EE 450 - Digital Signal Processing

Department of Electrical & Computer Engineering
San Diego State University

# Project 2

Date: May 17th, 2021

Name: Andrew Chung, RedID: 821542300

Given a noisy signal in the form of an audio file, the goal is to clean it up. Obviously, the existence of random signals prevents us from being able to completely clean up the signal, however we can still get it to sound good to the human ear. This can be done by designing as many FIR filters as necessary. I began my analysis of the noisy signal by first plotting the signal to determine the duration of the low and high frequency portions of the signal. We can see that the low frequency duration is 529200 (which we can also obtain from $f_s \cdot 12$) and the high frequency duration is 810000. Next, we must determine the noise signal frequencies to filter out. This is done by taking the Fast Fourier Transform (FFT) of the noisy signal, shifting it to center at 0, and then plotting the magnitude vs. frequency.

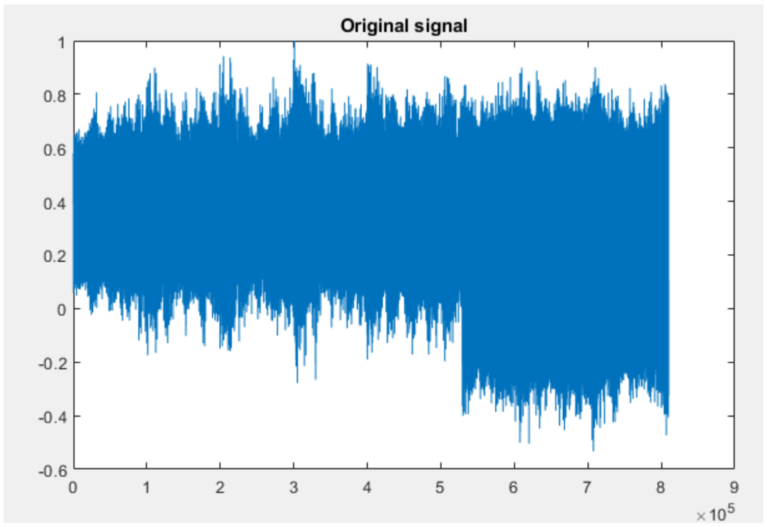

*Figure 1: Original signal*

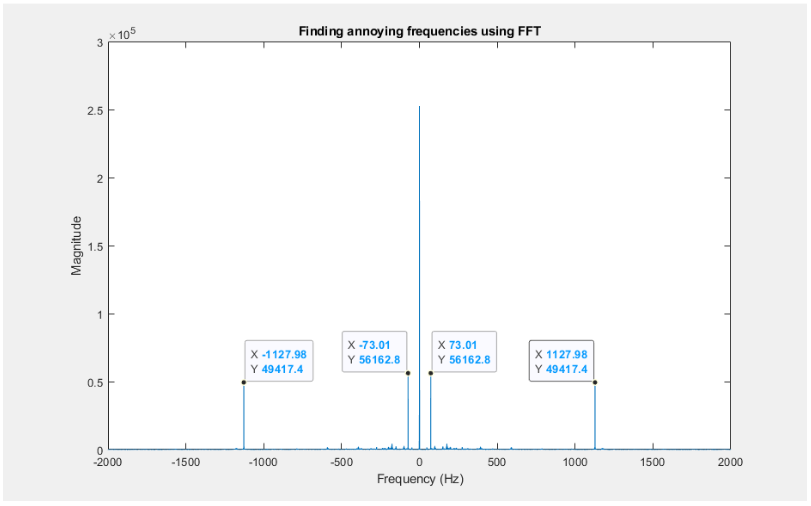

*Figure 2: Finding frequencies to filter out using the FFT of the noisy signal*

Here, we can easily see that the frequency of the first annoying signal is about 73Hz, which is expected as it is below 200Hz as given in the prompt. The frequency of the second signal is about 1128Hz, which is also expected as the prompt indicates it should be above 1000Hz.

The first filter I designed was a band-stop filter between frequencies 45Hz and 150Hz to filter out the first annoying signal at about 73Hz. The filter was tuned by trial and error. Since the FIR only needs to be run for the lower frequency duration of the signal, I pass only the low frequency portion of the signal through this FIR (the first 12 seconds of the original signal). I determined the order of the filter to be 1500 by trial and error as well. This was difficult because the signal would be clipped if the bandwidth was too large, and the lower pitch notes would be lost. The higher the bandwidth is, the lower the order of the FIR filter. Additionally, the lower the FIR filter order, the shorter the "startup" becomes.

