# ANALYSIS AND RESYNTHESIS OF THE CORNELL CHIMES

**A Design Project Report**

**Presented to the School of Electrical and Computer Engineering of Cornell University**

**In Partial Fulfillment of the Requirements for the Degree of**

**Master of Engineering, Electrical and Computer Engineering**

**Submitted by**

**Alexander Koenigsberger**

**MEng Field Advisor: Hunter Adams**

**MEng Outside Advisor: Bruce Land**

**Degree Date: January 2022**

# ABSTRACT

**Master of Engineering Program**

**School of Electrical and Computer Engineering**

**Cornell University**

**Design Project Report**

**Project Title:**

Analysis and Resynthesis of the Cornell Chimes

**Author:**

Alexander Koenigsberger

**Abstract:**
A bell produces a complicated sound. Its power spectrum can include dozens of irregularly-spaced modes, each of which decays at a different rate. As a result, they are difficult to synthesize and model using software. Bell-like sounds can be created using FM synthesis, but creating a parameterizable model of realistic bell music requires more sophistication. This was the objective for this project. To do so, each of the bell's modes was modeled by a decaying sinusoid, and each sinusoid was parameterized by its frequency, initial amplitude, and decay time. This method was validated against the bells of the Cornell Chimes and produced a synthesized bell sound realistic enough to fool Cornell students in a blind listening test.

# EXECUTIVE SUMMARY

This project consisted of two phases: searching for an adequate method to synthesize the sound of a bell, and then implementing that method to produce a resynthesis of the Cornell Chimes. Much was learned in each of the two phases.

This project required a substantial literature survey for bell synthesis methods. As a consequence of this survey, the project scope was narrowed. Rather than aiming to produce a general-purpose synthesizer for recreating the sound of any bell, the team endeavored to design a method for synthesis of the Cornell Chimes specifically. With the bells identified, the team next experimented with a few strategies for sound synthesis. Several known sound synthesis algorithms were explored and implemented. Direct Digital Synthesis with Frequency Modulation produced a nice bell-like sound, but it lacked resemblance to a real bell tower instrument. The literature survey uncovered two bell synthesis projects that provided valuable insight into alternative methods. In particular, a project from the Center of Computer Research in Music and Acoustics (CCRMA) at Stanford guided the team's efforts in the second semester.

After substantial research and experimentation in the first semester, the second semester focused on implementing the bell synthesis method to which the research and experimentation had converged. In particular, the team implemented a modal extraction algorithm in MATLAB and then used the acquired mode frequencies, amplitudes, and decay rates to resynthesize the bell. This method was informed by the research of Kermit Canfield-Dafilou at the CCRMA.

Ultimately, this project successfully resynthesized two bells from the Cornell Chimes. The simulated bells, modeled by only a few dozen parameters, were realistic enough to fool the average Cornell student, though not quite realistic enough to fool the Chimes players themselves.

# 1 Defining the Objective

This project was proposed as an open ended exploration of sound synthesis. As the author is an avid player of the Cornell Chimes, a set of 21 bells housed atop the university's McGraw Tower, the sound of bells was quickly chosen. The complex sound structure of a bell was thought to be an appropriate level of difficulty for an MEng project. Due to the rarity of bell synthesis projects and the author's inexperience, there was no certain approach in mind.

The initial idea for the project was to create a parameterizable synthesizer that could be adjusted to replicate the sound of any bell tower in the world. The earlier stages of this project, then, entailed identifying possible methods for bell synthesis that include tunable parameters. It was quickly discovered, however, that this was quite a naive and ambitious proposal. In the first semester, the team researched sound synthesis techniques, experimented with those techniques, and revised project objectives as a consequence of that research and experimentation. Thus, the next section of this report will take on a bit of a more narrative form.

Upon the discovery of a recent bell synthesis project, which shall be introduced later in this report, the minimum objective for this project was determined to be resynthesis of the Cornell Chimes. The resynthesis of an existing bell is a worthy goal because it creates a parameterized model that can be modified to apply interesting effects to the original, and it also seems quite impressive that a sound as complex as that of a bell can be created from just a few numbers. Additionally, it makes testing easier because the original bell sound is accessible.

