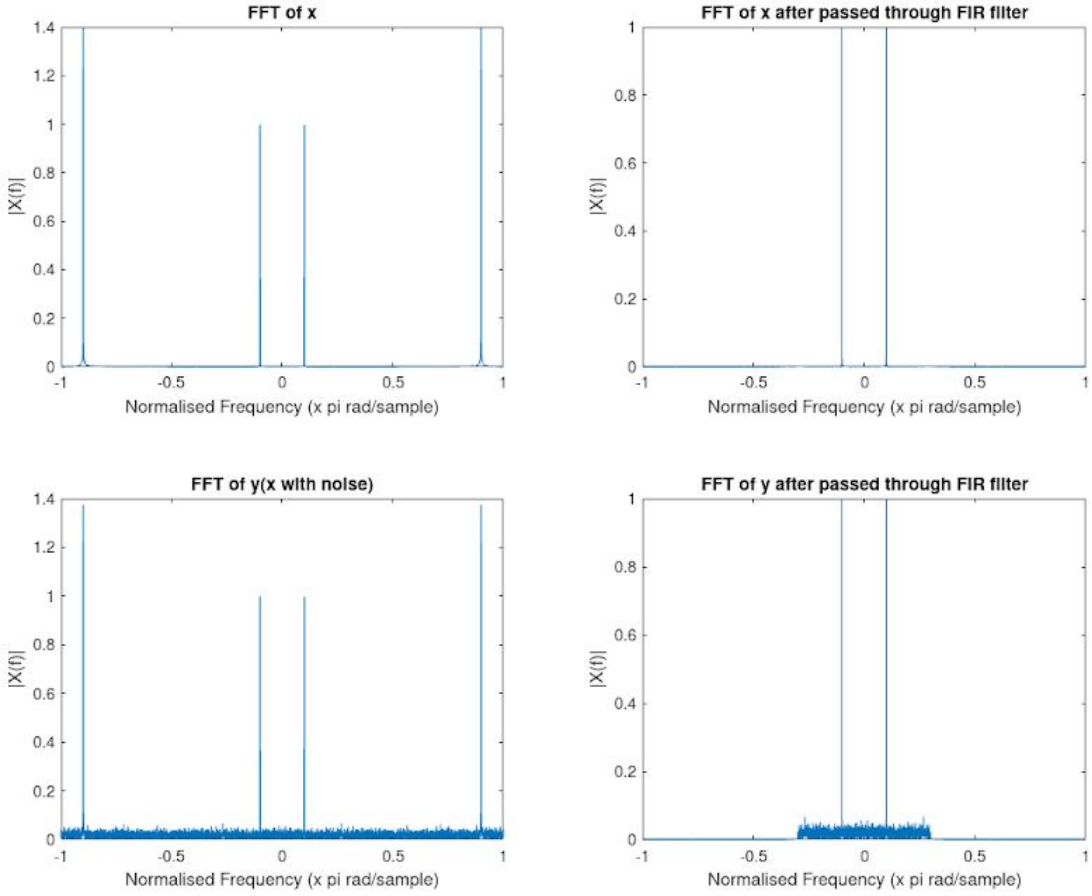

Fig.5_3_b:- FFT of signals when filter order N=512

Results(from plots):-

- Transition width :- the frequency band that exists between the lower bound(stop band cutoff :- 3dB + peak of first lobe in stopband) and upper bound(passband peak - 3dB).
- Peak of first lobe :- peak of first lobe in stop band.
- Maximum attenuation in stop band.
- Signal amplitude=10log(summation(x^2)).
- Noise amplitude=10log(summation(s^2)).
- SNR=signal amplitude-noise amplitude.

Rectangular window			
N	Signal Amplitude(dB)	Noise Amplitude(dB)	SNR(dB)
without filter	50.0016	45.9475	4.0541
8	44.0488	40.3786	3.6702
64	43.0273	40.9128	2.1145
512	43.0491	40.9551	2.094

Rectangular window			
N	Transition width	Peak of First Lobe	Maximum stopband attenuation
8	0.1744 pi	-20.528 dB	-33.9094 dB
64	0.0169 pi	-20.7161	-55.1212 dB
512	0.0026 pi	-21.0084	-65.8884 dB

Triangular Window			
N	Signal Amplitude(dB)	Noise Amplitude(dB)	SNR(dB)
without filter	50.0016	45.9475	4.0541
8	46.8029	43.1792	3.6237
64	43.1688	41.2085	1.9603
512	43.0793	40.9928	2.0865

Triangular Window			
N	Transition width	Peak of First Lobe	Maximum stopband attenuation
8	0.377 pi	-24.7438 dB	-33.8739 dB
64	0.0266 pi	no lobe	-39.0103 dB
512	0.0035 pi	no lobe	-58.0884 dB

Hanning Window			
N	Signal Amplitude(dB)	Noise Amplitude(dB)	SNR(dB)
without filter	50.0016	45.9305	4.0711
8	38.5473	35.2531	3.2942
64	43.0196	40.4618	2.5578
512	43.0278	40.7797	2.248

Hanning Window			
N	Transition width	Peak of First Lobe	Maximum stopband attenuation
8	0.5088 pi	-43.1577 dB	-54.361 dB
64	0.0588 pi	-43.94 dB	-130.683 dB
512	0.0073 pi	-44.395 dB	-189.748 dB

Hamming Window			
N	Signal Amplitude(dB)	Noise Amplitude(dB)	SNR(dB)
without filter	50.0016	46.0208	3.9807
8	39.0432	35.5719	3.4713
64	43.0202	40.4858	2.5343
512	43.0295	40.7594	2.27

Hamming window			
N	Transition width	Peak of First Lobe	Maximum stopband attenuation
8	0.5268 pi	-47.9964 dB	-54.7584 dB
64	0.0709 pi	-55.0686 dB	-77.9226 dB
512	0.0074 pi	-54.2108 dB	-87.7121 dB

Blackman Window			
N	Signal Amplitude(dB)	Noise Amplitude(dB)	SNR(dB)
without filter	50.0016	45.8955	4.1061
8	36.5065	33.295	3.2115
64	43.0112	40.1792	2.832
512	43.0277	40.6041	2.4236

Blackman Window			
N	Transition width	Peak of First Lobe	Maximum stopband attenuation
8	0.6939 pi	No lobe	-59.1536 dB
64	0.1018 pi	-76.1861 dB	-139.829 dB
512	0.0107 pi	-75.4508 dB	-194.72 dB

Discussion and observations:-

- The experiment presents a window methodology for the design of FIR filters, based on an approximation of the desired filter frequency response by using different window functions, with a simple procedure that starts from the usual design specifications.
- This experiment shows,with the increase in FIR filter order(N) the transition width may decrease and the maximum stop band attenuation increases but the ripples in the pass band may not be reduced(if there are any).Ripples are observed in case for triangular and rectangular windows.
- The hamming ,hanning,blackman windows eliminate the ripples in the pass band regardless of filter order.

- Experiment shows that for a particular window,The side peak lobe in the stop band doesn't depend on the filter order.
- With increasing filter order(N) the FIR filter tends to show the properties of ideal low pass filter when the sinc pulse is used to make a FIR filter.
- This experiment shows that the SNR value of the signal remains almost the same(for higher N) irrespective of the window used.
- This experiment concludes that Blackman window is better than other windows as it has high maximum stopband attenuation , low transition width and low side lobe peak in stopband comparatively.