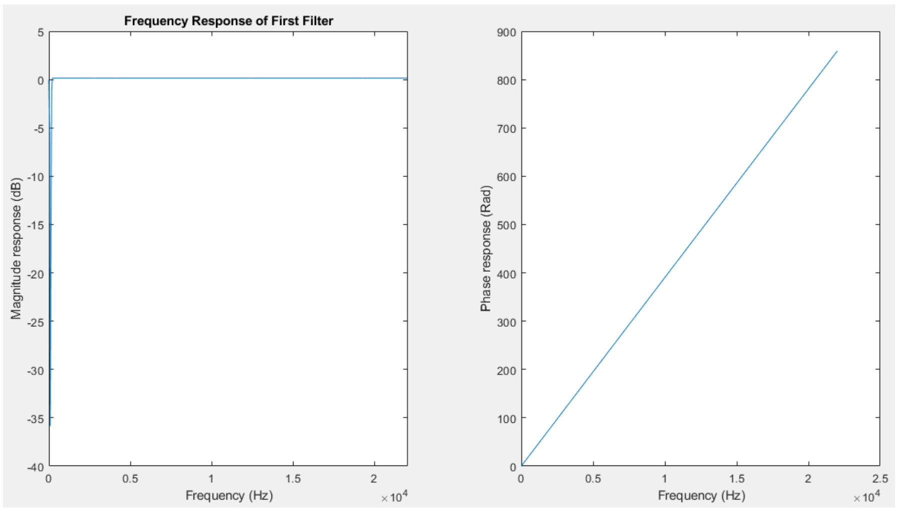

*Figure 3: Frequency Response of the First Filter*

The second filter I designed was also a band-stop filter between frequencies 1000Hz and 1300Hz to filter out the second annoying signal at about 1128Hz. Since the FIR only needs to be run for the higher frequency duration of the signal, I pass only the high frequency portion of the signal through this FIR. I determined the order of this filter to be 500 by trial and error.

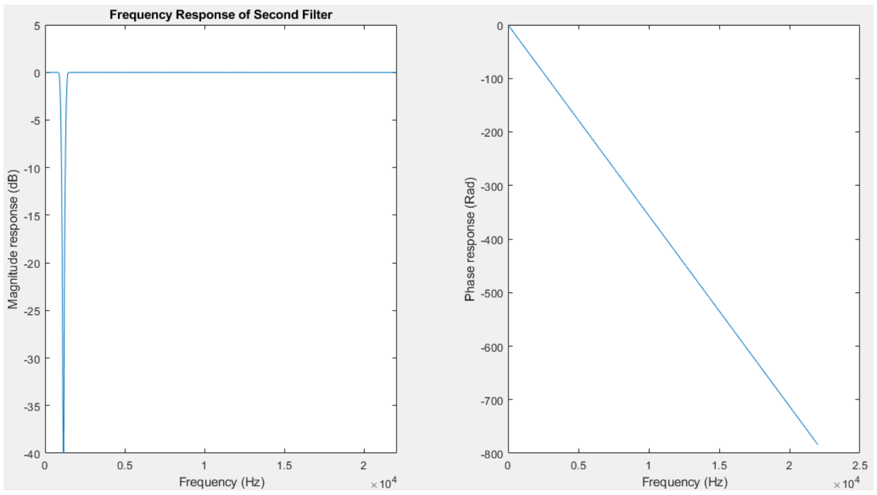

*Figure 4: Frequency Response of the Second Filter*

The third and final filter I designed was another band-stop filter between the frequencies 3500Hz and 5200Hz to filter out the white noise, which is a uniform mixture of random energy at every frequency. From the prompt, I knew the frequency range of the colored noise would be less than 5500Hz. While looking at the FFT of the original signal plotted logarithmically, using trial and error, and using the “play()” function to test the resulting signal, I found a range that filtered out this colored noise. Unlike the first two band-stop filters, the band-stop filter is applied to both the filtered first half signal and second half signal. The signals were combined with a “for” loop.

