**Group: B-3**

**Roll No.: 1909043, 1909044, 1909045, 1909046, 1909047, 1909048**

**Block Diagram of the Implemented Design:**

Determine the required operations with the user defined functions

**END**

Plot Magnitude Response, Phase Response,

Pole-Zeros (For Stability Check) and Display transfer function

Filtered Signal

Noisy Signal

Determine applicable Window and Filter coefficient

Input Parameters:

Filter-Type: Band Pass

Passband: 1500 – 2750 Hz Lower Stopband: 0-1KHz

Upper Stopband: 3.5-4.5KHz

Sampling rate: 8KHz

Passband Ripple: 0.5 dB

Stopband Attenuation: 40 dB

Calculate proper window function

**Start**

**Figure 3.1: Block Diagram of designed Band Pass FIR filter**

**Code:**

**main.m:**

% Calculate filter coefficient, window type and window function

[h] = filter\_coefficient('bpf',8000,0,0,1000,3500,1500,2750,25);

[win\_type,m] = window\_type(0.5,40);

[num,h\_win,w\_win] = window\_function(h,win\_type,m);

figure(1)

subplot(311)

plot(w\_win,20\*log10(abs(h\_win)))

grid;

xlabel('Frequency (Hz)');

ylabel('Magnitude Response (dB)');

subplot(312)

plot(w\_win, 180\*angle(h\_win)/pi); grid;

xlabel('Frequency (Hz)');

ylabel('Phase (degrees)');

subplot(313),

plot(w\_win,180\*unwrap(angle(h\_win))/pi);grid;

xlabel('Frequency (Hz)');

ylabel('Phase (degrees)');

figure(2)

subplot(1,1,1);

zplane(num,1);

title('Pole Zero plot(FIR Filter)');

grid on;

% Generate the input signal

fs=8000;

t = 0:1/fs:0.1; % Time vector from 0 to 0.1 seconds

input\_signal = sin(0.06\*pi\*t) + 3\*sin(0.14\*pi\*t);

B=reshape(h\_win.',1,[]); %change matrix to vector

% Filter the noisy signal using the FIR filter for the signal

noisy\_signal = input\_signal+sqrt(0.1)\*randn(size(input\_signal));

filtered\_signal = filter(h\_win,1,noisy\_signal);

% Plot the original signal, noisy signal, and filtered signal for the signal

figure(3);

subplot(3, 1, 1);

plot(t, input\_signal);

title('Original Signal');

xlabel('Time (s)');

ylabel('Amplitude');

subplot(3, 1, 2);

plot(t, noisy\_signal);

title('Noisy Signal');

xlabel('Time (s)');

ylabel('Amplitude');

subplot(3, 1, 3);

plot(t, filtered\_signal);

title('Filtered Signal');

xlabel('Time (s)');

ylabel('Amplitude');

**check\_digit.m:**

function x = check\_digit(m,n)

k = round(max(m,n));

if mod(k,2)==0

x = k+1;

else

x = k;

end

end

**filter\_coefficient.m:**

function [num] = filter\_coefficient(filter\_type,fs,fc\_low,fc\_up,f\_stop1,f\_stop2,f\_pass1,f\_pass2,n)

switch filter\_type

case 'lpf'

M=(n-1)/2;

wl = 2\*pi\*fc\_low/fs;

num = [];

for k = 0:1:M

pos = k+M+1;

if k == 0

num(pos) = wl/pi;

else

num(pos) = (sin(k\*wl)/(k\*pi));

new\_pos = pos-(2\*k);

num(new\_pos) = num(pos);

end

end

[hz,w]=freqz(num,[1],512,fs);

case 'hpf'

M=(n-1)/2;

wh = 2\*pi\*fc\_up/fs;

num = [];

for k = 0:1:M

pos = k+M+1;

if k == 0

num(pos) = (pi-wh)/pi;

else

num(pos) = -(sin(k\*wh)/(k\*pi));

new\_pos = pos-(2\*k);

num(new\_pos) = num(pos);

end

end

[hz,w]=freqz(num,[1],512,fs);

case 'bsf'