# 2 Method Exploration and Literature Search

The vast majority of this project's first semester was spent becoming familiar with well-known sound synthesis methods and assessing their effectiveness at synthesizing bell sounds, especially ones meant to sound like a specific tower bell. The team encountered a couple of classic papers on synthesis methods as well as emerging research on bell synthesis. This section will discuss each method tried in approximately the order in which it was encountered and experimented with.

## 2.1 Direct Digital Synthesis with Frequency Modulation

The first stop in the exploration of sound synthesis methods was to use Direct Digital Synthesis (DDS) with frequency modulation (FM). Having just learned about this algorithm in ECE 4760 - Digital Systems Design Using Microcontrollers - it seemed a good starting point [1]. In the first assignment for the class, an extremely accurate imitation of birdsongs was implemented using DDS with only one oscillator. The question was raised of what could be achieved by extending this algorithm or applying the FM cleverly enough. A search of ideas for modifications led to the original 1973 paper on FM Synthesis by John F. Chowning [2]. The notion of a carrier wave with a modulator wave was brought up, and so was the idea to parameterize such a system with the ratio of carrier to modulator frequency. This ratio is key when creating a sound, as small differences can greatly affect the result. Through experimentation, we found that using a ratio of 1.42 and an exponentially decaying amplitude

envelope makes a sound very much like struck metal or glass. This was quite an exciting result and the algorithm is fun to experiment with, but, while the one-carrier-one-modulator system produced a nice sound with such simplicity, there seem to be far too few degrees of freedom with this method for achieving a targeted bell sound.

### 2.2 Helsinki University of Technology Handbell Resynthesis Project

Among the first pieces of research found was a pair of papers from the Helsinki University of Technology. They appear to be from the same project in 2002, which entails accurate modeling and synthesis of handbells [3][4]. One of these papers is titled "Making of a Computer Carillon," which sounded especially exciting seeing how that is almost exactly what the vision for this project was at the time. However, both papers seem to focus mostly on how the authors used ARMA Modeling and inharmonic digital waveguides - neither of which were familiar and both of which looked too far above the team's skill level - to resynthesize handbells. In addition, the use of a program called BlockCompiler, which is now all but obsolete, speaks to its age. Another small part of the reason this project was dismissed is that it focuses on recreating a single bell, and at the time this was found the focus was still a general-purpose bell synthesizer. Lastly, we felt that the project had insufficient demonstrations of its results. Both papers linked to the same site that contained a recording of one handbell and also its resynthesis. The results are great, but they only show one bell. The titular "Computer Carillon" was unfortunately not found, which was a bit disheartening. A feeling of technical unpreparedness and concerns over this project's relevance led us to pass over it in the search for the right bell synthesis algorithm.

### 2.3 Karplus-Strong

The Karplus-Strong method is popular for the synthesis of string and percussive instruments [5]. While the bell is neither a string instrument nor a percussive instrument with a membrane, this algorithm seemed worth a try. An original implementation following a lab prompt from Julius Orion Smith's open-source Stanford course was attempted, but only worked if my excitation signal had an exponential decay envelope, which should not be the case. Instead of spending time making an implementation from scratch, the Karplus-Strong implementations from Bruce Land's "DSP for GCC" page were given to experiment with [6]. We eventually decided that 1-D synthesis was not worth spending too much time on.

### 2.4 Physical Synthesis

Bruce Land's embedded C-language implementation of sound synthesis using the finite-difference solution to the wave equation into MATLAB [7]. Not very much was done with this piece of code, for it was at this point where we decided that 1-D synthesis was probably not relevant to pursue. Regardless, it was still interesting to learn about this form of synthesis.

**2.4 Stanford CCRMA Modal Synthesis**

In the later part of the semester, a pair of recent projects from Stanford University's Center for Computer Research in Music and Acoustics (CCRMA) were found [8][9]. The first of these describes a method for modeling the sound of a carillon as a combination of modes, which are decaying sinusoids parameterized by their frequency, initial amplitude, and decay time. They describe the way they obtained each of these attributes, which is approximately as follows:

1. Find the peaks of the frequency domain signal to identify mode frequencies.
2. Apply a bandpass filter around each of the frequencies found in the previous step and use an RMS energy envelope to estimate the decay constant.
3. Create a matrix containing the decay rate of each partial, and use least squares to find the complex amplitudes.