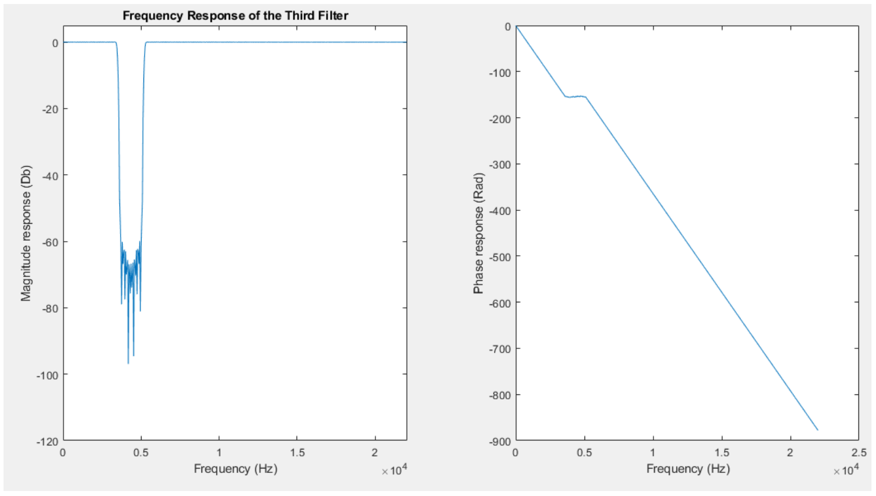

*Figure 5: Frequency Response of the Third Filter*

Finally, I plotted the filtered signal, along with a summary of the 3 signals before and after they enter the band-stop filters