M=(n-1)/2;

wl=2\*pi\*fc\_low/fs;

wh = 2\*pi\*fc\_up/fs;

num = []

for k = 0:1:M

pos = k+M+1;

if k == 0

num(pos) = pi-wh+wl/pi;

else

num(pos) = -(sin(k\*wh)/(k\*pi))+sin(k\*wl)/(k\*pi)

new\_pos = pos-(2\*k)

num(new\_pos) = num(pos)

end

end

[hz,w]=freqz(num,[1],512,fs);

case 'bpf'

M=(n-1)/2;

fl=(f\_stop1+f\_pass1)/2;

fh=(f\_stop2+f\_pass2)/2;

wl=2\*pi\*fl/fs;

wh = 2\*pi\*fh/fs;

num = [];

for k = 0:1:M

pos = k+M+1;

if k == 0

num(pos) = (wh-wl)/pi;

else

num(pos) = (sin(k\*wh)/(k\*pi))-sin(k\*wl)/(k\*pi);

new\_pos = pos-(2\*k);

num(new\_pos) = num(pos);

end

end

[hz,w]=freqz(num,[1],512,fs);

end

end

**window\_type.m:**

function [win\_type,M] = window\_type(passband\_ripple, stopband\_attenuation)

f\_stop1 = 1000;

f\_pass1 = 1500;

f\_pass2 = 2750;

f\_stop2 = 3500;

fs=8000;

if (passband\_ripple >= 0.7416) && (stopband\_attenuation > 0 && stopband\_attenuation <= 21) % Rectangular window

win\_type = 'rect';

N1\_rect = 0.9\*fs/ abs(f\_stop1-f\_pass1);

N2\_rect = 0.9\*fs/ abs(f\_stop2-f\_pass2);

N\_rect = check\_digit(N1\_rect,N2\_rect);

M = (N\_rect - 1) / 2;

elseif (passband\_ripple >= 0.0546) && (stopband\_attenuation > 21 && stopband\_attenuation <= 44) % Hanning window

win\_type = 'hann';

N1\_hann = 3.1\*fs/ abs(f\_stop1-f\_pass1);

N2\_hann = 3.1\*fs/ abs(f\_stop2-f\_pass2);

N\_hann = check\_digit(N1\_hann,N2\_hann);

M = (N\_hann - 1) / 2;

elseif (passband\_ripple >= 0.0194) && (stopband\_attenuation > 44 && stopband\_attenuation <= 53) % Hamming window

win\_type = 'hamm';

N1\_hamm = 3.3\*fs/ abs(f\_stop1-f\_pass1);

N2\_hamm = 3.3\*fs/ abs(f\_stop2-f\_pass2);

N\_hamm = check\_digit(N1\_hamm,N2\_hamm);

M = (N\_hamm - 1) / 2;

elseif (passband\_ripple >= 0.0017) && (stopband\_attenuation > 53 && stopband\_attenuation <= 74) % Blackman window

win\_type = 'black';

N1\_black = 5.5\*fs/ abs(f\_stop1-f\_pass1);

N2\_black = 5.5\*fs/ abs(f\_stop2-f\_pass2);

N\_black = check\_digit(N1\_black,N2\_black);

M = (N\_black - 1) / 2;

else

disp('Invalid parameters');

end

end

**window\_function.m:**function [num, h\_win, w\_win] = window\_function(h, win\_type, M)

fs = 8000;

w = ones(1, M); % Assuming M is the length of h

switch win\_type

case 'rect'

% Rectangular window

% No changes needed for rectangular window

case 'hann'

% Hanning window

for n = 0:M-1

w(n + 1) = 0.5 + 0.5 \* (1 - cos(2 \* pi \* n / M));

end

case 'hamm'

% Hamming window

for n = 0:M-1

w(n + 1) = 0.54 + 0.46 \* cos(2 \* pi \* n / M);

end

case 'black'

