Mathematical Methods in Physics

MATH 433

Analysis of Signals Generated By Acoustic and Electric Guitars

Kenneth Sharman

ID Number: 00300185

Professor:

Dr. Larry Bates

December 1, 2018

Contents

1 Introduction

In this paper we will analyze the sound generated by both acoustic and electric guitars.
First, the mathematical framework, for which this analysis is based upon, is presented in considerable detail.

The recorded signals of both an acoustic and an electric guitar will be analyzed using MATLAB, by investigating both the time and frequency domains independently. It is important to note that there are numerous pieces of free software that are specifically designed for such an analysis, however building the model from the ground up will prove to both test the mathematical theory and provide an introduction to data analysis in MATLAB.

The results of this analysis will be used to construct synthesized versions of the recordings and the effectiveness of this method will be discussed.

An alternative synthesis method, the Karplus-Strong algorithm, will be studied independently of the findings of the time and frequency domain analysis.
The paper will conclude with a discussion of the variables and factors omitted from the constructed models which result in less than realistic synthesized signals.

2 Pitch and Frequency

We begin by providing an example which illustrates the relationship between pitch and frequency-concepts which form the foundation of musical analysis.

2.1 The Relationship between Pitch and Frequency

A tuning fork is a tool that can be used to tune music instruments. Before electric tuners were popularized, this was the standard for making sure an instrument was in tune. The ideal tuning fork produces a pure tone having a single pitch which is associated with a single frequency. The recorded signal can be represented in both the time and frequency domains.

For example, the recorded signal (Track 1) of an E_3 tuning fork is shown in Figure 1. A tuning fork with a pitch of E_3 on the well-tempered scale ideally has frequency $\nu = 329.6$ Hz. The FFT founction in MATLAB (see section 2.5) was used to determine the frequency of this fork. Note that the recorded frequency is $\nu = 329.3$ Hz, so it is slightly out of tune.

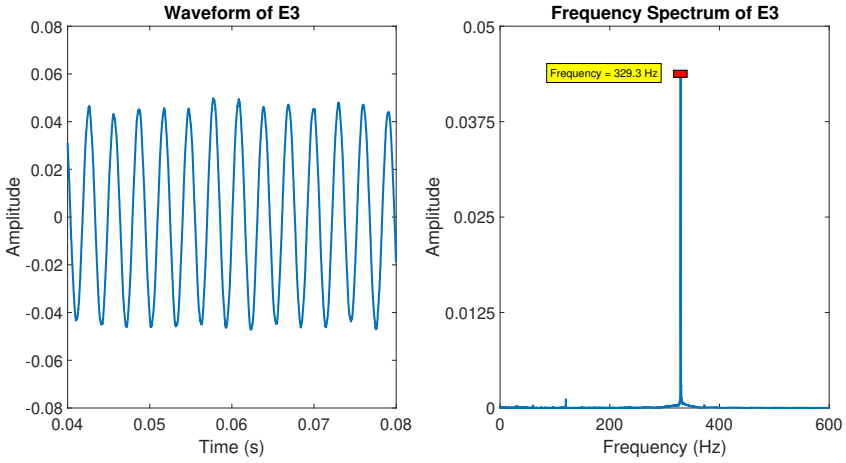

Figure 1: Time-Frequency Analysis of an E_3 Tuning Fork

The sound generated by musical instruments are superpositions of several pure tones (sinusoidal waves) which evolve over time. Thus, the signal is composed of several frequencies. The base frequency of a musical note is known as the *fundamental*. The other frequencies are all larger than the fundamental and are called *overtones*. If an overtone is an integer multiple of the fundamental then it is known as a *harmonic*. We will see that in the case of signals generated by guitars (or any other musical instrument for that matter) these overtones play a vital role in characterizing the sound produced.

2.2 Fourier Series

Musical notes can be mathematically described using the *Fourier Series*. The signal $f(t)$ of a note or a chord, defined on the interval $[0, T_0]$, has Fourier series

$$\frac{a_0}{2} + \sum_{n=1}^{\infty} \left[a_n \cos\Big(\frac{2\pi nt}{T_0}\Big) + b_n \sin\Big(\frac{2\pi nt}{T_0}\Big) \right] \tag{1}$$

with *Fourier coefficients* a_n, b_n given by

$$a_n = \frac{2}{T_0} \int_0^{T_0} f(t) \cos\Big(\frac{2\pi nt}{T_0}\Big) dt \qquad n = 0, 1, 2, 3, \ldots$$
$$b_n = \frac{2}{T_0} \int_0^{T_0} f(t) \sin\Big(\frac{2\pi nt}{T_0}\Big) dt \qquad n = 1, 2, 3, \ldots$$

Each term corresponding to a value of n has a period of T_0/n and frequency of n/T_0. Note that these frequencies are the harmonics associated with the fundamental frequency $\nu_0 = 1/T_0$. Equation 1 can be written in terms of complex exponentials using Euler's formulas,

$$c_0 + \sum_{n=1}^{\infty} \left[c_n e^{i2\pi nt/T_0} + c_{-n} e^{-i2\pi nt/T_0} \right]$$

where $c_0 = a_0/2$ and c_n and c_{-n} are the *complex Fourier coefficients*

$$c_n = \frac{1}{T_0} \int_0^{T_0} f(t) e^{-i2\pi nt/T_0} dt \qquad n = 0, \pm 1, \pm 2, \pm 3, \ldots \tag{2}$$

If $f(t)$ is the signal generated by a musical instrument, then $f(t)$ is real-valued, and the coefficients c_{-n} are the complex conjugates of c_n.

