Group: B-3

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

Block Diagram of the Implemented Design:

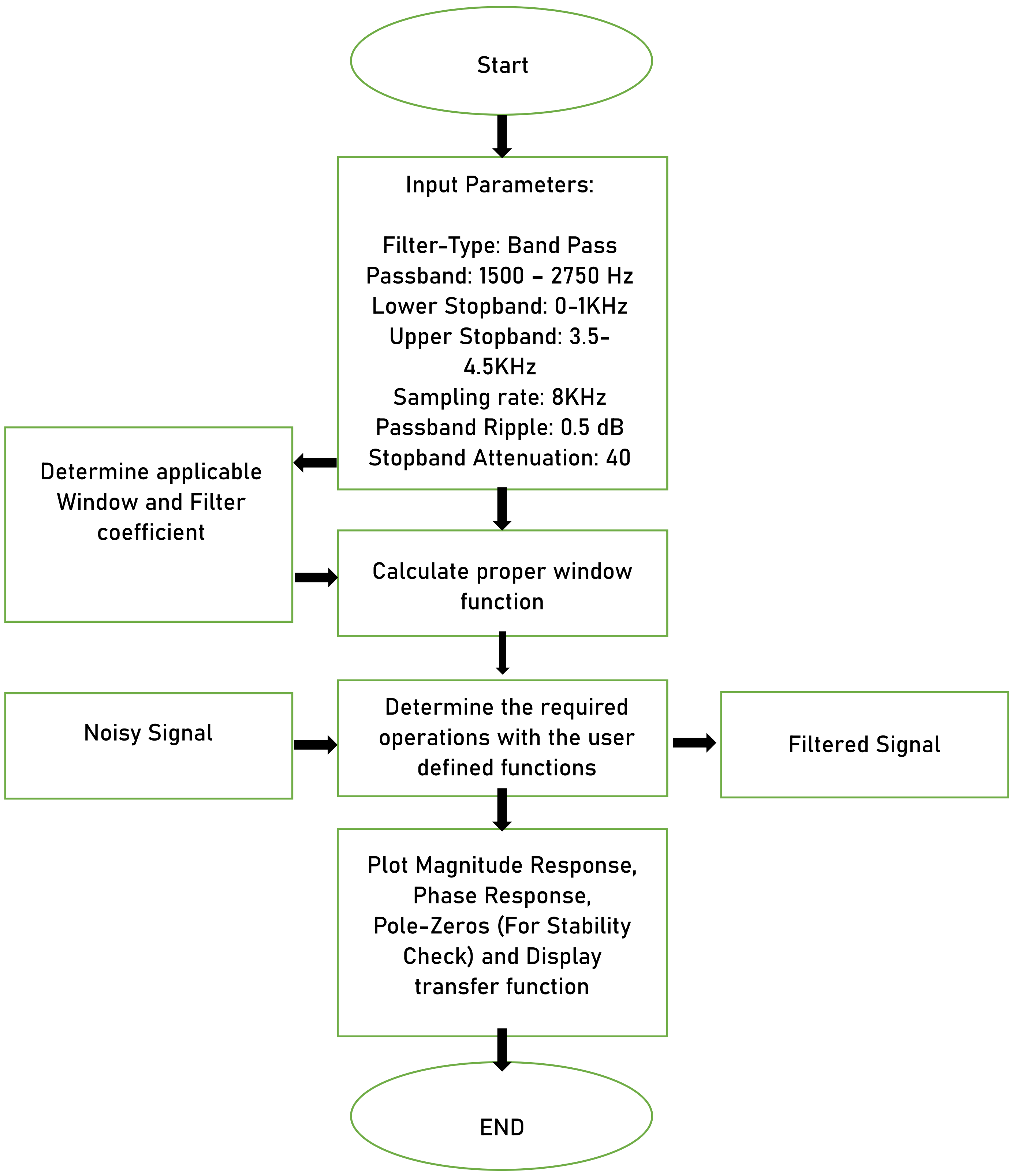
Start
Input Parameters:
Filter-Type: Band Pass
Passband: 1500 – 2750 Hz
Lower Stopband: 0-1KHz
Upper Stopband: 3.5-4.5KHz