% Blackman window

for n = 0:M-1

w(n + 1) = 0.42 + 0.5 \* cos(2 \* pi \* n / M) + 0.08 \* cos(4 \* pi \* n / M);

end

otherwise

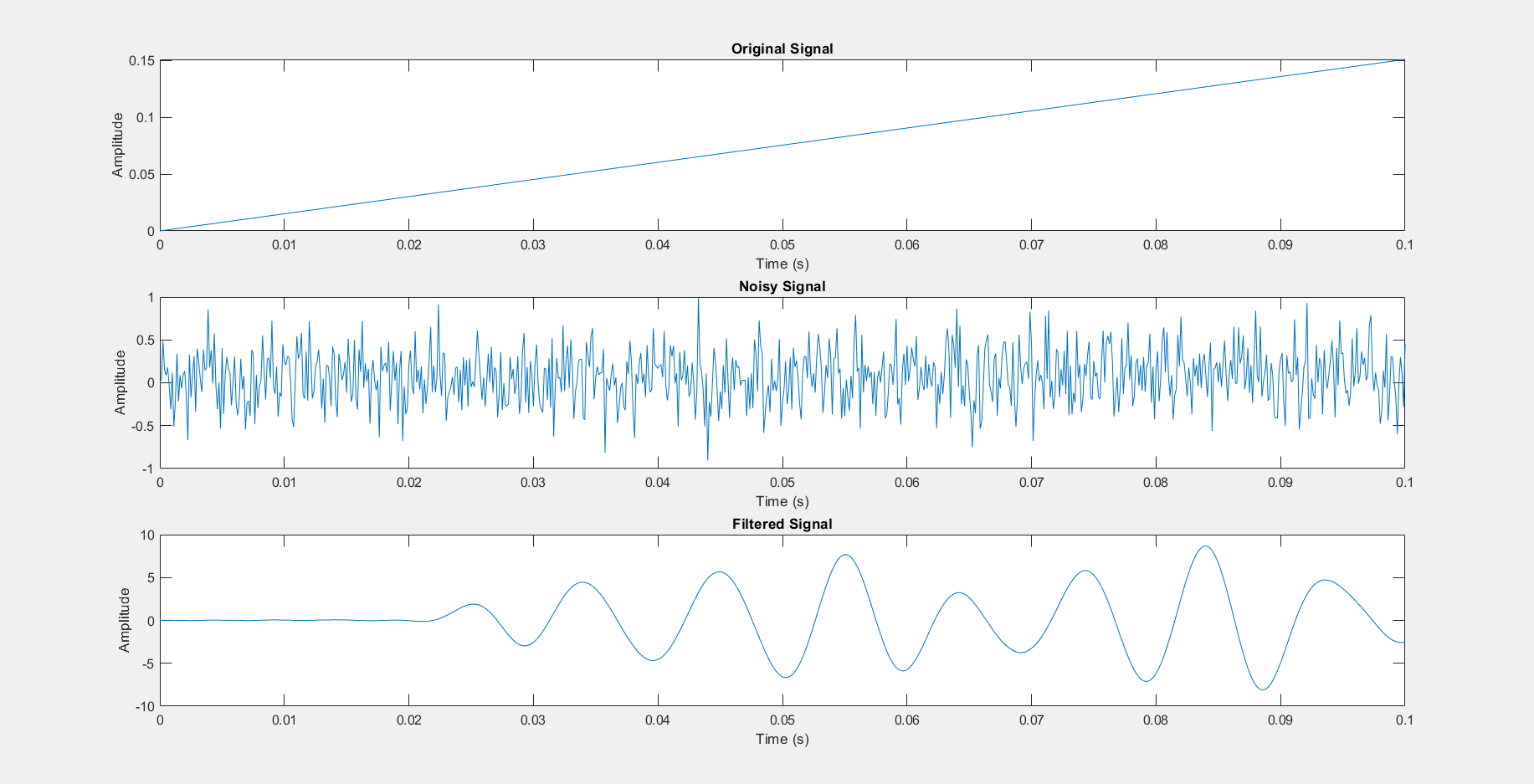
error('Unsupported window type');

