Chen, Samuel

Section: G2
Lab Partners:
N/A

Lab Report #7

MAE 107L – Dynamic Systems Laboratory
Fall 2017

OBJECTIVE

The objective of this laboratory report is to take a sample of multiple musical instruments, and to take the Fourier Transform of the musical samples in order to examine the magnitude spectrum of the musical instruments. This will be accomplished using various MATLAB commands, and the magnitude spectrum versus the frequency of the musical instruments will be plotted up until the Nyquist frequency. Then, using a Butterworth filter (high-pass, low-pass, or band-pass), the musical instrument samples are filtered based on the frequency spectrum generated by the Fourier Transform. Essentially, a Fourier Transform is taken to display the musical instrument spectrum, and then a Butterworth filter is applied based on the magnitude spectrum generated.

COMPOSITION DETAILS

The composition that I utilized was my own composition, which was inspired in part by the behavior of sinusoidal waves that we studied in this class (you may observe the rising and falling motion of the instrument notes mimicking a sinusoidal wave). The composition utilizes 7 different instruments – a female vocal line, a piano, first violins (plays the higher frequencies), second violins (plays the lower frequencies), a French horn (bassline), taiko drums (percussion), and a choir (background swells spanning a broad range of frequencies). The composition was composed using FL Studio 11 and various virtual instruments, shown as follows:

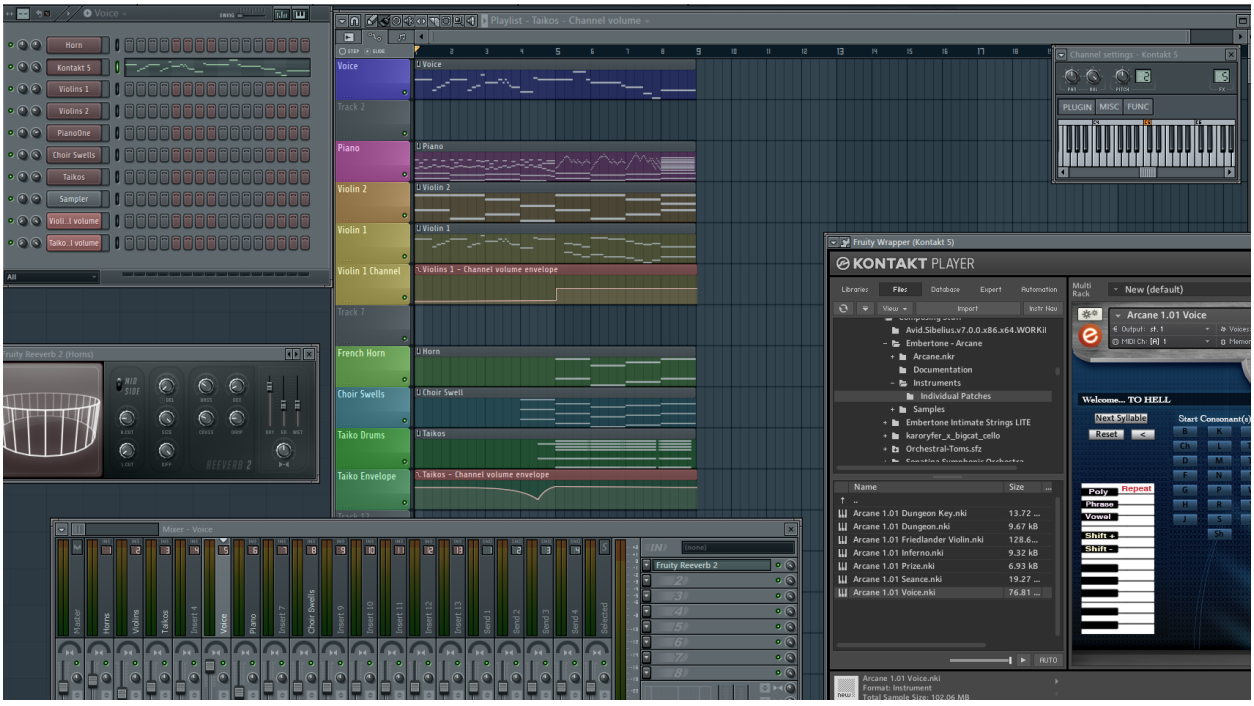

FIGURE 1: This figure shows the FL Studio 11 workspace that was used to compose the song.

Each individual instrument was then saved to a .WAV file and then loaded into MATLAB. The sampling frequency was 44100 Hz for the .WAV file (custom set using the composing program).

TIME PLOTS

After loading the music files into MATLAB, I then created a time vector using the sampling frequency (sample time is the inverse of sampling frequency) and then plotted the first five seconds of every instrument's amplitude versus this time vector.

It is important to note that the Taiko drums, choir, and French horn do not entre the piece until 15-17 seconds after the piece has started; as such, the time scale for these three graphs still shows 5 seconds, but offset to when the instrument actually enters the track.

We expect to see the waveform for the instrument, much like you would see in a music player. It is important to note that there are not actually two vectors being plotted, even if the figure may seem like it (because of the orange and blue colors). Rather, this is a feature of MATLAB's "audioread" function. What MATLAB is in fact displaying through the blue and orange distinction are the left and right hand channels. We exported the .WAV files in stereo, which means that there are left and right hand channels (what would play in the left and right ear on headphones and left and right speakers otherwise).

The time plots for the seven instruments are as follows:

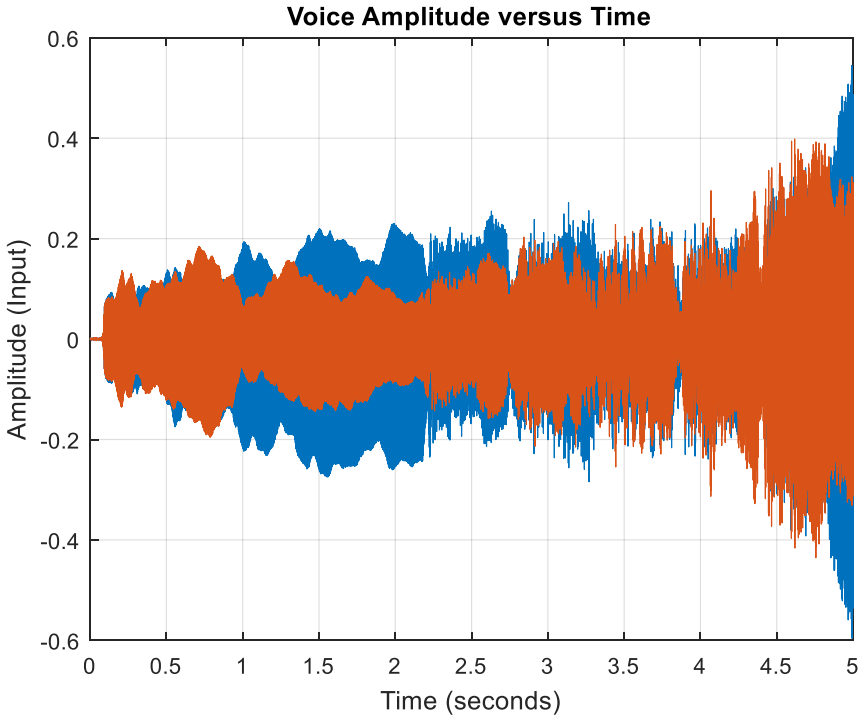

FIGURE 2: This figure shows the amplitude versus time graph for the female vocal instrument that was used.

