

CONTROLLED CHAOS AND
OTHER SOUND SYNTHESIS TECHNIQUES

A THESIS SUBMITTED TO THE FACULTY
IN CANDIDACY FOR THE DEGREE OF
BACHELOR OF SCIENCE

BY
KIMO JOHNSON

UNIVERSITY OF NEW HAMPSHIRE
DURHAM, NEW HAMPSHIRE

MAY 2000

CONTENTS

INTRODUCTION.....	1
ORGANIZATION OF THE THESIS.....	3
Chapter	
1. BACKGROUND INFORMATION.....	4
Physical Aspects of Sound.....	5
Psychological Aspects of Sound	8
2. CLASSIC SYNTHESIS TECHNIQUES.....	16
Additive Synthesis.....	17
Subtractive Synthesis.....	19
Amplitude Modulation.....	21
Frequency Modulation.....	23
3. SYNTHESIS USING CONTROLLED CHAOS	26
Original Synth.c Program	30
The Library	32
Instrument Modeling.....	34
4. CONCLUSIONS AND FUTURE DIRECTIONS	39
Appendix	
1. CD TRACK LIST	41
WORKS CITED	42

INTRODUCTION

Sound synthesis is a field that bridges the disciplines of music, physics, mathematics and psychoacoustics. Researchers from all of these fields study and use sound synthesis for different reasons. For example, musicians researching sound synthesis are often trying to find interesting sounds for compositional or artistic purposes. They look for sounds with a depth similar to those of natural instruments and their trained ears are often sensitive to sounds that are too artificial. Physicists and mathematicians are curious about how instruments work and the laws governing their behavior. They use sound synthesis to test their hypotheses and models of instruments. Researchers of psychoacoustics, on the other hand, are interested in how the ear and mind interact when we hear a sound. They use sound synthesis to help them examine human perception. This does not mean that musicians are not interested in perception or that psychoacousticians are not interested in how instruments work. On the contrary, when one begins to study a particular aspect of sound synthesis, one finds that questions related to the other fields of study need to be answered as well.

The goal of this thesis was to synthesize instrument sounds using a new synthesis technique derived from work in chaotic communications. My first step towards this goal was to learn classic synthesis techniques that have been both successful and unsuccessful. This information taught me about the problems that arise when trying to synthesize sound. This also put me face to face with the relationship between the physical and psychological interpretations of sound. I conducted several experiments to determine how I perceived certain sounds when I knew the physical components involved. These experiments gave me first hand knowledge about the intricacies of human perception. My interest in music was also important to my

research. I judged the synthesized instrument tones against my experience with natural instruments. While working on this thesis, I found it necessary to research the physics, mathematics and psychoacoustics involved in sound synthesis and I have included my findings in the Background Information chapter. The final step towards my goal was to use the knowledge and experience I had gained with other synthesis techniques to help me synthesize sounds using controlled chaos.

ORGANIZATION OF THE THESIS

The first section of this thesis provides background information about both the physical and psychological aspects of sound. I only discuss those aspects of sound that I feel are essential to sound synthesis. For instance, I do not discuss intensity and loudness of sound or sound location because, although they are important aspects of sound, I did not incorporate them into my synthesis techniques. I also discuss my experiments with perception of sound and the sound files I generated are on the included CD.

The second section of the thesis discusses my experiences with the classic synthesis techniques of additive synthesis, subtractive synthesis, amplitude modulation and frequency modulation. I have recorded examples of each type of synthesis on the CD to reinforce my descriptions of the synthesized sounds.

The third section of the thesis details my work with the synthesis of sound by controlling a chaotic system. I discuss each step I took in order to reach my current point in the research. Some musical examples from the different steps are also on the CD.

In the last section, I discuss the current state of my research and some possible future directions. There are still many options to explore using controlled chaos and I explain some that I am considering.

CHAPTER 1

BACKGROUND INFORMATION

Early composers and researchers of sound synthesis found that although they could make the computer play any waveform they specified, none of these waveforms sounded as interesting as natural instrument sounds. The sounds lacked depth and variability, so the researchers were forced to embrace the mathematical and physical sides of sound synthesis in an attempt to learn the important characteristics of natural sounds.

There were two approaches to researching this problem. The first was to analyze the instrument sound to learn which components were important to the overall effect. This usually involved using a Fourier Transform to look at the harmonic structure of the sound or looking at the signal in the time-domain to determine the proper Attack-Decay-Sustain-Release envelope. With this knowledge, those components could be controlled by different parameters possibly yielding interesting sounds.

Another approach was to learn how an instrument produced sound and try to model that process mathematically. For example, a brass instrument could be modeled by simulating the buzzing of the mouthpiece with a pulse train and the body with a filter bank. This method is more complex than analyzing the sound alone because there is often a strong coupling between the instrument body and the mouthpiece which is not taken into account in the simple model above.¹ There are also many other factors contributing to the sound of an instrument, including the player and the instrument itself. Consequentially, sound analysis usually comes before

1. Wayne Slawson, *Sound Color* (Los Angeles: University of California Press, 1985), 28.

instrument analysis when researching instrument sounds; although, both methods have their advantages and both provide valuable information.

The early pioneers of sound synthesis were also confronted with issues related to perception. A sound can be analyzed in two ways: physically and psychologically. The physical analysis includes the harmonic structure and the time-domain envelope described above. These components are determined to be important to the sound only through psychological analysis. In other words, the ear decides which parts of a sound are essential to its perceptual effect. When producing musical tones on a computer, the psychological effect of the sound is usually more important than the physical parameters describing it and often the relationship between the physics of a sound and its interpretation is unclear. In fact, a researcher named Roger Shepard demonstrated this by creating a sequence of tones that is interpreted differently by different people. Some hear an ascending sequence and others hear a descending sequence.² The field of psychoacoustics tries to understand how humans perceive and process sounds and this knowledge is invaluable to sound synthesis.