APPENDIX:

In the below codes part of the code is same for all the different window function,
The change in code for diff window functions is highlighted with **RED**
The common part of code for all different functions are in **BLACK**

With rectangular window:-

For part-A: defining the FIR filter

```
clear; %% clears the previously stored values
%% For N=8;
N= 8; %% filter length
hd = zeros(1,N); %% initializing the desired impulse response
k= ceil((N-1)/2); %% defining the centre for the impulse response
for i=1:N
if(i==k)
continue
end
hd(i)= (sin(0.3*pi*(i-k)))/(pi*(i-k)); %% defining the desired impulse response:sinc function
end
hd(k) = 0.3;
w = rectwin(N); %% defining the rectangular window
h = hd.*transpose(w); %% product of desired impulse response and window function:FIR filter.
omega = -pi:pi/10000:pi;
figure(1);
freqz(h,1,omega); %% frequency response of the filter.
%% For N=64
N= 64; %% filter length
hd = zeros(1,N); %% initializing the desired impulse response
k= ceil((N-1)/2); %% defining the centre for the impulse response
for i=1:N
if(i==k)
continue
end
hd(i)= (sin(0.3*pi*(i-k)))/(pi*(i-k)); %% defining the desired impulse response:sinc function
end
hd(k) = 0.3;
w = rectwin(N); %% defining the rectangular window
h = hd.*transpose(w); %% product of desired impulse response and window function:FIR filter.
omega = -pi:pi/10000:pi;
figure(2);
```

```
freqz(h,1,omega); %% frequency response of the filter.
%% For N=512
N= 512;
hd = zeros(1,N); %% initializing the desired impulse response
k= ceil((N-1)/2); %% defining the centre for the impulse response
for i=1:N
if(i==k)
continue
end
hd(i)= (sin(0.3*pi*(i-k)))/(pi*(i-k)); %% defining the desired impulse response:sinc function
end
hd(k) = 0.3;
w = rectwin(N); %% defining the rectangular window
h = hd.*transpose(w); %% product of desired impulse response and window function:FIR filter.
omega = -pi:pi/10000:pi;
figure(3);
freqz(h,1,omega); %% frequency response of the filter.
```

For part-B: filtering and SNR calculation

```
Np=10000; %% signal length
n=0:1:Np; %%discrete time sampling
x=2*cos(0.1*pi*n)+4*cos(0.9*pi*n); %%signal
s=2*transpose(randn(Np+1,1)); %%white noise
y=x+s; %% signal with the noise
freq=linspace(-pi,pi,length(x))/pi; %%frequency sampling for plotting in frequency domain

%%SNR calculation before passing through FIR filter
signal_amplitude = 10*log10(sum(abs(x).^2)); %% summed squared magnitude of the signal, x
noise_amplitude = 10*log10(sum(abs(s).^2)); %% summed squared magnitude of the noise, s
SNR = signal_amplitude-noise_amplitude; %% Formulae for SNR in db
%%Display the SNR and signal amplitude and noise amplitude
disp("Signal values before passing through filters")
disp("Signal amplitude:");
disp(signal_amplitude);
disp("Noise amplitude:");
disp(noise_amplitude);
disp("SNR Calculated value:")
disp(SNR);
%% For N=8
N= 8; %% filter length
hd = zeros(1,N); %% initializing the desired impulse response
k= ceil((N-1)/2); %% defining the centre for the impulse response
for i=1:N
if(i==k)
continue
end
hd(i)= (sin(0.3*pi*(i-k)))/(pi*(i-k)); %% defining the desired impulse response:sinc function
end
hd(k) = 0.3;
w = rectwin(N); %% defining the rectangular window
h = hd.*transpose(w); %% product of desired impulse response and window function:FIR filter.
xout=filtfilt(h,1,x); %% filtered output of the signal x when passed through FIR filter h.
sout=filtfilt(h,1,s); %% filtered output of the signal s when passed through FIR filter h
yout=filtfilt(h,1,y); %% filtered output of the signal y when passed through FIR filter h.
```

```
%%Actual frequency response of the signals(x and y) and Filtered frequency responses
figure(1);
subplot(221)
plot(freq,abs(fftshift(fft(x)))/Np);
xlabel("Normalised Frequency (x pi rad/sample)");
ylabel("|X(f)|");
title("FFT of x");
subplot(222)
plot(freq,abs(fftshift(fft(xout)))/Np);
xlabel("Normalised Frequency (x pi rad/sample)");
ylabel("|X(f)|");
title("FFT of x after passed through FIR filter");
subplot(223)
plot(freq,abs(fftshift(fft(y)))/Np);
xlabel("Normalised Frequency (x pi rad/sample)");
ylabel("|X(f)|");
title("FFT of y(x with noise)");
subplot(224)
plot(freq,abs(fftshift(fft(yout)))/Np);
xlabel("Normalised Frequency (x pi rad/sample)");
ylabel("|X(f)|");
title("FFT of y after passed through FIR filter");

%%SNR calculation before passing through FIR filter
signal_amplitude = 10*log10(sum(abs(xout).^2)); %% summed squared magnitude of the signal, xout
noise_amplitude = 10*log10(sum(abs(sout).^2)); %% summed squared magnitude of the noise, sout
SNR = signal_amplitude-noise_amplitude; %% Formulae for SNR in db
%%Display the SNR and signal amplitude and noise amplitude
disp("signal after passed through FIR filter with filter length N=8")
disp("Signal amplitude:");
disp(signal_amplitude);
disp("Noise amplitude:");
disp(noise_amplitude);
disp("SNR Calculated value:")
disp(SNR);
%% For N=64
N= 64; %% filter length
hd = zeros(1,N); %% initializing the desired impulse response
k= ceil((N-1)/2); %% defining the centre for the impulse response
for i=1:N
if(i==k)
continue
end
hd(i)= (sin(0.3*pi*(i-k)))/(pi*(i-k)); %% defining the desired impulse response:sinc function
end
hd(k) = 0.3;
w = rectwin(N); %% defining the rectangular window
h = hd.*transpose(w); %% product of desired impulse response and window function:FIR filter.
xout=filtfilt(h,1,x); %% filtered output of the signal x when passed through FIR filter h.
sout=filtfilt(h,1,s); %% filtered output of the signal s when passed through FIR filter h
yout=filtfilt(h,1,y); %% filtered output of the signal y when passed through FIR filter h.

%%Actual frequency response of the signals(x and y) and Filtered frequency responses
figure(2);
subplot(221)
plot(freq,abs(fftshift(fft(x)))/Np);
```

```
xlabel("Normalised Frequency (x pi rad/sample)");
ylabel("|X(f)|");
title("FFT of x");
subplot(222)
plot(freq,abs(fftshift(fft(xout)))/Np);
xlabel("Normalised Frequency (x pi rad/sample)");
ylabel("|X(f)|");
title("FFT of x after passed through FIR filter");
subplot(223)
plot(freq,abs(fftshift(fft(y)))/Np);
xlabel("Normalised Frequency (x pi rad/sample)");
ylabel("|X(f)|");
title("FFT of y(x with noise)");
subplot(224)
plot(freq,abs(fftshift(fft(yout)))/Np);
xlabel("Normalised Frequency (x pi rad/sample)");
ylabel("|X(f)|");
title("FFT of y after passed through FIR filter");

%%SNR calculation before passing through FIR filter
signal_amplitude = 10*log10(sum(abs(xout).^2)); %% summed squared magnitude of the signal, xout
noise_amplitude = 10*log10(sum(abs(sout).^2)); %% summed squared magnitude of the noise, sout
SNR = signal_amplitude-noise_amplitude; %% Formulae for SNR in db
%%Display the SNR and signal amplitude and noise amplitude
disp("signal after passed through FIR filter with filter length N=64")
disp("Signal amplitude:");
disp(signal_amplitude);
disp("Noise amplitude:");
disp(noise_amplitude);
disp("SNR Calculated value:")
disp(SNR);
%% For N=512
N= 512; %% filter length
hd = zeros(1,N); %% initializing the desired impulse response
k= ceil((N-1)/2); %% defining the centre for the impulse response
for i=1:N
if(i==k)
continue
end
hd(i)= (sin(0.3*pi*(i-k)))/(pi*(i-k)); %% defining the desired impulse response:sinc function
end
hd(k) = 0.3;
w = rectwin(N); %% defining the rectangular window
h = hd.*transpose(w); %% product of desired impulse response and window function:FIR filter.
xout=filtfilt(h,1,x); %% filtered output of the signal x when passed through FIR filter h.
sout=filtfilt(h,1,s); %% filtered output of the signal s when passed through FIR filter h
yout=filtfilt(h,1,y); %% filtered output of the signal y when passed through FIR filter h.

%%Actual frequency response of the signals(x and y) and Filtered frequency responses
figure(3);
subplot(221)
plot(freq,abs(fftshift(fft(x)))/Np);
xlabel("Normalised Frequency (x pi rad/sample)");
ylabel("|X(f)|");
title("FFT of x");
subplot(222)
```

```
plot(freq,abs(fftshift(fft(xout)))/Np);
xlabel("Normalised Frequency (x pi rad/sample)");
ylabel("|X(f)|");
title("FFT of x after passed through FIR filter");
subplot(223)
plot(freq,abs(fftshift(fft(y)))/Np);
xlabel("Normalised Frequency (x pi rad/sample)");
ylabel("|X(f)|");
title("FFT of y(x with noise)");
subplot(224)
plot(freq,abs(fftshift(fft(yout)))/Np);
xlabel("Normalised Frequency (x pi rad/sample)");
ylabel("|X(f)|");
title("FFT of y after passed through FIR filter");

%%SNR calculation before passing through FIR filter
signal_amplitude = 10*log10(sum(abs(xout).^2)); %% summed squared magnitude of the signal, xout
noise_amplitude = 10*log10(sum(abs(sout).^2)); %% summed squared magnitude of the noise, sout
SNR = signal_amplitude-noise_amplitude; %% Formulae for SNR in db
%%Display the SNR and signal amplitude and noise amplitude
disp("signal after passed through FIR filter with filter length N=512")
disp("Signal amplitude:");
disp(signal_amplitude);
disp("Noise amplitude:");
disp(noise_amplitude);
disp("SNR Calculated value:")
disp(SNR);
```