2.3 Parseval's Equality

An important result of Fourier Series is known as *Parseval's Theorem*, which states that the integral of the square of a function is equal to the sum (or integral) of the square of its transform [1].

$$\frac{1}{T_0} \int_0^{T_0} |f(t)|^2 dt = |c_0|^2 + \sum_{n=1}^{\infty} \left[|c_n|^2 + |c_{-n}|^2 \right]$$

The physical interpretation of this expression can be seen by first defining the *energy* of a function over $[0, T_0]$ as $\int_0^{T_0} f(t)^2 dt$ [2]. The above expression then states that the energy of $f(t)$ (in this case the energy of the sound signal) is equal to T_0 multiplied by the sum of the Fourier coefficients.

There are two simplifications that can be made at this step. Firstly, the constant $|c_0|^2$ is inaudible since our hearing responds to fluctuations in air pressure [1]. Thus, in the context of music sound analysis it provides no information. Secondly, since ${c_n}^2 = |c_{-n}|^2$ Parseval's Equality can be expressed as

$$\frac{1}{T_0}\int_0^{T_0} |f(t)|^2 dt \simeq \sum_{n=1}^{\infty} 2\,|c_n|^2 \tag{3}$$

The *Fourier series spectrum* $\{2|c_n|^2\}$ represents the energies (up to a constant) of the frequencies that comprise the sound generated by a musical instrument. In this form, we can see that by determining the Fourier coefficients of a given signal, we can determine the constituent frequencies.

Equation 3 provides us with a tool to compare the signals generated by different types of guitar signals, however the Fourier coefficients c_n have yet to be determined.

2.4 Fourier Transform

The Fourier Transform provides an explanation for the frequency spectrum shown in Figure 1 and the discrete version which will prove to be the key to determining the Fourier coefficients of the music signal. The definition of the Fourier Transform of a function $f(t)$ with frequency ν is

$$\mathfrak{F}[f(t)](\nu) = \int_{-\infty}^{\infty} f(t)e^{-i2\pi\nu t}dt$$

In signal processing this definition is often expressed as a function of angular frequency ω however musical notes are usually defined by linear frequencies and this notation will be used. As an example; the Fourier Transform of a sine wave with frequency A is computed [3].

$$\begin{aligned}
\mathfrak{F}_t[\sin(2\pi At)](\nu) &= \int_{-\infty}^{\infty} e^{-2\pi i\nu t}\left[\frac{e^{i2\pi At} - e^{-i2\pi At}}{2i}\right]dt \\
&= \frac{1}{2}i\int_{-\infty}^{\infty}\Big[-e^{-i2\pi(\nu-A)t} + e^{-i2\pi(\nu+A)t}\Big]dt \\
&= \frac{1}{2}i[\delta(\nu+A) - \delta(\nu-A)]
\end{aligned}$$

This equation states that the Fourier Transform of a sine function of frequency A is an impulse at $f = \pm A$.

Referring back to the tuning fork example, Figure 1 shows this impulse as a spike at the frequency $\nu = 329.3$ Hz. MATLAB was used to transform the signal from the time domain to the frequency domain using the FFT function (more to come on FFT). Recall that since the sound signal is real-valued the FFT result is conjugate symmetric ($|c_n|^2 = |c_{-n}|^2$) and left left-hand side of the spectrum can be omitted without loss of information. Hence, we see a single spike in the spectrum, corresponding to the pure tone of the tuning fork.

2.5 FFT

The method used here for computing the Fourier series spectrum is known as the Fast Fourier Transform (FFT). The reason it is called "fast" is because it provides approximations to the Fourier Series coefficients, via an algorithm that is computationally inexpensive. These approximations are known as discrete Fourier Transforms (DFTs).

For a positive integer N, let $t_k = k\ T_0/N$ for $k = 0, 1, 2, \ldots, N-1$ and let $\Delta t = T_0/N$. The coefficient c_n expressed in Equation 2 can be approximated by

$$c_n \approx \frac{1}{T_0} \sum_{k=0}^{N-1} f(t_k) e^{-i2\pi n t_k/T_0} \Delta t = \frac{1}{N} \sum_{k=0}^{N-1} f(t_k) e^{-i2\pi nk/N} = F[n]$$

Thus, the Fourier Series spectrum $\{2|c_n|^2\}$ can be approximated by the set of DFTs, $\{2|F[n]|^2\}$ for $n = 1, 2, 3, \ldots$

The FFT can easily be computed for an input signal using MATLAB. Analyzing the Frequency domain of both electric and acoustic guitar signals will serve to be the basis of the analysis section of this report.

3 Signal Acquisition

We will compare the signal generated by a Fender American Telecaster electric guitar to that of a Martin & Co. flat top acoustic.

Both guitars were recording playing an A_3 note (with known frequency $\nu = 220$ Hz), using the Steinberg CI1 USB audio interface and Ableton Live 9 recording software. The Telecaster was directly connected to the interface, and the acoustic was recorded using Shure SM58 dynamic microphone. No manipulation of the audio recordings was performed with the exception of trimming the file to isolate the note. Both recordings were exported to WAV format using 44100 sample rate and 16 bit depth. These recordings are labeled Track 2 and 3. Finally, the WAV files were imported into MATLAB for analysis.