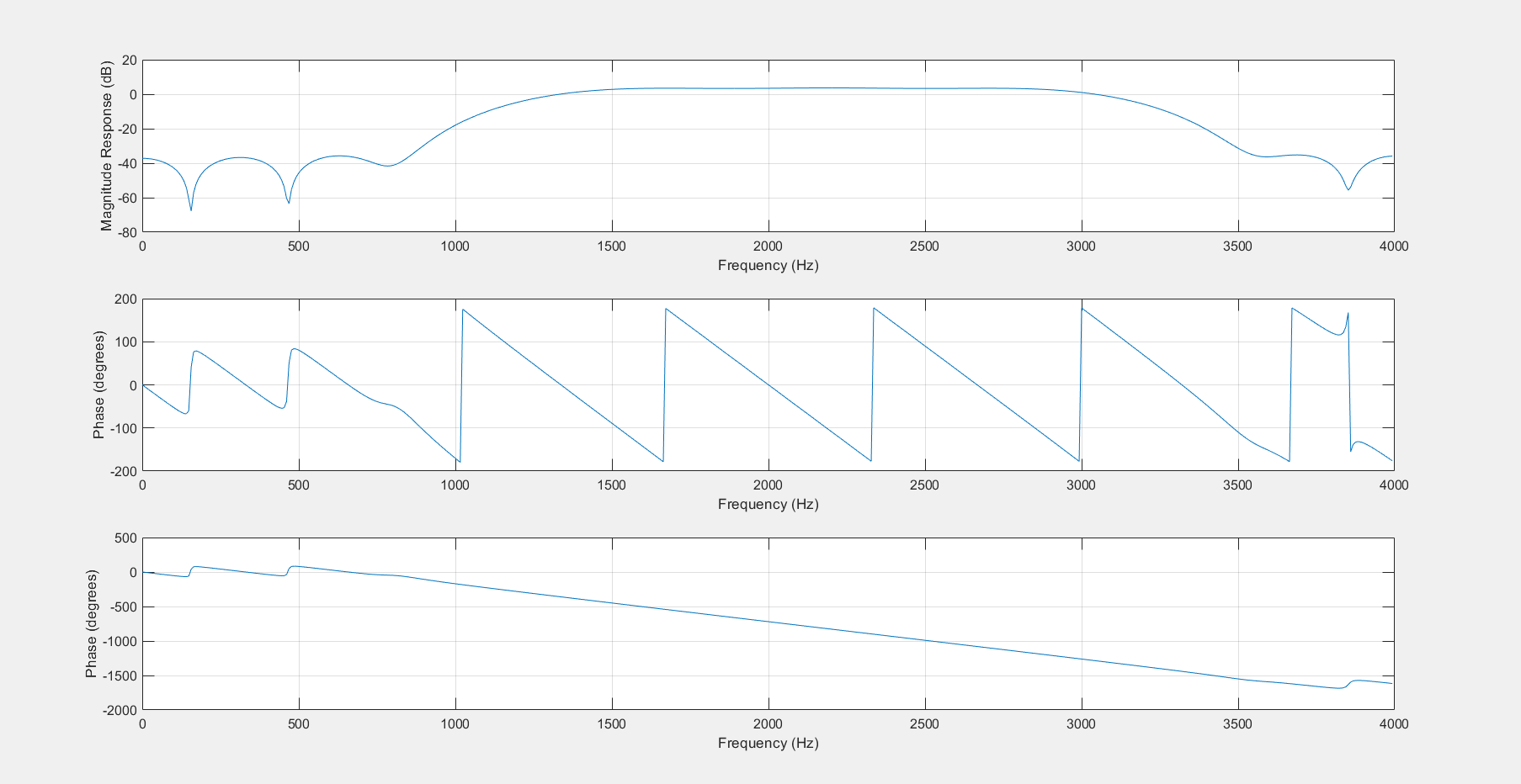
end

num = h .\* w;

