An Analysis of FIR Filters on Audio Files

Michael Dickerson
ECE Department
Missouri University of Science and Technology
Rolla, Missouri
mad5r8@umsystem.edu

Josh Gervich
ECE & CS Department
Missouri University of Science and Technology
Rolla, Missouri
jcghnm@umsystem.edu

Jack Kufa
ECE & CS Department
Missouri University of Science and Technology
Rolla, Missouri
jhkmw3@umsystem.edu

Jordan Marrow
ECE Department
Missouri University of Science and Technology
Rolla, Missouri
jdmrvf@umsystem.edu

***Abstract*— An FIR filter was used to filter out certain elements of an audio clip. Two FIR filters were made from scratch in MATLAB to filter out the bass and hi-hats of an audio file. Then, a built-in MATLAB Butterworth filter was used to filter out the high pitched synths of the same audio file. The made-from-scratch filter worked well at filtering out the hi hat and bass from the audio file. The built in Butterworth filter didn't work as well as the made-from-scratch filter because high pitched synths were taken out at the cost of some of the vocals.**

I. Introduction

FIR and IIR filters attenuate certain frequencies of a signal while leaving other frequencies mostly unchanged. Because of this, these filters can be used to filter out certain parts of audio. This is a functionality which is extremely useful for industries such as the music industry. If certain frequencies are undesired, like very high pitched ones for example, then these filters can remove those. This report will function as a record of an experiment, in which two FIR filters are designed and used to filter out components of a song sample. These filters will use the Buttersworth and Hanning windowing functions, and will remove the base and hi-hats from the 3 second music sample taken from "GONE GONE / THANK YOU" by Tyler The Creator. The Hann filter will be created modularly using the fft and power spectrum of the sample to determine specifications. The Buttersworth filter will be implemented using a built-in function of matlab. Throughout this report, why and how those filters were used will be discussed and the results will be compared.

II. An Overview of Music Selection

The song snippet is sampled from "GONE GONE / THANK YOU" by Tyler The Creator. The snippet contains vocals, a low-pitched beat, and high pitched synths. This song was selected due to the distinct tones present.

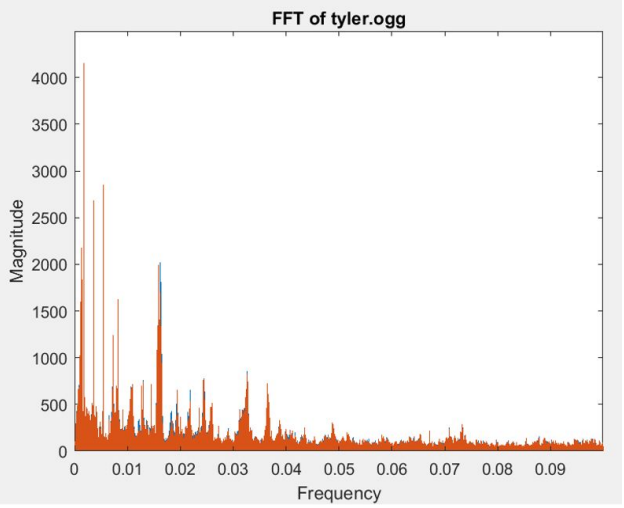

Figure 1. Graphical representation of sample's Fourier Transform.

There were notable spikes in the FFT of the sample due to different frequencies produced by the different instruments in the sample. In addition, there was a good amount of noise between these peaks. This was expected from the FFT, and no strong conclusions may be drawn from it.

To reduce the amount of noise and get a clearer idea of the specific frequencies instruments from the sample occupy, a power spectrum can be used.

