ME173 Final Project

Modeling Acoustic Timbres with FM Synthesis

Quy Chau

ME173, University of California, Berkeley

By studying the fundamentals of timbre as well as frequency modulation in digital synthesis, I produced a computer algorithm in MATLAB to automatically analyze the harmonics and time response of an instrument sample and recreate the instrument with a simple FM digital synthesizer.

Special thanks to Eric Nguyen for helping me understand frequency modulation and fast Fourier transform, as well as assisting with MATLAB coding.

A - Introduction

Timbre and FM

For applications in the fields of sound engineering and music the harmonic characteristics of the sound wave is integral. Harmonics of spectrally complex sources like a plucked string, a pipe blown through, and other common instruments can be described by the multiple frequencies and relative weight of these frequencies that result from resonant excitation or impulse. Most instruments can produce multiple frequencies at once that also change in amplitude over time. Therefore, to describe the timbre of an instrument it is necessary to identify two components of the resulting waveform: (1) the spectral spread, including harmonic and partial overtones and (2) the temporal activity of the amplitude.

Timbre is traditionally designed with consideration to both of these qualities. For example, a guitar string which produces a harmonically rich spectrum also has a modest “sustain” time and amplitude decaying slowly. In contrast, the spectra of drums are sparsely populated and contain a rapidly decaying amplitude, but negligible “attack” (initiation). However, while timbre is often created and analyzed for specific instruments by considering the physical parameters of the instrument and qualitatively judging the tones produced, there has been few attempts to reproduce timbre through emulation of the original waveform.

Frequency Modulation (FM) has long been used in a variety of signals applications, most notably radio signals (FM radio). After John Chowning identified the usefulness of frequency modulation in synthesizing audio spectra [1], it has since been the basis of common analog and digital synthesizers on the market today. Through a simple model of a sinusoidal wave modulated by a second wave, both with their own particular frequencies and amplitude envelopes, the automative synthesis of complex timbres has been realized with striking ease. The reason for this simplicity lies in the aforementioned components of timbre. Harmonics are manipulated through the ratio of the frequency belonging to the carrier wave

(wave that produces the sound) and the frequency belonging to the modulator (wave that modulates the carrier), as well as the relative “weights” of these frequencies. “Strong” modulation is embodied by the *index of modulation* I, which dictates the amplitudes of the frequency peaks in the resulting spectrum. Finally, the temporal quality is described by the change of the amplitude of the carrier over time as well as the change of the modulation index over time. However, we will see that the modulation index also dictates the harmonic spectrum of an FM-produced waveform.

A2 - Theory

FM Synthesis

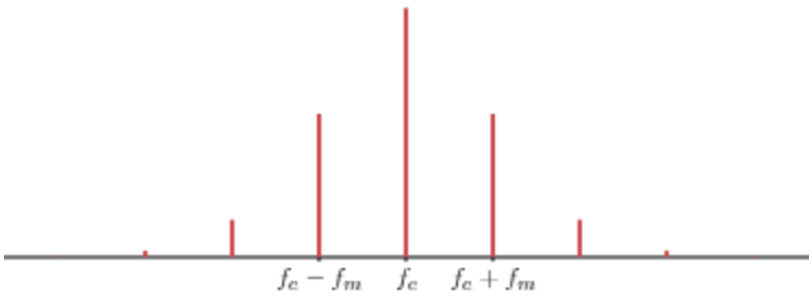
The basic idea of FM synthesis arises from the fact that if a simple sine wave with amplitude A and frequency ω_c:

$$p(t) = Asin(\omega_c t)$$

is **modulated** by a second sine wave with amplitude I and frequency ω_m in the same order of magnitude as the first frequency (e.g. audible range):

$$p(t) = Asin(\omega_c t + Isin(\omega_m t))$$

The result is a waveform where both the carrier frequency and the modulator frequency can be found in the resulting spectrum obtained by Fourier transform in the frequency domain:

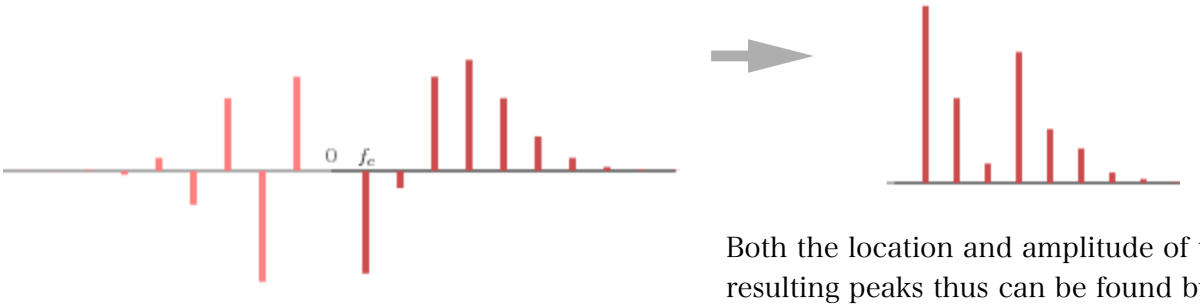


More specifically, the carrier frequency is present while frequencies at integer multiples of the modulator frequency to the right and left can be symmetrically found about this point. These frequencies are called **sidebands**. When $f_c : f_m$ is a rational ratio, the resulting spectrum with be harmonic. The number of peaks with significant amplitude is proportional to the **index of modulation, I** in the equation above. The entire equation formulated by Chowning [1], with the effects of phase included by Bate [2], is as follows:

$$p(t) = Asin(\omega_c t + \phi_c - Icos(\omega_m t + \phi_m))$$

The effects of phase are not discussed here; by setting $\varphi_c = 0$ and $\varphi_m = \pi/2$ we have the standard model proposed by Chowning.