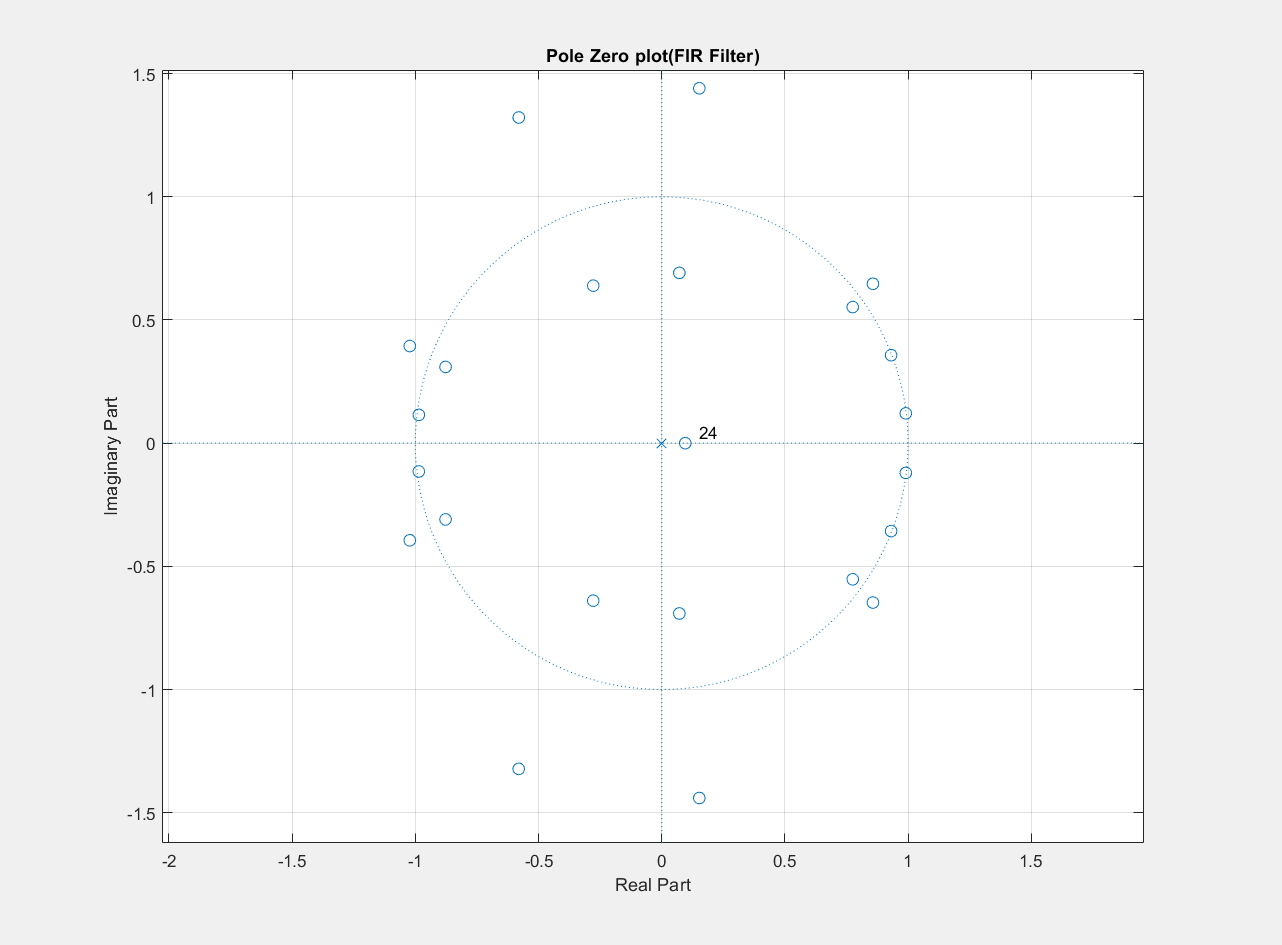
[h\_win, w\_win] = freqz(num, [1], 512, fs);

End

**Obtained figures:**

**Figure 3.2: Waveform of the original signal and filtered signal**

**Figure 3.3: Magnitude Response & Phase Response of the designed FIR ­filter**

****

**Figure 3.4: Pole-Zeros Plot of the FIR ­Filter (Stability Check)**