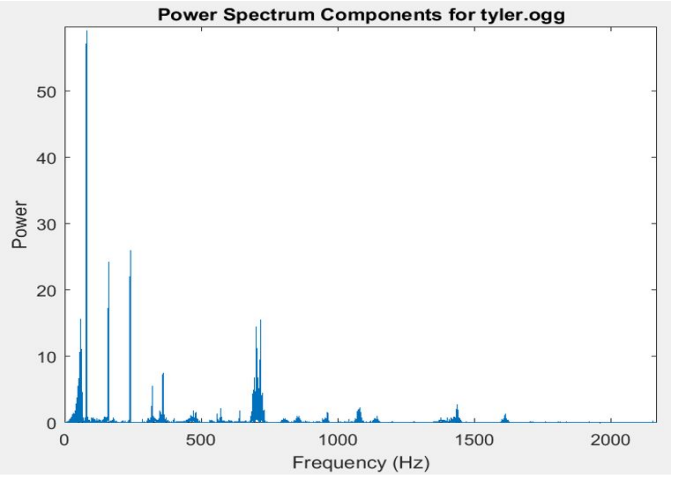

Figure 2. The graphical representation of sample's power spectrum.

In the power spectrum of the sample, spikes had a greater amplitude relative to the noise. Notable spikes occurred at about 60Hz, 80Hz, 160Hz, 240Hz, 360Hz, and a short band of spikes around 708Hz. In addition, there was a small amount of power in signals well into the KHz range.

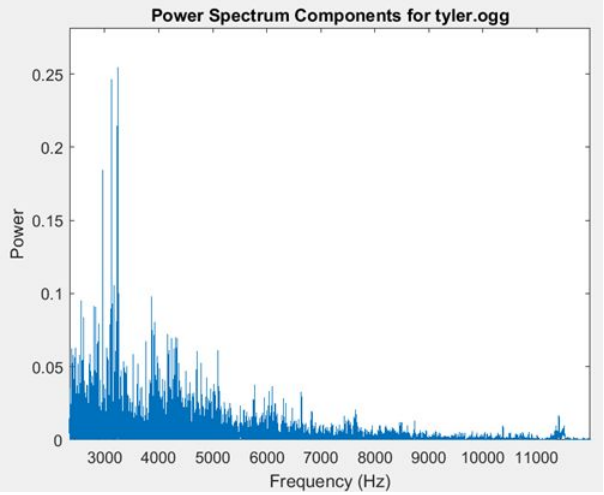

Figure 3. The 2.8kHz-12kHz power spectrum.

These can be roughly grouped together and correlated to specific instruments. This was done using the ear and approximate ranges for the instruments listed, from Heroic Academy's short course on audio engineering. [1]

Range (Hz)	**Instrument**
50-170	Bass Line
330-390	Vocals
700-720	Synthesizer
1300-10000	Hi-Hat

Table 2. The frequency analysis of song elements.

These associations were verified by the results of the filters in the upcoming sections. While noise was less prevalent in the power spectrum, it was not altogether removed. There were frequent variations before 500 Hz, and a final spike at about 750 Hz, before the variance began to stabilize. This noise was likely from two sources. Some of the power outside of peaks was due to frequencies produced by an instrument besides its primary frequency, or "harmonics." [2] It should also be noted that part of the frequencies were simply noise produced by the recording and processing of the sample.

III. FIR Design and Application

A. Target Specifications

Two filters were constructed. The first was constructed with the intent of extracting the hi-hat from the sample. Based on the associations made in section III, the filter would have to keep the 1.3-10kHz range and stop the rest. During the design of the filter, however, it was discovered that the majority of the distinctive hi-hat sound was above 4.5kHz, and the filter was adjusted accordingly. The goal of the second filter was to extract the bass drum, the lowest part of the bass line. This filter had to extract the 0-50Hz range and attenuate the other frequencies. The mode of this filter was on the order of 300; this was to provide greater roll off rates between the pass and stop band and also increase the sharpness of the cutoff. The following graphs showcase the magnitude response for both the high-pass and low-pass filter:

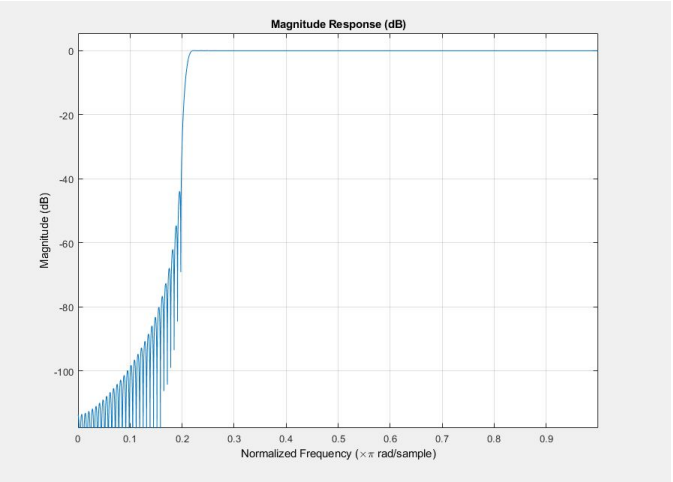

Figure 4. The magnitude response for the high-pass Hanning filter.

The stopband attenuation of the high-pass filter was less than -40 decibels for normalized frequencies that were less than 0.2.

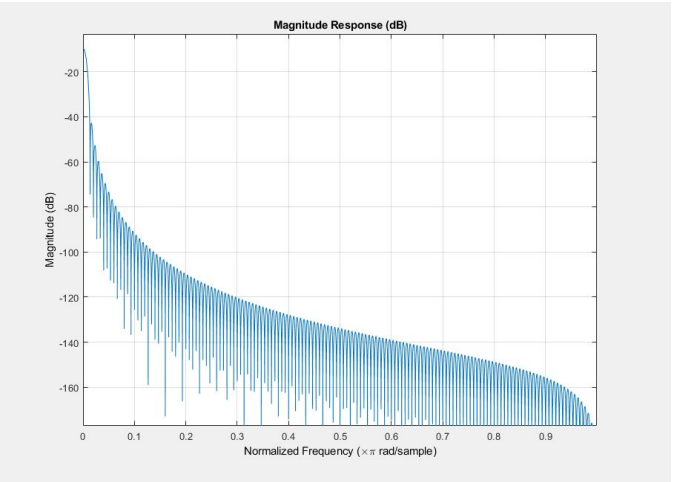

Figure 5. The magnitude response for the low-pass Hanning filter.

The stopband attenuation of the low-pass filter was less than -60 decibels for normalized frequencies that were greater than 0.02.

B. Filter Selection

The Hann window function was selected not only because it is easy to implement from scratch, but also because it is effective. The function has a sinusoidal shape with wide peaks and low side lobes that touch zero at both ends, eliminating all discontinuity. These properties were useful for measuring and targeting noise where robust side lobes do not present a problem in filtering. In most cases where this condition was present, the Hann windowing function was satisfactory, boasting good frequency resolution and reduced spectral leakage. The sound sample chosen has infrequent and definitive peaks, meaning that with a Hann windowing function it was easy to make distinctions between musical components and filter out unwanted sound.

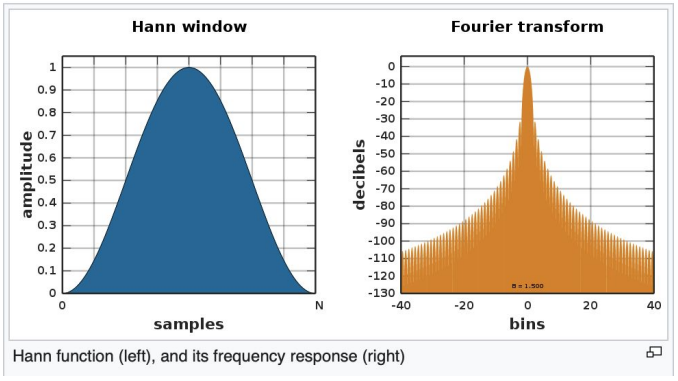

Figure 6. The graphical representation of the Hann window and its associated frequency response

C. Level of Success

The Hann filter was very successful in its implementation. This basis was founded on the results of two different experiments on the sound sample. One was a high-pass filter design while the other was a low-pass filter design. With the high-pass design, the Hanning filter was able to separate the hi-hats (high-pitched drums,) which occurred at the highest frequency in the sample, from the rest of the sample. With the low-pass design, there was less of a successful result. Occurring at the lowest frequency of the sample was the bass line of the sample, and although the filter was able to isolate this component, there was still a hint of vocals as well. It is to be assumed that the lowest frequency of the vocals aligns with the frequency of the base, and this is the reasoning for its inclusion. Nevertheless, the addition of vocals in the resulting sound was miniscule and the filter proved to be effective at encapsulating the low frequency.