With Triangular window:-

For part-A: defining the FIR filter

```
clear; %% clears the previously stored values
%% For N=8;
N= 8; %% filter length
hd = zeros(1,N); %% initializing the desired impulse response
k= ceil((N-1)/2); %% defining the centre for the impulse response
for i=1:N
if(i==k)
continue
end
hd(i)= (sin(0.3*pi*(i-k)))/(pi*(i-k)); %% defining the desired impulse response:sinc function
end
hd(k) = 0.3;
w = zeros(1,N);
for i=1:N
w(i)=1-2*(i-1-(N-1)/2)/(N-1); %% defining the triangular window
end
h = hd.*w; %% product of desired impulse response and window function:FIR filter.
omega = -pi:pi/10000:pi;
figure(1);
freqz(h,1,omega); %% frequency response of the filter.
%% For N=64
N= 64; %% filter length
hd = zeros(1,N); %% initializing the desired impulse response
```

```
k= ceil((N-1)/2); %% defining the centre for the impulse response
for i=1:N
if(i==k)
continue
end
hd(i)= (sin(0.3*pi*(i-k)))/(pi*(i-k)); %% defining the desired impulse response:sinc function
end
hd(k) = 0.3;
w = zeros(1,N);
for i=1:N
w(i)=1-2*(i-1-(N-1)/2)/(N-1); %% defining the triangular window
end
h = hd.*w; %% product of desired impulse response and window function:FIR filter.
omega = -pi:pi/10000:pi;
figure(2);
freqz(h,1,omega); %% frequency response of the filter.
%% For N=512
N= 512; %% filter length
hd = zeros(1,N); %% initializing the desired impulse response
k= ceil((N-1)/2); %% defining the centre for the impulse response
for i=1:N
if(i==k)
continue
end
hd(i)= (sin(0.3*pi*(i-k)))/(pi*(i-k)); %% defining the desired impulse response:sinc function
end
hd(k) = 0.3;
w = zeros(1,N);
for i=1:N
w(i)=1-2*(i-1-(N-1)/2)/(N-1); %% defining the triangular window
end
h = hd.*w; %% product of desired impulse response and window function:FIR filter.
omega = -pi:pi/10000:pi;
figure(3);
freqz(h,1,omega); %% frequency response of the filter.
```

