Table of Contents

[project.m 1](#_Toc69165578)

[sound files 5](#_Toc69165579)

project.m\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

%% Q1

F=[500 ,1500, 2000, 3000]; % band pass edge frequencies

A=[0, 1, 0]; % a stop band(0), pass band(1) and a stop band(0)

Dev = [0.01, 0.01, 0.001]; % Ripples in bands

[N,Fi,Ai,W]=firpmord(F,A,Dev,8000);

% filter length / approximate order: N, normalized frequency band edges: Fi, frequency band amplitudes: Ai, and weights W that meet input specifications

B1=firpm(N,Fi,Ai,W);

% B1 gives the filter coefficients

freqz(B1,1); % displaying the magnitude and phase responses of the filter

N;

B1;

plot(abs(B1))

plot(angle(B1))

plot(abs(freqz(B1,1)))

plot(angle(freqz(B1,1)))

%verify

freqz(B1,1,1024,8000);

xline(500,'--k',sprintf('fs1=%0.0f', 500))

xline(1500,'--k',sprintf('fp1=%0.0f', 1500))

xline(2000,'--k',sprintf('fp2=%0.0f', 2000))

xline(3000,'--k',sprintf('fs2=%0.0f', 3000))

yline(-20\*log10(0.01),'--k',sprintf('Rs1=%0.2f', -20\*log10(0.01)))

yline(-20\*log10(0.001),'--k',sprintf('Rs2=%0.2f', -20\*log10(0.001)))

%% Q2

[y,fs]=audioread('music\_noisy.wav'); % reading the music with noise from the .wav file

sound(y,fs); %listening to the corrupted music file, y: input signal, fs: sample frequency of y

plot(abs(fft(y))); %Fourier transform of the noisy music

xlabel('Samples');

title('Fourier transform of the signal ');

freqz(y,1,1024,fs); % displaying the magnitude and phase responses of the noisy music

%% TESTING PHASE

%used only high pass

% --------

fHp = fir1(256,5564/(fs/2),'high');

w = 0:0.001\*pi:pi;

xHpf = filter(fHp,1,y);

sound(xHpf,fs)

plot(abs(fft(xHpf)));

freqz(xHpf,1,1024,fs);

%%%%%%%%% removes frequncies in the music as well...no base anymore...not l

%%%%%%%%% loud enough

%% trying different values

fbs = fir1(64, [1087, 1097]/(fs/2), 'stop');

fbs2 = fir1(64, [2719, 2743]/(fs/2), 'stop');

fbs\_Combine = conv(fbs, fbs2);

xbsf = filter(fbs,1,y);

sound(xbsf,fs)

plot(abs(fft(xbsf)));

fbs = fir1(256, [9302, 9356]/(fs/2), 'stop');

fbs2 = fir1(256, [2734, 2777]/(fs/2), 'stop');

fbs\_Combine = conv(fbs, fbs2);

test = filter(fbs2,1,y);

sound(test,fs)

plot(abs(fft(test)));

freqz(test,1,1024,fs);

%% good but could do better

%-----------------------

fbs = fir1(256, [500, 1500]/(fs/2), 'stop');

fbs2 = fir1(256, [2500, 3000]/(fs/2), 'stop');

fbs\_Combine = conv(fbs, fbs2);

test = filter(fbs\_Combine,1,y);

sound(test,fs)

plot(abs(fft(test)));

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

%% Final Time Domain Method with 3 filters

%------

fhp1 = fir1(400, 274/(fs/2), 'high'); % high pass filter

freqz(fhp1,1);

title('High Pass Filter Mag and Phase responses');

fbs2 = fir1(400, [900, 1500]/(fs/2), 'stop'); % band stop filter 1

freqz(fbs2,1);

title('Band Stop Filter1 Mag and Phase responses');

fbs3 = fir1(400, [2500, 3000]/(fs/2), 'stop'); % band stop filter 2

freqz(fbs3,1);

title('Band Stop Filter2 Mag and Phase responses');

filter\_Combine = conv(fhp1, fbs2); % convolution of 1st 2 filters

filter\_Combine = conv(filter\_Combine, fbs3); % convolution with the remaining filters

freqz(filter\_Combine,1);

title('Combined Filters Mag and Phase responses');

filtered\_music = conv(filter\_Combine, y); % convolution of the signal with filters

plot(abs(fft(filtered\_music))); % Fourier transform of the filtered music

title('Fourier transform Plot of the input signal after removing the noise');

%test2 = filter(fbs\_Combine2,1,y);

sound(filtered\_music,fs)

plot(abs(fft(filtered\_music)));

% freqz(filtered\_music,1,1024,fs);

sound(filtered\_music,fs) % listen to verify

audiowrite('Filtered\_music\_td\_method.wav',filtered\_music,fs); % store the new filtered music

%another way

tmp = conv(fbs, y);

tmp2 = conv(tmp, fbs2);

plot(abs(fft(tmp2)));

sound(tmp2, fs);

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

%% Final Frequency Domain Method with 3 filters

%-------------

[y,fs]=audioread('music\_noisy.wav');

fhp1 = fir1(400, 274/(fs/2), 'high'); % high pass filter

fbs = fir1(256, [900, 1500]/(fs/2), 'stop');

fbs2 = fir1(256, [2500, 3000]/(fs/2), 'stop');

filter\_Combine = conv(fhp1, fbs2); % convolution of 1st 2 filters

filter\_Combine = conv(filter\_Combine, fbs3);

fft\_y = fft(y); % Fourier transform of the input signal (music with noise)

fft\_fhp1 = fft(fhp1, length(fft\_y))'; % Fourier transform of first filter

fft\_fbs2 = fft(fbs2, length(fft\_y))'; % Fourier transform of second filter

fft\_fbs3 = fft(fbs3, length(fft\_y))'; % Fourier transform of third filter

filtered\_music = fft\_y.\*fft\_fhp1.\*fft\_fbs2.\*fft\_fbs3; % multiply input signal with the filters in their frequency domains

plot(abs(filtered\_music));

title('Fourier transform Plot of the input signal after removing the noise');

% freqz(filtered\_music,1,1024,fs);

filtered\_music\_time\_domain = ifft(filtered\_music); % inverse fourier transform of the final filtered signal to listen it later

sound(filtered\_music\_time\_domain,fs) % listen to verify

audiowrite('Filtered\_music\_fd\_method.wav',filtered\_music\_time\_domain,fs); % store the new filtered music

sound files\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Noisy music: [music\_noisy.wav](https://drive.google.com/file/d/1gAtWMpiYq1FV0v0BgtWDSePDaTjp44Iz/view?usp=sharing)

Music with no noise – Filtered using Time Domain(td) Method: [Filtered\_music\_td\_method.wav](https://drive.google.com/file/d/1_AiXGNxISbXtBDj0L9DaaEQmUaMBgbuM/view?usp=sharing)

Music with no noise – Filtered using Frequency Domain(fd) Method: [Filtered\_music\_fd\_method.wav](https://drive.google.com/file/d/1RUKLi9QHd0SY7EUQS-9is5FCEawzbUiM/view?usp=sharing)

Both the methods give similar results