4 Time and Frequency Domain Analysis

4.1 Time Domain

The first step in this analysis is to plot the waveform of each signal. Plots are shown for both the full signal and a magnified view of the attack phase (region where the signal starts- the pick hits the string). The corresponding MATLAB code can be found in Section 9.1.

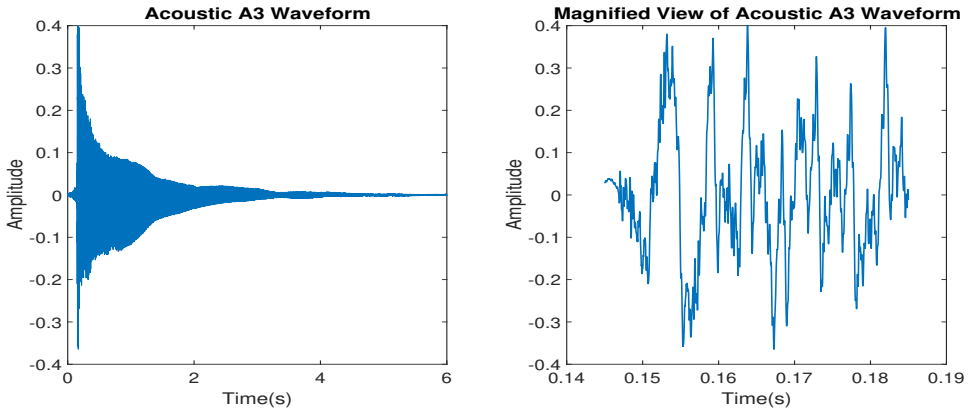

Figure 2: Waveform of A_3 played on Acoustic Guitar

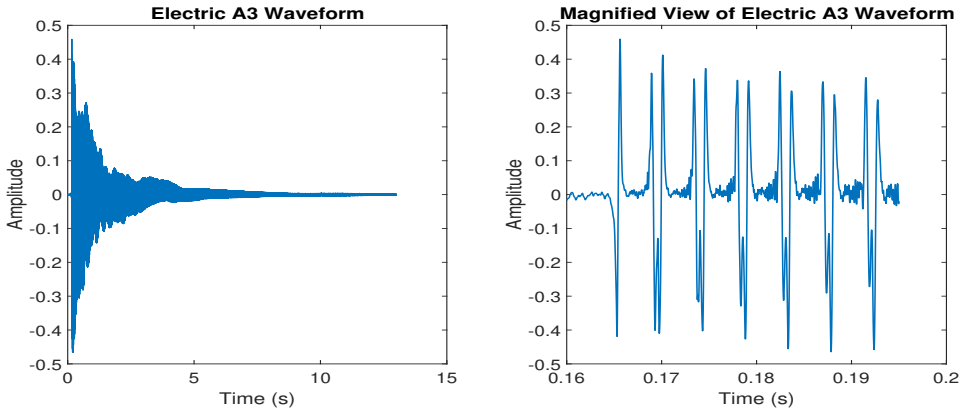

Figure 3: Waveform of A_3 played on Electric Guitar

There are several observations of note here. The first is that, unsurprisingly, the acoustic signal damps out much quicker than the electric signal. From experience of playing both guitars it is almost common knowledge that (when plugged into an amp) the sustain of an electric is much longer than an acoustic. This is due to the fact that the signal of an electric guitar is almost entirely generated by the vibrations of the strings, while the acoustic signal is dependent on the system of strings, soundboard, and acoustic cavity, which as a whole tends to damp out much faster [4].

From the magnified view of both waveforms, it is appears that the acoustic signal has a lot more going on. As mentioned, the acoustic signal is made up of more than just the vibrating strings, but the recording process must also be considered.

Since the electric was plugged directly into the recording interface, no background noise or reflected sound waves (from surfaces within the room) are present in this recording. Factoring these considerations into the recording of the acoustic, it becomes apparent as to why this signal appears to be somewhat distorted, or at least much more complicated than the electric.

4.2 ADSR Envelope

The first step of synthesizing signals to imitate these recordings can be made with information provided by these plots. Surely, if a signal is to be successfully generated that resembles a guitar, then the amplitude of the waveform should vary in proportion to the recording. This is what is referred to as the ADSR envelope, which describes the amplitude of the signal during the attack, decay, sustain, and release phases of the signal.

MATLAB was used to determine this envelope by dividing both signals into attack, decay/ sustain, and release phases, as differentiating between the decay and sustain regions was not straightforward. In theory, the envelope function in MATLAB could have determined the envelope for the whole signal with one function call, however it was found that the function required a different number of samples per interval, to successfully fit the envelope curve to each of the phases.

Typically, the envelope has an upper and a lower portion, however for simplicity only the positive amplitude was considered here. The envelopes for both signals are shown below and the corresponding MATLAB code can be found in Section 9.2.