For part-B: filtering and SNR calculation

```
Np=10000; %% signal length
n=0:1:Np; %%discrete time sampling
x=2*cos(0.1*pi*n)+4*cos(0.9*pi*n); %%signal
s=2*randn(1,Np+1); %%white noise
y=x+s; %% signal with the noise
freq=linspace(-pi,pi,length(x))/pi; %%frequency sampling for plotting in frequency domain

%%SNR calculation before passing through FIR filter
signal_amplitude = 10*log10(sum(abs(x).^2)); %% summed squared magnitude of the signal, x
noise_amplitude = 10*log10(sum(abs(s).^2)); %% summed squared magnitude of the noise, s
SNR = signal_amplitude-noise_amplitude; %% Formulae for SNR in db
%%Display the SNR and signal amplitude and noise amplitude
disp("Signal values before passing through filters")
disp("Signal amplitude:");
disp(signal_amplitude);
disp("Noise amplitude:");
```

```
disp(noise_amplitude);
disp("SNR Calculated value:")
disp(SNR);
%% For N=8
N= 8; %% filter length
hd = zeros(1,N); %% initializing the desired impulse response
k= ceil((N-1)/2); %% defining the centre for the impulse response
for i=1:N
if(i==k)
continue
end
hd(i)= (sin(0.3*pi*(i-k)))/(pi*(i-k)); %% defining the desired impulse response:sinc function
end
hd(k) = 0.3;
w = zeros(1,N);
for i=1:N
w(i)=1-2*(i-1-(N-1)/2)/(N-1); %% defining the triangular window
end
h = hd.*w; %% product of desired impulse response and window function:FIR filter.
xout=filtfilt(h,1,x); %% filtered output of the signal x when passed through FIR filter h.
sout=filtfilt(h,1,s); %% filtered output of the signal s when passed through FIR filter h
yout=filtfilt(h,1,y); %% filtered output of the signal y when passed through FIR filter h.

%%Actual frequency response of the signals(x and y) and Filtered frequency responses
figure(1);
subplot(221)
plot(freq,abs(fftshift(fft(x)))/Np);
xlabel("Normalised Frequency (x pi rad/sample)");
ylabel("|X(f)|");
title("FFT of x");
subplot(222)
plot(freq,abs(fftshift(fft(xout)))/Np);
xlabel("Normalised Frequency (x pi rad/sample)");
ylabel("|X(f)|");
title("FFT of x after passed through FIR filter");
subplot(223)
plot(freq,abs(fftshift(fft(y)))/Np);
xlabel("Normalised Frequency (x pi rad/sample)");
ylabel("|X(f)|");
title("FFT of y(x with noise)");
subplot(224)
plot(freq,abs(fftshift(fft(yout)))/Np);
xlabel("Normalised Frequency (x pi rad/sample)");
ylabel("|X(f)|");
title("FFT of y after passed through FIR filter");

%%SNR calculation before passing through FIR filter
signal_amplitude = 10*log10(sum(abs(xout).^2)); %% summed squared magnitude of the signal, xout
noise_amplitude = 10*log10(sum(abs(sout).^2)); %% summed squared magnitude of the noise, sout
SNR = signal_amplitude-noise_amplitude; %% Formulae for SNR in db
%%Display the SNR and signal amplitude and noise amplitude
disp("signal after passed through FIR filter with filter length N=8")
disp("Signal amplitude:");
disp(signal_amplitude);
disp("Noise amplitude:");
disp(noise_amplitude);
```

```
disp("SNR Calculated value:")
disp(SNR);
%% For N=64
N= 64; %% filter length
hd = zeros(1,N); %% initializing the desired impulse response
k= ceil((N-1)/2); %% defining the centre for the impulse response
for i=1:N
if(i==k)
continue
end
hd(i)= (sin(0.3*pi*(i-k)))/(pi*(i-k)); %% defining the desired impulse response:sinc function
end
hd(k) = 0.3;
w = zeros(1,N);
for i=1:N
w(i)=1-2*(i-1-(N-1)/2)/(N-1); %% defining the triangular window
end
h = hd.*w; %% product of desired impulse response and window function:FIR filter.
xout=filtfilt(h,1,x); %% filtered output of the signal x when passed through FIR filter h.
sout=filtfilt(h,1,s); %% filtered output of the signal s when passed through FIR filter h
yout=filtfilt(h,1,y); %% filtered output of the signal y when passed through FIR filter h.

%%Actual frequency response of the signals(x and y) and Filtered frequency responses
figure(2);
subplot(221)
plot(freq,abs(fftshift(fft(x)))/Np);
xlabel("Normalised Frequency (x pi rad/sample)");
ylabel("|X(f)|");
title("FFT of x");
subplot(222)
plot(freq,abs(fftshift(fft(xout)))/Np);
xlabel("Normalised Frequency (x pi rad/sample)");
ylabel("|X(f)|");
title("FFT of x after passed through FIR filter");
subplot(223)
plot(freq,abs(fftshift(fft(y)))/Np);
xlabel("Normalised Frequency (x pi rad/sample)");
ylabel("|X(f)|");
title("FFT of y(x with noise)");
subplot(224)
plot(freq,abs(fftshift(fft(yout)))/Np);
xlabel("Normalised Frequency (x pi rad/sample)");
ylabel("|X(f)|");
title("FFT of y after passed through FIR filter");

%%SNR calculation before passing through FIR filter
signal_amplitude = 10*log10(sum(abs(xout).^2)); %% summed squared magnitude of the signal, xout
noise_amplitude = 10*log10(sum(abs(sout).^2)); %% summed squared magnitude of the noise, sout
SNR = signal_amplitude-noise_amplitude; %% Formulae for SNR in db
%%Display the SNR and signal amplitude and noise amplitude
disp("signal after passed through FIR filter with filter length N=64")
disp("Signal amplitude:");
disp(signal_amplitude);
disp("Noise amplitude:");
disp(noise_amplitude);
disp("SNR Calculated value:")
```

```
disp(SNR);
%% For N=512
N= 512; %% filter length
hd = zeros(1,N); %% initializing the desired impulse response
k= ceil((N-1)/2); %% defining the centre for the impulse response
for i=1:N
if(i==k)
continue
end
hd(i)= (sin(0.3*pi*(i-k)))/(pi*(i-k)); %% defining the desired impulse response:sinc function
end
hd(k) = 0.3;
w = zeros(1,N);
for i=1:N
w(i)=1-2*(i-1-(N-1)/2)/(N-1); %% defining the triangular window
end
h = hd.*w; %% product of desired impulse response and window function:FIR filter.
xout=filtfilt(h,1,x); %% filtered output of the signal x when passed through FIR filter h.
sout=filtfilt(h,1,s); %% filtered output of the signal s when passed through FIR filter h
yout=filtfilt(h,1,y); %% filtered output of the signal y when passed through FIR filter h.

%%Actual frequency response of the signals(x and y) and Filtered frequency responses
figure(3);
subplot(221)
plot(freq,abs(fftshift(fft(x)))/Np);
xlabel("Normalised Frequency (x pi rad/sample)");
ylabel("|X(f)|");
title("FFT of x");
subplot(222)
plot(freq,abs(fftshift(fft(xout)))/Np);
xlabel("Normalised Frequency (x pi rad/sample)");
ylabel("|X(f)|");
title("FFT of x after passed through FIR filter");
subplot(223)
plot(freq,abs(fftshift(fft(y)))/Np);
xlabel("Normalised Frequency (x pi rad/sample)");
ylabel("|X(f)|");
title("FFT of y(x with noise)");
subplot(224)
plot(freq,abs(fftshift(fft(yout)))/Np);
xlabel("Normalised Frequency (x pi rad/sample)");
ylabel("|X(f)|");
title("FFT of y after passed through FIR filter");

%%SNR calculation before passing through FIR filter
signal_amplitude = 10*log10(sum(abs(xout).^2)); %% summed squared magnitude of the signal, xout
noise_amplitude = 10*log10(sum(abs(sout).^2)); %% summed squared magnitude of the noise, sout
SNR = signal_amplitude-noise_amplitude; %% Formulae for SNR in db
%%Display the SNR and signal amplitude and noise amplitude
disp("signal after passed through FIR filter with filter length N=512")
disp("Signal amplitude:");
disp(signal_amplitude);
disp("Noise amplitude:");
disp(noise_amplitude);
disp("SNR Calculated value:")
disp(SNR);
```

With Hanning window:-

For part-A: defining the FIR filter

```
clear; %% clears the previously stored values
%% For N=8;
N= 8; %% filter length
hd = zeros(1,N); %% initializing the desired impulse response
k= ceil((N-1)/2); %% defining the centre for the impulse response
for i=1:N
if(i==k)
continue
end
hd(i)= (sin(0.3*pi*(i-k)))/(pi*(i-k)); %% defining the desired impulse response:sinc function
end
hd(k) = 0.3;
w = zeros(1,N);
for i=1:N
w(i)=0.5-0.5*cos(2*pi*((i-1)/(N-1))); %% defining the hanning window
end
h = hd.*w; %% product of desired impulse response and window function:FIR filter.
omega = -pi:pi/10000:pi;
figure(1);
freqz(h,1,omega); %% frequency response of the filter.
%% For N=64
N= 64; %% filter length
hd = zeros(1,N); %% initializing the desired impulse response
k= ceil((N-1)/2); %% defining the centre for the impulse response
for i=1:N
if(i==k)
continue
end
hd(i)= (sin(0.3*pi*(i-k)))/(pi*(i-k)); %% defining the desired impulse response:sinc function
end
hd(k) = 0.3;
w = zeros(1,N);
for i=1:N
w(i)=0.5-0.5*cos(2*pi*((i-1)/(N-1))); %% defining the hanning window
end
h = hd.*w; %% product of desired impulse response and window function:FIR filter.
omega = -pi:pi/10000:pi;
figure(2);
freqz(h,1,omega); %% frequency response of the filter.
%% For N=512
N= 512; %% filter length
hd = zeros(1,N); %% initializing the desired impulse response
k= ceil((N-1)/2); %% defining the centre for the impulse response
for i=1:N
if(i==k)
continue
end
hd(i)= (sin(0.3*pi*(i-k)))/(pi*(i-k)); %% defining the desired impulse response:sinc function
end
hd(k) = 0.3;
w = zeros(1,N);
```

```
for i=1:N
w(i)=0.5-0.5*cos(2*pi*((i-1)/(N-1))); %% defining the hanning window
end
h = hd.*w; %% product of desired impulse response and window function:FIR filter.
omega = -pi:pi/10000:pi;
figure(3);
freqz(h,1,omega); %% frequency response of the filter.
```