The relative amplitude of the peaks are dictated by the Bessel function of the first kind of order n, $J_n(I)$ where n is the integer multiple of the modulator frequency away from the carrier frequency (e.g. $f_c + 2f_m$, $f_c - 2f_m$) and I is the index of modulation [1]. However, when the sidebands extend into the negative frequency domain, they reflect to the positive side of the real one-sided FFT spectrum:



Both the location and amplitude of the resulting peaks thus can be found by knowing the carrier, modulator frequencies, and modulation index beforehand, but not the other way around. Therefore for the purposes of this project Bessel functions will be ignored and Carson's rule [5] will be used to estimate the index from the generated f_c and f_m:

$$I \approx f_d / f_m$$

Here f_c is the peak variation of the carrier frequency, and is thus represented by the bandwidth of the signal. The difficulty of robustly finding these frequencies and the modulation index from a single-sided FFT spectrum is high, and may be impossible. Because the modulation index is changing over time in a typical FM synthesizer, the positions and amplitudes of the frequencies cannot be predicted. This project will make use of automative guess-and-check to validate its results for picking certain f_c , f_m , and I.

Timbre

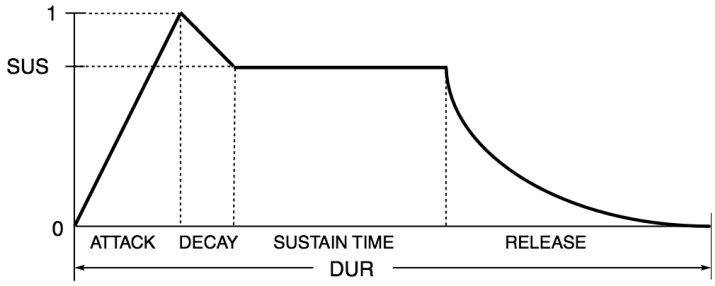
The most succinct measure I found to describe the timbre of an instrument is Pollard and Jansson's **tristimulus** model. The model is a direct music analogy to the concept of RGB values for color; the stimulus for the sense of "sound" is dictated by the mix of three major harmonic "categories". The first category is the fundamental frequency, the second category is the 2, 3, 4 overtones, and the third category is the rest of the overtones. The value of each category is the weight (amplitude) of the frequency in the category divided by the total weights of present harmonics.

$$T1 = a_1 / \sum_{i=1}^{N} a_i \qquad T2 = (a_2 + a_3 + a_4) / \sum_{i=1}^{N} a_i \qquad T3 = (\sum_{i=5}^{N} a_i) / (\sum_{i=1}^{N} a_i)$$

I will primarily use these three values in determining the "timbre" of a sample input for the purpose of producing a matching "timbre" for the output using least-squares.

An envelope is the time profile of the amplitude of a wave. Typically in music and digital synthesis it is described by four distinct periods: attack, decay, sustain, and release:

For the amplitude and index of modulation envelopes, a similar “least-squares error” approach is used where the ADSR envelope with the least discrepancy with the original waveform is picked, among a series of FM-generated waveforms with varying carrier and modulator frequencies.



B - The prototype product: Automated timbre-modeling FM synthesizer

By utilizing the ability of computing software such as MATLAB to quickly determine the spectral components of an instrument’s timbre (using Fast Fourier Transform) and producing a simple FM model with consideration to just the harmonics and temporals, I have successfully automated the timbre reproduction of simple instrument tones with varying degrees of accuracy. The FM model acts as a synthesizer in that it outputs a waveform that mimics the original instrument, given a note to play.

While individual classes of instruments can be easily distinguished by the software model (e.g. guitars vs. woodwinds), the exact tone quality of each instrument distinctive from other instruments of the same class is not clearly reproduced. However, the spectral content for harmonic instruments is the “best” that can be found in regards to the tristimulus timbral components. For inharmonic instruments that produce multiple frequencies that are not rational multiples of each other (e.g. a bell), the model still captures the inharmonious relationship between frequencies but most likely does not accurately capture timbral content because the tristimulus coefficients are based on harmonics.

Multiple plots of the waveforms, fast Fourier transform, and amplitude envelopes, as well as the tristimulus timbral components of the spectra, are provided by the code. These are purely for analytical purposes and to assess the validity of the algorithm. The full code can be found in Appendix A.

mainFM.m

The function first prompts the user for an input .WAV file. This WAV file should be no longer than 5 seconds, should be a sample of an instrument producing a single tone/note (e.g. middle C), and as little noise as possible. The WAV should start in silence and then capture the initiation of the instrument after about half a second, and should not end before the note is done. The note’s velocity should not change over time, but an instrument may either be impulsed or driven. The WAV file should be at least 2 seconds long.

Once a WAV file is entered, the program will call generator.m to analyze and generate the common FM synthesizer parameters of carrier and modulator frequencies, the amplitude

and modulation index envelopes, and the total time a note should be played. Note that a note from this synthesizer will only play as long as the input file and will attempt to match an envelope based solely on the input file's amplitude and duration.

While generator.m is processing it should produce various "trial" sounds that represent the algorithm testing each possible output timbre with the input timbre. This is purely for fun and does not impact the algorithm except for its speed.

Once generator.m has generated these parameters and displayed useful plots/data, the main function will then ask the user to input a note of the form [A-G][#]*[b]*[\d.]+, for example A0, Eb4, C#3, etc. The generator parameters will then be passed to operator.m along with the pitch (translated from note to frequency by noteConvert.m).

generator.m

Part I: Amplitude and Index Envelopes

First the function reads the WAV file, plots it, and then produces an amplitude envelope. This amplitude envelope is based off a standard Attack-Decay-Sustain-Release model, and it finds the optimal times for each section based on trial-and-error until the least-squares error with the original WAV's amplitude profile at 100 even time slices is minimized. The attack is the exception; it is found by calculating the time between the first non-negligible sound in the file and the first instance of the maximum amplitude. This is to avoid unnecessary nesting of loops. Attack and Sustain are straight lines while Decay and Release are exponential decays.