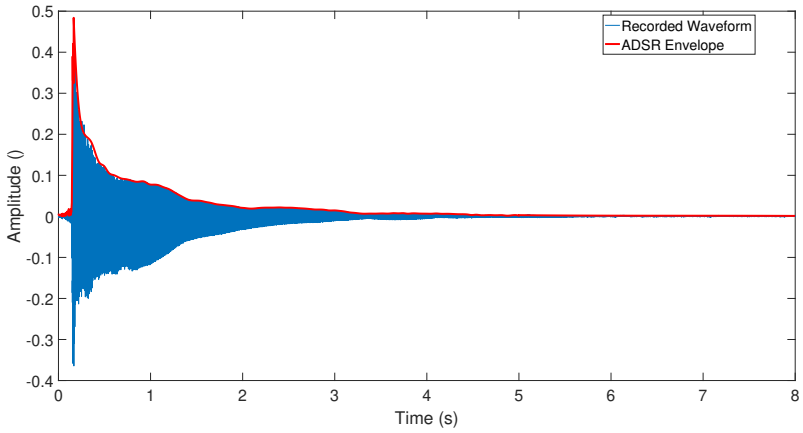

Figure 4: Envelope of A_3 played on Acoustic Guitar

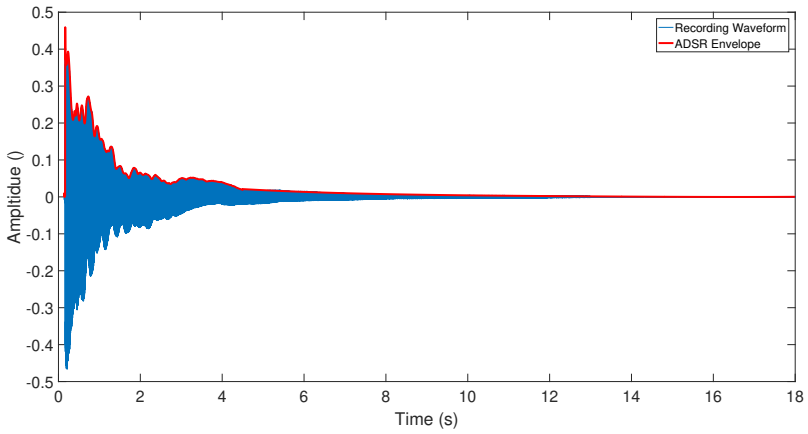

Figure 5: Envelope of A_3 played on Electric Guitar

Note that the envelope for the acoustic guitar is not quite as tight to the signal as that of the electric guitar. This is simply due to the behavior of the envelope function. Curve fitting can be a difficult process, and after experimenting with the number of samples per interval used in the envelope function, the displayed result was determined to give the best fit possible to the data.

Either of these envelopes can be used to properly scale the amplitude of a synthesized signal which hopefully will result in a realistic guitar sound. However, this is only one piece of the puzzle. To determine how the signal will be constructed, the frequency domain must first be analyzed.

4.3 Frequency Domain

Using the results from Section 2.5 the FFT function in MATLAB will provide the frequencies contained in both recordings. It is expected that the fundamental frequencies in both will be relatively close to $\nu = 220$ Hz, where the error in this value is due to each guitar being slightly out of tune. Below are the frequency domain plots for both the acoustic and electric guitars. Various Frequencies are highlighted. The MATLAB code used to produce these plots is found in Section 9.1.

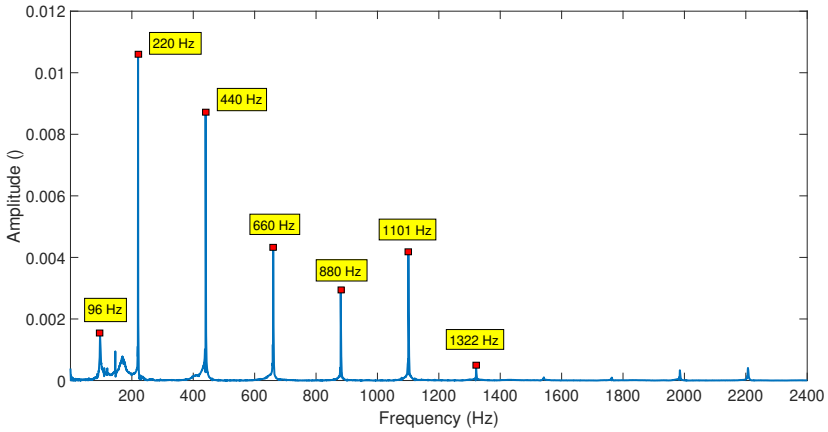

Figure 6: Frequency Spectrum of A_3 played on Acoustic Guitar

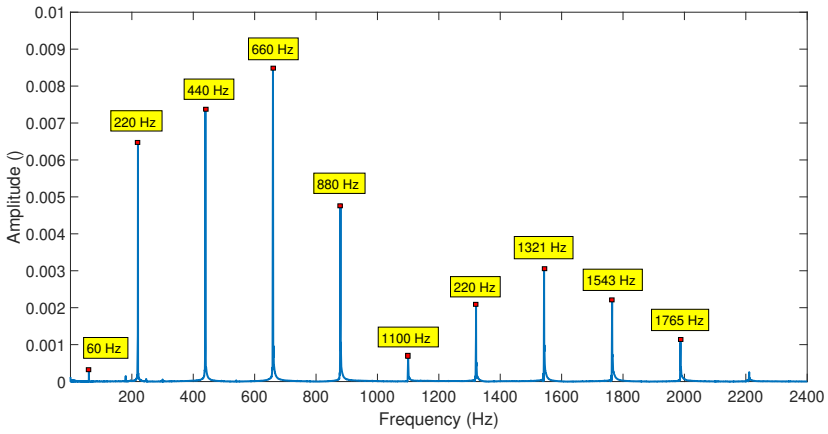

Figure 7: Frequency Spectrum of A_3 played on Electric Guitar

There are a few surprises in these plots. The most notable is in the spectrum of the electric guitar. The fundamental $\nu = 220$ Hz did not have the largest amplitude- in fact the first two harmonics have larger amplitudes.