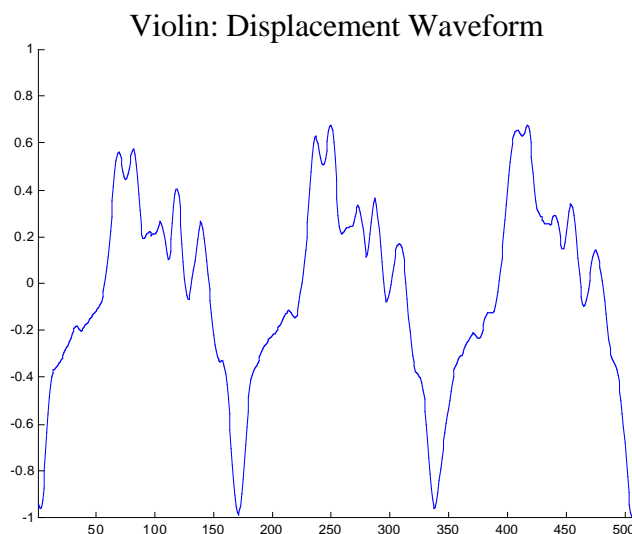
The following sections describe my research in both the physical and psychological aspects of sound.

Physical Aspects of Sound

Sound is a vibration transferred through a medium. In the case of instrument sounds, the vibration comes from the instrument and the medium is the air. From a physical point of view, the vibrations of the instrument's body push the surrounding air molecules, creating patterns of

2. Roger Shepard, Pitch Perception and Measurement, *On Music, Cognition, and Computerized Sound*, ed. Perry R. Cook (Cambridge: The MIT Press, 1999), 159.

condensations and rarefactions that directly relate to the vibrations of the body. The air molecules near the instrument push other air molecules, passing the pattern of vibrations outward from the source in the form of a longitudinal wave. This wave carries the pattern of vibrations through the air at approximately 343 meters per second.³



The displacement of an air molecule due to a waveform can be plotted in time in a graph like the diagram above. The diagram shows the displacement pattern of air molecules when a violin is playing middle C; this sound is Track 1 on the CD. The y-axis of the diagram corresponds to the amplitude of the displacement and the x-axis corresponds to time. The other physical properties of sound, such as period, frequency and phase, can be described in relation to these two dimensions.

The period of a waveform is the amount of time it takes for the waveform to complete a full cycle. In terms of displacement, this is the amount of time it takes for an air molecule to

3. David Halliday, Robert Resnick, and Jearl Walker, *Fundamentals of Physics* (New York: John Wiley & Sons, Inc., 1997), 427.

move forward and backward to end up in the same location.⁴ The diagram of the violin middle C shows three full cycles. The time is expressed in samples along the x-axis and one period is about 170 samples. Since the sample rate of the recording is 44100 samples per second, the period of this waveform is about .0038 seconds.

More often, when discussing musical sounds, we talk about the frequency of a waveform. The frequency of a waveform is how many full cycles of the waveform we will hear in one second. Its value is one divided by the period and it has units of cycles per second, or Hertz (Hz). Using the calculation above for the period, the frequency of the violin middle C is about 263 Hz.

Phase is a measure of how much of the period has been completed at a certain time. It can also be thought of as a value that keeps track of location in the cycle relative to a starting and ending displacement. Its value is related to both period and time and it is measured in degrees. Zero degrees corresponds to a starting displacement. 180 degrees corresponds to the displacement halfway between the starting displacement and the location where the period repeats. At 360 degrees, the period is complete and the displacement is the same as it was at zero degrees.⁵

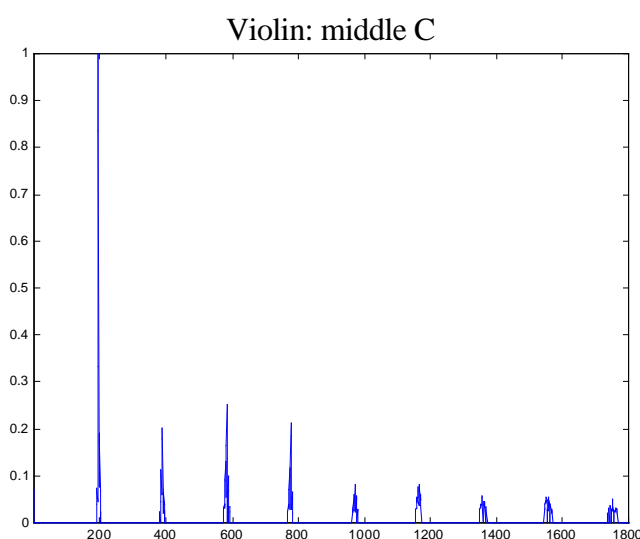
Most sounds we hear are the combination of several frequencies with varying amplitudes and phases. These component frequencies can be divided into two groups: harmonic and nonharmonic partials. Harmonic partials are all the frequencies that are integer multiples of a fundamental frequency. Nonharmonic partials is a general category for all the frequencies present in a sound that are not integer multiples of the fundamental. Musical sounds have both

4. Hubert S. Howe, Jr., *Electronic Music Synthesis* (New York: W. W. Norton & Company Inc., 1975), 4.

5. Ibid., 5.

harmonic and nonharmonic partials, although the nonharmonic partials typically occur during the attack of the note and quickly decrease in amplitude.⁶

It is often useful to view a sound in terms of its frequency components. Fourier's Theorem tells us that any periodic signal can be represented in terms of sine waves that are integer multiples of a fundamental frequency.⁷ For example, the diagram below represents the middle C on a violin in the frequency domain. In this picture, the fundamental is the first peak from the left and it is the strongest in amplitude; although, this is not the case for many instrument tones. The diagram also shows that all of the peaks are the same distance apart and that distance is equal to the frequency of the fundamental.