A similar approach is attempted with the index of modulation. The "original" modulation index envelope was constructed by considering 20 time slices, and representing the index by the bandwidth for each time slice. In comparison, the original WAV amplitude is linearly constructed with 100 time slices to find the amplitude envelope. Because the calculation for modulation index at each time slice depends on analyzing the FFT of the input WAV, it proved better to limit the number of time slices and get a more rough idea of the

Part II: Carrier and Modulator frequencies

To find the carrier and modulator frequency the code first computes the FFT of the spectrum, plots it, and finds the magnitude and location of the peaks by passing the waveform to findCM.m. It then iterates through all peaks to find the timbrally optimal carrier and modulator frequencies. For every iteration, it assigns one peak to the carrier frequency and one to the modulator frequency, and then uses the amplitude envelope calculated in part I and a simple exponential envelope for the index of modulation to output a T-second long WAV file by calling operator2.m each iteration. It finally reads the WAV file created by the operator and analyzes the timbral spread with findCM.m, writing the new parameters if the error is minimal.

findCM.m

This function simply does FFT on a sample and returns the frequency location and magnitude of "peaks", where a peak is defined as being at least 1/10th of the maximum magnitude of the FFT. It also returns the location of the maximum frequency found and an

array containing the three tristimulus values, obtained from the equations for T1, T2, and T3 listed in the theory section. The weights are calculated as the natural log of the peak magnitudes.

operator.m

Operator.m will simply output the waveform dictated by the FM model equation

$$p(t) = A(t))sin(\omega_c t + I(t))sin(\omega_m t))$$

for the duration specified by the input WAV file. It will save an output WAV to sample_output.m in the MATLAB directory and play the sample. Finally, it displays a smoothed FFT plot for the sample. operator2.m is the same, but it does not plot the FFT.

C - Analysis

The most important aspect of the generator.m algorithm is that it strives to attain maximum **timbral accuracy**, where timbre is quantified as a set of tristimulus values T1, T2, and T3. It does not attempt to find the most accurate carrier or modulator frequencies and index of modulation. If given a WAV file generated explicitly from an FM synthesizer, it will not be able to accurately find the carrier and modulator frequencies or the index of modulation envelope that was originally used in the synthesis.

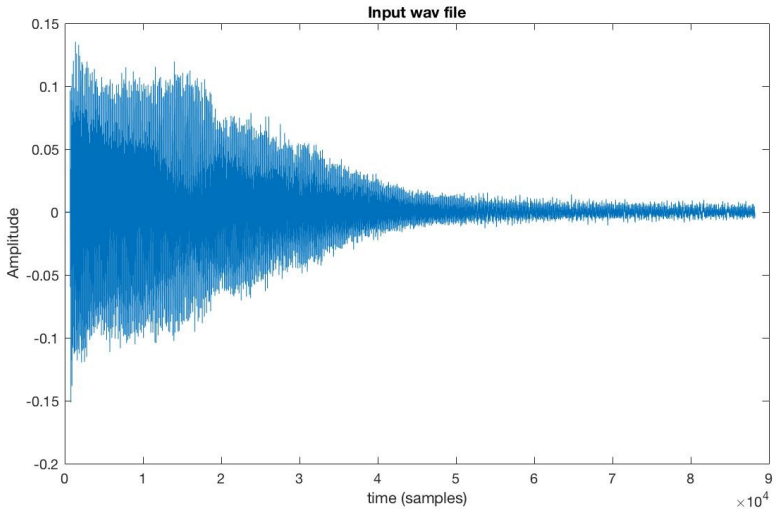
For testing the algorithm, eight distinct WAV files were used. Four WAV files are sampled instrument sounds, pulled either from the web as a sampled instrument to be used in a digital sound production software (e.g. Ableton, Logic Pro, FL studio) or recorded (by a third party). The other four WAV files are generated using Ableton's Operator, which is a digital FM synthesizer that uses the basic FM model found here. Each of these instruments, whether sampled or synthesized, is recorded in Ableton by playing a single note (usually C4) with a MIDI keyboard for the duration of about 8 beats at 120 BPM, with the note off on the first and last beats.

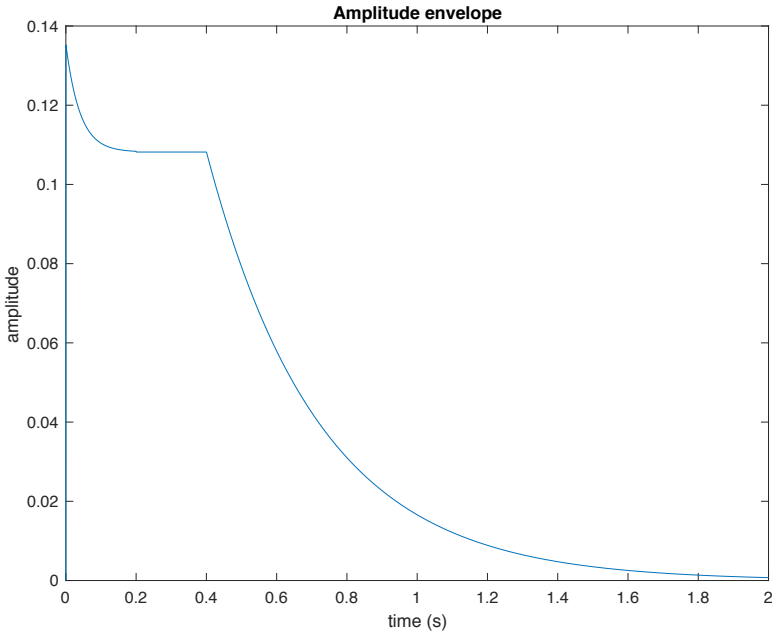
Analysis 1: Sampled instrument

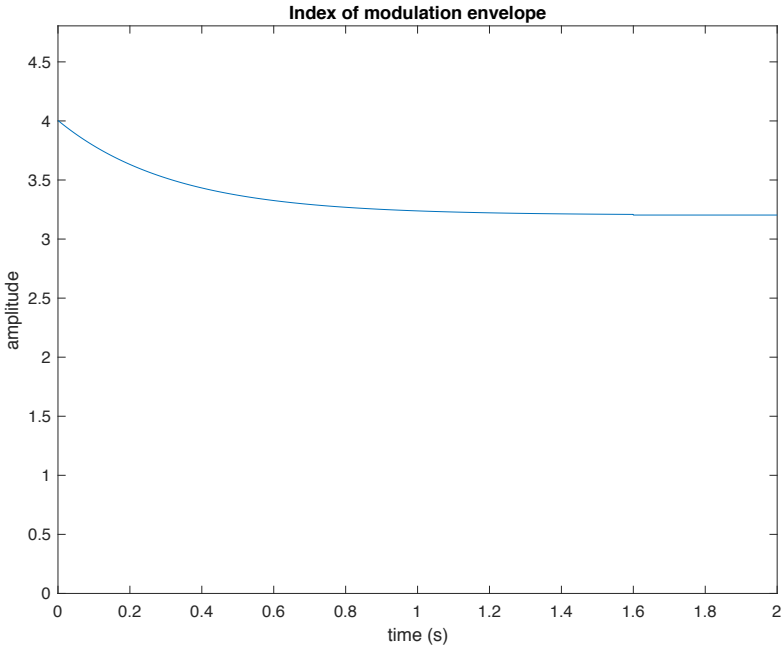
First we will analyze a realistic application of the software; playing a short recorded sample of an instrument and outputting a realistic FM-generated sound. Playing guitar_e_b.wav, which is produced by plucking the E string (E3) on a guitar, we hear a fairly clean impulse response from the string that is rich in harmonics. Before inputting any note to play, we input the file into the program and receive these output plots and data:

```
FFT peak frequencies:
   1.0e+03 *

    0.3288    0.6569    1.3140    1.6425
```

```
Tristimulus coefficients:
    0.2526    0.7474         0

output frequency peaks:
   1.0e+03 *

    0.3283    0.6573    0.9850
1.3135

output tristimulus
coefficients:
    0.2696    0.7304         0

output tristimulus square
error
    0.0241

amplitude error:
    0.0719

index of modulation error:
    1.8710

input fc:
  328.7688

input fm:
  328.1073

fm/fc:     0.9980
```