Sampling rate: 8KHz
Passband Ripple: 0.5 dB
Stopband Attenuation: 40
Determine applicable Window and Filter coefficient
Calculate proper window function
Noisy Signal
Determine the required operations with the user defined functions
Filtered Signal
Plot Magnitude Response, Phase Response, Pole-Zeros (For Stability Check) and Display transfer function
END

Figure 3.1: Block Diagram of designed Band Pass FIR filter

Code:

main.m:

```matlab
% 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:

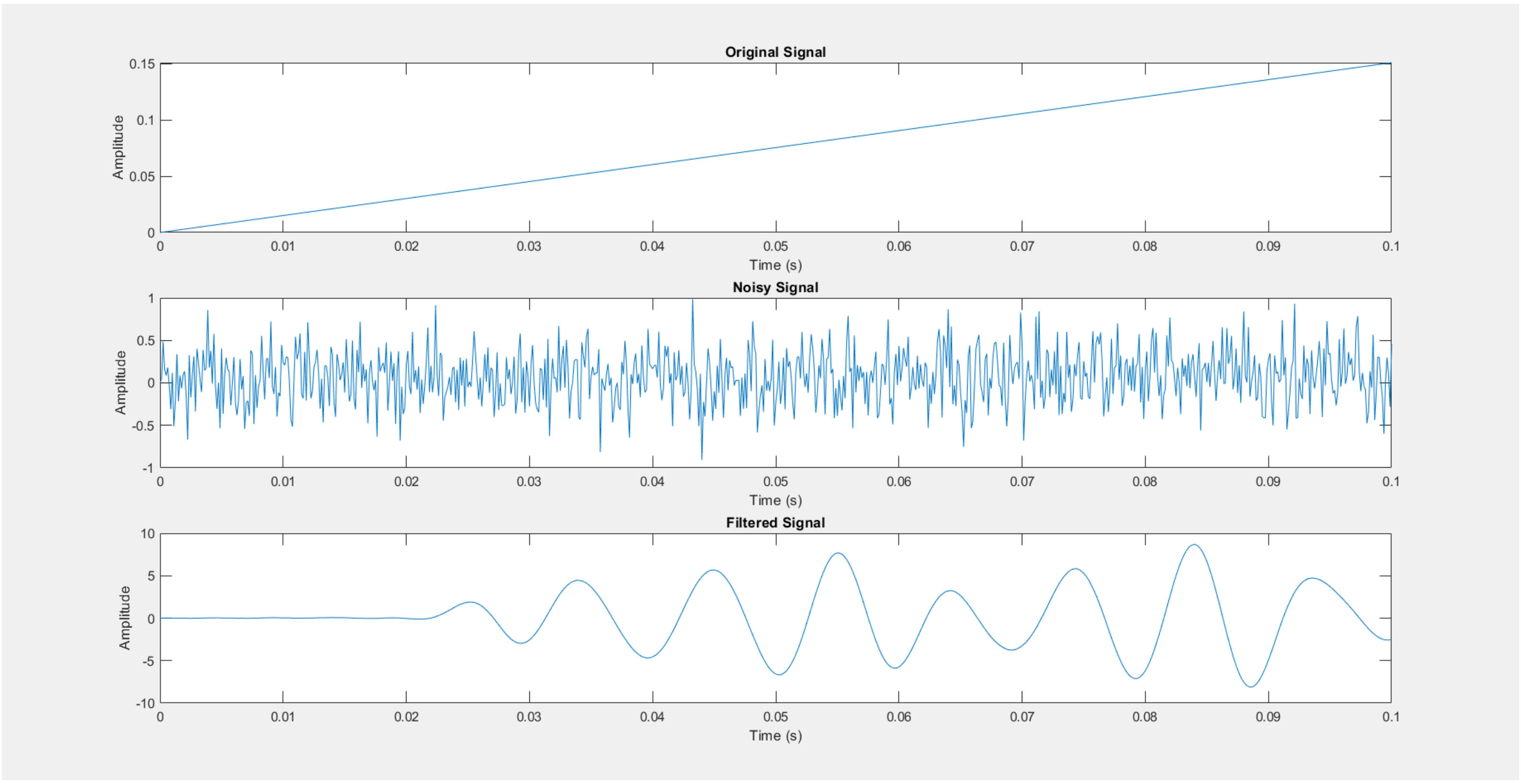