This method was actually applied on recordings of each bell of the Lurie Carillon at the University of Michigan, which are publicly available. The acquired parameters are made available by the researchers as well. With these modal parameters, the paper describes creative uses for them such as retuning the harmonics as well as synthesizing the sound of bells that the physical instrument does not have. This sounded extremely promising a guide for this MEng project, for the Cornell Chimes is missing one of the bells in its range: the low C#. This project demonstrates that the synthesis of real tower bells is achievable.

The other paper in the project outlines a sophisticated method of recording the bells of Stanford's carillon and creating a detailed modal model of the sound similar to the other paper. It also goes so far as to model the belfry reverberation and bell-clapper interaction at different clapper velocities. This led to an accurate parameterized model of the Stanford carillon. The means to take measurements on the Cornell Chimes in a similar way are unobtainable, as it is unclear what audio equipment is required and, more importantly, the bells of the Cornell Chimes are housed in a cage that prevents access to their clappers. While interesting, this particular paper lies outside the scope of this MEng project so it was dismissed as a model for subsequent work in favor of its preceding paper. Regardless, a promising guide was found through these projects.

*Interpolation Proof of Concept*

The use of modal parameters for interpolation especially stood out because of its implications for the Cornell Chimes. The Chimes range from C4 to A5 and contain every note in between except for the C#4.[1] With the knowledge obtained from the CCRMA project, being able to interpolate the sound of the missing low C# sounded like a worthy goal.

Before entertaining that idea too much, a simple interpolation method was attempted using the modal parameters obtained from the Lurie Carillon. We wrote a MATLAB script that creates the decaying sinusoids out of these parameters and adds them together to create a tone

[1] These are the pitches written in the music. Bell instruments are often transposed, and the Cornell Chimes' true pitch is approximately a half-step higher than the written note. So the true range of the bells' strike tones is C#4 to A#5. For simplicity, the notes will be referred to as they are written, not as they sound.

using Equation 1. As expected, the results sounded extremely similar to the recordings of the Lurie Carillon. Most interestingly, simply averaging the parameters of corresponding modes across bells that are a whole step apart results in a new sound that is almost exactly like the semitone in between them. Because modal parameters were gathered for all the bells and the Lurie Carillon is chromatic, this could be tested rather easily. This demonstrated that resynthesizing the low C# was quite possible as long as the parameters for the low C and D were collected.

$$y(t) = \sum_{i=1}^{\#\ of\ modes} A_i * sin(2\pi f_i t) * exp(-t/\tau_i)$$

Equation 1: The model for the resynthesized bell sound, where $A$ is initial amplitude, $f$ is the frequency, and $\tau$ is the decay time constant. These are the three estimated parameters.

## 3 Arrival at a New Goal

The idea for this project at the beginning of the first semester was rather lofty, but after a thorough exploration of signal processing methods and previous bell synthesis project, we had arrived at a new set of goals. Given that interpolations, and likely extrapolations, were possible, the ideal minimum stopping point at the end of semester two was to extend the 21-bell Cornell Chimes into a two-octave carillon by interpolating the low C# and extrapolating the upper range to C6. If that is completed with enough time, a next activity would be to build an embedded system that can play the resynthesized sounds in real time, for the CCRMA papers mention that their systems can be implemented in real time but do not say how. Additionally, another bonus goal would be to manipulate the modal parameters in some other creative way such as retuning harmonics. However, it was established that completing the resynthesis of one bell at minimum would be a satisfactory stopping point for the MEng project. This ended up having to be the case because finalizing the method for achieving passable resynthesis took much longer than anticipated, and so did the modal extraction itself because decay time and initial amplitude required estimations from plots done by hand.

## 4 Design of Modal Parameter Estimation and Resynthesis Method

The second semester was focused on implementing a variation of the modal parameter estimation method and gathering as much modal data as time permitted. The algorithm was implemented in MATLAB. Following closely to the method described in [8], the algorithm has three parts: frequency estimation, decay time estimation, and initial amplitude estimation.