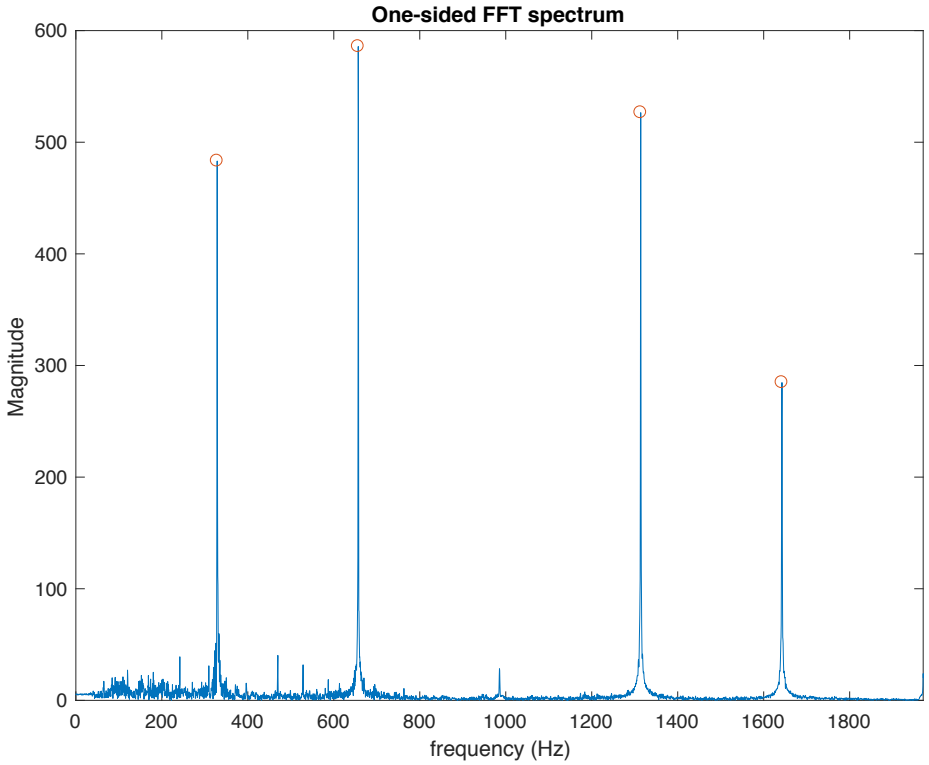


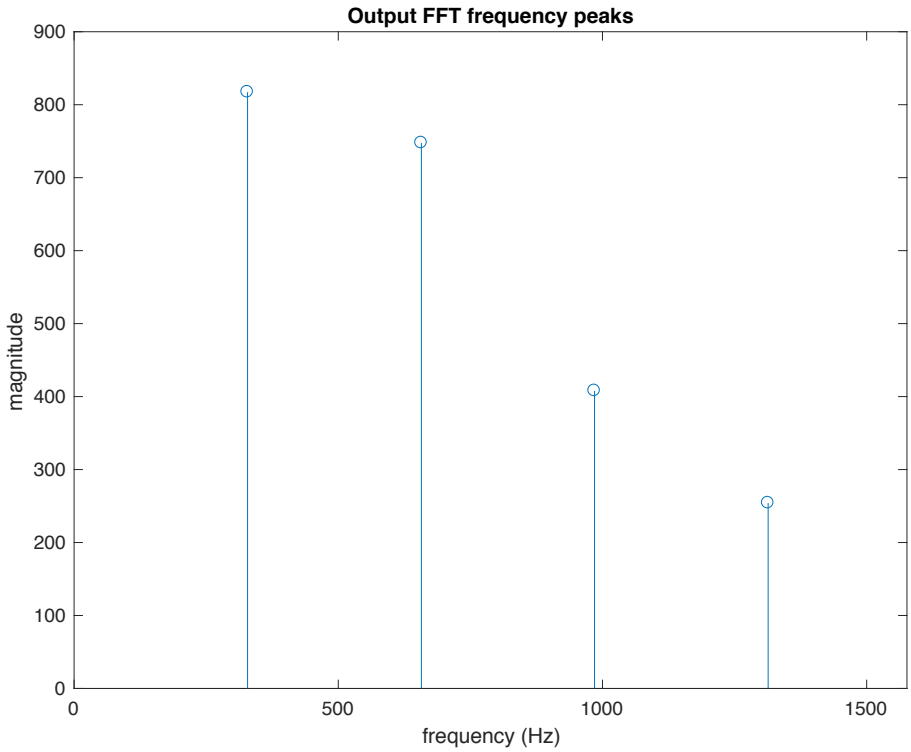


The amplitude envelope does a very nice job at portraying the sudden attack, initial dip and brief sustain, and release. The total least-squares error (0.0719) is about half of the maximum amplitude (0.15) , which is very decent.

It is more difficult to quantify the index of modulation “error” since the “desired” waveform is calculated from 20 highly error-susceptible calculations of the bandwidth from 20 different FFTs, which in turn is just an approximation of the index of

modulation (normalized by the modulator frequency) by Carson's rule. However, testing the least-squares fit of the ADSR envelope with the desired envelope reveals a modest error of 1.871 (compare to maximum modulation index of 4).