For part-B: filtering and SNR calculation

```
Np=10000; %% signal length
n=0:1:Np; %%discrete time sampling
x=2*cos(0.1*pi*n)+4*cos(0.9*pi*n); %%signal
s=2*randn(1,Np+1); %%white noise
y=x+s; %% signal with the noise
freq=linspace(-pi,pi,length(x))/pi; %%frequency sampling for plotting in frequency domain

%%SNR calculation before passing through FIR filter
signal_amplitude = 10*log10(sum(abs(x).^2)); %% summed squared magnitude of the signal, x
noise_amplitude = 10*log10(sum(abs(s).^2)); %% summed squared magnitude of the noise, s
SNR = signal_amplitude-noise_amplitude; %% Formulae for SNR in db
%%Display the SNR and signal amplitude and noise amplitude
disp("Signal values before passing through filters")
disp("Signal amplitude:");
disp(signal_amplitude);
disp("Noise amplitude:");
disp(noise_amplitude);
disp("SNR Calculated value:")
disp(SNR);
%% For N=8
N= 8; %% filter length
hd = zeros(1,N); %% initializing the desired impulse response
k= ceil((N-1)/2); %% defining the centre for the impulse response
for i=1:N
if(i==k)
continue
end
hd(i)= (sin(0.3*pi*(i-k)))/(pi*(i-k)); %% defining the desired impulse response:sinc function
end
hd(k) = 0.3;
w = zeros(1,N);
for i=1:N
w(i)=0.5-0.5*cos(2*pi*((i-1)/(N-1))); %% defining the hanning window
end
h = hd.*w; %% product of desired impulse response and window function:FIR filter.
xout=filtfilt(h,1,x); %% filtered output of the signal x when passed through FIR filter h.
sout=filtfilt(h,1,s); %% filtered output of the signal s when passed through FIR filter h
yout=filtfilt(h,1,y); %% filtered output of the signal y when passed through FIR filter h.

%%Actual frequency response of the signals(x and y) and Filtered frequency responses
figure(1);
subplot(221)
plot(freq,abs(fftshift(fft(x)))/Np);
xlabel("Normalised Frequency (x pi rad/sample)");
ylabel("|X(f)|");
title("FFT of x");
```

```
subplot(222)
plot(freq,abs(fftshift(fft(xout)))/Np);
xlabel("Normalised Frequency (x pi rad/sample)");
ylabel("|X(f)|");
title("FFT of x after passed through FIR filter");
subplot(223)
plot(freq,abs(fftshift(fft(y)))/Np);
xlabel("Normalised Frequency (x pi rad/sample)");
ylabel("|X(f)|");
title("FFT of y(x with noise)");
subplot(224)
plot(freq,abs(fftshift(fft(yout)))/Np);
xlabel("Normalised Frequency (x pi rad/sample)");
ylabel("|X(f)|");
title("FFT of y after passed through FIR filter");

%%SNR calculation before passing through FIR filter
signal_amplitude = 10*log10(sum(abs(xout).^2)); %% summed squared magnitude of the signal, xout
noise_amplitude = 10*log10(sum(abs(sout).^2)); %% summed squared magnitude of the noise, sout
SNR = signal_amplitude-noise_amplitude; %% Formulae for SNR in db
%%Display the SNR and signal amplitude and noise amplitude
disp("signal after passed through FIR filter with filter length N=8")
disp("Signal amplitude:");
disp(signal_amplitude);
disp("Noise amplitude:");
disp(noise_amplitude);
disp("SNR Calculated value:")
disp(SNR);
%% For N=64
N= 64; %% filter length
hd = zeros(1,N); %% initializing the desired impulse response
k= ceil((N-1)/2); %% defining the centre for the impulse response
for i=1:N
if(i==k)
continue
end
hd(i)= (sin(0.3*pi*(i-k)))/(pi*(i-k)); %% defining the desired impulse response:sinc function
end
hd(k) = 0.3;
w = zeros(1,N);
for i=1:N
w(i)=0.5-0.5*cos(2*pi*((i-1)/(N-1))); %% defining the hanning window
end
h = hd.*w; %% product of desired impulse response and window function:FIR filter.
xout=filtfilt(h,1,x); %% filtered output of the signal x when passed through FIR filter h.
sout=filtfilt(h,1,s); %% filtered output of the signal s when passed through FIR filter h
yout=filtfilt(h,1,y); %% filtered output of the signal y when passed through FIR filter h.

%%Actual frequency response of the signals(x and y) and Filtered frequency responses
figure(2);
subplot(221)
plot(freq,abs(fftshift(fft(x)))/Np);
xlabel("Normalised Frequency (x pi rad/sample)");
ylabel("|X(f)|");
title("FFT of x");
subplot(222)
```

```
plot(freq,abs(fftshift(fft(xout)))/Np);
xlabel("Normalised Frequency (x pi rad/sample)");
ylabel("|X(f)|");
title("FFT of x after passed through FIR filter");
subplot(223)
plot(freq,abs(fftshift(fft(y)))/Np);
xlabel("Normalised Frequency (x pi rad/sample)");
ylabel("|X(f)|");
title("FFT of y(x with noise)");
subplot(224)
plot(freq,abs(fftshift(fft(yout)))/Np);
xlabel("Normalised Frequency (x pi rad/sample)");
ylabel("|X(f)|");
title("FFT of y after passed through FIR filter");

%%SNR calculation before passing through FIR filter
signal_amplitude = 10*log10(sum(abs(xout).^2)); %% summed squared magnitude of the signal, xout
noise_amplitude = 10*log10(sum(abs(sout).^2)); %% summed squared magnitude of the noise, sout
SNR = signal_amplitude-noise_amplitude; %% Formulae for SNR in db
%%Display the SNR and signal amplitude and noise amplitude
disp("signal after passed through FIR filter with filter length N=64")
disp("Signal amplitude:");
disp(signal_amplitude);
disp("Noise amplitude:");
disp(noise_amplitude);
disp("SNR Calculated value:")
disp(SNR);
%% For N=512
N= 512; %% filter length
hd = zeros(1,N); %% initializing the desired impulse response
k= ceil((N-1)/2); %% defining the centre for the impulse response
for i=1:N
if(i==k)
continue
end
hd(i)= (sin(0.3*pi*(i-k)))/(pi*(i-k)); %% defining the desired impulse response:sinc function
end
hd(k) = 0.3;
w = zeros(1,N);
for i=1:N
w(i)=0.5-0.5*cos(2*pi*((i-1)/(N-1))); %% defining the hanning window
end
h = hd.*w; %% product of desired impulse response and window function:FIR filter.
xout=filtfilt(h,1,x); %% filtered output of the signal x when passed through FIR filter h.
sout=filtfilt(h,1,s); %% filtered output of the signal s when passed through FIR filter h
yout=filtfilt(h,1,y); %% filtered output of the signal y when passed through FIR filter h.

%%Actual frequency response of the signals(x and y) and Filtered frequency responses
figure(3);
subplot(221)
plot(freq,abs(fftshift(fft(x)))/Np);
xlabel("Normalised Frequency (x pi rad/sample)");
ylabel("|X(f)|");
title("FFT of x");
subplot(222)
plot(freq,abs(fftshift(fft(xout)))/Np);
```

```
xlabel("Normalised Frequency (x pi rad/sample)");
ylabel("|X(f)|");
title("FFT of x after passed through FIR filter");
subplot(223)
plot(freq,abs(fftshift(fft(y)))/Np);
xlabel("Normalised Frequency (x pi rad/sample)");
ylabel("|X(f)|");
title("FFT of y(x with noise)");
subplot(224)
plot(freq,abs(fftshift(fft(yout)))/Np);
xlabel("Normalised Frequency (x pi rad/sample)");
ylabel("|X(f)|");
title("FFT of y after passed through FIR filter");

%%SNR calculation before passing through FIR filter
signal_amplitude = 10*log10(sum(abs(xout).^2)); %% summed squared magnitude of the signal, xout
noise_amplitude = 10*log10(sum(abs(sout).^2)); %% summed squared magnitude of the noise, sout
SNR = signal_amplitude-noise_amplitude; %% Formulae for SNR in db
%%Display the SNR and signal amplitude and noise amplitude
disp("signal after passed through FIR filter with filter length N=512")
disp("Signal amplitude:");
disp(signal_amplitude);
disp("Noise amplitude:");
disp(noise_amplitude);
disp("SNR Calculated value:")
disp(SNR);
```