Psychological Aspects of Sound

The psychological aspects of a sound are the dimensions associated with the perception of sound waves. Two such aspects are pitch and timbre. Pitch is the perceptual measure of

6. Ibid., 7.

7. Erik Spjut, *An Introduction to Signal Processing with Csound*, *The Csound Book*, ed. Richard Boulanger (Cambridge: The MIT Press, 2000), 400.

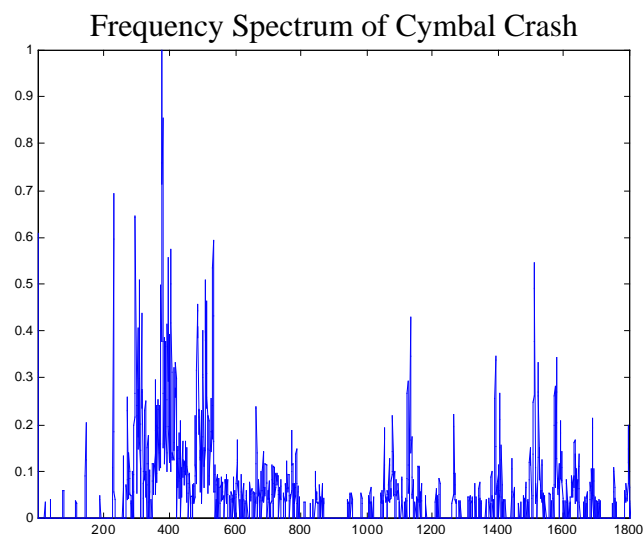
frequency and is relatively simple to understand when compared to timbre. Timbre is often defined as the quality of a sound that allows it to be distinguished from other sounds of the same pitch.⁸

Pitch corresponds closely to the physical measure of frequency. All frequencies, however, are not perceived as pitches. Most people hear frequencies as pitches from around 20 cycles per second to 16,000 cycles per second. Below this limit, frequencies are heard as individual pulses and frequencies above this limit are inaudible. This range of frequencies depends on the individual and usually gets smaller with age.⁹

In musical tones, as discussed above, there are often many frequencies present. Pitch is perceived when the frequencies in the sound are integer multiples of a fundamental frequency. For example, compare the frequency spectrum of the violin middle C in the section above, to the frequency spectrum of the cymbal crash below. The different frequency components of the violin note appear to be evenly spaced at integer multiples of the fundamental, whereas, the frequency components of the cymbal crash are not. We perceive the multiple frequencies in the violin note as a single pitch. The multiple frequencies present in the cymbal crash are not perceived as a single pitch. Tracks 1 and 2 on the CD are the violin middle C and the cymbal crash.

8. Howe, 23.

9. Ibid., 18.



An interesting fact is that the fundamental frequency does not need to be present in the musical tone for pitch to be perceived. As long as three sequential harmonics of a fundamental are present, the fundamental will be perceived as the pitch of a sound.¹⁰ This fact surprised me so I created several sounds that were combinations of sine waves at different frequencies. These sounds are Track 3 on the CD. The first sound and last sounds are sine waves at 110 Hz; these serve as reference pitches. The second sound is a combination of sine waves at 330 Hz, 550 Hz and 880 Hz. These are all integer multiples of the fundamental 110 Hz. When I listen to this combination, it does not have the same pitch as the reference pitch. I hear the individual component frequencies separately. The third sound is a combination of sine waves at 220 Hz, 330 Hz and 440 Hz. These are sequential integer multiples of 110. I hear this sound as having the same pitch as the reference pitch although the reference pitch is not present in the sound. This is a good demonstration of pitch perception. A fundamental pitch is perceived if there are at least three sequential harmonics of that fundamental present in a sound.

10. John Pierce, *Introduction to Pitch Perception, On Music, Cognition, and Computerized Sound*, 59.

One aspect of pitch particularly important to music is the perceived similarity of pitches in a 2:1 ratio. This ratio is called an octave and is important to music because pitches an octave apart are considered to be the same note in a different register. For example, the note A corresponds to pitches with fundamentals at 55 Hz, 110 Hz, 220 Hz, 440 Hz, 880 Hz and so on. The octave is the basis for the equal tempered tuning system, a fundamental part of Western music.¹¹

Another psychological dimension of a musical sound is its timbre. This is the quality of a sound that allows a listener to distinguish it from other sounds of the same pitch. In other words, timbre associates a sound to the source that produced the sound.¹² The perception of timbre is more complicated than pitch but two aspects of the sound, related to the spectrum, seem to be important. The first is distribution of the harmonics and their relative strengths.¹³ The second aspect deals with regions of strong amplitudes called formants.¹⁴

The overtone structure of a sound is related to its perceived timbre. For example, timbre tends to get brighter and harsher as the higher partials are increased in amplitude. An overtone structure with only odd harmonics will sound hollow in comparison to a sound with all the harmonics. To demonstrate these basic observations, I created three sounds with different overtone structures and recorded them to Track 4 on the CD. The first sound is a fundamental and nine harmonics with the relative amplitudes decreasing from one harmonic to the next highest harmonic. The second sound is the same as the first, only the amplitudes of the harmonics are increasing. Since the upper harmonics are stronger in the second sound, its timbre