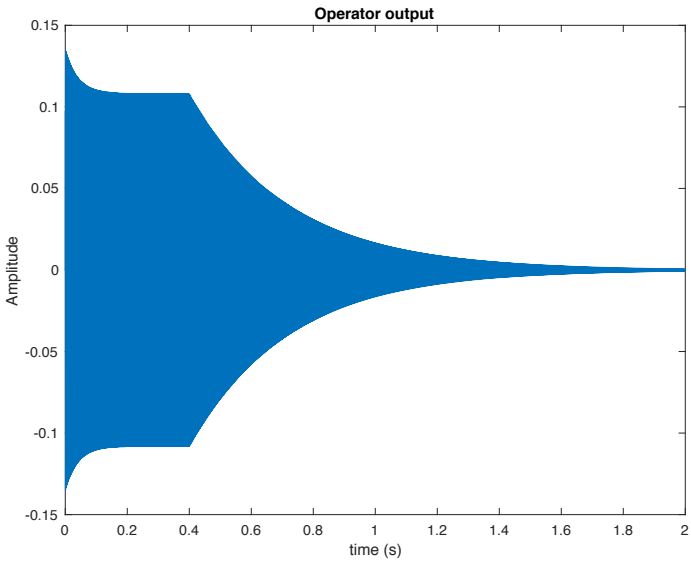
Note that the output FFT is just reading the FFT of a waveform generated with the specified “input fc” and “input fm” found by error minimization, as well as the amplitude envelope. However, the program replaces the modulation index envelope with a simple exponential decay because I am not confident in its accuracy.

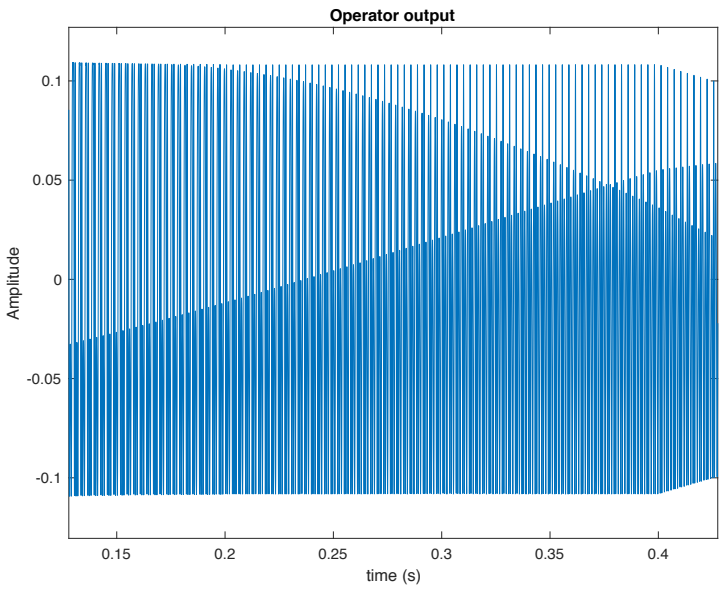
As expected, the tones present are harmonic multiples of the fundamental, which is at frequency 328.8 (note E4). Furthermore, we see that the tristimulus values are weighted about 25% on the fundamental and 75% on the 2nd, 3rd, and 4th overtones, with an extremely low error of 0.0241.

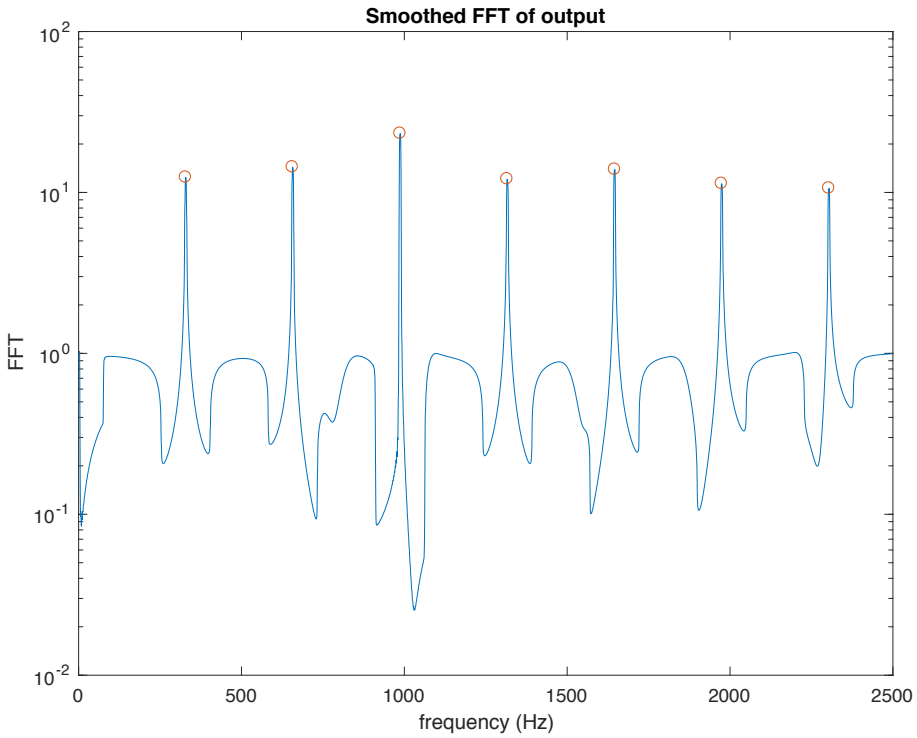
However, looking at the FFT plots we see that the output wave does not account for the gap in the original FFT at ~1000 Hz, or the “3rd” harmonic. Possible reasons for this discrepancy include an inaccurate FFT, as we see a small but negligible peak at 1000 Hz in the original sample. It more likely attests to the limitations of the simple FM model in capturing the complex spectrum; if we assume an accurate FFT, any “gaps” or unevenness in the spread of peaks relates to the reflection of negative frequencies. An FM model with the right index of modulation, carrier, and modulator frequencies could theoretically find a correct complex spectra to emulate these gaps, but understanding the emphasis of the program on timbral accuracy rather than spectral accuracy, we are also limited on assessing the tone based primarily on the tristimulus values. Though the fundamental is weighted less heavily than the second overtone in the original sample, but more in the output, the relative decreasing contour of the peaks is captured.