With Hamming window:-

For part-A: defining the FIR filter

```
clear; %% clears the previously stored values
%% For N=8;
N= 8; %% filter length
hd = zeros(1,N); %% initializing the desired impulse response
k= ceil((N-1)/2); %% defining the centre for the impulse response
for i=1:N
if(i==k)
continue
end
hd(i)= (sin(0.3*pi*(i-k)))/(pi*(i-k)); %% defining the desired impulse response:sinc function
end
hd(k) = 0.3;
w = zeros(1,N);
for i=1:N
w(i)=0.54-0.46*cos(2*pi*((i-1)/(N-1))); %% defining the hamming window
end
h = hd.*w; %% product of desired impulse response and window function:FIR filter.
omega = -pi:pi/10000:pi;
figure(1);
freqz(h,1,omega); %% frequency response of the filter.
%% For N=64
N= 64; %% filter length
hd = zeros(1,N); %% initializing the desired impulse response
k= ceil((N-1)/2); %% defining the centre for the impulse response
for i=1:N
```

```
if(i==k)
continue
end
hd(i)= (sin(0.3*pi*(i-k)))/(pi*(i-k)); %% defining the desired impulse response:sinc function
end
hd(k) = 0.3;
w = zeros(1,N);
for i=1:N
w(i)=0.54-0.46*cos(2*pi*((i-1)/(N-1))); %% defining the hamming window
end
h = hd.*w; %% product of desired impulse response and window function:FIR filter.
omega = -pi:pi/10000:pi;
figure(2);
freqz(h,1,omega); %% frequency response of the filter.
%% For N=512
N= 512; %% filter length
hd = zeros(1,N); %% initializing the desired impulse response
k= ceil((N-1)/2); %% defining the centre for the impulse response
for i=1:N
if(i==k)
continue
end
hd(i)= (sin(0.3*pi*(i-k)))/(pi*(i-k)); %% defining the desired impulse response:sinc function
end
hd(k) = 0.3;
w = zeros(1,N);
for i=1:N
w(i)=0.54-0.46*cos(2*pi*((i-1)/(N-1))); %% defining the hamming window
end
h = hd.*w; %% product of desired impulse response and window function:FIR filter.
omega = -pi:pi/10000:pi;
figure(3);
freqz(h,1,omega); %% frequency response of the filter.
```

For part-B: filtering and SNR calculation

```
Np=10000; %% signal length
n=0:1:Np; %%discrete time sampling
x=2*cos(0.1*pi*n)+4*cos(0.9*pi*n); %%signal
s=2*randn(1,Np+1); %%white noise
y=x+s; %% signal with the noise
freq=linspace(-pi,pi,length(x))/pi; %%frequency sampling for plotting in frequency domain

%%SNR calculation before passing through FIR filter
signal_amplitude = 10*log10(sum(abs(x).^2)); %% summed squared magnitude of the signal, x
noise_amplitude = 10*log10(sum(abs(s).^2)); %% summed squared magnitude of the noise, s
SNR = signal_amplitude-noise_amplitude; %% Formulae for SNR in db
%%Display the SNR and signal amplitude and noise amplitude
disp("Signal values before passing through filters")
disp("Signal amplitude:");
disp(signal_amplitude);
disp("Noise amplitude:");
disp(noise_amplitude);
disp("SNR Calculated value:")
disp(SNR);
%% For N=8
```

```
N= 8; %% filter length
hd = zeros(1,N); %% initializing the desired impulse response
k= ceil((N-1)/2); %% defining the centre for the impulse response
for i=1:N
if(i==k)
continue
end
hd(i)= (sin(0.3*pi*(i-k)))/(pi*(i-k)); %% defining the desired impulse response:sinc function
end
hd(k) = 0.3;
w = zeros(1,N);
for i=1:N
w(i)=0.54-0.46*cos(2*pi*((i-1)/(N-1))); %% defining the hamming window
end
h = hd.*w; %% product of desired impulse response and window function:FIR filter.
xout=filtfilt(h,1,x); %% filtered output of the signal x when passed through FIR filter h.
sout=filtfilt(h,1,s); %% filtered output of the signal s when passed through FIR filter h
yout=filtfilt(h,1,y); %% filtered output of the signal y when passed through FIR filter h.

%%Actual frequency response of the signals(x and y) and Filtered frequency responses
figure(1);
subplot(221)
plot(freq,abs(fftshift(fft(x)))/Np);
xlabel("Normalised Frequency (x pi rad/sample)");
ylabel("|X(f)|");
title("FFT of x");
subplot(222)
plot(freq,abs(fftshift(fft(xout)))/Np);
xlabel("Normalised Frequency (x pi rad/sample)");
ylabel("|X(f)|");
title("FFT of x after passed through FIR filter");
subplot(223)
plot(freq,abs(fftshift(fft(y)))/Np);
xlabel("Normalised Frequency (x pi rad/sample)");
ylabel("|X(f)|");
title("FFT of y(x with noise)");
subplot(224)
plot(freq,abs(fftshift(fft(yout)))/Np);
xlabel("Normalised Frequency (x pi rad/sample)");
ylabel("|X(f)|");
title("FFT of y after passed through FIR filter");

%%SNR calculation before passing through FIR filter
signal_amplitude = 10*log10(sum(abs(xout).^2)); %% summed squared magnitude of the signal, xout
noise_amplitude = 10*log10(sum(abs(sout).^2)); %% summed squared magnitude of the noise, sout
SNR = signal_amplitude-noise_amplitude; %% Formulae for SNR in db
%%Display the SNR and signal amplitude and noise amplitude
disp("signal after passed through FIR filter with filter length N=8")
disp("Signal amplitude:");
disp(signal_amplitude);
disp("Noise amplitude:");
disp(noise_amplitude);
disp("SNR Calculated value:")
disp(SNR);
%% For N=64
N= 64; %% filter length
```

```
hd = zeros(1,N); %% initializing the desired impulse response
k= ceil((N-1)/2); %% defining the centre for the impulse response
for i=1:N
if(i==k)
continue
end
hd(i)= (sin(0.3*pi*(i-k)))/(pi*(i-k)); %% defining the desired impulse response:sinc function
end
hd(k) = 0.3;
w = zeros(1,N);
for i=1:N
w(i)=0.54-0.46*cos(2*pi*((i-1)/(N-1))); %% defining the hamming window
end
h = hd.*w; %% product of desired impulse response and window function:FIR filter.
xout=filtfilt(h,1,x); %% filtered output of the signal x when passed through FIR filter h.
sout=filtfilt(h,1,s); %% filtered output of the signal s when passed through FIR filter h
yout=filtfilt(h,1,y); %% filtered output of the signal y when passed through FIR filter h.

%%Actual frequency response of the signals(x and y) and Filtered frequency responses
figure(2);
subplot(221)
plot(freq,abs(fftshift(fft(x)))/Np);
xlabel("Normalised Frequency (x pi rad/sample)");
ylabel("|X(f)|");
title("FFT of x");
subplot(222)
plot(freq,abs(fftshift(fft(xout)))/Np);
xlabel("Normalised Frequency (x pi rad/sample)");
ylabel("|X(f)|");
title("FFT of x after passed through FIR filter");
subplot(223)
plot(freq,abs(fftshift(fft(y)))/Np);
xlabel("Normalised Frequency (x pi rad/sample)");
ylabel("|X(f)|");
title("FFT of y(x with noise)");
subplot(224)
plot(freq,abs(fftshift(fft(yout)))/Np);
xlabel("Normalised Frequency (x pi rad/sample)");
ylabel("|X(f)|");
title("FFT of y after passed through FIR filter");

%%SNR calculation before passing through FIR filter
signal_amplitude = 10*log10(sum(abs(xout).^2)); %% summed squared magnitude of the signal, xout
noise_amplitude = 10*log10(sum(abs(sout).^2)); %% summed squared magnitude of the noise, sout
SNR = signal_amplitude-noise_amplitude; %% Formulae for SNR in db
%%Display the SNR and signal amplitude and noise amplitude
disp("signal after passed through FIR filter with filter length N=64")
disp("Signal amplitude:");
disp(signal_amplitude);
disp("Noise amplitude:");
disp(noise_amplitude);
disp("SNR Calculated value:")
disp(SNR);
%% For N=512
N= 512; %% filter length
hd = zeros(1,N); %% initializing the desired impulse response
```

```
k= ceil((N-1)/2); %% defining the centre for the impulse response
for i=1:N
if(i==k)
continue
end
hd(i)= (sin(0.3*pi*(i-k)))/(pi*(i-k)); %% defining the desired impulse response:sinc function
end
hd(k) = 0.3;
w = zeros(1,N);
for i=1:N
w(i)=0.54-0.46*cos(2*pi*((i-1)/(N-1))); %% defining the hamming window
end
h = hd.*w; %% product of desired impulse response and window function:FIR filter.
xout=filtfilt(h,1,x); %% filtered output of the signal x when passed through FIR filter h.
sout=filtfilt(h,1,s); %% filtered output of the signal s when passed through FIR filter h
yout=filtfilt(h,1,y); %% filtered output of the signal y when passed through FIR filter h.

%%Actual frequency response of the signals(x and y) and Filtered frequency responses
figure(3);
subplot(221)
plot(freq,abs(fftshift(fft(x)))/Np);
xlabel("Normalised Frequency (x pi rad/sample)");
ylabel("|X(f)|");
title("FFT of x");
subplot(222)
plot(freq,abs(fftshift(fft(xout)))/Np);
xlabel("Normalised Frequency (x pi rad/sample)");
ylabel("|X(f)|");
title("FFT of x after passed through FIR filter");
subplot(223)
plot(freq,abs(fftshift(fft(y)))/Np);
xlabel("Normalised Frequency (x pi rad/sample)");
ylabel("|X(f)|");
title("FFT of y(x with noise)");
subplot(224)
plot(freq,abs(fftshift(fft(yout)))/Np);
xlabel("Normalised Frequency (x pi rad/sample)");
ylabel("|X(f)|");
title("FFT of y after passed through FIR filter");

%%SNR calculation before passing through FIR filter
signal_amplitude = 10*log10(sum(abs(xout).^2)); %% summed squared magnitude of the signal, xout
noise_amplitude = 10*log10(sum(abs(sout).^2)); %% summed squared magnitude of the noise, sout
SNR = signal_amplitude-noise_amplitude; %% Formulae for SNR in db
%%Display the SNR and signal amplitude and noise amplitude
disp("signal after passed through FIR filter with filter length N=512")
disp("Signal amplitude:");
disp(signal_amplitude);
disp("Noise amplitude:");
disp(noise_amplitude);
disp("SNR Calculated value:")
disp(SNR);
```