In these figures you can see the changes in sound pressure from the original audio file after the filters were applied:

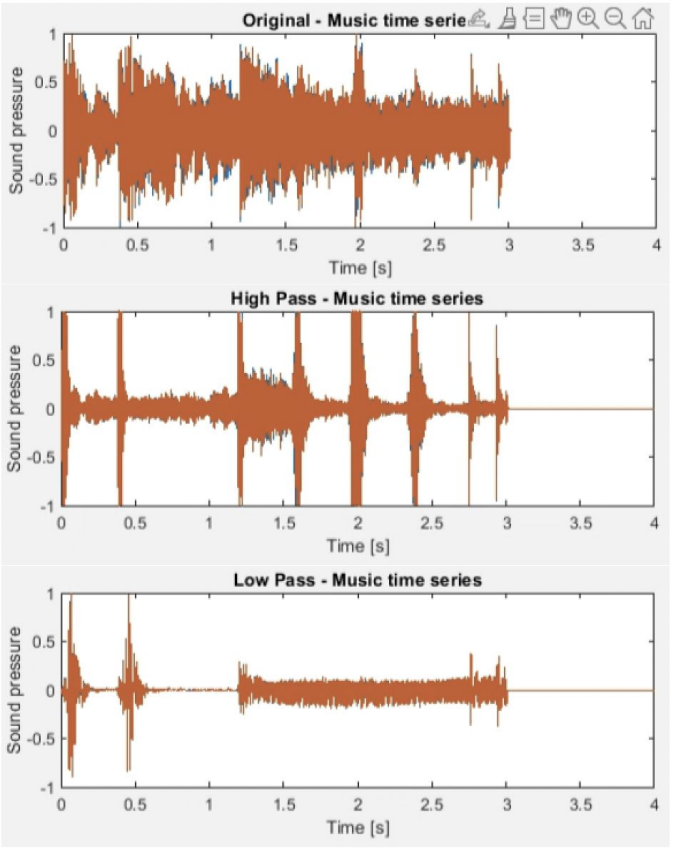

Figure 7. The sound pressures of the sample before and after being filtered.

Notice the difference of the filtered sound profiles. The high-pass filter attempted to maintain the high frequencies while minimizing the low frequencies. The low-pass filter tried to do the opposite, removing most of the high frequencies while preserving the low frequencies.

IV. Built-In Filtering Function

A. Target Specifications

The built in filtering functions were used to remove the synthesizer present in the sample. This filter needed to remove the 700-720Hz range and leave other frequencies unattenuated.

B. Filter Selection

In order to remove the high-pitched synths, a Butterworth low-pass filter was used. The butterworth filter has the advantage of producing a smooth and flat frequency response, meaning little to no ripple is present in the passband and the stopband rolls off towards zero. However, as a low-pass filter, effectively everything past the cutoff frequency was removed. This means that the hi-hat was also removed. A minimum stopband attenuation of 15-20dB was desired, due to the greatest peak in the target stopband being about 18dB.

In MATLAB, the built-in 'butter' function can be used to design a low-pass, high-pass, band-pass, or even bandstop filter. In this case, a low-pass version was used because a low-pass filter can reduce the intensity of the higher frequencies, and since the primary concern of this filter was to remove one of the highest frequency elements, a low-pass filter was an appropriate solution. The low-pass filter used was an 11th order filter, because the higher the order, the closer the butterworth filter becomes to an ideal low-pass filter. In this case, a 12th order ended up being too large, and would end up completely distorting the sound file.