The program also fails to capture the higher peak at 1642.5 Hz. This is simply because it attempts to give the best spread for the first four overtones, since the sample reports T3 = 0 for any harmonics higher than the 5th, and thus gives what it believes to be the fifth overtone a weight of 0.

Because we know the note played was note E4 (329.63 Hz), we now enter in E4 to the program to hear an output note and plots of the output waveform and FFT:







The waveform pretty much perfectly conforms to the amplitude envelope, as expected, because the frequencies are very high; in this 2 second time slice, the waveform output from the model will oscillated rapidly enough so that only the amplitude changing over time is visible. Interestingly, if we zoom in we can see not just the rapid sinusoid but also contours of much slower periodic waveforms that seem to create a sense of "density" or layer in the output. This is most likely the effect of the modulation index on the waveform frequency over time, giving the waveform the illusion of being multi-layered.

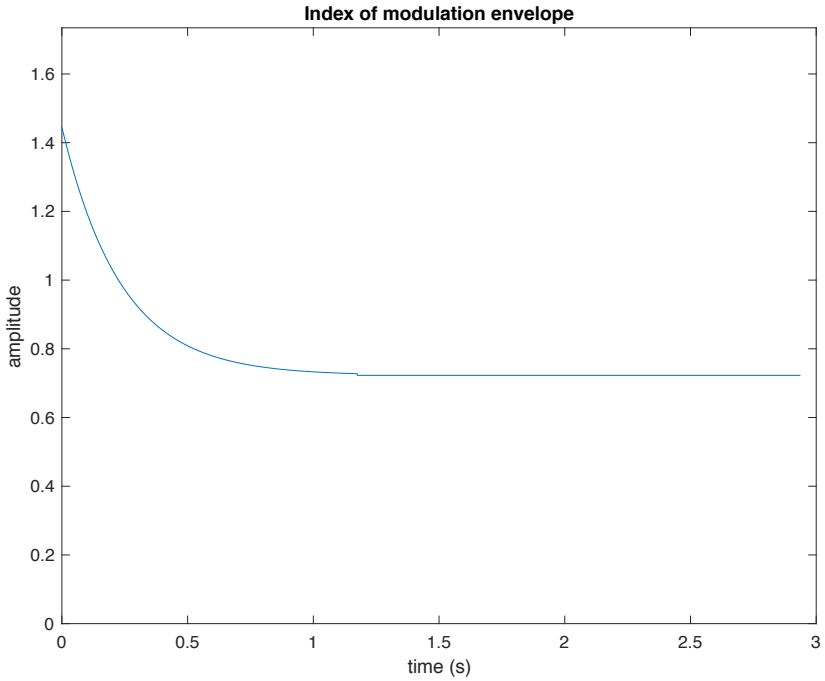
Here we see the smoothed FFT of the actual sound produced. The peaks are indeed all at overtones of the fundamental frequency, with no overtones missing, but with curiously more emphasis on the higher overtones than previously. This discrepancy is due to the use of the computed modulation index envelope rather than a simple exponential model in the final output, and as we see in the original envelope there is much higher index and thus bandwidth throughout the envelope, giving much weight to the higher harmonics. Also, the rising and falling profile of the peaks from the original FFT is captured here, yet curiously the strongest harmonic is the missing one at ~1000 Hz.

Of course, the output note sounds distinctly like a plucked guitar string, though how "close" it is to the actual instrument is up to the listener's judgement. Playing it at different frequencies reveals that it may perform more realistically at higher or lower notes, though that is again up to the listener's discretion.