11. Howe, 19.

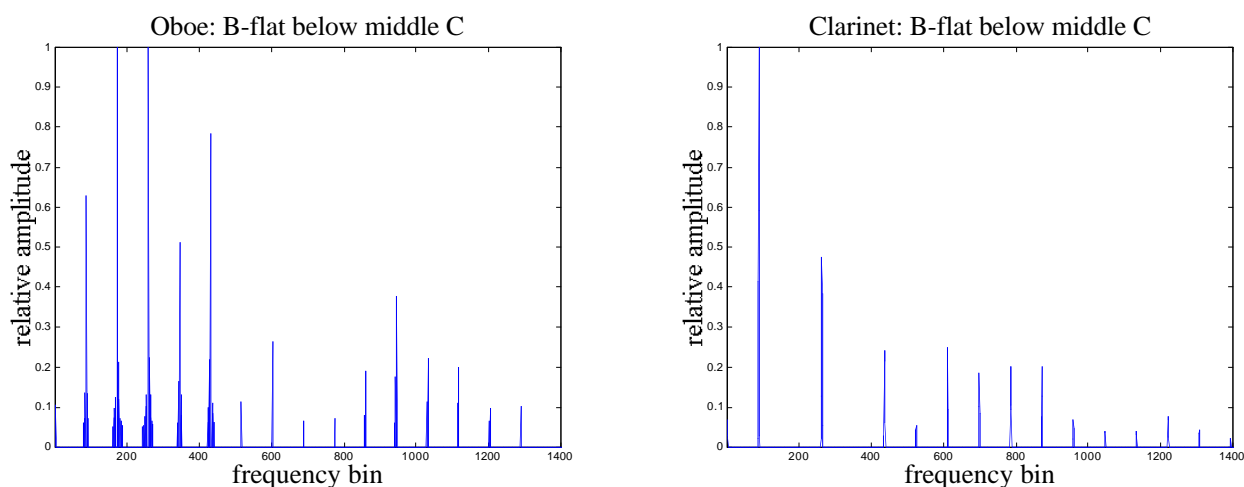
12. Ibid., 23.

13. Ibid.

14. Max Matthews, *Introduction to Timbre, On Music, Cognition, and Computerized Sound*, 82.

is brighter than that of the first sound. The third sound has nine harmonics with equal amplitude and the final sound has only the odd harmonics of the third sound. Since only the odd harmonics are present in the last sound, its timbre is more hollow than that of the third sound. These general observations can also be applied to the overtone structure of instrument sounds.

The overtone structure, or spectrum, of a sound can be found by taking the Fourier Transform of the signal. The Fourier Transforms of the same pitch on two different instruments are plotted below. Using the general observations above, the oboe should sound harsher than the clarinet because it has relatively strong upper harmonics. The clarinet should sound more hollow in comparison because its spectrum is missing the first two even harmonics. Track 5 on the CD contains the oboe B-flat followed by the clarinet B-flat.



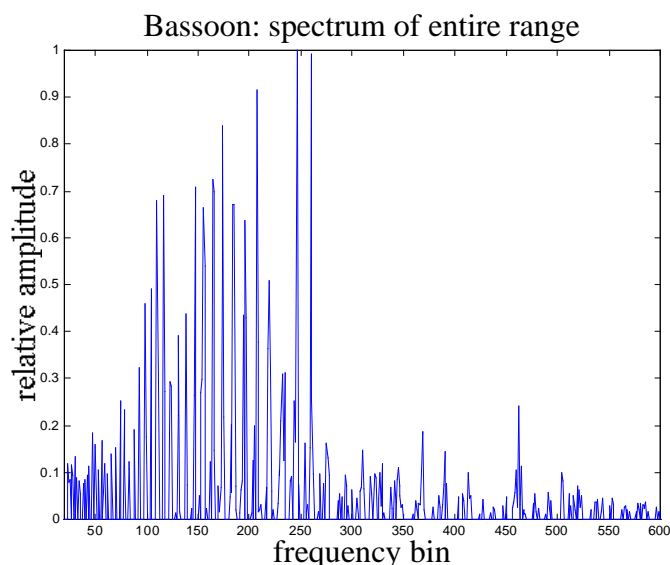
The plots above show the amplitudes for different frequency components of the sound. After looking at these plots, several questions arise. Does this pattern of harmonics remain the same from note to note? If this pattern of harmonics happens at a higher frequency, will the timbre be the same?

The answers to these questions are mixed. To research the second question, I created three square waves at frequencies of 110 Hz, 165 Hz and 220 Hz. The waves all have the same

pattern of harmonics; the fundamental has an amplitude of 1 and the n^{th} harmonic has a relative amplitude of $1/n$. These three square waves are Track 6 on the CD. To my ears, these waveforms all have the same timbre. I then tried the same technique using a pattern of harmonics derived from a bassoon sound. I made three notes with this pattern of harmonics: the first at 110 Hz, the second at 165 Hz and the third at 220 Hz. I then compared these to the actual sound of a bassoon playing those pitches. This is Track 7 on the CD. The first sound is the synthesized bassoon and the second sound is the real bassoon. The 110 and 165 Hz notes sound similar to the actual bassoon but the 220 Hz note sounds pinched in comparison. The further away from the frequency of the original note, the less the synthesized note sounds like a bassoon.

The property of a bassoon's timbre that identifies it as a bassoon is its formant structure. A formant is a fixed frequency area in which the loudest partials of a tone occur.¹⁵ An instrument may have several formants so they are collectively called the formant structure of the instrument. The formant structure affects different partials of different notes thus changing the overtone structure from note to note. We perceive notes from the same formant structure as belonging to the same instrument so these notes have the same timbre. Below is a plot of the spectrums of every note in the bassoon's normal range, almost three octaves of notes. This shows where the bassoon has strong and weak harmonics. Around the 250th frequency bin, the bassoon always has a strong harmonic, whereas, harmonics near the 300th frequency bin are always weak in comparison. This suggests that the filtering action of the body of the bassoon is independent of the harmonics produced by the double reed. Experiments involving the formant structure of the bassoon are detailed in the section on subtractive synthesis.