With Blackmnan window:-

For part-A: defining the FIR filter

```
clear; %% clears the previously stored values
%% For N=8;
N= 8; %% filter length
hd = zeros(1,N); %% initializing the desired impulse response
k= ceil((N-1)/2); %% defining the centre for the impulse response
for i=1:N
if(i==k)
continue
end
hd(i)= (sin(0.3*pi*(i-k)))/(pi*(i-k)); %% defining the desired impulse response:sinc function
end
hd(k) = 0.3;
w = zeros(1,N);
for i=1:N
w(i)=0.42-0.5*cos(2*pi*((i-1)/(N-1)))+0.08*cos(4*pi*((i-1)/(N-1))); %% defining the Blackmnan window
end
h = hd.*w; %% product of desired impulse response and window function:FIR filter.
omega = -pi:pi/10000:pi;
figure(1);
freqz(h,1,omega); %% frequency response of the filter.
%% For N=64
N= 64; %% filter length
hd = zeros(1,N); %% initializing the desired impulse response
k= ceil((N-1)/2); %% defining the centre for the impulse response
for i=1:N
if(i==k)
continue
end
hd(i)= (sin(0.3*pi*(i-k)))/(pi*(i-k)); %% defining the desired impulse response:sinc function
end
hd(k) = 0.3;
w = zeros(1,N);
for i=1:N
w(i)=0.42-0.5*cos(2*pi*((i-1)/(N-1)))+0.08*cos(4*pi*((i-1)/(N-1))); %% defining the blackmnan window
end
h = hd.*w; %% product of desired impulse response and window function:FIR filter.
omega = -pi:pi/10000:pi;
figure(2);
freqz(h,1,omega); %% frequency response of the filter.
%% For N=512
N= 512; %% filter length
hd = zeros(1,N); %% initializing the desired impulse response
k= ceil((N-1)/2); %% defining the centre for the impulse response
for i=1:N
if(i==k)
continue
end
hd(i)= (sin(0.3*pi*(i-k)))/(pi*(i-k)); %% defining the desired impulse response:sinc function
end
hd(k) = 0.3;
w = zeros(1,N);
for i=1:N
w(i)=0.42-0.5*cos(2*pi*((i-1)/(N-1)))+0.08*cos(4*pi*((i-1)/(N-1))); %% defining the blackmnan window
end
h = hd.*w; %% product of desired impulse response and window function:FIR filter.
omega = -pi:pi/10000:pi;
```

```
figure(3);
freqz(h,1,omega); %% frequency response of the filter.
```