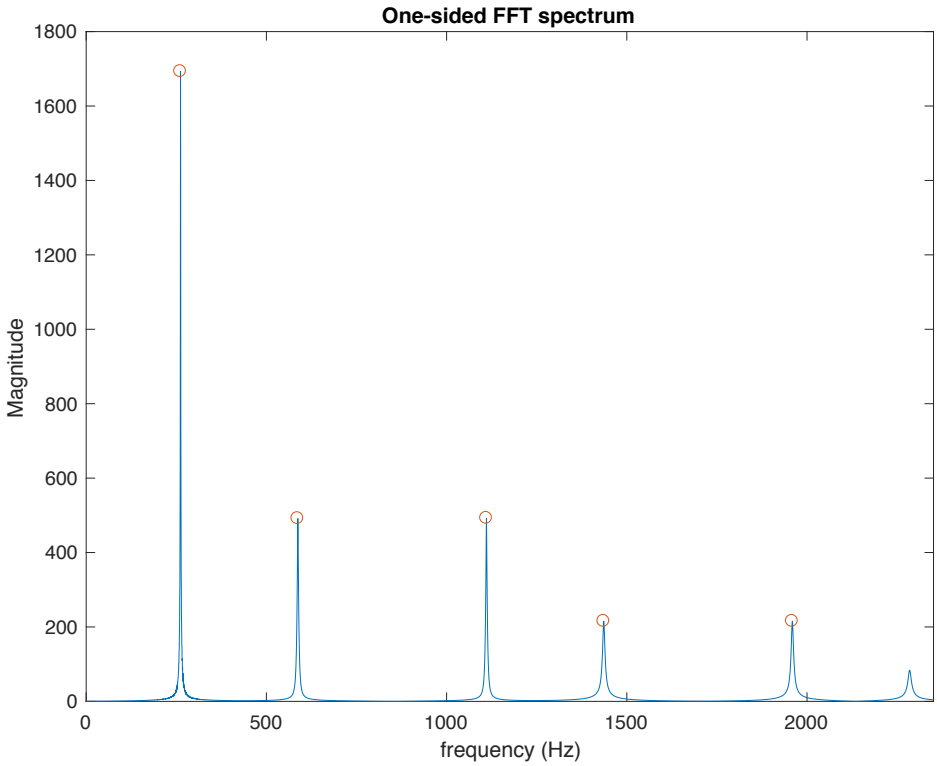
Analysis II: FM-generated instrument

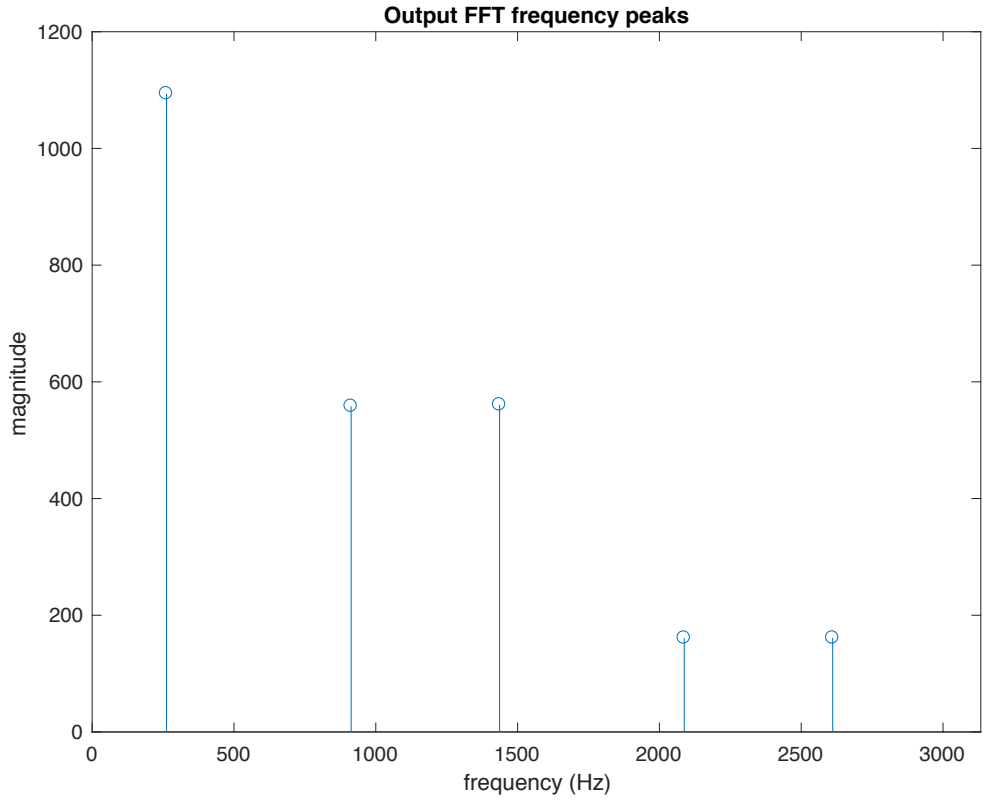
For the purposes of contrast, we now look at a waveform first generated by Ableton's Operator FM synth and recorded in software, so that it is as "pure" a sound as possible. It is

a “bell” that was obtained by following Chowning’s suggestions for the parameters of a bell instrument. The note played was C4 (middle C, frequency 261.6 Hz).




```
FFT peak frequencies:
   1.0e+03 *

    0.2617    0.5872
1.1107    1.4361    1.9592

Tristimulus coefficients:
    0.2432    0.5811
0.1757

output frequency peaks:
   1.0e+03 *

    0.2622
0.9131    1.4366
2.0875    2.6110

output tristimulus
coefficients:
    0.2347
0.5948    0.1705

output tristimulus
square error
    0.0169

amplitude error:
    0.4857

index of
modulation error:
   39.5434

input fc:
  261.7361

input fm:
   1.1744e+03

fm/fc:
    4.4869
```





Chowning suggests that the spectrum of a bell is initially rich and inharmonic [3], but then one frequency begins to dominate over time, so the index of modulation should have an exponential envelope. As expected, the index of modulation decreases over time exponentially. However, it sustains at a pretty high value, suggesting that the sample bell was not the same kind of bell envisioned by Chowning.

Looking at the FFT spectrum, it is very clear that the tones are inharmonic, appearing as non-evenly spaced peaks at irregular intervals. The ratio between modulator and carrier frequency is not obviously rational, so we can assume that it is irrational, giving the inharmonic timbre of the bell. However, the output peaks do not match very well except for the fundamental frequency, revealing that the tristimulus error minimization method may not favor inharmonic spectra as much as harmonic spectra. The spread of peaks with respect to the tristimulus values and peak contour is again very accurate, with error of only 0.0169.

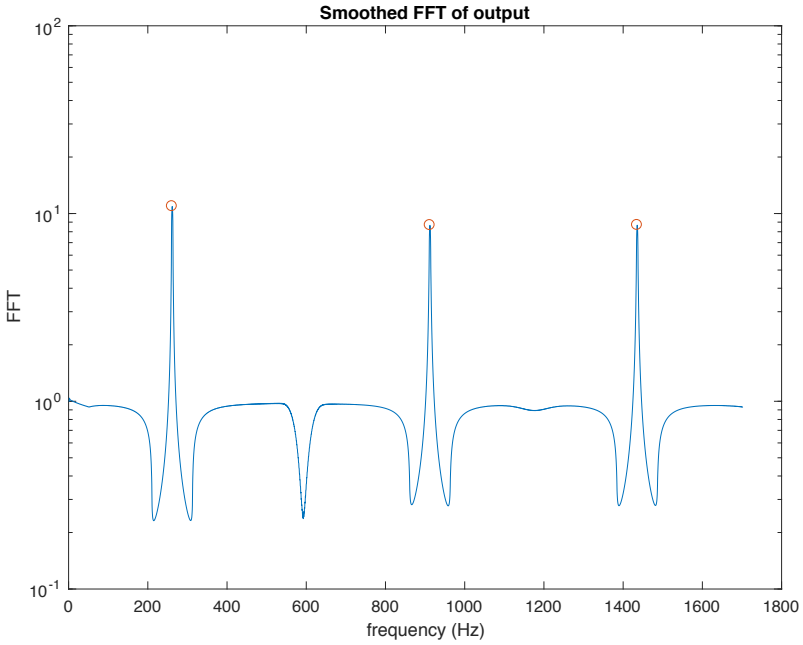
Playing the correct note C4 reveals more about the actual spectrum:

```
actual output peaks:
   1.0e+03 *

    0.2615    0.9124    1.4356
```

The resulting spectrum is less harmonically rich than the designed output FFT, showing that the index of modulation may have spread the sidebands less but in a more equally weighted fashion.

The actual sounded note is distinctly bell-like, and playing it at various pitches demonstrates the sustain-heavy “ringing” of a typical bell sound. This most likely can be attributed to the persistence of the index of modulation even after exponential decay, which is not in Chowning’s original formulation.