15. Howe, 24.

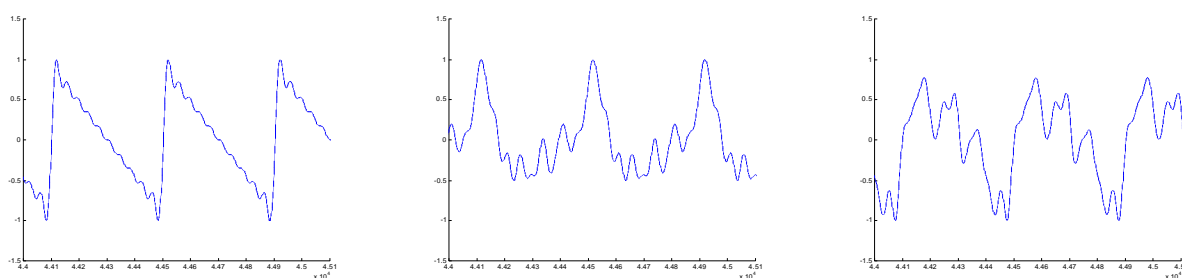


So far I have discussed overtone and formant structure as being important aspects of timbre. If these were the only important elements of an instrument sound then we could synthesize these sounds by creating waveforms with all the harmonics at the correct amplitudes. The first two notes of Track 7 suggest that there is something more to an instrument sound than the amplitudes of its harmonics. Both notes have the same overtone structure but the synthesized sound sounds lifeless compared to the real sound.

There are several reasons why synthesizing sounds by copying only the overtone structure fails to reproduce convincing instrument sounds. The first is the assumption that instrument tones are made up of only harmonic partials. This certainly is not true for the attack of a sound. For example, in a trumpet note, many nonharmonic partials are present for a short duration at the beginning of the note. These are very important to our perception of the whole note. To demonstrate the importance of attacks, I took recordings of 5 instruments and a human voice and removed the attacks from the notes. This experiment can be found on Track 8 on the CD. The sounds in order are piano, oboe, trumpet, clarinet, voice and violin. Although, I can distinguish one instrument from another, they are difficult to identify without the attacks.

The second reason why attempts to synthesize a sound from its spectrum have failed is that the Fourier Transform of a sound gives the average of the amplitudes of the harmonics over the length of the signal. As a result, variations in the amplitudes of the harmonics will not show up in the spectrum.¹⁶ A simple way to regain some of the lost variations is to divide the signal into pieces and find the spectrum of the individual pieces. The trade off is frequency resolution.

A question that often arises when trying to synthesize instrument sounds is the importance of phase. As an experiment, I created some waveforms made from the first ten harmonics of a sawtooth wave. The phases of the harmonics were chosen to be different for each waveform and the resulting time-domain plots look very different.



Each waveform was played back at three different frequencies: 110 Hz, 220 Hz and 440 Hz. On Track 9, you will hear the three waveforms above from left to right at each of the frequencies. To my ears, the second waveform sounds a little muffled and the first and third sound identical. The second waveform still has the same timbre as the others. It seems reasonable to conclude that phase has little effect on the perceived timbre of a sound.¹⁷

16. Howe, 23.

17. John Pierce, *Sound Waves and Sine Waves, On Music, Cognition, and Computerized Sound*, 47.

CHAPTER 2

CLASSIC SYNTHESIS TECHNIQUES

This chapter discusses my experiences with the classic frequency-domain synthesis techniques. Frequency-domain techniques can be divided into two subcategories: linear and nonlinear. Linear techniques do not generate any frequencies that were not in the input signals; whereas, nonlinear techniques can generate new frequencies. Two important linear techniques are additive synthesis and subtractive synthesis. The most common nonlinear techniques are frequency modulation and amplitude modulation.¹⁸ The following sections detail my experiences with these techniques.

Frequency-domain synthesis is based on Fourier's Theorem that any periodic signal can be expressed as a sum of sine waves at different frequencies, each with its own amplitude and phase. As discussed above, phase seems to make little difference in the resulting timbre so it is usually left out of the synthesis techniques.

Two important linear frequency-domain techniques are additive synthesis and subtractive synthesis. Additive synthesis starts with simple sine waves and sums them at different amplitudes to create a spectrum similar to the instrument being modeled.¹⁹ Subtractive synthesis usually starts with noise or waveforms with many harmonics and then shapes the waveforms to create the necessary spectrum.²⁰ These techniques are often adequate for sounds where the harmonics do not vary over time. Unfortunately, the harmonics of most instrument tones vary considerably, so additive and subtractive synthesis techniques often sound lifeless.