### 4.1 Frequency Estimation

The frequencies of the modes present in the bell sound are found at the peaks of the frequency domain signal. In other words, find where there are peaks on the output of an FFT. This is conducted on a recording of the bells.

To preprocess the signal, a high pass filter is applied at about half an octave before the hum tone (an octave below the fundamental, or strike tone) to discard unwanted spikes at low frequency. Then, an FFT is taken starting at about 10ms after the bell is struck. This prevents noise from any transients during the striking of the bell. The CCRMA paper employed this technique, which is where the idea came from. The length of the FFT is about a half-second, or just over 20,000 samples. This must be kept relatively short and taken close to the beginning of the signal so that the faster-decaying modes are detected.

*Peak Picking for Frequency Estimation*

To find the locations of all of the peaks, the `findpeaks` function in MATLAB is used. It is able to return the locations of any peaks it finds, and contains optional parameters so that it only returns locations of peaks that meet certain criteria. The 'MinPeakDistance' does not choose peaks that are within the given distance from another chosen peak. The 'MinPeakHeight' option only chooses peaks whose value at the peak is above the given value. The 'MinPeakProminence' option only chooses peaks that stand a certain height above its highest nearest adjacent valley. The peak-picking was conducted on the logarithm of the absolute value of the FFT result. For an unknown reason, the noise floor of this plot decreases at higher frequencies. Because of this, simply setting a 'MinPeakHeight' will not capture all of the modes, so a decreased height but increased prominence filter will detect more peaks at higher frequencies.

The peak picking was not completely perfect. Sometimes it catches a peak or two that looks like it is not great enough to be above the noise, so the peak outputs are verified by graphing the outputs with the peaks marked and checking each visually.

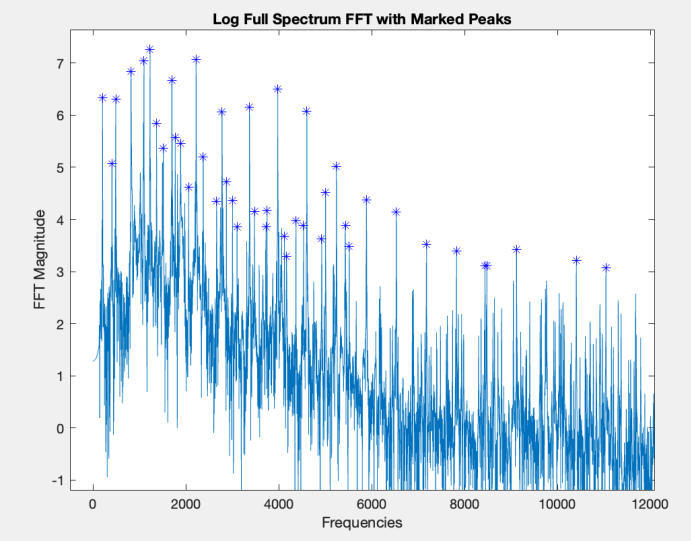


Figure 1: Peak-Picking results for the low F with min Distance of 10, min Height of 3, and min Prominence of 3.

### 4.2 Decay Time Estimation

The decay time can be estimated in one of two ways:

- Taking a series of short-time FFTs across the duration of the sample and plotting the intensity of any certain mode's frequency bin over time.
- Running a narrow band-pass filter around the mode's frequency and computing the 10ms root-mean-squared average of the band-passed time domain signal.

With either of the plots generated above, the slope of the natural log of the resulting signal is the decay time constant. Both processes yield similar results, which is expected because of Parseval's Theorem. Since the log-linear region varies in length across modes - in other words, the relevant region where the mode is present before decaying away - the estimation of the slope must be done by visual inspection of the plot. No way to automate this process was conceived.