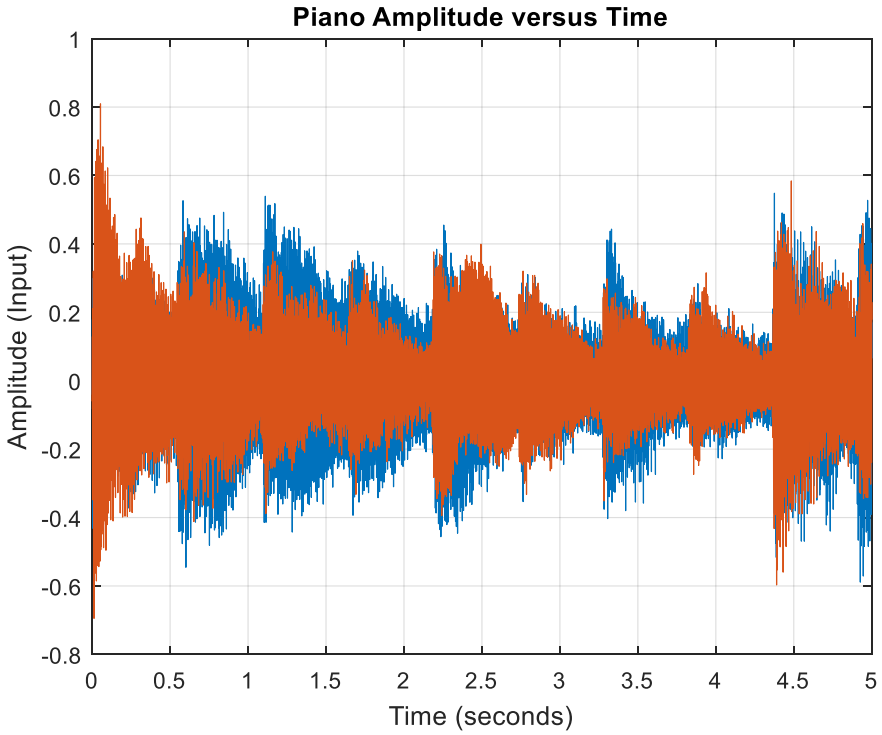

FIGURE 3: This figure shows the amplitude versus time for the piano instrument that was used within the track.

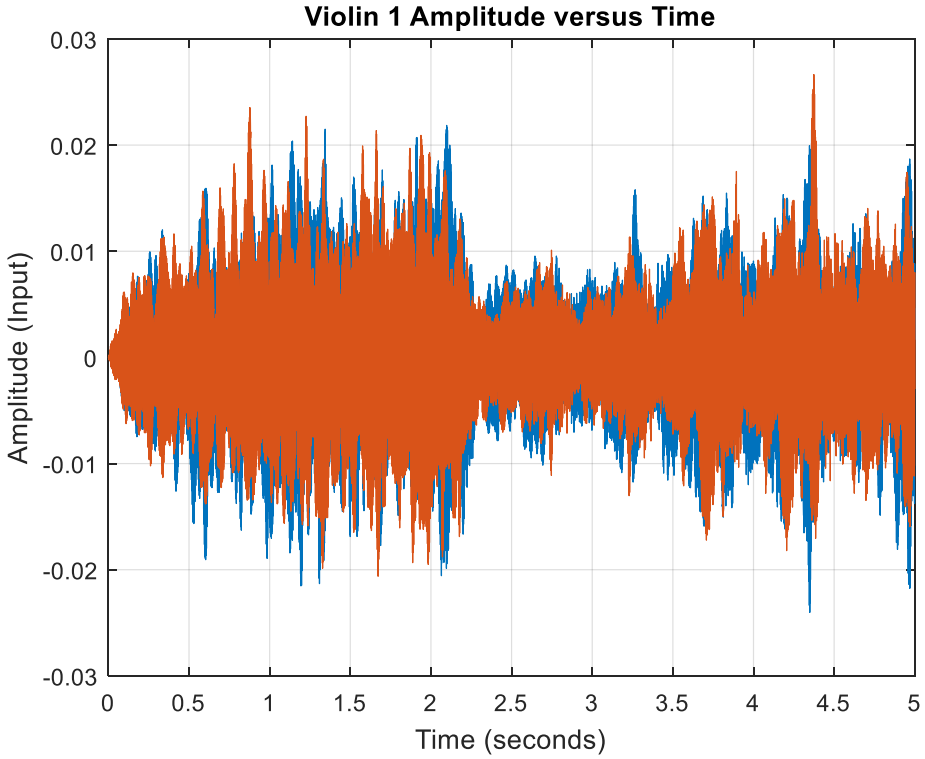

FIGURE 4: This figure shows the amplitude versus time for the violin 1 instrument that was used within the track.

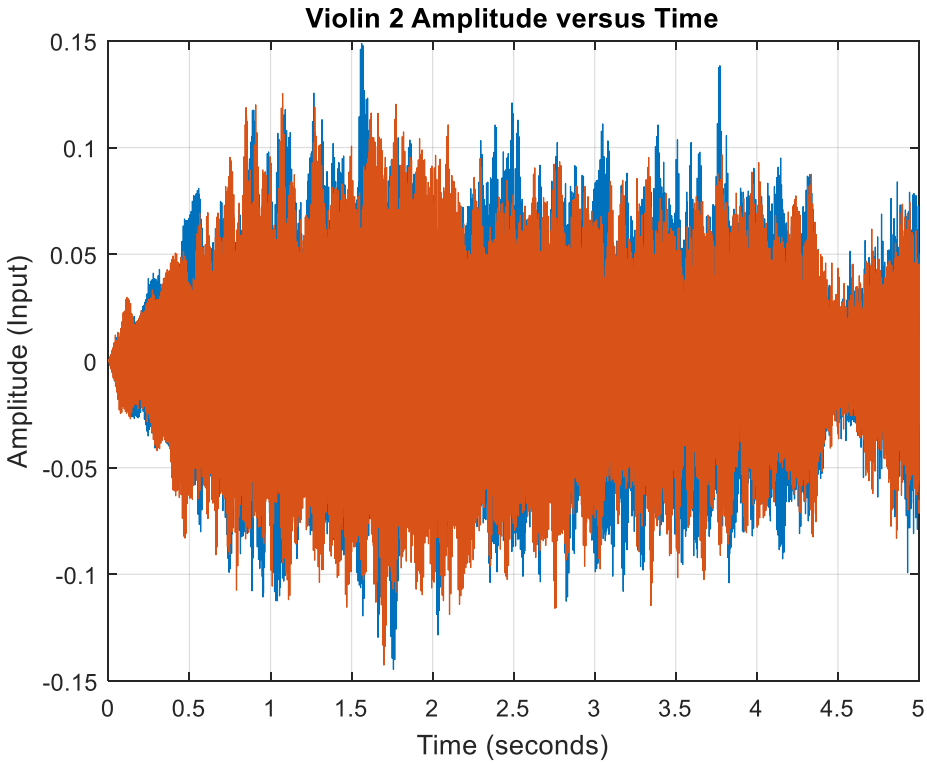

FIGURE 5: This figure shows the amplitude versus time for the violin 2 instrument that was used within the track.

