**Experiment 8: Linear Phase F. I. R. Filter Design using Windowing Method**

|  |  |
| --- | --- |
| **Name** | Prathamesh Mane |
| **UID no. & Branch** | **2022200078 (B1)** |
| **Experiment No.** | 8 |

Code:

% Linear Phase LPF Design using Windowing Method

clc;

clear;

close all;

disp("Step 1: Accept user input specifications");

Fs = input('Enter the sampling frequency (Hz): ')

Fp = input('Enter passband frequency (Hz): ')

Fs\_stop = input('Enter stopband frequency (Hz): ')

Ap = input('Enter passband attenuation (dB): ')

As = input('Enter stopband attenuation (dB): ')

N = input('Enter the filter order (N): ')

if mod(N, 2) == 0

warning('Filter order should be odd for linear phase. Incrementing N by 1.')

N = N + 1;

end

disp("Step 2: Select appropriate window function");

if As <= 21

window\_type = 'rectwin';

elseif As <= 44

window\_type = 'hann';

elseif As <= 53

window\_type = 'hamming';

else

window\_type = 'blackman';

end

fprintf('Selected window: %s\n', window\_type);

disp("Step 3: Normalize frequencies");

Wp = Fp / (Fs / 2)

Ws = Fs\_stop / (Fs / 2)

Wn = [Wp];

disp("Step 4: Design the LPF using the selected window");

b = fir1(N-1, Wn, 'low', window(window\_type, N));

disp("Step 5: Frequency response");

[H, f] = freqz(b, 1, 1024, Fs);

figure;

subplot(2, 1, 1);

plot(f, 20\*log10(abs(H)));

grid on;

xlabel('Frequency (Hz)');

ylabel('Magnitude (dB)');

title('Magnitude Spectrum');

xlim([0, Fs / 2]);

ylim([-70, 5]);

hold on;

yline(-Ap, 'r--', 'Passband Attenuation (Ap)');

yline(-As, 'g--', 'Stopband Attenuation (As)');

hold off;

subplot(2, 1, 2);

plot(f, angle(H));

grid on;

xlabel('Frequency (Hz)');

ylabel('Phase (radians)');

title('Phase Spectrum');

disp("Step 6: Impulse response");

figure;

stem(b, 'filled');

grid on;

xlabel('Samples');

ylabel('Amplitude');

title('Impulse Response of LPF');

disp('Observe the phase spectrum for linearity. Linear phase indicates symmetry in the impulse response.');

disp("Step 1: Load recorded audio");

[audio\_signal, Fs] = audioread('prathamesh\_rec.wav'); % Load the audio file

audio\_signal = audio\_signal(:, 1); % Convert to mono if stereo

disp("Step 2: Add 7.8 kHz noise");

t = (0:length(audio\_signal)-1) / Fs; % Time vector for the audio signal

noise\_freq = 7800; % Noise frequency in Hz

noise = 0.1 \* sin(2 \* pi \* noise\_freq \* t)'; % Generate sinusoidal noise

noisy\_signal = audio\_signal + noise; % Add noise to the original signal

disp("Step 3: noisy audio for 12 seconds");

sound(noisy\_signal, Fs);

pause(12); % Play noisy audio for 12 seconds

disp("Step 4: Pass the noisy signal through the LPF");

filtered\_signal = filter(b, 1, noisy\_signal); % Apply the designed filter

disp("Step 5: Play filtered audio for 12 seconds");

sound(filtered\_signal, Fs);

pause(12); % Play filtered audio for 12 seconds

disp("Step 6: Plot the signals");

t\_plot = (0:1/Fs:12-1/Fs); % Time vector for plotting (first 12 seconds)

figure;

subplot(3, 1, 1);

plot(t\_plot, audio\_signal(1:Fs\*12));

grid on;

xlabel('Time (s)');

ylabel('Amplitude');

title('Original Signal');

subplot(3, 1, 2);

plot(t\_plot, noisy\_signal(1:Fs\*12));

grid on;

xlabel('Time (s)');

ylabel('Amplitude');

title('Noisy Signal (with 7.8 kHz Noise)');

subplot(3, 1, 3);

plot(t\_plot, filtered\_signal(1:Fs\*12));

grid on;

xlabel('Time (s)');

ylabel('Amplitude');

title('Filtered Signal');

disp('Observe the plots to analyze the effect of filtering.');