This was not the case with the acoustic, where the fundamental is the dominant frequency. The second highlight is found in the low end of the spectra. It was expected that the fundamental would be the lowest frequency found in these signals (and we know what the fundamental is because it was tuned accordingly), however this is not the case. The low end of the acoustic spectrum is fairly busy with what appears to be a range of frequencies. This characteristic of the acoustic spectrum will be further discussed in the Synthesis section.

To get a better comparison we will plot both spectra, where the amplitude of each frequency is expressed as a the percentage of the largest amplitude in the respective spectrum.

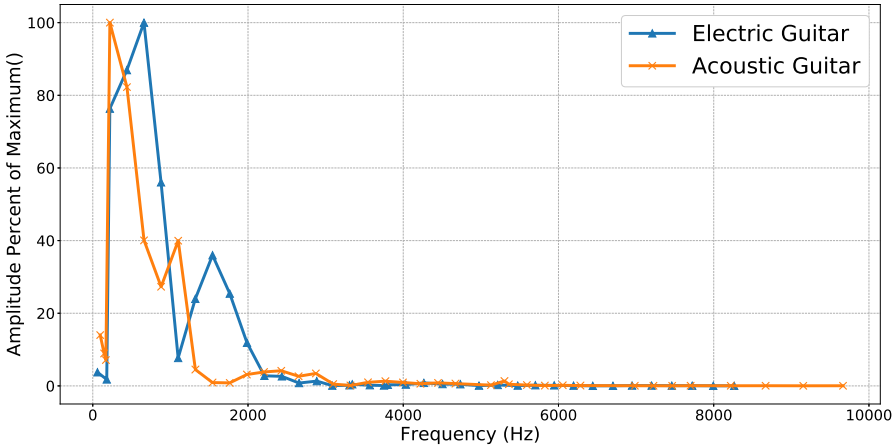

Figure 8: Frequency Spectra Expressed as Percent of Largest Amplitude

As a generalization, the electric guitar signal appears to be dominated by mid and high range frequencies, whereas the acoustic signal has an increased dependency at the low end. This may explain why acoustic guitars tends to have a warmer feeling to their sound than an electric.

A careful inspection of the spectra was performed and a list of the frequencies and amplitudes was compiled for each signal. It is unclear exactly what range of frequencies should be considered when synthesizing a similar signal, hence it is unclear as to what range is responsible for creating the sound we would recognize as either an acoustic or an electric guitar. Due to this, it was determined that all observable frequencies should be considered, at the very minimum as a starting point. The table of all frequencies present in each spectrum is found in Section 8.

5 Sound Synthesis

5.1 Signal Model

At this point, the time-domain ADSR envelope and constituent frequencies have been determined for each signal. It is expected that if the analysis was done without error, this information should be sufficient to synthesize a signal that resembles that of an acoustic and an electric guitar.

A simple model will be used to construct both signals. An array of time values will be instantiated with length equal to that of the corresponding guitar signal. A summation of sine waves will be iteratively implemented, with frequencies and amplitudes corresponding to those observed in the Fourier spectrum. The independent variable will be the time. This signal can be represented mathematically as follows

$$f(t) = \sum_{i=1}^{N} A_i \; sin(2\pi\nu_i t) \qquad \text{summation is over the observable frequencies}$$

Once the frequencies are proportionally set, the time envelope for the corresponding recording was applied to the signal by way of element-wise multiplication. This should scale the amplitude of the signal to match that of the recorded guitar. This model was implemented in MATLAB and the corresponding code can be found in Section 9.3.

5.2 Interpretation of Results

The synthesized electric guitar A_3 note is saved as Track 4 and the acoustic version is Track 5.

The electric guitar sound seems to be more or less on point. An electric tuner was plugged into the output of the computer to verify that the constructed signal was, at the least, in-tune. Comparing to the recording revealed that there are some strong differences between the signals, however the first impression was that the synthesized signal did indeed resemble an electric guitar. Perhaps the only odd aspect is how much the signal fluctuates during the first few seconds. Having gone through the process of attempting to fit the envelope correctly, it is clear this step may be responsible for this effect.

Synthesizing the acoustic guitar sound was not as successful. The initial reaction was that the sound was very muffled and missing the sharp attack that is characteristic of an acoustic. Further, the signal sounds very artificial for the sustain and release phases.

An attempt was made to isolate the annoying humming that can be heard during these sections, by removing certain frequencies from the signal, however it was determined that the hum is a result of all the frequencies added together.

During this process of removing one or some of the frequencies, it was found that the low end of the spectrum actually results in a realistic feature of the signal. The frequencies below the fundamental are responsible for the "hollow body" sound of the acoustic. It almost sounds like the body of the acoustic is being struck- which is appropriate since the sound wave is indeed striking the soundboard.

Track 6 is the synthesized signal containing only the low end frequencies. It should be noted that either a relatively good quality speaker or headphones are required to properly hear this signal. After listening to variations of the signal some several hundred times, this component becomes extremely prominent to the point where it is impossible not to notice even when playing a few notes on the acoustic. This discovery alone makes the exercise worthwhile, as a key differentiating characteristic between the guitar sounds can be found in the low end dependence.

A considerable amount of experimentation was done with the acoustic envelope, only to obtain very similar results. Similarly, various combinations of the frequencies were tested, as well as adjusted to perfect multiples of the fundamental. None of these attempts resulted in a more realistic synthesized signal.