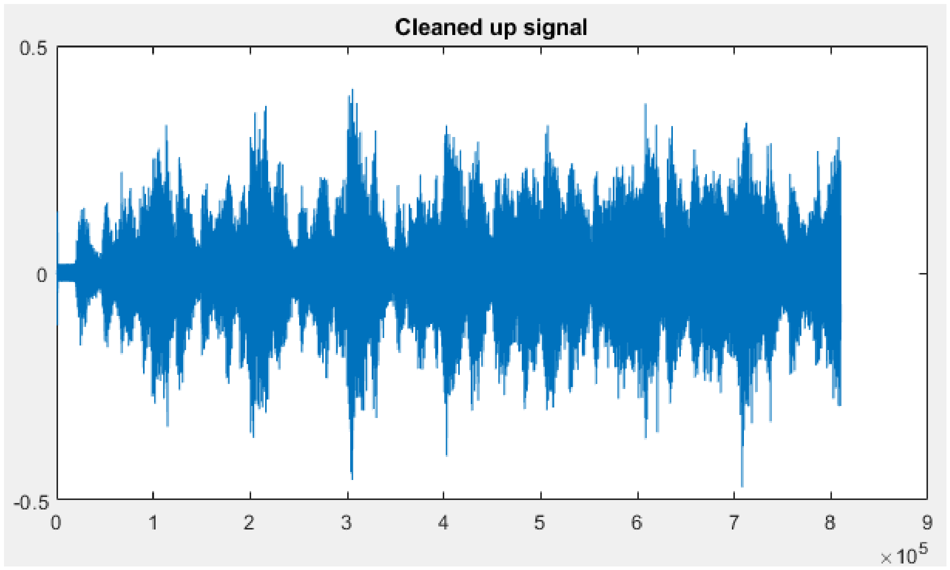

*Figure 6: Cleaned up signal*

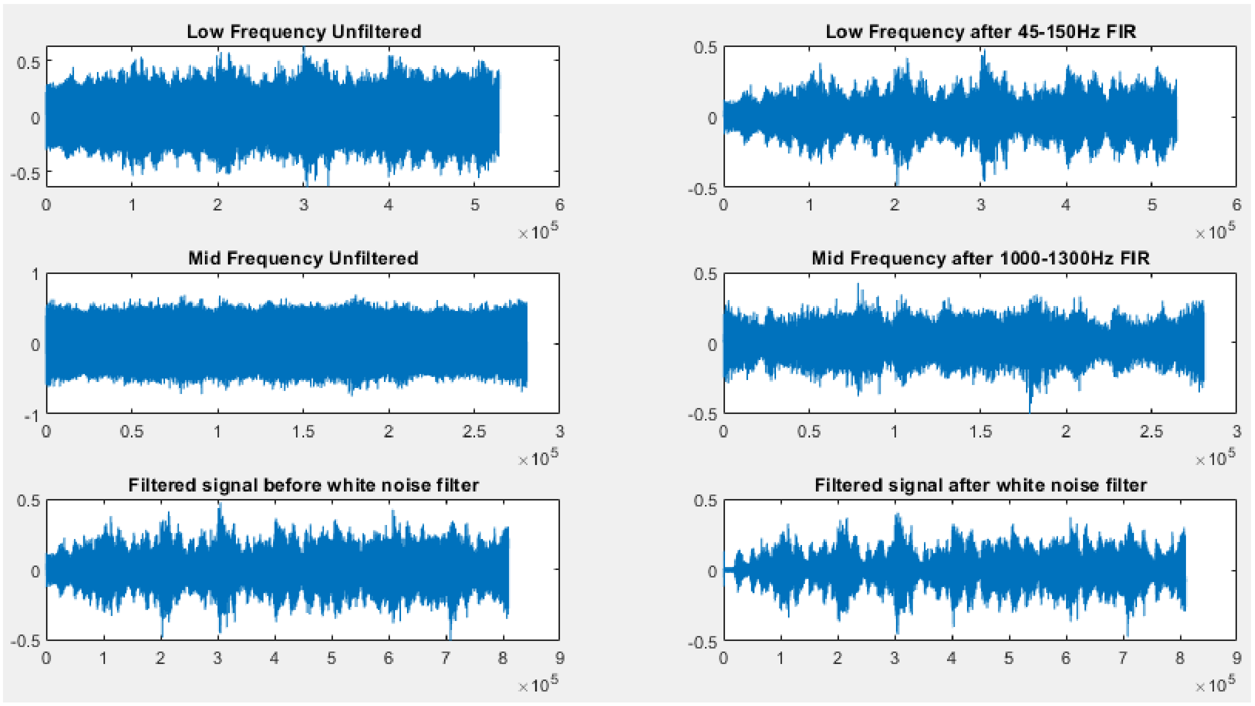

*Figure 7: Before and after effects of the filters*

## **Appendix A** - MATLAB Code

```
% EE-450 Project 2

%{
One can use MATLAB as a very good environment to work on audio signals. MATLAB has
powerful capabilities to read and write audio wave files (.wav) and play them back. An audio signal in
MATLAB is a column or row vector containing real numbers between -1 and +1.

The goal of this project is to design as many FIR filters as needed to clean up the entire audio file.
Obviously, it is impossible to completely clean up the audio file because of the existence of random
signal but the file can be cleaned to obtain an acceptable quality.
%}

clear, clc;

% ---------- Reading and writing audio files in MATLAB -----------

% reading a 'wav' file
% x = column vector w/ audio signal
% fs = sampling frequency of audio signal
[x,fs] = audioread('imperial_march_noisy.wav');

figure(1)
plot(x)
title('Original signal')

%{
Since the valid range of the numbers in a voice file is between -1 and +1, the "wavwrite"
command will clip the signal exceeding this range, from both negative and positive sides, to keep the
signal between -1 and +1. This will cause a significant distortion in the audio signal and has to be
prevented. In order to prevent this from happening you should create (manipulate) the audio signal in
such a way that it will not exceed -1 and +1.
%}


% -------------- Analyzing the spectrum of the signal ---------------

lowFreq = 529200;       % Low frequency signal duration (fs * 12)
highFreq = 810000;      % High frequency signal duration (visual inspection)

xLowFreq = x(1:lowFreq);                % Low frequency part of original signal (x)
xHighFreq = x(lowFreq:highFreq);        % High frequency part of original signal (x)
lowFreqMean = mean(xLowFreq);           % Mean of low frequency part of x
highFreqMean = mean(xHighFreq);         % Mean of high frequency part of x
xLowFreq = xLowFreq - lowFreqMean;      % Ofsetting low frequency part of x
xHighFreq = xHighFreq - highFreqMean;   % Offsetting high frequency part of x

% Determining noise signal frequencies
y=fft(x);                   % Fast Fourier Transfrom of noisy signal
yShifted = fftshift(y);     % Plot centered at 0
yLength = length(y);        % Length of the signal
magSpec = abs(yShifted);    % Magnitude spectrum
freqAxis = (-yLength/2:yLength/2-1)*(fs/yLength);   % x-axis in Hertz (Hz)

figure(2)
plot(freqAxis, magSpec);    % Plot
xlim([-2000 2000])          % To view the frequencies easier
xlabel('Frequency (Hz)')
ylabel('Magnitude')
title('Finding annoying frequencies using FFT')
```

```matlab
figure(3)
plot(20*log(abs(y)));
title('Finding the colored noise')

% ----------- Designing linear-phase FIR filters in MATLAB --------------

% Design of low-frequency band stop filter, applied to low frequency
wc1 = 45;                   % Lower lim