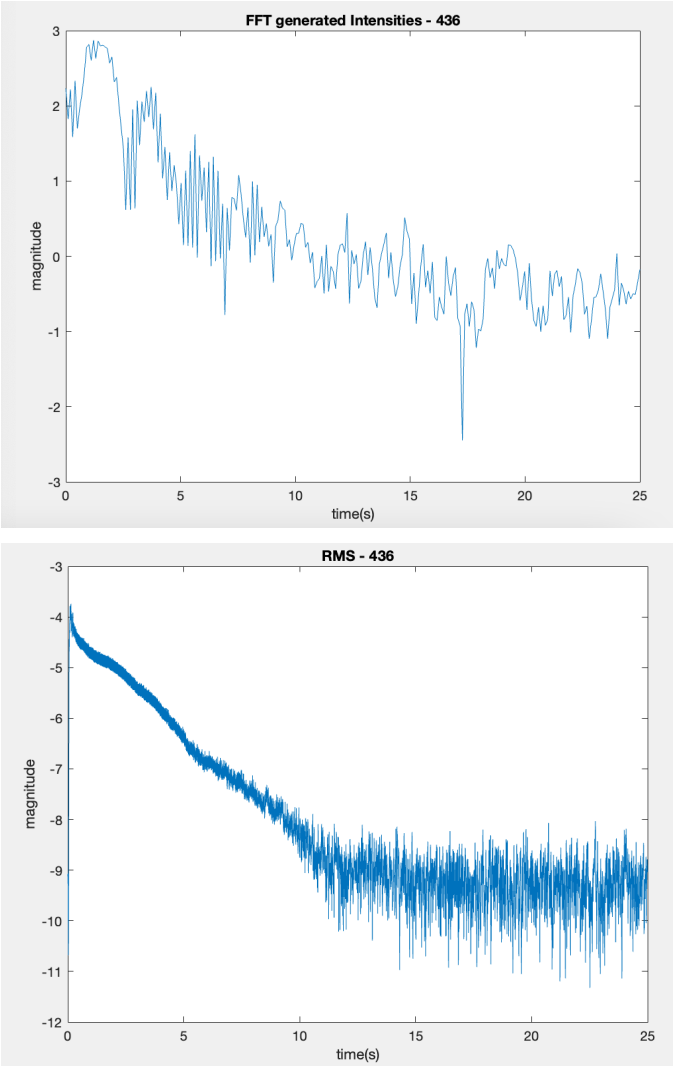


Figure 2: Outputs for Decay Time estimation for the minor 3rd overtone of low F, 436 Hz. Top plot is with sliding FFTs. Bottom plot is with RMS.

### 4.3 Initial Amplitude Estimation

The initial amplitude of the mode was estimated by simply reading the initial amplitude from the plot of the band-passed time domain signal. This is quite a simple method, which on the surface appears as though it should work. It strays from the method of the CCRMA researchers and comments on this will be made in a later section.

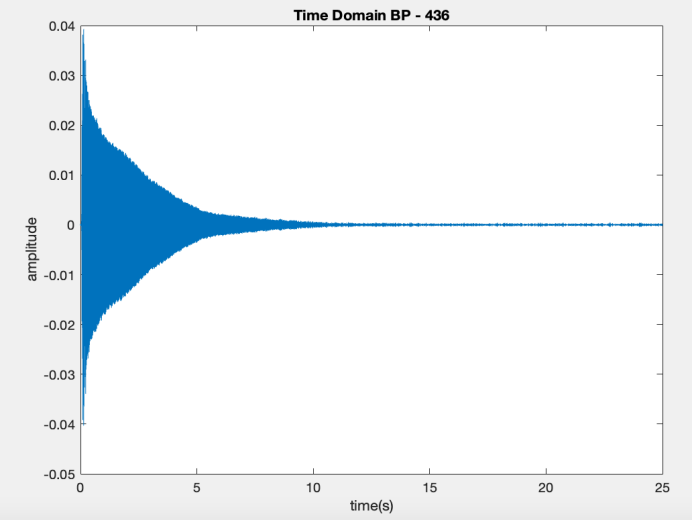


Figure 3: Low F band-passed at 436 Hz.

### 4.4 Resynthesis

The resynthesis functionality used was written when experimenting with the dataset of modes for the Lurie Carillon. A helper function was written that generates a sinusoid with a given frequency, initial amplitude, and decay time as in Equation 1. The modal parameters were presented in CSV format, so the parameters for each mode could be used to generate a sinusoid and add it to an array that holds the resulting tone. For a clearer explanation, a quick pseudocode is shown below:

```
for each mode {
      freq, amp, decay = read_from_csv()
      mode = create_decaying_sinusoid(freq, amp, decay)
      tone = tone + mode
}
```

To be able to recycle this code from the Lurie experiments, the gathered modal parameters were placed into a .xlsx file that could be read using the `readmatrix` function in MATLAB.