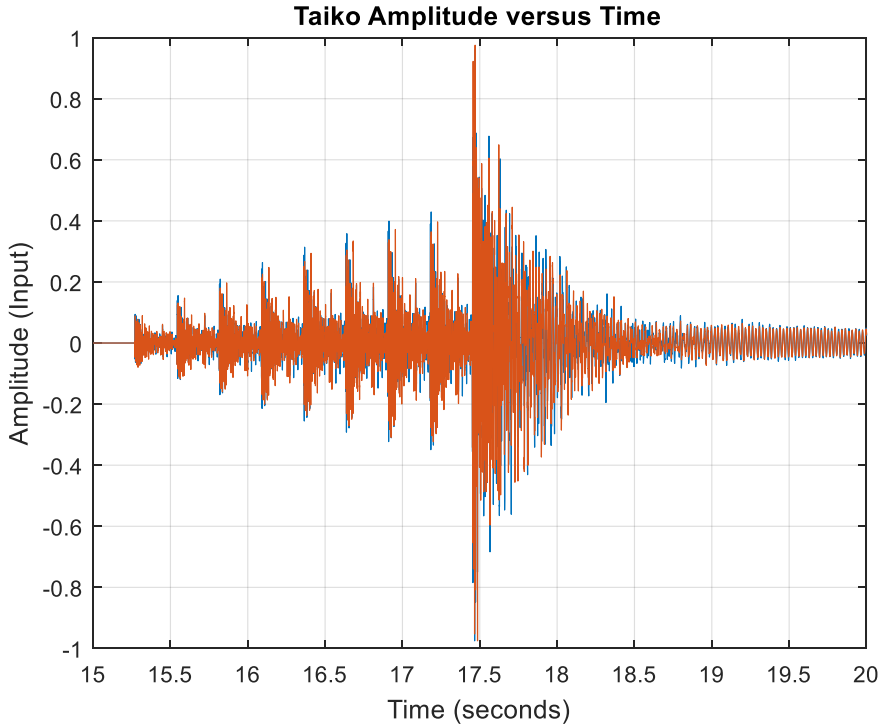

FIGURE 6: This figure shows the amplitude versus time for the taiko instrument that was used within the track.

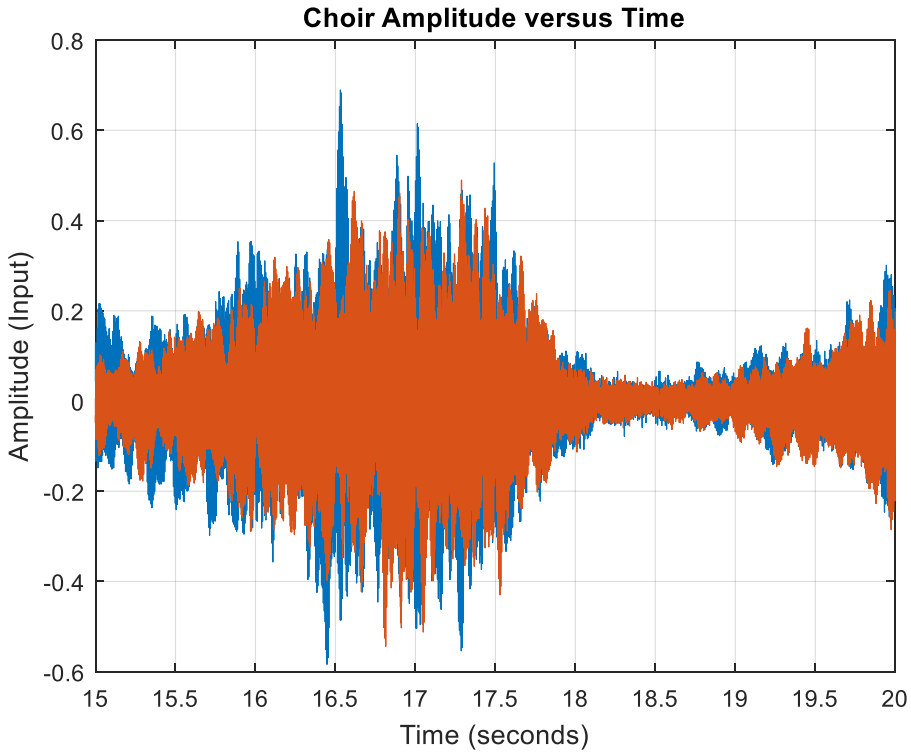

FIGURE 7: This figure shows the amplitude versus time for the choir instrument that was used within the track.

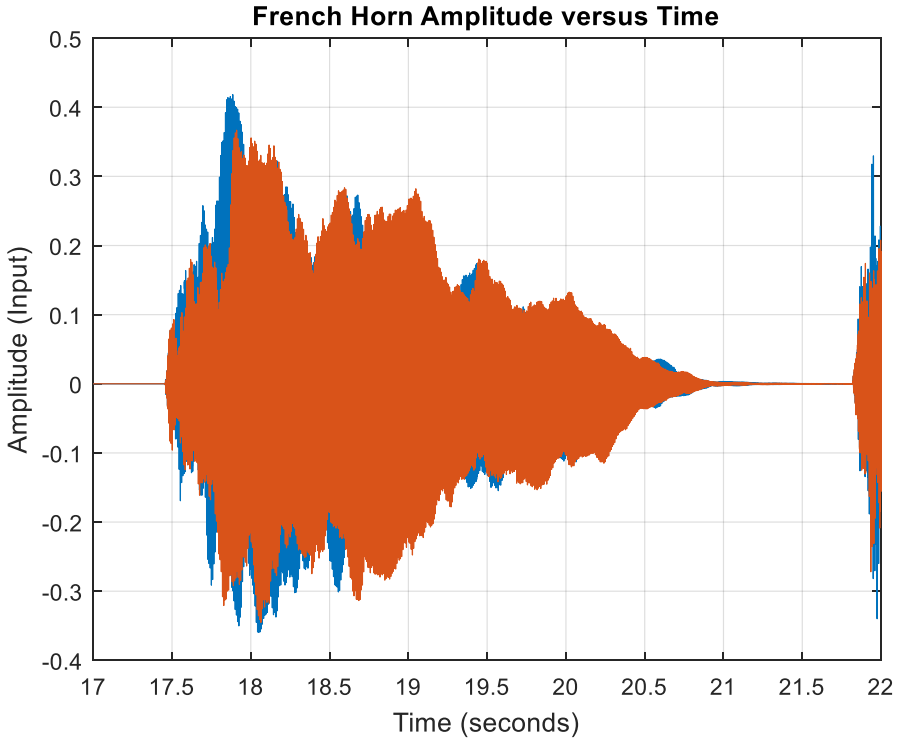

FIGURE 8: This figure shows the amplitude versus time for the French horn instrument that was used within the track.

We note again that the blue and orange don't mean that there were two vectors being graphed, but rather that the "audioread" function in MATLAB was graphing the left and right channels of the .WAV file since the .WAV file was stereo.

Now that we have the input and the time properly sorted out, we can actually take the Fourier Transform using "fft" and graph out the spectrum magnitude versus frequency.

FOURIER TRANSFORM PLOTS

After plotting the input amplitude versus the time, a Fourier Transform can now be taken of all seven instruments to see where the magnitude is greatest depending on the frequency. We can then use this information in order to create our Butterworth filter.

We took the Fourier Transform, noting to cut off the graph after the Nyquist frequency, which was 22,050 samples/second, or half of the 44,100 samples/second sampling frequency that we had for the .WAV file. The Fourier Transform graph for the seven instruments are as follows:

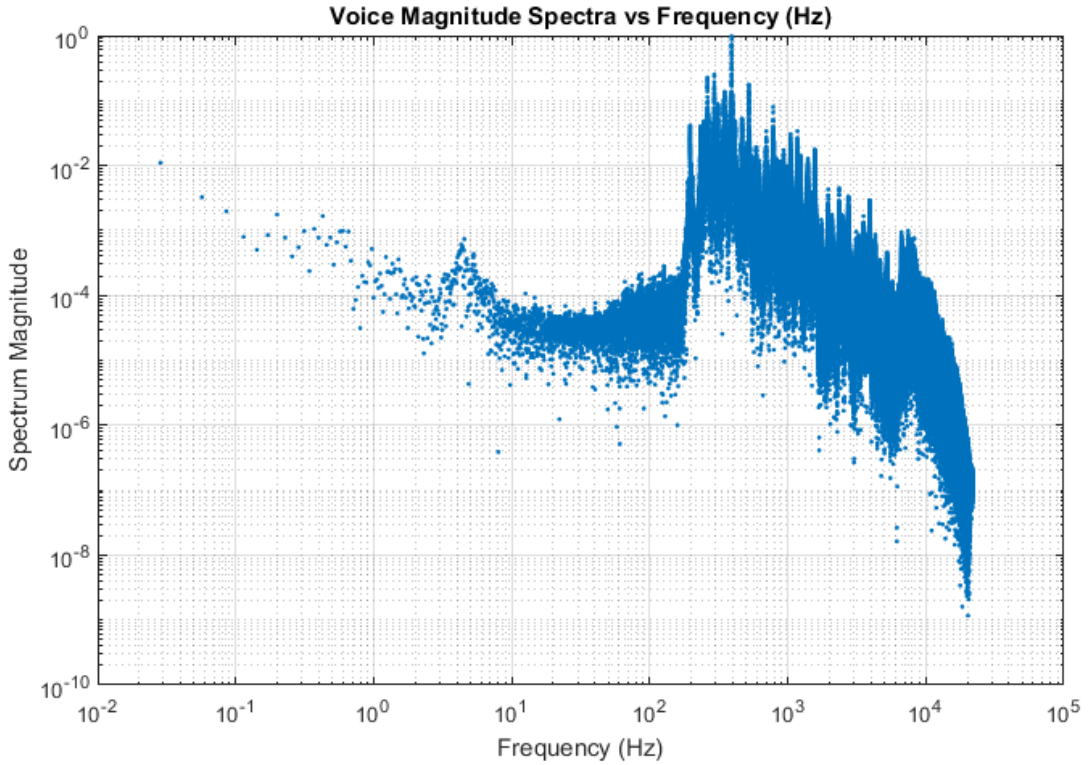

FIGURE 9: This figure shows the Fourier Transform magnitude spectrum for the female vocal instrument that was used. Note that the amplitude spikes around 400 Hz to 10,000 Hz.

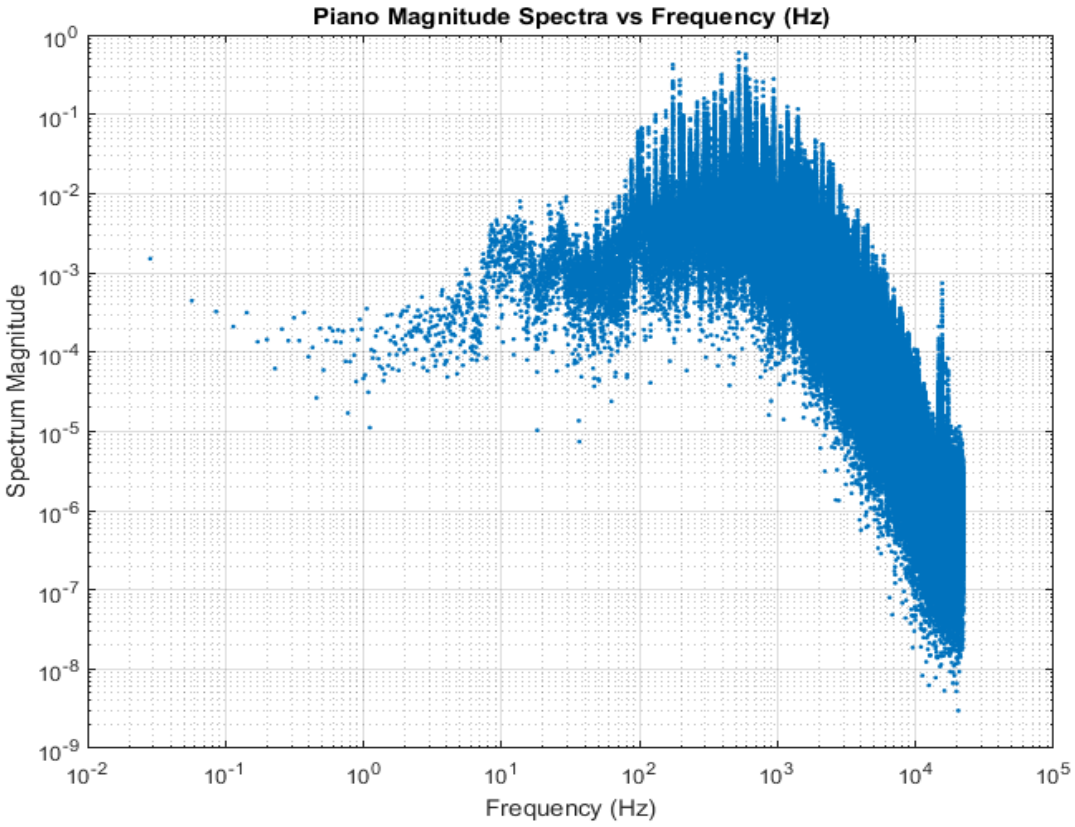

FIGURE 10: This figure shows the Fourier Transform magnitude spectrum for the piano instrument that was used. Notice that the piano has a broad range (typical of an instrument with 88 keys), but that the piano has an amplitude spike mostly around 100 Hz to 10,000 Hz.

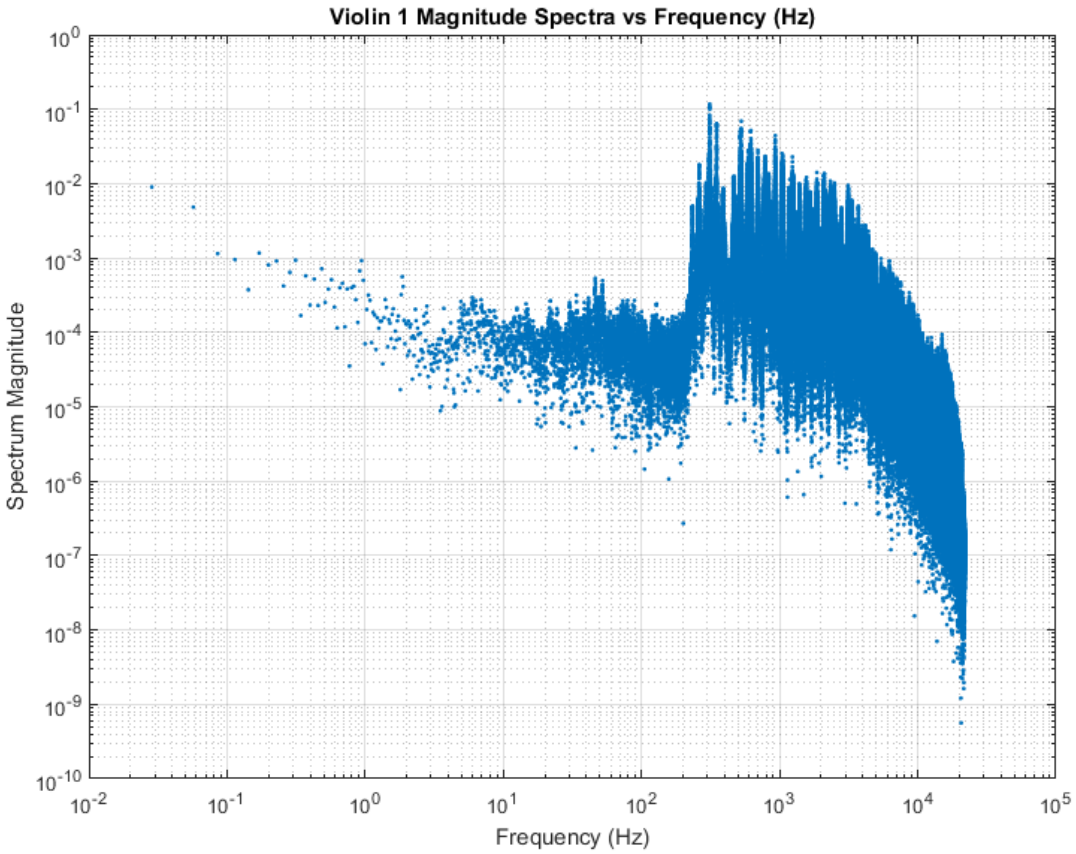

FIGURE 11: This graph shows the Fourier Transform for the Violin 1 instrument that was used within the song. Note that the magnitude spikes around 400 Hz and dies down around 4000 Hz.