wc2 = 150;                  % Upper lim
wn1 = [wc1 wc2]/(fs/2);     % Cutoff frequency
N1 = 1500;                  % Order of the filter
h1 = fir1(N1,wn1,'stop');   % Band-Stop linear-phase FIR filter
x1 = filter(h1,1,xLowFreq); % Low freq portion after 40-150Hz FIR
[H1,w1] = freqz(h1,1);      % Frequency response

figure(4)
subplot(1,2,1)
plot(w1*fs/(2*pi),20*log10(abs(H1)))
title('Frequency Response of First Filter')
xlabel('Frequency (Hz)')
ylabel('Magnitude response (dB)')
axis([0,fs/2,-40,5])
subplot(1,2,2)
plot(w1*fs/(2*pi),unwrap(angle(H1)))
xlabel('Frequency (Hz)')
ylabel('Phase response (Rad)')

% Design of mid-frequency band stop filter, applied to high frequency
wc3 = 1000;                 % Lower lim
wc4 = 1300;                 % Upper lim
wn2 = [wc3 wc4]/(fs/2);     % Cutoff frequency
N2 = 500;                   % Order of the filter
h2 = fir1(N2,wn2,'stop');   % Band-Stop linear-phase FIR filter
x2 = filter(h2,1,xHighFreq);% Mid freq portion after 40-150Hz FIR
[H2,w2] = freqz(h2,1);      % Frequency response

figure(5)
subplot(1,2,1)
plot(w2*fs/(2*pi),20*log10(abs(H2)))
title('Frequency Response of Second Filter')
xlabel('Frequency (Hz)')
ylabel('Magnitude response (dB)')
axis([0,fs/2,-40,5])
subplot(1,2,2)
plot(w2*fs/(2*pi),unwrap(angle(H2)))
xlabel('Frequency (Hz)')
ylabel('Phase response (Rad)')

x3=x1;                      % Combining both filtered signals
for i=lowFreq:highFreq
    x3(i)=x2(i-lowFreq+1);
end

% Design of high-frequency band stop filter, applied to entire signal
wc5 = 3500;                 % Lower lim
wc6 = 5200;                 % Upper lim
wn3 = [wc5 wc6]/(fs/2);     % Cutoff frequency
N3 = 400;                   % Order of the filter
h3 = fir1(N3,wn3,'stop');   % Band-Stop linear-phase FIR filter
x4 = filter(h3,1,x3);       % High freq portion after 40-150Hz FIR
[H3,w3] = freqz(h3,1);      % Frequency response
```

```
figure(6)
subplot(1,2,1)
plot(w3*fs/(2*pi),20*log10(abs(H3)))
title('Frequency Response of the Third Filter')
xlabel('Frequency (Hz)')
ylabel('Magnitude response (Db)')
axis([0,fs/2,-120,5])
subplot(1,2,2)
plot(w3*fs/(2*pi),unwrap(angle(H3)))
xlabel('Frequency (Hz)')
ylabel('Phase response (Rad)')

%------------------------ More Plots ---------------------------

figure(7)
plot(x4)                      % Cleaned up signal
title('Cleaned up signal')

figure(8)
subplot(3,2,1)
plot(xLowFreq)
title('Low Frequency Unfiltered')
subplot(3,2,2)
plot(x1)
title('Low Frequency after 40-150Hz FIR')
subplot(3,2,3)
plot(xHighFreq)
title('Mid Frequency Unfiltered')
subplot(3,2,4)
plot(x2)
title('Mid Frequency after 1000-1300Hz FIR')
subplot(3,2,5)
plot(x3)
title('Filtered signal before white noise filter')
subplot(3,2,6)
plot(x4)
title('Filtered signal after white noise filter')

%-----------------Writing "wav" file and playback-----------------

%writing a "wav" file from an audio signal
audiowrite('imperial_march_filtered.wav', x4, fs);

% Playback an audio signal in MATLAB
P = audioplayer(x4,fs);
play(P);
```