Playing the bell at lower frequencies tends to increase the number of peaks, while playing it at higher pitches usually results in only one peak.



Analysis III - All instruments

Table of the resulting carrier/modulator ratio, amplitude error, and tristimulus values:

Input WAV file	Source	Tristimulus Coefficients							Amp-litude error	FC : FM
		Desired			Actual			Error (norm)		
		T1	T2	T3	T1	T2	T3			
cello.wav	sample	0.1956	0.5130	0.2913	0.2913	0.5290	0.2936	0.0244	2.5461	0.9950
guitar_e_b.wav	sample	0.2526	0.7474	0	0.2696	0.7304	0	0.0241	0.0719	0.9980
piano.wav	sample	0.2973	0.7027	0	0.3189	0.6811	0	0.0305	0.7533	0.5006
piccolo.wav	sample	0.5618	0.4382	0	0.5476	0.4524	0	0.0200	1.9839	2.0042
bellt.wav	FM synth	0.2432	0.5811	0.1757	0.2347	0.5948	0.1705	0.0169	0.4857	4.4869
recorder.wav	FM synth	0.3303	0.6697	0	0.3116	0.6884	0	0.0264	0.5200	0.6667
horn.wav	FM synth	0.1415	0.4519	0.4065	0.1430	0.4579	0.3991	0.0097	0.4795	0.3335
wooddrum.wav	FM synth	1	0	0	1	0	0	0	0.3200	0

It seems that the most harmonically rich samples are the bell and horn, which surprisingly are synthesized samples. Furthermore, most instruments have fundamental frequencies weighted lower than 0.3, suggesting that the timbral spread is not as heavily favored towards the fundamental for most complex tones. However, this result could have been a consequence of taking the natural logarithm of the frequency peak magnitudes in tristimulus weight calculation, which levels the frequency peaks greatly.

In all cases the error is less than 0.05, or 5%, suggesting that the algorithm is harmonically accurate. This is, however, only by design, as the algorithm chooses its parameters to minimize this error, but it does suggest that in almost all cases an instrument sample's timbral content, measured by harmonic spread, can be quite accurately modeled by FM synthesis. Even in cases where it is hard to judge if the quantification of timbre by the tristimulus model is accurate, the actual sound output by the program verifies that the program can in fact reproduce at least the general tone particular to the sample's class of instrument (e.g. string vs. percussion).

D - Conclusion

The beauty of FM synthesis lies in its deceptively simple yet infinitely complex model. With one waveform and essentially two parameters - frequency and amplitude - a pressure wave can only describe two qualities of a musical note: pitch and loudness. With two waveforms and four parameters, it is possible to produce an infinite range of musical timbres. Therefore, attempting to find and replicate a sound as "accurately" as possible with this type of synthesis is akin to finding a needle in a haystack, and may not even be possible. Besides the fact that it is impossible to quantitatively and exactly characterize the tone quality of an instrument's sound, even finding "correct" carrier and modulator frequencies as well as amplitude and modulation index envelopes will never fully describe the sound of an input waveform.

For the purposes of replicating an instrument's timbre in the digital production of music, the most common and accurate method is of course sampling the instrument (i.e. just playing back a recorded version of the original instrument for a particular note). However, it is of particular use to sound producers to use a synthesizer to produce realistic acoustic timbres. With this in mind I set out to create an algorithm for computing the "closest" FM-generated waveform to a recorded or sampled instrument as possible.

The automative software developed in this project was fairly successful in capturing most of the harmonic, timbral characteristics of short single-tone instrument samples from a collection of strings, woodwinds, brass, and percussive instruments. However, what the computer and I were unable to achieve was systematic processing of peak heights and evaluation of the mathematical relationships between frequency peaks and the index of modulation to reverse engineer a simple FM waveform. The parameters were entirely generated through a process of trial and error, with the error comparison relying mostly on

the basic properties of the input waveform rather than the fundamental relations in FM theory. Although I did attempt to automatically reproduce FM-generated sounds by calculating the most likely carrier and modulator frequency positions, it proved difficult and perhaps impossible to pinpoint the right positions based solely on peak frequency positions relative to each other and their magnitudes in the FFT. Therefore I moved to a more empirical solution, finding the least-squares error in each parameter of amplitude, modulation index, and tristimulus timbre values.

A more theory-inclined solution, which could increase the efficiency of the program but compromise the timbral accuracy, would involve calculating the possible carrier and modulator frequency positions as well as the index of modulation by use of the Bessel functions. Knowing the relative weights of the (unreflected) peaks through the FFT, it is possible to calculate the index of modulation by finding intersection points from the Bessel functions of the first kind of order 0 to N, where N is the distance in units of modulator frequency from the carrier frequency. However, it is too risky to rely on the FFT being a perfect representative of the timbre of the sound. Furthermore, in the case that the spectrum is harmonic and negative frequencies reflect onto the existing peaks, it will be much more difficult to extract the expected frequencies from the peak magnitudes because the Bessel functions only dictate the magnitudes of the unreflected peaks relative to the carrier peak. My previous attempts to characterize the spectra on the basis of harmonic vs. inharmonic and finding possible carrier/modulator frequencies was fairly successful, until I found that the possibility of reflected sidebands from the negative frequency domain complicated the process. Moreover, the various inaccuracies associated with the WAV recording and the FFT made it less appealing to rely on human computations and more appealing to automate the process through rapid “prototyping” of an FM model.

E - References

[1] Chowning, John. *The Synthesis of Complex Audio Spectra by Means of Frequency Modulation,* J. Audio Engineering Society, 21 (7), 1973. 526-534.

[2] Bate, John A. *The Effect of Modulator Phase on Timbres in FM Synthesis. Computer Music Journal, Vol. 14, No. 3 (Autumn, 1990), pp. 38-45.*

[3] Austin, David. *No Static at All: Frequency modulation and music synthesis*. American Mathematical Society. 2017.

[4] Pollard, H.F., and E. V. Jansson (1982). *A Trisitmulus Method for the Specification of Musical Timbre*. *Acustica* 51:162-71.

[5] J.R. Carson, "Notes on the theory of modulation", Proc. IRE, vol. 10, no. 1 (Feb. 1922), pp. 57-64.

Appendix A