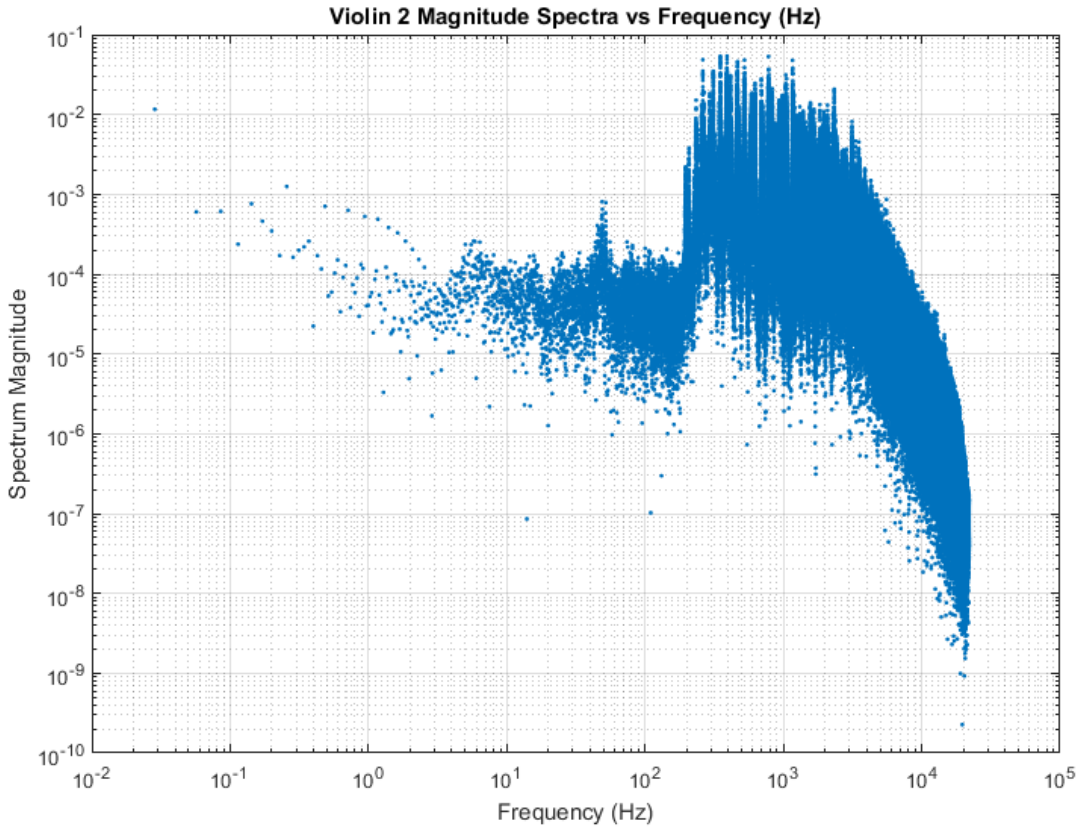

FIGURE 12: This graph shows the Fourier Transform for the Violin 2 instrument used within the song. Note that even though this was the same exact instrument as Violin 1 (both violins), the magnitude spikes around 200 Hz to 4000 Hz.

As stated in the figure, the magnitude for Violin 2 spikes differently than Violin 1, even though they are the same instrument. This is because Violin 2 mostly plays on the lower frequencies (lower strings on an actual violin), whereas Violin 1 plays on the higher frequencies (higher strings on an actual violin). This is just how classical music is arranged, with the harmony parts mostly falling to Violin 2. As such, since the Violin 2 instrument is playing tones at lower frequencies, the Fourier Transform magnitude spectrum will naturally shift to reflect that, and as such the magnitude spikes around 200 Hz to 4000 Hz, versus the 400 Hz to 4000 Hz of the Violin 1 instrument.

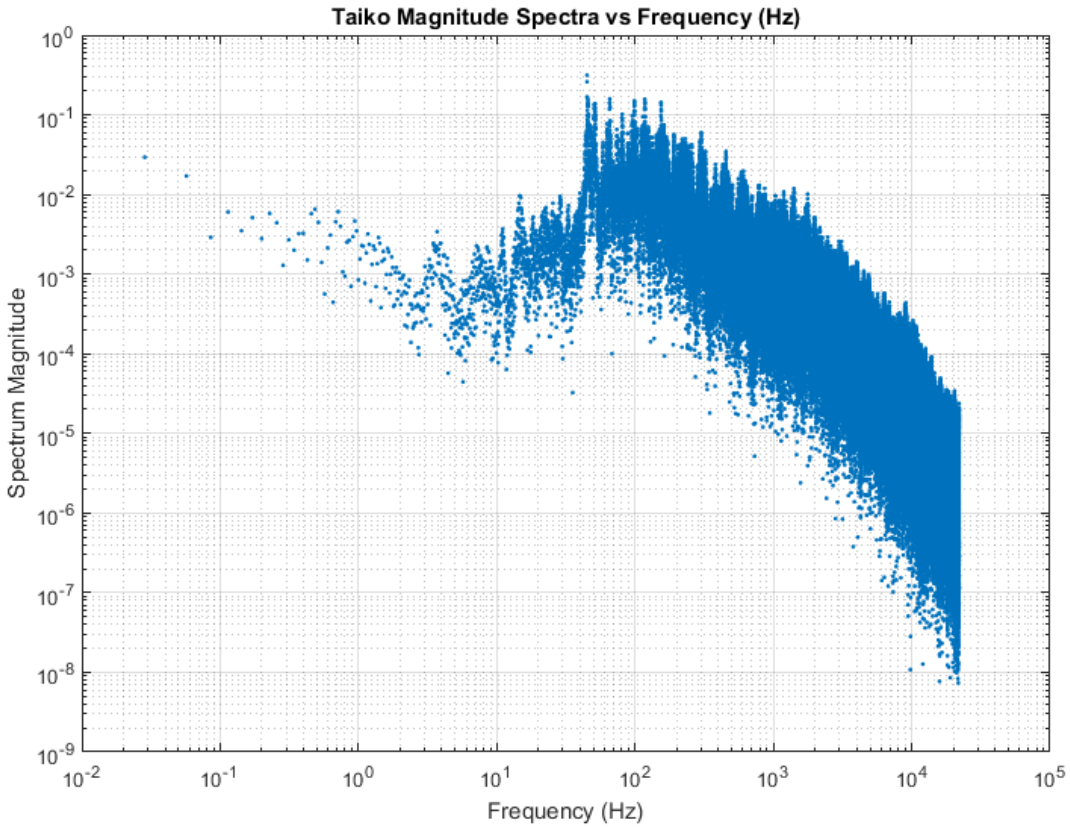

FIGURE 13: This figure shows the Fourier Transform magnitude spectrum for the Taiko instrument that was used. Notice that the taiko has a lower frequency at which the magnitude spikes. This is because it is a percussion instrument. The Taiko has ranges from around 40 Hz to 2000 Hz, since it is a percussion instrument on the low-end of the percussion register and range.