18. Rajmil Fischman, "A Survey of Classic Synthesis Techniques in Csound," *The Csound Book*, 223.

19. Ibid., 224.

20. Ibid., 230-1.

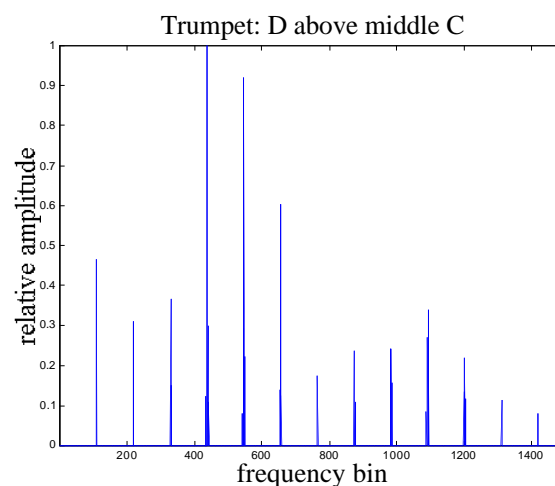
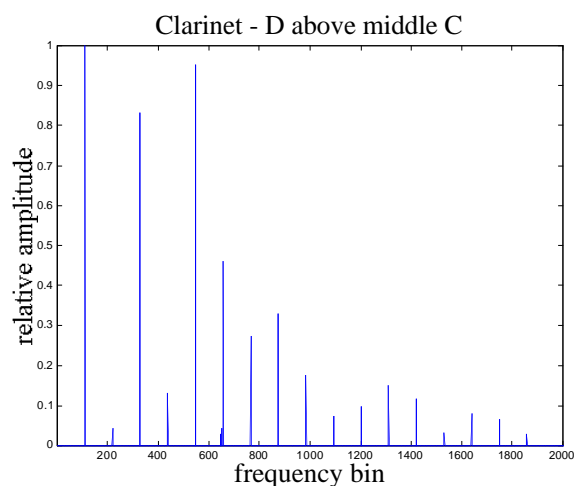
Nonlinear techniques can easily produce dynamic spectra and sound more realistic than linear techniques. Two classic nonlinear techniques are amplitude modulation and frequency modulation. Both techniques use a signal, called the modulator, to change another signal, called the carrier, over time. In amplitude modulation, the modulator changes the amplitude of the carrier over time.²¹ In frequency modulation, the modulator changes the frequency of the carrier over time.²² Both techniques are fairly simple to implement and can produce decent results from a small amount of information.

Additive Synthesis

Additive synthesis combines sine waves of different frequencies and amplitudes to create a sound. This is the most basic type of synthesis, but is useful for demonstrating many issues relating to timbre.

21. Ibid., 237.

22. Ibid., 249-50.



My first experiment with additive synthesis was to look at the FFT of instrument sounds and try to recreate the sounds by summing sine waves at the appropriate amplitudes. Above are the FFTs of the same note on two different instruments. I created waveforms that were sums of sine waves with the amplitudes set to values from the FFT. I then multiplied each signal in the time domain by an envelope function to simulate the attack, sustain, decay and release of the note. The clarinet sounds are Track 10 on the CD and the trumpet sounds are Track 11. In both tracks, the real instrument sound is first and the synthesized instrument is second. For the clarinet, the resulting sound is close to the original. The differences I notice are that the upper harmonics are more noticeable in the synthesized sound and it is too steady. For the trumpet, on the other hand, the synthesized sound is less convincing. It is close for the sustained portion of the note but lacks the sharpness of the attack. The real sound also has a wavering quality that the synthesized tone cannot have without adding more parameters.

More realistic sounds can be obtained by analyzing how the amplitudes of the frequency components of the real instrument sounds vary over time. To do this, I looked at windows of 2048 samples, took the FFT and recorded the amplitudes at 30 harmonics. I then used these values to control the amplitude envelopes of the sine waves that were producing the harmonics.

The results of this method can be found in Track 12 on the CD. The sound is closer to the real instrument but the amount of data required for the note makes the method unappealing.

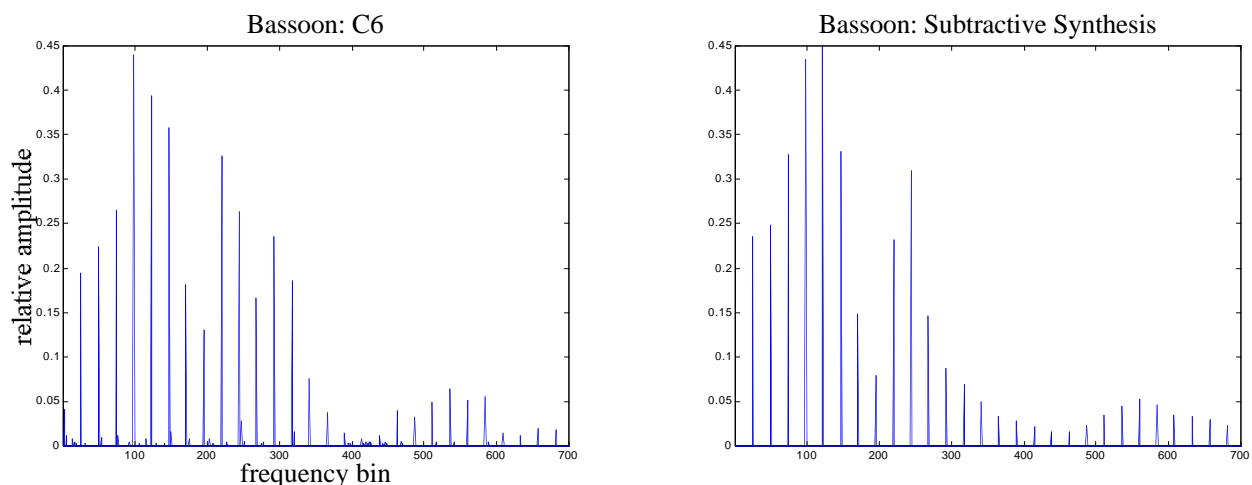
Subtractive Synthesis

Subtractive synthesis is the opposite of additive synthesis. While additive synthesis constructs complex tones by combining frequencies, subtractive synthesis starts with all frequencies, or all harmonics of a fundamental, at equal amplitudes and uses filters to shape the spectrum. Filtering a noisy source is similar to the way many instruments produce sounds. As explained in the discussion on timbre, the double reed of the bassoon produces a buzzing sound, almost like a pulse train. The body of the bassoon filters this buzzing to create the characteristic sound.