In addition, white noise was added to the resynthesis and made such that the resulting bell tone starts about once second into the resulting audio in order to mimic an actual recording sample.

# 5 Results

The evaluation of results for this project is quite interesting because it is not measured with numerical data, but rather consensus of subjective listening experience. Once refined after the first semester, the original goal of this project was to create a convincing resynthesis of the Cornell Chimes that is as close as possible to indistinguishable from the real recording. In addition, a further goal was to interpolate the missing low C# from the Cornell Chimes. Even further, it was hoped that there would be time to design and construct an embedded system that could play back the bell resynthesis in real time. At the end of the year, however, the stretch goals were not achieved because the modal parameter extraction algorithm and the parameter acquisition took far longer than expected. Resynthesis of the low F and low G of the Cornell chimes was achieved, but not quite to the originally desired quality in terms of similarity to the original. However, according to most listeners, the resynthesized bell sounds very much like a real bell, but perhaps not the exact Cornell Chimes. Further discussion of the results follows below.

## 5.1 Resynthesis Quality

Several resynthesis experiments were conducted with varying degrees of success. Different numbers of modes were tried, and it was surprisingly found that adding more modes does not always yield a better result. This is quite contrary to the Lurie dataset, where increasing the number of modes included in the resynthesis always increased the quality of the sound. Also, adding soft random noise to the resynthesis makes it sound much more realistic because it sounds like the noise detected by microphones in a quiet environment.

The first resynthesis implemented was the low G. An initial set of peak-picking parameters without prominence was used to find 16 modes, which seems very low compared to the Lurie Carillon bell of the same pitch. Due to the noise floor of the frequency domain signal decreasing at higher frequencies, modes higher than about 4kHz were not detected at all, so the parameters were adjusted to lower the height and also consider prominence, which led to 40 modes being detected, including more mid-range frequencies and higher frequencies. Surprisingly, the 16-mode synthesis sounded more similar to the Cornell Chimes. The 40-mode resynthesis seemed like it had too many high frequencies that were too prominent, as it sounded excessively metallic or sparkly in the beginning. This, however, makes it sound more complex.

*Three-Chime Experiment*

An experiment was run on several groups of listeners. The 16-mode resynthesis, the original recording, and the 40-mode resynthesis were played, and then listeners were told only one of the three was the sound of the real bell. When played from a laptop speaker for a room of about twenty or so people, no one could easily identify which was the real recording. The most common guess was that the 40-mode recording was the real bell, likely because it sounded more different from the other two. The tones were played for a couple of the Cornell chimesmasters,

and they were able to quickly identify the sound of the real recording, although they asked for a few repeats of the tones before making the choice.

Interestingly, one listener said that the 40-mode synthesis sounded the most pleasant and bell-like. While a bit different from the other two tones, it does indeed still sound like a bell. With that said, although a near-perfect resynthesis of the chimes was not achieved, the sound of a bell in general was convincingly created using only a few dozen numbers as parameters, which is interesting in itself. The low F was also resynthesized in a similar manner with similar results.

*Reasons for the Results*

The question arises regarding why more modes makes the resynthesis sound more dissimilar. It could well be in the modal parameter estimation algorithm itself, as it is a variant of the CCRMA method that works very well. The CCRMA method takes complex numbers into account when calculating the amplitudes, whereas this project does not. Implementing the CCRMA method for amplitudes shall be considered future work. It also may have to do with the recording hardware. The low G from an iPhone produced different amplitude results than when recorded using a more professional microphone.

**5.2 Interpolation**

Since parameters were gathered for the low F and low G, an interpolation of the F# was attempted in the same way as the Lurie Carillon experiment in the Fall, a simple average of each parameter across corresponding modes in ascending order by frequency, but the results were rather disheartening. While the strike tone sounded about correct, the other modes sounded too dissonant and less bell-like. The idea was that the similar bells would have similar modal patterns, but that was not the case. The differences in frequency between corresponding modes varied too much. Whether this is due to an error in the modal parameter estimation or real differences between the bells is unknown.