Figure 3.2: Waveform of the original signal and filtered signal

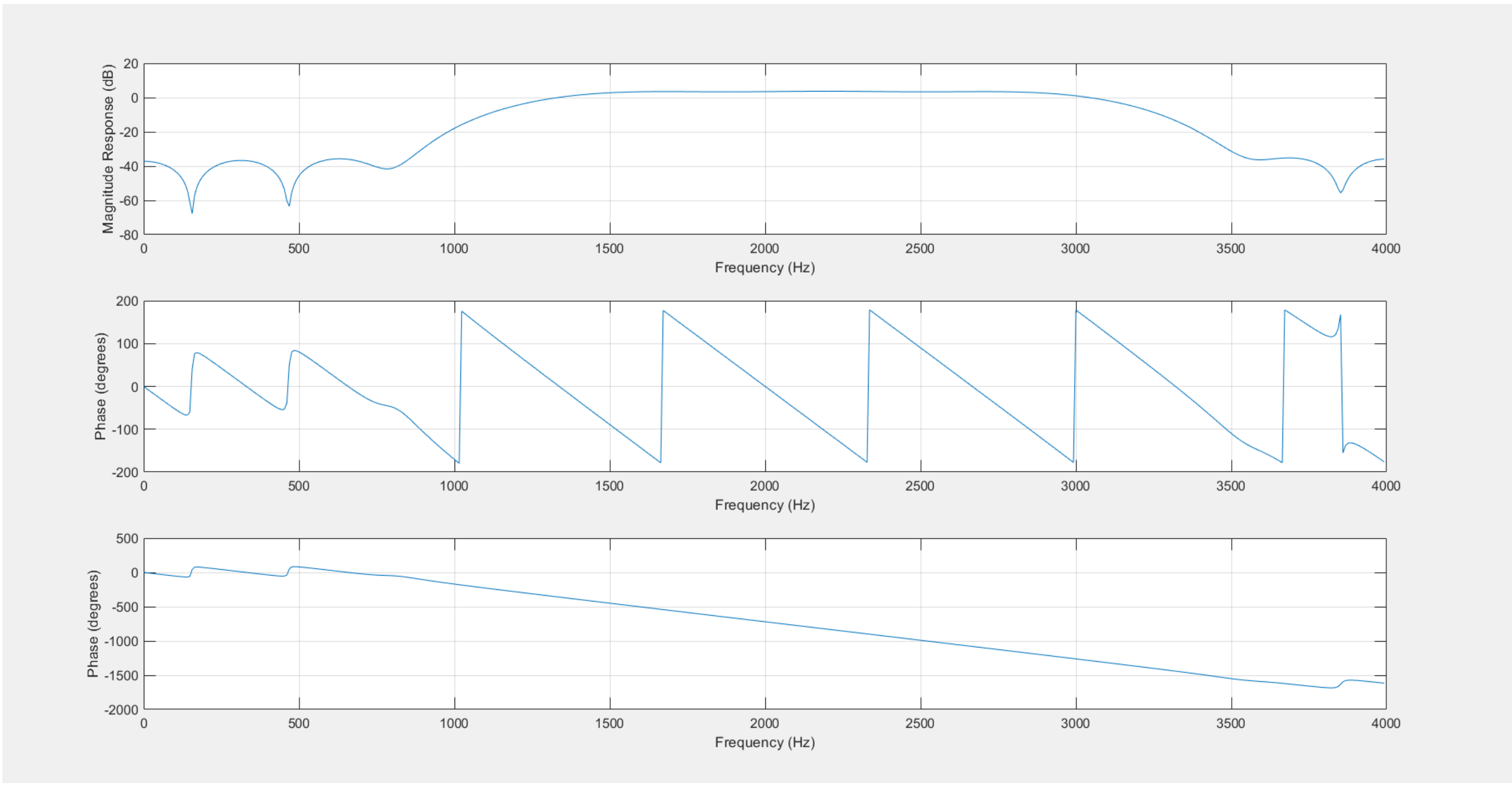

Figure 3.3: Magnitude Response & Phase Response of the designed FIR filter

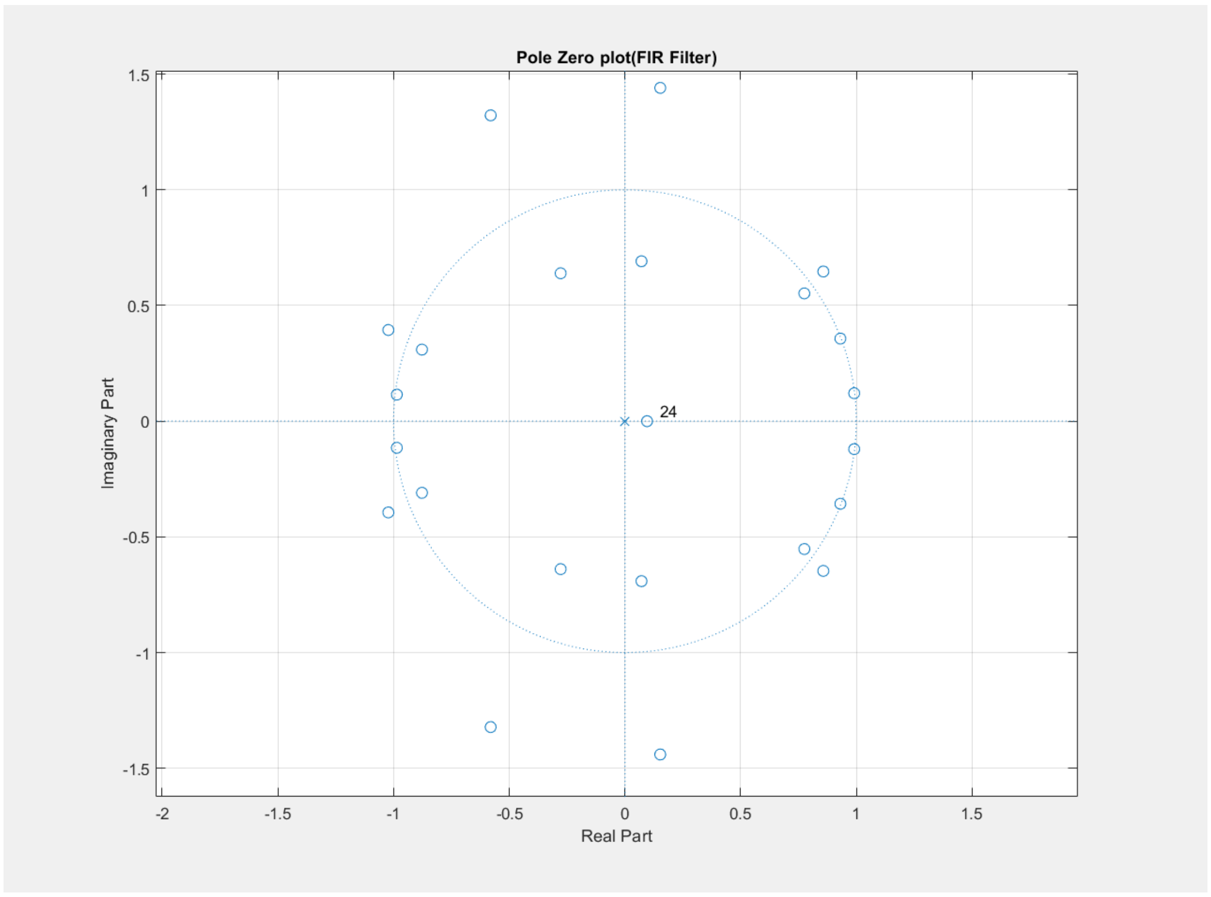

Figure 3.4: Pole-Zeros Plot of the FIR Filter (Stability Check)