To simulate the filtering mechanism of the body, I looked at the spectrum of the note I was trying to simulate, shown in the diagram below to the left. Looking at the spectrum, I saw three regions with locally high amplitudes: near frequency bins 130, 220 and 560. I decided to use three bandpass filters centered in those regions. A bandpass filter lets frequencies pass within a bandwidth above or below a center frequency; other frequencies are attenuated. More precisely, the bandpass filter shapes the amplitudes of frequencies around the center so that one bandwidth away, the amplitudes have been reduced to half power.²³

For the first of the three filters, I placed the center half way between the fourth and fifth peaks, at 330 Hz. For a bandwidth, I chose 120 Hz because this would let the sixth harmonic pass but not the seventh or higher. The center of the second filter was placed half way between peaks 9 and 10, 620 Hz, with a bandwidth of 65 Hz. This would capture the ninth and tenth

harmonics. The third filter was used for the upper harmonics. It was placed at 1448 Hz and it had a bandwidth of 260 Hz.



To simulate the reed, I created a pulse train of 337 sine waves at equal amplitude. The sine waves were all harmonics of C6, 65.41 Hz. The pulse train was run through the three filters connected in parallel and the three outputs were scaled and summed. This produced the waveform with the set of harmonics in the diagram above to the right. The harmonics are not exactly the same as the original but they are similar. The original and synthesized bassoon sounds are Track 13 on the CD. Listening to the synthesized sound, it sounds somewhat like a bassoon but not as close as an additive synthesis version.

The advantage of subtractive synthesis shows when the synthesized instruments are used to play a scale. For the scale using additive synthesis, I created a waveform with the amplitudes of the harmonics taken from the note C6 on the bassoon. I then resampled this note so that I could get one octave of the C major scale. For the subtractive synthesis instrument, I created

pulse trains at each note of the scale and passed these through the same filter bank. These scales are on the CD on Track 14 for the additive scale and Track 15 for the subtractive scale. A scale of recorded bassoon notes is on Track 16. The timbre of the additive instrument becomes pinched as it ascends the scale; whereas, the timbre of the subtractive instrument seems to stay fairly constant. This is probably due to the fact that the subtractive instrument simulates the way the bassoon works, while the additive instrument only simulates the way the instrument sounds on one note. If I created a filter bank with more than three filters, I could potentially get an instrument that sounded even more like a bassoon.

Amplitude Modulation

Amplitude modulation is a technique that uses one signal, the modulator, to control the amplitude of another signal, the carrier. A simple case of amplitude modulation involves sine waves as both the carrier and modulator. The modulator wave changes the amplitude of the carrier wave by multiplication. Mathematically, this is represented by the equation below where s is the output signal, f_c is the frequency of the carrier and f_m is the frequency of the modulator.²⁴

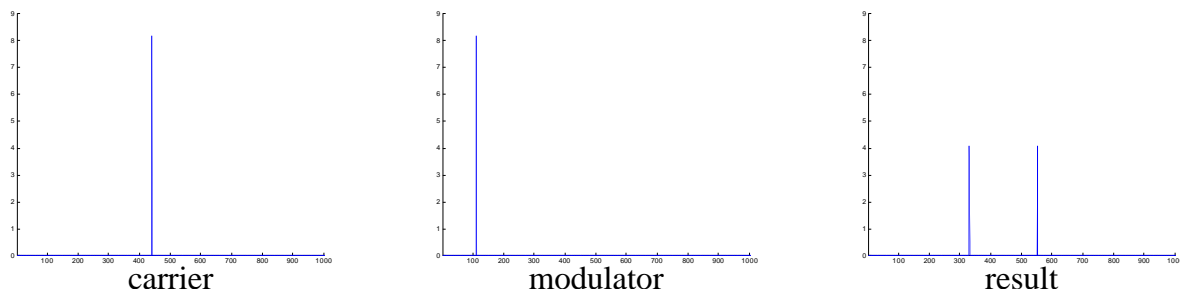
$$s(t) = \sin(2\pi f_c t) \sin(2\pi f_m t)$$

Since the equation above is a product of two sines, it can be transformed into the following equation:

$$s(t) = \frac{1}{2} [\cos(2\pi [f_c + f_m] t) - \cos(2\pi [f_c - f_m] t)]$$

24. Fischman, 237.

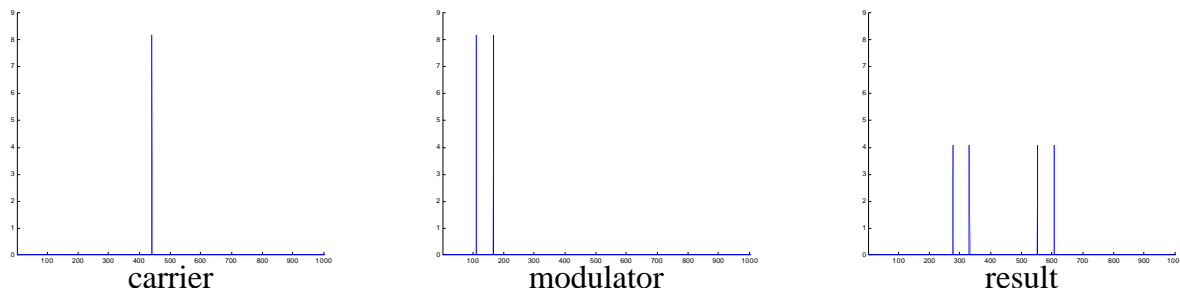
In other words, the resulting waveform will have frequency components above and below the carrier by a distance equal to the frequency of the modulator. This process is clearly nonlinear because the frequencies $f_c + f_m$ and $f_c - f_m$ are not present in either the carrier or the modulator. These new frequencies are called sidebands because they are distributed evenly on both sides of the carrier.²⁵ The diagrams below show what amplitude modulation with simple sine waves does to an output signal. In this example, the carrier has a frequency of 440 Hz and the modulator has a frequency of 110 Hz. The resulting sidebands are at 330 Hz and 550 Hz. The three waveforms are on Track 17 on the CD. The first is the carrier, then the modulator and finally the result. The result waveform is heard as two separate pitches because there are only two frequencies present. To be perceived as a single pitch, three harmonics of a fundamental need to be present.