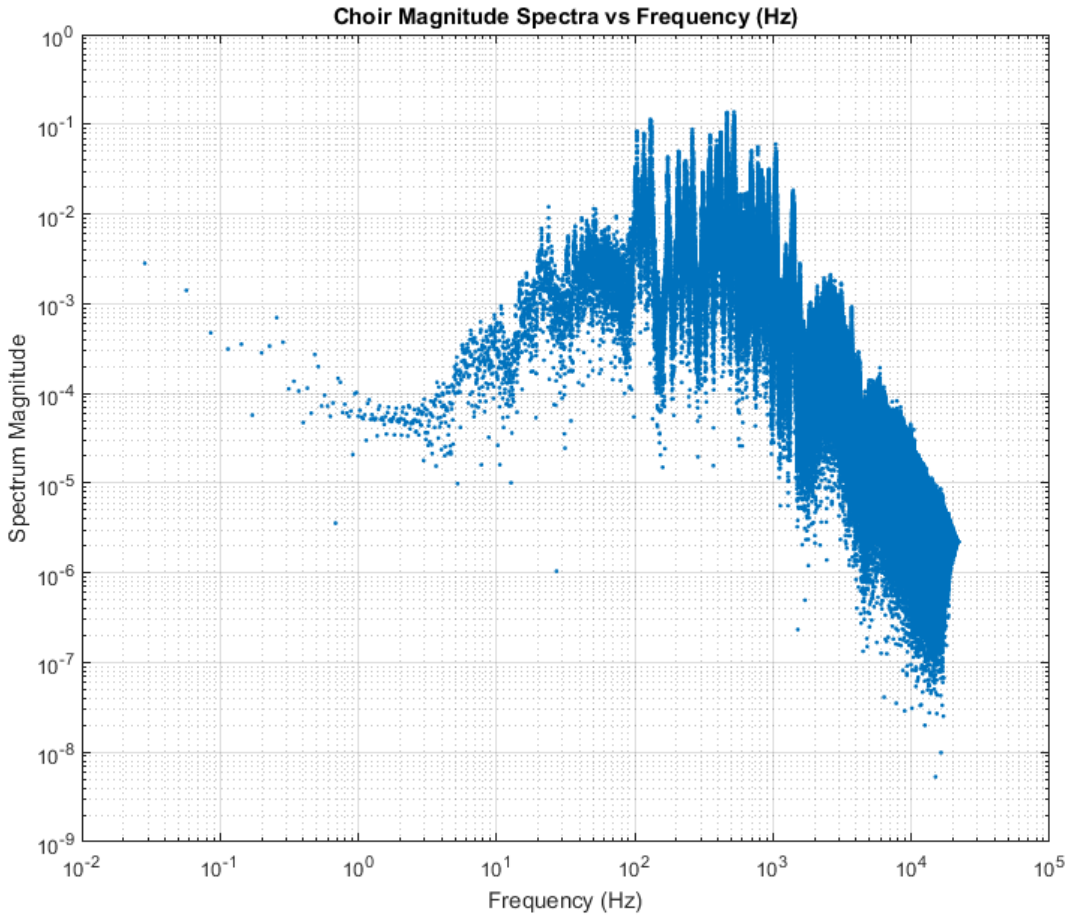

FIGURE 14: This figure shows the Fourier Transform magnitude spectrum for the choir instrument that was used within this song. Notice that the magnitude spikes around 100 Hz to 1000 Hz. Note that there is a broad range of magnitudes because I used both male and female voices for this choir at the edges of both of the human vocal ranges. This would cause a large spectrum of frequencies that the instrument could play.

An interesting thought in this case would be that if I just used male voices and separated, there would not be as many frequency spikes in the higher range because the female voices would be absent.

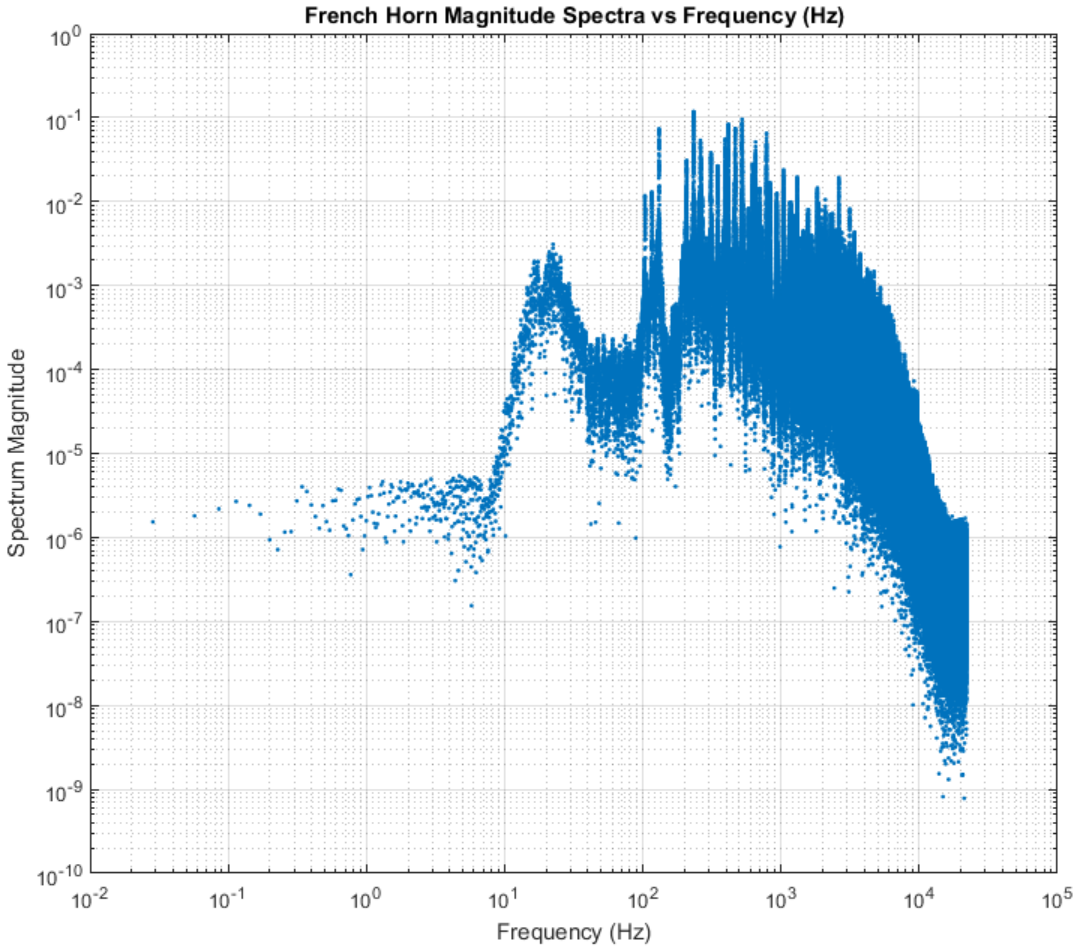

FIGURE 15: This figure shows the French Horn Fourier Transform graph for the instrument used within this song. Note that the French Horn is within the tenor range of an instrument (so middle of the orchestra).

This tenor range for the French Horn is reflected in the 100 to 1000 Hz magnitude spikes that it has. There is also an interesting spike around 20 Hz. We note that there is this spike at 20 Hz because I actually created this spike – before the instrument was exported as a .WAV file, I did an experiment where I used a pre-built equalizer within the composition software to make it so that the lower frequency of the instrument would be a lot louder. This can be seen in the following figure:

FIGURE 16: This figure shows the equalizer that was put on the French Horn before it was exported as a .WAV file. Note that the Fourier Transform graph follows the general shape of this graph.

This accounts for why there was a spike in the low frequencies – because the instrument was already preset to do so.

FILTERING

We note that it was a difficult task to filter out exactly which frequencies that we needed from each instrument because most of the instruments utilized very large ranges of frequencies. As such, we did not want to cut off part of the instrument's range. This necessitated the use of many band-pass filters. We will discuss the filtering ideas for each of these instruments in detail:

Voice

The voice track was filtered using a fourth order band-pass filter from 80 Hz to 10,000 Hz. The reason the voice track went down to 80 Hz was because there was a "reverb" effect applied on the voice that registered in the low frequencies. If the band-pass filter had a higher minimum frequency, then the reverb effect would have been lost or cut off. This was the range of the voice that was used.