To conclude this section, a final attempt was made at producing the most realistic sound possible. Both synthesize notes were imported into Ableton Live 9 and "Small Room Reverberation" was added. The resulting WAV file is labeled Track 7. The reverberation made the sounds slightly more realistic and seemed to enhance the low-end of the acoustic (or perhaps that is all I can hear now).

It appears that the envelope and frequency analysis of the signal is too simplistic to synthesize a realistic guitar sound. Initially the plucked string is rich in harmonics. As the string's energy decreases there are fewer harmonics. The signal synthesis described above simply applied the envelope to all frequencies, and they damped out at the same rate.

That fact that each frequency has its own envelope may be a key characteristic of a realistic signal. In the next section, the analysis will make a slight detour to investigate a method that accounts for this frequency dependent damping.

6 Karplus-Strong String Synthesis

The *Karplus-String algorithm*, named after its inventors, Kevin Karplus and Alex Strong (1983) is an efficient method for synthesizing the sound of plucked string instruments. The algorithm is relatively straightforward, and improvements have been made on its over the years [5]. The method will be implemented and the synthesized signal of an acoustic guitar will be compared to the previous attempts made in this report.

6.1 Karplus-String Algorithm

First, assume that the vibration of the string is an LTI system. The Karplus-String loop with a delay line (buffer) of N samples, initialized to the length of one period of the fundamental frequency f, with sample rate f_s

$$N = \frac{f_s}{f}$$

For simplicity, we will use the floor function as the length must be an integer value. Note that the buffer size represents the pitch of the signal.

The algorithm starts with the buffer full of random numbers, which represent the initial energies transferred to the string when it is plucked. Every time a value is read from the buffer, it is averaged with the previous value. This filter corresponds to the following difference equation

$$y[n] = 0.5(x[n] + x[n-1])$$

The output is fed back into the buffer which evolves the waveform. Due to the nature of the averaging process, the signal changes less and less with each iteration. This acts as a low-pass filter, as it limits the number of high frequencies, which is precisely how a string evolves over time. This behavior was missing from the previous attempt to synthesize the signal.

The block diagram of this algorithm can be represented as follows [6]

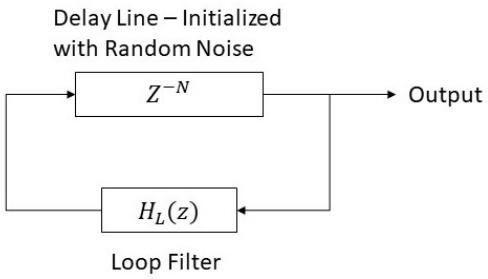

Figure 9: The Karplus-Strong Block Diagram

The closed-loop transfer function has the form

$$\frac{Y(z)}{X(z)} = \frac{G(z)}{1 + H_L(z)G(z)}$$

where $G(z)$ is the delay line and $H_L(z)$ is the low-pass filter.
Here,

$$G(z) = z^{-N} \qquad H_L(z) = \frac{1 + z^{-1}}{2}$$

Thus, the closed-loop transfer function of the Karplus-Strong algorithm is

$$H(z) = \frac{z^{-N}}{1 - 0.5(z^{-N} + z^{-N-1})}$$

This version of the Karplus-Strong algorithm was implemented in MATLAB (code can be found in section) and a signal was generated with frequency $f = 220$ Hz. The resulting WAV file is labeled Track 8.

6.2 Comparison of Synthesis Methods

The signal generated by the Karplus-Strong method certainly sounds like an acoustic guitar. It is clearly a massive improvement over the first synthesize signal. The signal does not have the humming feature and the attack portion of the envelope successfully mimics the pluck of an acoustic guitar.

The release portion of the envelope is not present, however it would not be difficult to include. In the previous section, the envelope of the recorded signal was manually fit and applied to the synthesized signal. In theory, this would have given the synthesized signal a realistic component, however defining the phases of the ASDR envelope resulted in a signal that fluctuated in a less than realistic way. There are numerous pieces of software that are designed specifically for this purpose, and with a bit a fine tuning my code could be improved, however the envelope was omitted from the Karplus-Strong signal to so that it simply represents the output of the algorithm.

This second attempt at synthesizing a guitar signal was motivated by the lack of frequency dependent damping. The simple low-pass averaging filter accomplishes this task, however the decay of each harmonic is not specifically addressed. This means that we paid no attention to how fast the individual overtones decay.

As mentioned earlier, improvements have been made to this algorithm, most of which specifically address this issue [7]. Needless to say, analyzing the recorded signal to determine the decay of each frequency and implementing this in MATLAB would prove to be quite the endeavor.

Without going down that path, we can conclude that there are indeed alternative methods to synthesize guitar signals, that are vastly superior to the frequency/ ASDR envelope method employed in the first section.

7 Conclusion

In the analysis of the time and frequency domains of the recorded signals, we were able to establish several characteristics that differentiate between the acoustic and the electric signals. A note produced by an electric guitar has a cleaner waveform (more closely resembles a sinusoidal wave) and a longer amount of sustain than that of an acoustic. Oddly, we found that the first and second harmonics in the electric signal, had amplitudes larger than the fundamental frequency. This was not the case with the acoustic guitar and possibly explains why an electric guitar has a more treble-like sound.

While both recordings featured frequencies lower than the fundamental, the acoustic guitar had an increased dependency on the low end. This was determined to be a defining characteristic of the acoustic sound, and it is most likely explained by the reverberation of the sound wave within the acoustic cavity.