The hope was that a convincing F# interpolation, which could be compared to an original recording, would indicate that a similarly acquired interpolation of the low C# would be accurate. Since the F# interpolation was not of adequate quality, interpolation of the low C# was not attempted.

**5.3 Stretch Goals**

Initially, the hope was to gather modal parameters for all 21 bells of the Cornell Chimes and use them to conjure new bell sounds and create an embedded system that plays them in real time. However, both the modal extraction algorithm and the gathering of the parameters took far longer than expected. The modal extraction algorithm, while it seems simple in retrospect, turned out to be difficult to polish up. Even more impactful, though, was the length of time it takes to gather the modal parameters of a bell. Since each mode requires visual inspection of multiple plots, they had to be done one-by-one, all by eye. This was a time-consuming process that took at least an hour for each of the bells. Fine-tuning the peak picking parameters, estimating the

parameters, and then noting them, was a long, repetitive process. If each bell takes an hour, and there are 21 bells, that would take well over a student's work week, maybe two. For these reasons, expansion of the Cornell Chimes or the implementation of an embedded system was not attempted.

### 5.4 Summary

The bell resynthesis did not sound quite exactly like the Cornell Chimes, but it definitely sounded like a bell. In other words, it is "a bell" rather than "the bell," which is still interesting. It may point to a potential error in the algorithm, but has also been a test of sorts for the human ear. It seems like it is quite easy to fool, as only 16 modes together can sound like a bell that is probably far more complex than that in reality. Further progress was prevented by unexpected time consumption and unsatisfactory interpolation results, but an interesting overall result was achieved nonetheless.

## 6 Future Work

There are a number of things that can be done from where this project left. Firstly, the quality of the synthesis could be improved so that it is more identical to the Cornell Chimes. Secondly, the remainder of the dataset of modal parameters can be gathered with good quality. From there, one can use the dataset for creative and experimental uses such as synthesizing bells that do not exist with convincing accuracy. A real-time embedded system can be created that plays the sounds back. A computational advantage of modeling bells as a combination of modes is that much less storage space is required, as each bell model requires only a few dozen parameters whereas playing back a recording requires storing many thousands of samples. In addition, interactively modifying the parameters somehow and applying effects to them sounds like an interesting endeavor, but that is a stretch. The purpose of these kinds of projects is educational, but also to produce a creative tool.

## 7 References


[1] https://people.ece.cornell.edu/land/courses/ece4760/PIC32/DDS/DDS.html

[2] JO. M.. Chowning, "The Synthesis of Complex Audio Spectra by Means of Frequency Modulation," J. Audio Eng. Soc., vol. 21, no. 7, pp. 526-534, (1973 September.)

[3] Karjalainen M, Välimäki V and Esquef P A A 2002 Efficient modeling and synthesis of bell-like sounds ¨ DAFx: Proc. Int. Conf. on Digital Audio Effects (Hamburg, Germany, 2002)

[4] Karjalainen M, Välimäki V and Esque ¨ fPAA 2003 Making of a computer carillon Proc. Stockholm Music Acoustics Conf. (Stockholm, Sweden, 2003)

[5] Karplus, Kevin, and Alex Strong. "Digital Synthesis of Plucked-String and Drum Timbres." *Computer Music Journal* 7, no. 2 (1983): 43-55. doi:10.2307/3680062.

[6] https://people.ece.cornell.edu/land/courses/ece4760/Math/avrDSP.htm

[7] https://people.ece.cornell.edu/land/courses/ece4760/PIC32/index_sound_synth.html

[8] Elliot Kermit Canfield-Dafilou and Kurt James Werner, “Modal audio effects: A carillon case study,” in Proceedings of the International Conference on Digital Audio Effects, 2017.

[9] Rau M, Das O, Canfield-Dafilou EK. “Improved carillon synthesis.” In: Proceedings of the 22nd international conference on digital audio effects, Birmingham, UK, September 2–6, 2019; 2019. p. 1–8.