Piano

The piano track was filtered using a band-pass filter from 100 Hz to 10,000 Hz. We again note that there is a wide range in frequencies because the piano has 88-keys that span from 27 Hz to 4000 Hz. As such, we cut off some of the lower notes on the piano that were not used, and kept some of the higher frequencies because the piano, in this case, was used as a mid to high range instrument.

Violins

Violins were filtered using a band-pass filter from 200 Hz to 10,000 Hz. The violin has a frequency range from 200 Hz to 660 Hz, and we kept the higher frequencies because the violins were used as a higher instrument. We note that an equivalent would have just been to use a high-pass filter with a cutoff at 200 Hz. We also used a low-pass filter for the second violins because we wanted to preserve the lower frequency sound.

Taiko

The taiko was filtered using a band-pass filter from 40 to 10,000 Hz. The main reason that the taiko was filtered was because there was a strange, empty, thumping sound that accompanied the taiko on the low frequency. This was filtered out using a band-pass filter. In addition, we kept some of the higher frequencies because there were taikos utilized that played in the higher frequencies that would have echos. We note that this would have been equivalent to using a high-pass filter with a cutoff frequency at 40 Hz. We have graphed the taiko graph below as follows:

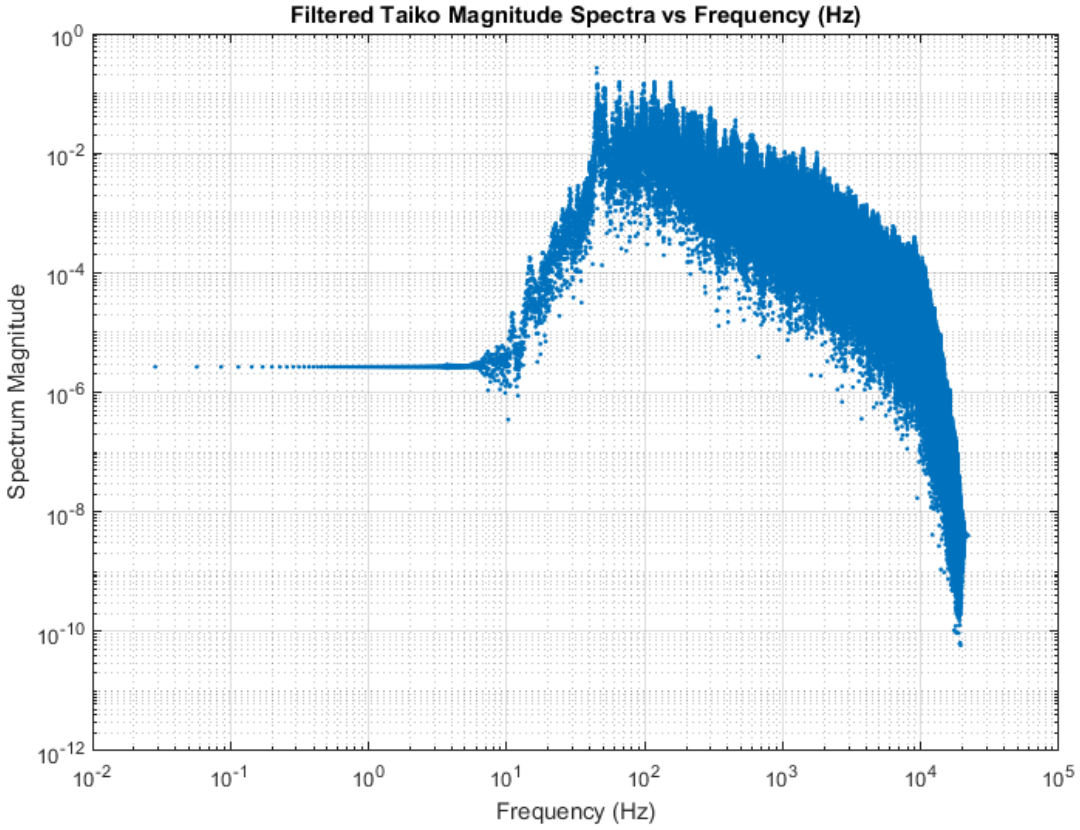

FIGURE 17: This graph shows the filtered Taiko instrument magnitude spectra versus the frequency.

Choir:

The choir was filtered using a low-pass filter with a cut-off at 2000 Hz. This was meant to emphasize the bass of the choir, because I felt that there wasn't enough bass within the piece and thus converted the choir into a male choir to use as a bass instrument.

This also preserved the vocal integrity of the choir and the range at which the choir could sing.

French Horn

The French horn is in the middle range of frequencies in terms of instruments, but I wanted to utilize it again as a bass instrument so I used a low-pass filter with a cut-off frequency at 3000 Hz. This is a higher cut-off frequency than the choir because the French horn actually plays at higher frequencies than the male choir sings, since male choirs are more bass-heavy and French horns are in the tenor range, which is above the bass range (the order goes bass, tenor, alto, soprano).

Overall Piece

In terms of the overall mixing of the piece, I changed a few of the volumes of the instruments because I wanted to bring them out (especially the voice). I also changed the playback on MATLAB to play at 85% volume, since 100% would clip on headphones.

CONCLUSION

Overall, this experiment was concerned with taking .WAV output files from multiple musical instruments, and then graphing various quantities related to them. First, the input amplitude was graphed versus the time to observe the waveform. Then, a Fourier Transform of each individual instrument was taken to look at the magnitude spectrum of the instrument. Based on the magnitude spectrum, a Butterworth filter was applied to the instrument in order to make the sound quality of the piece better. Finally, all the instruments were brought together to form a coherent piece that was edited based on the principles of Fourier Transforms. Overall, we can say that this experiment was a success because we did indeed manage to take the Fourier Transform of each individual instrument sound, and applied filters accordingly. MATLAB was able to play back the song, which can be listened to using the code in the Appendix, or the code and the files attached separately.

APPENDIX

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
% MAE 107 LAB 7
% Samuel Chen
% 704-453-763
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%%%% NOTE:
% I made my own song -- has seven instruments which are: voice, violin1,
% violin2, choir, taiko, horn, and piano
% inspired by sinusoidal waves

% opening ritual
close all; clc; clear all;

% import audio files
voice_mtx = audioread('voice.wav');
piano_mtx = audioread('piano.wav');
violin1_mtx = audioread('violin1.wav');
violin2_mtx = audioread('violin2.wav');
taiko_mtx = audioread('taiko.wav');
choir_mtx = audioread('choir.wav');
horn_mtx = audioread('horn.wav');

% define variables we will use later
sample_rate = 44100;  % typical WAV sampling
sample_time = 1/sample_rate;

% get the length of all of the matrices
voice_length = length(voice_mtx);
piano_length = length(piano_mtx);
violin1_length = length(violin1_mtx);
violin2_length = length(violin2_mtx);
taiko_length = length(taiko_mtx);
choir_length = length(choir_mtx);
horn_length = length(horn_mtx);
% we need the lengths to convert to time and also later to truncate
% to the nyquist frequency

% Get the time for all of the samples
voice_time = sample_time.*(1:voice_length);
piano_time = sample_time.*(1:piano_length);
violin1_time = sample_time.*(1:violin1_length);
violin2_time = sample_time.*(1:violin2_length);
taiko_time = sample_time.*(1:taiko_length);
choir_time = sample_time.*(1:choir_length);
horn_time = sample_time.*(1:horn_length);