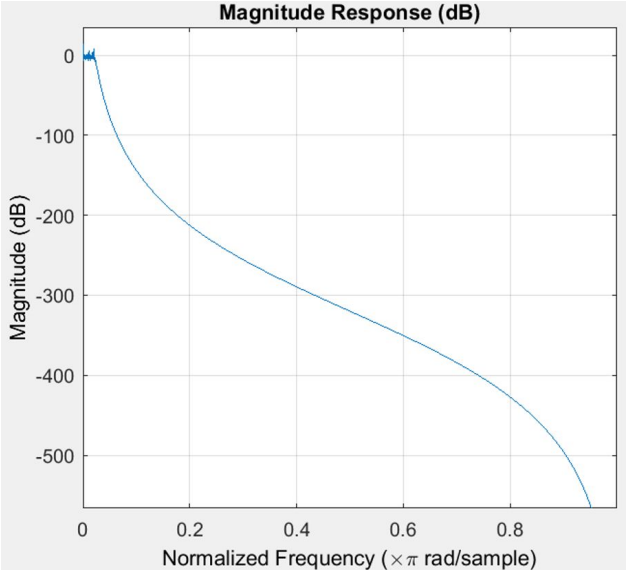

Figure 8. The magnitude response of the butterworth filter.

The magnitude response properly showcases the properties of a low-pass butterworth filter. The transition band is very smooth but has a relatively low slope.

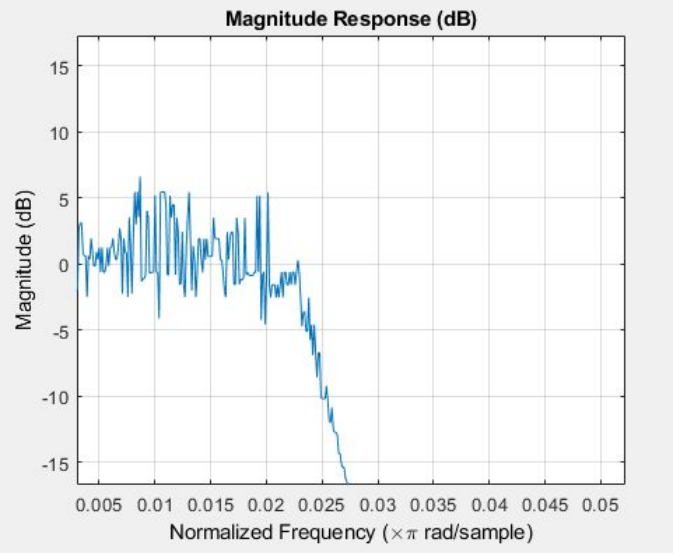

Figure 9. The passband ripples.

A closer look at the ripples in the passband show that there is very minimal fluctuation, consistent with the properties of a butterworth filter. The peak ripple in the passband is about 5 dB. The minimum stopband attenuation of 15dB occurs at 650Hz. The stopband levels out somewhat before taking another dive, and has very minimal ripple throughout.

C. Level of Success

At first, when the initial (cutoff) frequency was set to 700 Hz and the order was set to 12, the filter was unsuccessful at removing the synthesizer. In order to fix this, adjustments were made; the initial frequency was decreased to 500 Hz, and the order was decreased to 11. This necessary change likely has something to do with the frequencies in the 500-700 range contributing to the synthesizer's sound profile.

After implementing this change, the filter worked moderately well. It was very effective at filtering out the high-pitched synth. However, it also affected the vocals, and caused additional distortion as a result of the Gibbs' phenomenon, as evident from the overshoot on the edge of the stopband. The biggest consequence of using a low-pass butterworth filter was the lack of precision available past the cutoff frequency. With the low-pass butterworth filter, in order to remove the synthesizer, the hi-hats also had to be removed.

V. Conclusion

Overall, the made-from-scratch low and high-pass filters worked better than the built in filter in terms of how the audio file needed to be changed. The made-from-scratch filters filtered out the bass and hi-hat sounds very effectively. The built in function did end up filtering out the high pitched synth but at the cost of the vocals being distorted and the hi-hat being removed. It can be concluded that building a filter from scratch can better fit as a solution to the problem.

VI. References

P. MacFarlane and D. Schnibbe, “Harmonics,” 11-Jan-2016. [Online]. Available: https://www.guitarlessonworld.com/lessons/harmonics/. [Accessed: 05-Dec-2020].