It seems as though analyzing the time and frequency domains independently severely limited how realistic of a sound we were able to synthesize. Implementing the straightforward Karplus-

Strong algorithm produced a sound that closely resembles an acoustic guitar, however it is completely independent of the mathematical model and signal analysis we did earlier in the report. Ideally, the initial model could be improved upon to produce better results, and this would give a stronger validation to the theory of the Fourier Coefficient description. Unfortunately, there is a huge number of factors involved when generating a note from a musical instrument, and the model we described would have to be vastly extended.

When it comes to a guitar, these factors include (but are not limited to) the damping of the left and right hands, the angle of pick attack, the positioning of the electric guitar pickups, and the fact that the guitar string is not a perfect harmonic oscillator. This list literally goes on and on. Including these factors in the mathematical model would pose such a daunting challenge, that it may very well be impossible- at least for a human. With the all advances being made in AI, perhaps this is an area where we will see significant progress in the coming years.

8 Appendix A: Frequency Spectra

8.1 Acoustic Guitar Spectrum

Amplitudes of Frequencies in A_3 played on an Acoustic Guitar			
Frequency (Hz)	**Amplitude** 10^{-5} ()	**Frequency (Hz)**	**Amplitude** 10^{-5} ()
96.13	148.30	3996	10.12
146.1	94.80	4221	6.33
168.6	75.39	4448	8.90
220.3	1060.00	4673	6.69
440.3	871.80	5130	2.23
660.5	424.70	5304	14.27
880.5	289.40	5361	4.74
1101	423.80	5593	2.52
1322	47.66	5824	0.86
1542	9.80	6055	1.82
1764	8.85	6282	0.92
1985	33.23	6983	0.54
2207	40.71	7223	0.39
2429	44.17	7465	0.36
2652	28.09	7703	0.69
2874	36.37	8219	0.42
3098	5.38	8669	0.49
3320	1.19	9154	0.32
3546	10.50	9666	0.42
3770	13.75		

8.2 Electric Guitar Spectrum

Amplitudes of Frequencies in A_3 played on an Electric Guitar			
Frequency (Hz)	**Amplitude** 10^{-5} ()	**Frequency (Hz)**	**Amplitude** 10^{-5} ()
60	31.68	3801	2.57
180	15.02	4033	3.07
219.9	644.2	4266	6.97
439.5	733.8	4506	4.33
659.7	844.00	4736	3.91
879.6	473.00	4976	0.68
1100	64.94	5217	2.13
1321	202.30	5475	0.80
1543	303.70	5702	1.55
1765	214.50	5945	1.30
1987	100.20	6194	0.57
2211	23.44	6442	0.19
2436	22.16	6702	0.24
2658	6.64	6948	0.73
2886	11.10	7200	0.73
3086	0.65	7460	0.57
3307	1.49	7723	0.47
3343	3.82	7994	0.24
3569	1.57	8264	0.27
3752	0.89		

9 Appendix B: Matlab Code

The first two code files have time values which are specified by "magic numbers". These numbers were determined by trial and error, and are specific to the recording itself.

9.1 Time and Frequency Plots

Code used to plot the time and frequency domains for the recordings of both the acoustic and electric guitars.

Listing 1: M-File Code for Time and Frequency Domain Plots

```
[audio_in,fs]=audioread(acoustic_recording);
t = linspace(0,length(audio_in)/fs, length(audio_in));
T0 = length(audio_in); % number of points of audio file

% Waveform Plot
figure; subplot(1,2,1) % first plot
t_domain = find(t>0 & t<6.0); % time domain for plot
plot(t(t_domain),audio_in(t_domain));
xlabel('Time(s)'); ylabel('Amplitude')
title('Acoustic A3 Waveform')

subplot(1,2,2) % second plot
t_domain = find(t>0.145 & t<.185); % time domain for plot
plot(t(t_domain),audio_in(t_domain));
xlabel('Time(s)'); ylabel('Amplitude')
title('Magnified View of Acoustic A3 Waveform')

% Frequency Plot
figure;
audio_fft = fft(audio_in, T0); % compute fft of audio file
amplitude = 2*abs(audio_fft(1:T0/2+1))/T0; % magnitude of rhs of fft
%amplitude = amplitude / max(amplitude)
frequency = fs/2*linspace(0,1,T0/2+1); % nyquist frequency

plot(frequency,amplitude)
axis([0 2400 0 1.2*10^-2]); xticks(linspace(200,2400,12));
xlabel('Frequency (Hz)'); ylabel('Amplitude ()');
```

9.2 ADSR Envelope

Code used to determine the ADSR envelope of the acoustic recording. The electric guitar envelope was determined using the same process. It should be noted that simply calling the envelope function for the release phase resulted in a "bumpy" release so a polynomial was fit to the envelope of this region which smoothed out the envelope.