%%% INPUT PLOTS %%%

figure(1)
```

```
plot(voice_time, voice_mtx);
xlim([0 5]);
grid on;
xlabel('Time (seconds)'); ylabel('Amplitude (Input)');
title('Voice Amplitude versus Time');

figure(2)
plot(piano_time, piano_mtx);
xlim([0 5]);
grid on;
xlabel('Time (seconds)'); ylabel('Amplitude (Input)');
title('Piano Amplitude versus Time');

figure(3)
plot(violin1_time, violin1_mtx);
xlim([0 5]);
grid on;
xlabel('Time (seconds)'); ylabel('Amplitude (Input)');
title('Violin 1 Amplitude versus Time');

figure(4)
plot(violin2_time, violin2_mtx);
xlim([0 5]);
grid on;
xlabel('Time (seconds)'); ylabel('Amplitude (Input)');
title('Violin 2 Amplitude versus Time');

figure(5)
plot(taiko_time, taiko_mtx);
xlim([15 20]);   % note here that in the song, the taiko doesn't start
                 % until 15 seconds in so the first 15 seconds starts here
grid on;
xlabel('Time (seconds)'); ylabel('Amplitude (Input)');
title('Taiko Amplitude versus Time');

figure(6)
plot(choir_time, choir_mtx);
xlim([15 20]);  % same thing with choir, doesn't start until 15 seconds
grid on;
xlabel('Time (seconds)'); ylabel('Amplitude (Input)');
title('Choir Amplitude versus Time');

figure(7)
plot(horn_time, horn_mtx);
xlim([17 22]);  % horn starts later at 17 seconds (about half the song);
grid on;
xlabel('Time (seconds)'); ylabel('Amplitude (Input)');
title('French Horn Amplitude versus Time');


%%%% SECOND PART - FOURIER TRANSFORM

nyquist = 22050;  % needed later

% take the step size
```

```
voice_step = sample_rate./voice_length;
piano_step = sample_rate./piano_length;
violin1_step = sample_rate./violin1_length;
violin2_step = sample_rate./violin2_length;
taiko_step = sample_rate./taiko_length;
choir_step = sample_rate./choir_length;
horn_step = sample_rate./horn_length;
% these are the step sizes after convering to frequency
% note through dimensional analysis (samples/sec) divided by (samples)
% which equals 1/sec which equals Hz

% take the fourier transform
voice_fft = (fft(voice_mtx)).*sample_time;
piano_fft = (fft(piano_mtx)).*sample_time;
violin1_fft = (fft(violin1_mtx)).*sample_time;
violin2_fft = (fft(violin2_mtx)).*sample_time;
taiko_fft = (fft(taiko_mtx)).*sample_time;
choir_fft = (fft(choir_mtx)).*sample_time;
horn_fft = (fft(horn_mtx)).*sample_time;
% fourier transform times the sample time


% truncate to the nyquist frequency
voice_length_new = 0.5*voice_length;
piano_length_new = 0.5*piano_length;
violin1_length_new = 0.5*violin1_length;
violin2_length_new = 0.5*violin2_length;
taiko_length_new = 0.5*taiko_length;
choir_length_new = 0.5*choir_length;
horn_length_new = 0.5*horn_length;
% corresponds to 22050
% this means rate of 44100 in the half, so through dimensional analysis
% and logic we can just take half of our samples and that should truncate
% right or very close to the Nyquist frequency


% truncate the fourier transform to the nyquist frequency
voice_fft = voice_fft(1:voice_length_new);
piano_fft = piano_fft(1:piano_length_new);
violin1_fft = violin1_fft(1:floor(violin1_length_new));  % floor for
fractions
violin2_fft = violin2_fft(1:floor(violin2_length_new));
taiko_fft = taiko_fft(1:floor(taiko_length_new));
choir_fft = choir_fft(1:choir_length_new);
horn_fft = horn_fft(1:horn_length_new);

% truncate omega to the nyquist frequency and establish it
% note that the matrix dimensions have to agree
voice_om = voice_step.*(1:voice_length_new);
piano_om = piano_step.*(1:piano_length_new);
violin1_om = violin1_step.*(1:floor(violin1_length_new));  % add floor or
else
                                                          % we have
                                                          % fractions
violin2_om = violin2_step.*(1:floor(violin2_length_new));
taiko_om = taiko_step.*(1:floor(taiko_length_new));
```

```
choir_om = choir_step.*(1:choir_length_new);
horn_om = horn_step.*(1:horn_length_new);


%%% FOURIER TRANSFORM PLOTS
figure(8)
loglog(voice_om, abs(voice_fft), '.');
grid on;
xlabel('Frequency (Hz)'); ylabel ('Spectrum Magnitude');
title('Voice Magnitude Spectra vs Frequency (Hz)');

figure(9)
loglog(piano_om, abs(piano_fft), '.');
grid on;
xlabel('Frequency (Hz)'); ylabel ('Spectrum Magnitude');
title('Piano Magnitude Spectra vs Frequency (Hz)');

figure(10)
loglog(violin1_om, abs(violin1_fft), '.');
grid on;
xlabel('Frequency (Hz)'); ylabel ('Spectrum Magnitude');
title('Violin 1 Magnitude Spectra vs Frequency (Hz)');

figure(11)
loglog(violin2_om, abs(violin2_fft), '.');
grid on;
xlabel('Frequency (Hz)'); ylabel ('Spectrum Magnitude');
title('Violin 2 Magnitude Spectra vs Frequency (Hz)');

figure(12)
loglog(taiko_om, abs(taiko_fft), '.');
grid on;
xlabel('Frequency (Hz)'); ylabel ('Spectrum Magnitude');
title('Taiko Magnitude Spectra vs Frequency (Hz)');

figure(13)
loglog(choir_om, abs(choir_fft), '.');
grid on;
xlabel('Frequency (Hz)'); ylabel ('Spectrum Magnitude');
title('Choir Magnitude Spectra vs Frequency (Hz)');

figure(14)
loglog(horn_om, abs(horn_fft), '.');
grid on;
xlabel('Frequency (Hz)'); ylabel ('Spectrum Magnitude');
title('French Horn Magnitude Spectra vs Frequency (Hz)');



%%% FILTERS
% start the filters based on the spectrum magnitude
[voice_num, voice_den] = butter(4, [80 10000]./nyquist, 'bandpass');
voice_filtered = filter(voice_num, voice_den, voice_mtx);
```

```
[piano_num, piano_den] = butter(4, [100 10000]./nyquist, 'bandpass');
piano_filtered = filter(piano_num, piano_den, piano_mtx);

[violin1_num, violin1_den] = butter(4, [200 10000]./nyquist, 'bandpass');
violin1_filtered = filter(violin1_num, violin1_den, violin1_mtx);

[violin2_num, violin2_den] = butter(4, 6000/nyquist, 'low');
violin2_filtered = filter(violin2_num, violin2_den, violin2_mtx);

[taiko_num, taiko_den] = butter(4, [40 10000]./nyquist, 'bandpass');
taiko_filtered = filter(taiko_num, taiko_den, taiko_mtx);

[choir_num, choir_den] = butter(4, 2000/nyquist, 'low');
choir_filtered = filter(choir_num, choir_den, choir_mtx);

[horn_num, horn_den] = butter(4, 3000/nyquist, 'low');
horn_filtered = filter(horn_num, horn_den, horn_mtx);

% lengths were not the same for some reason
violin1_filtered = violin1_filtered(1:1546006,:);
violin2_filtered = violin2_filtered(1:1546006,:);
taiko_filtered = taiko_filtered(1:1546006,:);
taiko_mtx = taiko_mtx(1:1546006,:);

% I like the original sound for the horn so I didn't use the
% filtered version

my_comp = (1.1)*voice_filtered + (0.9)*piano_filtered + violin1_filtered
+ ...
    1.05*violin2_filtered + (0.9)*taiko_filtered + choir_filtered + horn_mtx;


sound(0.85*my_comp, 44100);

% filtered taiko graph for show
taiko_filtered_ftt = (fft(taiko_filtered))*sample_time;
taiko_filtered_ftt = taiko_filtered_ftt(1:length(taiko_om));

figure(15)
loglog(taiko_om, abs(taiko_filtered_ftt), '.');
grid on;
xlabel('Frequency (Hz)'); ylabel ('Spectrum Magnitude');
title('Filtered Taiko Magnitude Spectra vs Frequency (Hz)');
```