T. van Doorne April 12th, “Mixing Instruments & Synths - A Step-by-step Guide,” *Heroic Academy*, 12-Apr-2018. [Online]. Available: https://heroic.academy/how-to-mix-music-part-5-mixing-instruments-synths/. [Accessed: 05-Dec-2020].

National Instruments Corp. (2019). Understanding FFTs and Windowing. *National Instruments*. Retrieved December 3, 2020, from https://download.ni.com/evaluation/pxi/Understanding%20FFTs%20and%20Windowing.pdf

Tyler the Creator, (2019). Gone Gone / Thank You, *Columbia.* Sampled December 3, 2020, from https://www.youtube.com/watch?v=pVInBRkoKgY

VII. Appendix

FIR_from_scratch.m

```
%Headers
clear all % this clears all your variables
load handel;

%%Definition of audio file
[x,Fs]=audioread('tyler_original.ogg'); % reads in the file
m = length(x);              % calculates the length of the signal
m1 = pow2(nextpow2(m));   % chooses the next higher power of 2

%%Sound pressure spectrum of original audio
figure(1);
subplot(2,1,1)
plot((1:m)/Fs,x);
%formatting
xlabel('Time [s]');
ylabel('Sound pressure');
title('Original - Music time series');
xlim([0 4])
ylim([-1 1])

%%Play original audio
p8 = audioplayer(x,Fs); %define audio object
playblocking(p8); %play audio object

%%FFT and power spectrum of original audio
X = fft(x,m1);               % takes the fft of signal
f = (0:m1-1)*(Fs/m1);        % sets your frequency variable range
power = abs(X).^2/m1;        % calculates the power of the signal
figure(2);                   % creates a new plot window
plot(f(1:floor(m1/2)), power(1:floor(m1/2)));      % plots the power
xlabel('Frequency (Hz)');           % labels the horizontal axis
ylabel('Power');                    % labels the vertical axis
title('Original Power Spectrum for tyler_orginal.ogg');  % graph title
xlim([0 3000])
ylim([0 60])

%%High-pass Hann Filter definition
Fc = 4600; %frequency cut-off
fc = Fc/Fs; %normalized cut-off frequency
M = 300; %filter order
n = 0:1:M;
w = 0.5.*(1-cos(2*n*pi/M)); %hann filter algorithm
%high-pass filter
hs = 2*(0.5)*sinc(2*(0.5)*(n-M/2))-2*fc*sinc(2*fc*(n-(M/2)));
hw = hs.*w; %windowed impulse response
HW = fft(hw,m1); %fft of signal
HWdB = 20*log10(abs(HW)); %convert to dB
```

```
HWdB2 = HWdB(1:(m1/2)); %shorten dB to plot
n1 = 0:1:(m1/2)-1; %new plot index
figure(3);
plot((n1/m1),HWdB2);
%formatting
title('Filter Magnitude in dB');
xlabel('Noramlized frequency');
ylabel('dB');

%%Apply Hann Filter to original audio
Y = X.*(HW.');
power2 = abs(Y).^2/m1; %convert to power
figure(4);
plot(f(1:floor(m1/2)), power2(1:floor(m1/2)));
%formatting
title('HP Filtered Power Spectrum for tyler_orginal.ogg');
xlabel('Frequency (Hz)');
ylabel('Power');
xlim([0 3000])
ylim([0 60])

%%Inverse FFT to retrieve filtered audio
fil = ifft(Y); %inverse fft
mfil = length(fil); %calculates the length of the signal

%%Play filtered audio
xtrafil = fil*3.6; % Increase volume
player = audioplayer(xtrafil, Fs);
play(player,[1 (get(player, 'SampleRate')*3)]);
audiowrite('tyler_scratchfilter_highpass.ogg',xtrafil,Fs)

%%Sound pressure spectrum of filtered audio
figure(5);
subplot(2,1,1)
%formatting
plot((1:mfil)/Fs,xtrafil);
xlabel('Time [s]');
ylabel('Sound pressure');
title('High-pass - Music time series');
xlim([0 4])
ylim([-1 1])

%%Low-pass Hann Filter definition
Fc = 50; %frequency cut-off
fc = Fc/Fs; %normalized cut-off frequency
M = 300; %filter order
n = 0:1:M;
w = 0.5.*(1-cos(2*n*pi/M)); %hann filter algorithm
%low-pass filter
hs = 2*fc*sinc(2*fc*(n-(M/2)));
hw = hs.*w; %windowed impulse response
HW = fft(hw,m1); %fft of signal
HWdB = 20*log10(abs(HW)); %convert to dB
HWdB2 = HWdB(1:(m1/2)); %shorten dB to plot
```

```
n1 = 0:1:(m1/2)-1; %new plot index
figure(6);
plot((n1/m1),HWdB2);
%formatting
title('Filter Magnitude in dB');
xlabel('Normalized frequency');
ylabel('dB');

%Apply Hann Filter to original audio
Y = X.*(HW.');
power2 = abs(Y).^2/m1; %convert to power
figure(7);
plot(f(1:floor(m1/2)), power2(1:floor(m1/2)));
%formatting
title('LP Filtered Power Spectrum for tyler_orginal.ogg');
xlabel('Frequency (Hz)');
ylabel('Power');
xlim([0 3000])
ylim([0 60])

%Inverse FFT to retrieve filtered audio
fil = ifft(Y); %inverse fft
mfil = length(fil); %calculates the length of the signal

%Play filtered audio
xtrafil = fil*3.6; % Increase volume
player = audioplayer(xtrafil, Fs);
play(player,[1 (get(player, 'SampleRate')*3)]);
audiowrite('tyler_scratchfilter_lowpass.ogg',xtrafil,Fs)

%Sound pressure spectrum of filtered audio
figure(8);
subplot(2,1,1)
%formatting
plot((1:mfil)/Fs,xtrafil);
xlabel('Time [s]');
ylabel('Sound pressure');
title('Low-pass - Music time series');
xlim([0 4])
ylim([-1 1])
```