Listing 2: M-File Code for Envelope of Acoustic Guitar Signal

```
function signal_envelope=acoustic_envelope(t,audio_in)
% Function is customized for acoustic_high_a.wav recording.
% Divides signal into Attack, Decay/ Sustain, and Release section for which
% an envelope is determined and returned. The samples/interval are different
% each section to provide best possible envelope fit.
% Paramters
%   audio_in:   acoustic_high_a.wav signal
%   t:          time values
```

```
9
10 % Attack Phase (a)
11 attack_region = t>=0.0 & t<.165; % attack region data points
12 y_a = abs(audio_in(attack_region)); % attack region signal
13 np_a = 100; % number of samples per interval
14 [up_a,~] = envelope(y_a,np_a,'peak'); % only upper envelope is considered
15 up_a = abs(up_a); % pretend any negative values didn't happen
16
17 % Decay/ Sustain (ds)
18 ds_region = t>=0.165 & t<5; % ds region data points
19 y_ds = audio_in(ds_region);
20 np_a = 2000;
21 [up_ds,~] = envelope(y_ds,np_a,'peak');
22 up_ds = abs(up_ds);
23
24 % Release Phase (r)
25 r_region = t>=5 & t<=8; % release region data points
26 t_r = t(r_region); % release region data points
27 y_r = audio_in(r_region);
28 np_a = 3000;
29 [up_r,~] = envelope(y_r,np_a,'peak');
30 up_r = abs(up_r);
31 up_r(1:500) = up_r(500); % Smooth transition at boundary
32
33     % It was found that envelope produced a "bumpy" release so a
34     % poloynomial was fit to represent average release
35 p=polyfit(t_r,up_r,3); % 3rd order polynomail found to give good fit
36 y_fit = polyval(p,t_r); % calc release vals of signal
37
38 % Combine Results
39 up = [up_a up_ds y_fit]; % concat each section of signal envelope
40
41 signal_envelope = up; % return envelope
```

9.3 Signal Synthesis

Code used to synthesize acoustic guitar signal. Writes result to WAV file. The synthesized electric guitar signal was done using the same process.

Listing 3: M-File Code for Acoustic Guitar Signal Synthesis

```
1  [audio_in,fs]=audioread(acoustic_recording); % read file
2  T0 = length(audio_in); % number of points of audio file
3  t = linspace(0,T0/fs, T0); % independent variable
4
5  envelope = acoustic_envelope(t,audio_in); % determine envelope of signal
6
7  duration=8; % set duration to match guitar recording
8  signal = 0; % declare output signal variable
9
10 % all frequencies observed in acoustic frequency spectrum
11 freqs = [96.13, 146.1, 168.6, 220.3, 440.3, 660.5, 880.5, 1101, 1322, ...
12     1542, 1764, 1985, 2207, 2429, 2652, 2874, 3098, 3320, 3546, 3770, ...
13     3996, 4221, 4448, 4673, 5130, 5304, 5361, 5593, 5824, 6055, 6282, ...
14     6983, 7223, 7465, 7703, 8219, 8669, 9154, 9666];
15
16 % ampltidues of frequencies observed in acoustic frequency spectrum
17 amps = 10^-5*[148.3, 94.8, 75.39, 1060, 871.8, 424.7, 289.4, 423.8, ...
```

```
    47.66, 9.8, 8.847, 33.23, 40.71, 44.17, 28.09, 36.37, 5.377, 1.189, ...
    10.5, 13.75, 10.12, 6.331, 8.902, 6.686, 2.227, 14.27, 4.743, 2.52, ...
    0.8555, 1.816, 0.9218, 0.5397, 0.3872, 0.3638, 0.6866, 0.4186, ...
    0.4894, 0.3214, 0.4163];
% add A*sin(2pi*v*t) for all frequencies, v
for n = 1:numel(freqs)
    signal = signal + amps(n) * sin(2*pi*freqs(n)*t);
end

signal = signal.*envelope; % element-wise multiplication of signal & envelope
signal = signal / max(signal); % normalize Signal

audiowrite(synth_acoustic,signal,fs); % write signal to WAV file
```

9.4 Karplus-Strong Algorithm

Code used to implement the Karplus-Strong algorithm. Based off of code sourced from this reference article [8].

Listing 4: M-File Code for Karplus-Strong Algorithm

```
function signal=karplus_strong(f,duration)
%   Karplus-Strong algorithm
%   f = fundamental frequency
%   duration = length out output signal

fs = 44100; % sampling frequency
N = floor(fs/f); % integer buffer delay line length

x = zeros(1, duration*fs); % initialize signal input to zero

b = [zeros(1,N) 1]; % delay block
a = [0 zeros(1,N-1) -.5 -.5]; % low-pass averaging filter
zi = rand(1, max( max(size(a)), max(size(b))) -1);
% intialize buffer to random noise

% funbction filters input x using transfer function defined by b/a
signal = filter(b, a, x, zi);
```

References

[1] James S. Walker Jeremy F. Alm. Time-frequency analysis of musical instruments. *SIAM REVIEW*, 44(3):457–476, 2002.

[2] B.P. Lathi and Roger A. Green. *Essentials of Digital Signal Processing*. Cambridge University Press.

[3] Wolfram MathWorld. Fourier transform–sine. http://mathworld.wolfram.com. Accessed: November 15, 2018.

[4] Rossing T.D. Fletcher N. H. *The Physics of Musical Instruments*. Springer Science+Business Media, Inc.

[5] David A. Jaffe and Julius O. Smith. *Extensions of the Karplus-String Plucked-String Algorithm*. Center for Computer Research in Music and Acoustics.

[6] Nicholas P. Donaldson. An electric guitar plucked string model for realtime control with distortion and feedback. *www.music.mcgill.ca/ gary/courses/projects/*618_2009*/NickDonaldson/#Loop*. Accessed: Number 23, 2018.

[7] C. R. Sullivan. *Extending the Karplus-Strong Algorithm to Synthesize Electric Guitar Timbres with Distortion and Feedback*. Center for COmputer Research in Music and Acoustics.

[8] Steven Sanders. Synthesizing a guitar using physical modeling techniques. www.ee.columbia.edu/ ronw/dsp/#sources. Accessed: Number 17, 2018.