For part-B: filtering and SNR calculation

```
Np=10000; %% signal length
n=0:1:Np; %%discrete time sampling
x=2*cos(0.1*pi*n)+4*cos(0.9*pi*n); %%signal
s=2*randn(1,Np+1); %%white noise
y=x+s; %% signal with the noise
freq=linspace(-pi,pi,length(x))/pi; %%frequency sampling for plotting in frequency domain

%%SNR calculation before passing through FIR filter
signal_amplitude = 10*log10(sum(abs(x).^2)); %% summed squared magnitude of the signal, x
noise_amplitude = 10*log10(sum(abs(s).^2)); %% summed squared magnitude of the noise, s
SNR = signal_amplitude-noise_amplitude; %% Formulae for SNR in db
%%Display the SNR and signal amplitude and noise amplitude
disp("Signal values before passing through filters")
disp("Signal amplitude:");
disp(signal_amplitude);
disp("Noise amplitude:");
disp(noise_amplitude);
disp("SNR Calculated value:")
disp(SNR);
%% For N=8
N= 8; %% filter length
hd = zeros(1,N); %% initializing the desired impulse response
k= ceil((N-1)/2); %% defining the centre for the impulse response
for i=1:N
if(i==k)
continue
end
hd(i)= (sin(0.3*pi*(i-k)))/(pi*(i-k)); %% defining the desired impulse response:sinc function
end
hd(k) = 0.3;
w = zeros(1,N);
for i=1:N
w(i)=0.42-0.5*cos(2*pi*((i-1)/(N-1)))+0.08*cos(4*pi*((i-1)/(N-1))); %% defining the blackmnan  window
end
h = hd.*w; %% product of desired impulse response and window function:FIR filter.
xout=filtfilt(h,1,x); %% filtered output of the signal x when passed through FIR filter h.
sout=filtfilt(h,1,s); %% filtered output of the signal s when passed through FIR filter h
yout=filtfilt(h,1,y); %% filtered output of the signal y when passed through FIR filter h.

%%Actual frequency response of the signals(x and y) and Filtered frequency responses
figure(1);
subplot(221)
plot(freq,abs(fftshift(fft(x)))/Np);
xlabel("Normalised Frequency (x pi rad/sample)");
ylabel("|X(f)|");
title("FFT of x");
subplot(222)
plot(freq,abs(fftshift(fft(xout)))/Np);
xlabel("Normalised Frequency (x pi rad/sample)");
ylabel("|X(f)|");
title("FFT of x after passed through FIR filter");
```

```
subplot(223)
plot(freq,abs(fftshift(fft(y)))/Np);
xlabel("Normalised Frequency (x pi rad/sample)");
ylabel("|X(f)|");
title("FFT of y(x with noise)");
subplot(224)
plot(freq,abs(fftshift(fft(yout)))/Np);
xlabel("Normalised Frequency (x pi rad/sample)");
ylabel("|X(f)|");
title("FFT of y after passed through FIR filter");

%%SNR calculation before passing through FIR filter
signal_amplitude = 10*log10(sum(abs(xout).^2)); %% summed squared magnitude of the signal, xout
noise_amplitude = 10*log10(sum(abs(sout).^2)); %% summed squared magnitude of the noise, sout
SNR = signal_amplitude-noise_amplitude; %% Formulae for SNR in db
%%Display the SNR and signal amplitude and noise amplitude
disp("signal after passed through FIR filter with filter length N=8")
disp("Signal amplitude:");
disp(signal_amplitude);
disp("Noise amplitude:");
disp(noise_amplitude);
disp("SNR Calculated value:")
disp(SNR);
%% For N=64
N= 64; %% filter length
hd = zeros(1,N); %% initializing the desired impulse response
k= ceil((N-1)/2); %% defining the centre for the impulse response
for i=1:N
if(i==k)
continue
end
hd(i)= (sin(0.3*pi*(i-k)))/(pi*(i-k)); %% defining the desired impulse response:sinc function
end
hd(k) = 0.3;
w = zeros(1,N);
for i=1:N
w(i)=0.42-0.5*cos(2*pi*((i-1)/(N-1)))+0.08*cos(4*pi*((i-1)/(N-1))); %% defining the blackmnan window
end
h = hd.*w; %% product of desired impulse response and window function:FIR filter.
xout=filtfilt(h,1,x); %% filtered output of the signal x when passed through FIR filter h.
sout=filtfilt(h,1,s); %% filtered output of the signal s when passed through FIR filter h
yout=filtfilt(h,1,y); %% filtered output of the signal y when passed through FIR filter h.

%%Actual frequency response of the signals(x and y) and Filtered frequency responses
figure(2);
subplot(221)
plot(freq,abs(fftshift(fft(x)))/Np);
xlabel("Normalised Frequency (x pi rad/sample)");
ylabel("|X(f)|");
title("FFT of x");
subplot(222)
plot(freq,abs(fftshift(fft(xout)))/Np);
xlabel("Normalised Frequency (x pi rad/sample)");
ylabel("|X(f)|");
title("FFT of x after passed through FIR filter");
subplot(223)
```

```
plot(freq,abs(fftshift(fft(y)))/Np);
xlabel("Normalised Frequency (x pi rad/sample)");
ylabel("|X(f)|");
title("FFT of y(x with noise)");
subplot(224)
plot(freq,abs(fftshift(fft(yout)))/Np);
xlabel("Normalised Frequency (x pi rad/sample)");
ylabel("|X(f)|");
title("FFT of y after passed through FIR filter");

%%SNR calculation before passing through FIR filter
signal_amplitude = 10*log10(sum(abs(xout).^2)); %% summed squared magnitude of the signal, xout
noise_amplitude = 10*log10(sum(abs(sout).^2)); %% summed squared magnitude of the noise, sout
SNR = signal_amplitude-noise_amplitude; %% Formulae for SNR in db
%%Display the SNR and signal amplitude and noise amplitude
disp("signal after passed through FIR filter with filter length N=64")
disp("Signal amplitude:");
disp(signal_amplitude);
disp("Noise amplitude:");
disp(noise_amplitude);
disp("SNR Calculated value:")
disp(SNR);
%% For N=512
N= 512; %% filter length
hd = zeros(1,N); %% initializing the desired impulse response
k= ceil((N-1)/2); %% defining the centre for the impulse response
for i=1:N
if(i==k)
continue
end
hd(i)= (sin(0.3*pi*(i-k)))/(pi*(i-k)); %% defining the desired impulse response:sinc function
end
hd(k) = 0.3;
w = zeros(1,N);
for i=1:N
w(i)=0.42-0.5*cos(2*pi*((i-1)/(N-1)))+0.08*cos(4*pi*((i-1)/(N-1))); %% defining the blackmnan  window
end
h = hd.*w; %% product of desired impulse response and window function:FIR filter.
xout=filtfilt(h,1,x); %% filtered output of the signal x when passed through FIR filter h.
sout=filtfilt(h,1,s); %% filtered output of the signal s when passed through FIR filter h
yout=filtfilt(h,1,y); %% filtered output of the signal y when passed through FIR filter h.

%%Actual frequency response of the signals(x and y) and Filtered frequency responses
figure(3);
subplot(221)
plot(freq,abs(fftshift(fft(x)))/Np);
xlabel("Normalised Frequency (x pi rad/sample)");
ylabel("|X(f)|");
title("FFT of x");
subplot(222)
plot(freq,abs(fftshift(fft(xout)))/Np);
xlabel("Normalised Frequency (x pi rad/sample)");
ylabel("|X(f)|");
title("FFT of x after passed through FIR filter");
subplot(223)
plot(freq,abs(fftshift(fft(y)))/Np);
```

```
xlabel("Normalised Frequency (x pi rad/sample)");
ylabel("|X(f)|");
title("FFT of y(x with noise)");
subplot(224)
plot(freq,abs(fftshift(fft(yout)))/Np);
xlabel("Normalised Frequency (x pi rad/sample)");
ylabel("|X(f)|");
title("FFT of y after passed through FIR filter");

%%SNR calculation before passing through FIR filter
signal_amplitude = 10*log10(sum(abs(xout).^2)); %% summed squared magnitude of the signal, xout
noise_amplitude = 10*log10(sum(abs(sout).^2)); %% summed squared magnitude of the noise, sout
SNR = signal_amplitude-noise_amplitude; %% Formulae for SNR in db
%%Display the SNR and signal amplitude and noise amplitude
disp("signal after passed through FIR filter with filter length N=512")
disp("Signal amplitude:");
disp(signal_amplitude);
disp("Noise amplitude:");
disp(noise_amplitude);
disp("SNR Calculated value:")
disp(SNR);
```