Appendix 1. Code for FIR filter built from scratch

built_in_filter.m

```
%% Read in the file
clearvars;
close all;
[x,fs] = audioread('tyler_orginal.ogg');

%% Plot both audio channels
N = size(x,1); % Determine total number of samples in audio file

%% Plot power spectrum
df = fs / N;
w = (-(N/2):(N/2)-1)*df;
y = fft(x(:,1), N) / N; % Normalize freq
y2 = fftshift(y);
figure;
plot(w,abs(y2));
title('Power Spectrum');
subtitle('tyler_orginal.ogg');
ylabel('Power');
xlabel('Normalized frequency');

%% Create 11th order lowpass filter
n = 11;
initial_freq = 500 / (fs/2);
[b,a] = butter(n, initial_freq, 'low');
h = fvtool(b,a);

%% Filter the signal
f_out = filter(b, a, x);

%% Construct audioplayer object and play
p = audioplayer(f_out, fs);
p.play;

%% write to file
audiowrite('tyler_butterworth_lowpass.ogg',f_out,fs)
```

Appendix 2. Code for filter built using MALAB's butter function

2_signalAnalysis.m

```
%Michael Dickerson
%EE3410 Signal Analysis

%1a
clear all             % this clears all your variables
[x,Fs]=audioread('tyler_original.ogg'); % reads in the file
m=length(x);          % calculates the length of the signal
m1=pow2(nextpow2(m));   % chooses the next higher power of 2
X=fft(x,m1);          % takes the fft of signal
f=(0:m1-1)*(Fs/m1);   % sets your frequency variable range
figure(1); % open window for fft
stem((f./Fs),abs(X),'Marker', 'none'); % plot fft
xlabel('Frequency');
ylabel('Magnitude');
title('FFT of tyler_orginal.ogg');
power=abs(X).^2/m1;   % calculates the power of the signal figure
figure(2);% creates a new plot window
plot(f(1:floor(m1/2)),power(1:floor(m1/2)))        % plots the power
xlabel('Frequency (Hz)');
ylabel('Power');
title('Power Spectrum Components for tyler_orginal.ogg');
```

Appendix 3. Code for analysis of original song's signal