The process described above can also be applied to signals with multiple frequency components. In my next experiment I used the same carrier and a modulator with two frequency components. Below are the diagrams of the frequency content of the carrier, modulator and resulting signal. In this case, the modulator has frequency components of 110 Hz and 165 Hz and the carrier is a 440 Hz sine wave. The resulting sidebands are at 275 Hz, 330 Hz, 550 Hz and 605 Hz. The sounds of the carrier, modulator and result can be found on Track 18.

25. Ibid.

Although all four sidebands are integer multiples of 55 Hz, it is difficult to perceive a pitch in the result because it does not have three sequential harmonics.



In order to produce harmonic sounds with amplitude modulation, the component frequencies of the resulting signal need to be evenly displaced by a constant frequency width and there need to be at least three sequential harmonics of a fundamental. This conforms to the research done concerning pitch in the chapter on background information. For sine waves, if the carrier to modulator ratio is an integer or one over an integer, there is a good change the resulting signal will be harmonic.²⁶ For my final experiment with amplitude modulation, I used a 440 Hz carrier and a modulator with frequencies of 110 Hz, 220 Hz and 330 Hz. The resulting signal had frequencies of 110 Hz, 220 Hz, 330 Hz, 550 Hz, 660 Hz and 770 Hz. Since all of the sine waves are integer multiples of 110 Hz and there are at least three sequential harmonics present, the perceived pitch is 110 Hz. These sounds can be found on Track 19.

Frequency Modulation

Synthesis by frequency modulation can achieve dynamic changes to the spectrum of a sound without a large amount of control data. Similar to amplitude modulation, a simple form of frequency modulation can be achieved with sine waves using the equation below. Instead of

26. Ibid., 240.

multiplying the modulator and carrier, the modulator is added to the phase argument of the carrier. In the equation below, the output signal is s , α is the carrier frequency times 2π , β is the modulator frequency times 2π and I is the amplitude of the modulation.²⁷

$$s(t) = \sin(\alpha t + I \sin \beta t)$$

To rewrite the equation above in a simple form is not as easy as it was for amplitude modulation. John Chowning, the father of FM synthesis, showed that it could be reduced to the equation below where n is an integer and J_n are Bessel functions. Basically, this formula shows that frequency modulation will produce many sidebands with amplitudes given by $J_n(I)$.²⁸

$$s(t) = \sum_{n=0} J_n(I) [\sin([\alpha - n\beta]t) + \sin([\alpha + n\beta]t)]$$

When the index and modulating frequencies are small, FM synthesis can be thought of as a vibrato. The first example is a 220 Hz carrier modulated by a 4 Hz modulator with the index set to 4. This produces a vibrato on the carrier because the frequency of the carrier will change between 216 Hz and 224 Hz four times per second. For the second example, I changed the modulator to 6 Hz with an index of 6. This produces a wider, faster vibrato. These sounds can be found on Track 20 on the CD.

Interesting sounds occur when the modulating frequency is much higher and when the index varies over time. Varying the index over time is accomplished by putting an envelope around the modulating signal. For the next sets of tones, I varied the carrier and modulator frequencies and put an exponential envelope around the modulating signal. By changing only

27. Ibid., 249-50.

28. Ibid., 250.

these parameters it is possible to produce sounds that resemble multiple instruments. Track 21 has FM versions of a trumpet, bassoon, clarinet and bell.²⁹

The advantage of FM synthesis over additive or subtractive synthesis is that fact that dynamic spectra are now possible using only a few parameters. The attacks and decays are also more realistic. The disadvantage is that it is difficult to obtain the correct pattern of harmonics for sustained part of the note.

29. Russell Pinkston, *FM Synthesis in Csound*, *On The Csound Book*, 264.

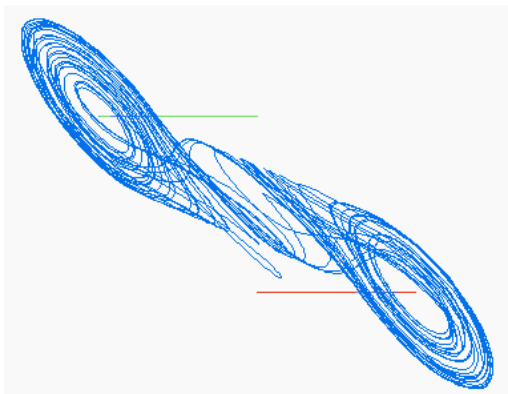
CHAPTER 3

SYNTHESIS USING CONTROLLED CHAOS

The field of electronic music is constantly borrowing ideas from other fields and applying them to music. FM synthesis is a good example of this; it had been used in radio transmission for years before Chowning used the equations for musical purposes.³⁰ The synthesis technique described in this chapter continues this tradition of borrowing from other fields. It was originally developed for chaotic communications but it has proven useful for sound synthesis as well.

Work on this project began in the spring of 1999 as an independent study project proposed to Dan Hussey and me by Professor Kevin Short. He wanted us to use a technique he and a graduate student, Andy Parker, had developed for chaotic communications to produce musical sounds. At this point, there was no talk of synthesizing instrument sounds; the initial goal was to create any kind of musical sound. We were both involved with electronic and computer music so naturally the project interested us.

We spent the first few weeks of that semester learning about the technique of controlling a chaotic attractor to produce periodic waveforms. The technique starts with a chaotic system



called the double-scroll attractor, pictured to the left.

An attractor is a fixed region in space where all trajectories end up after a certain amount of time.

Once a trajectory reaches the attractor, it traces around it and never leaves. The picture to the left was

30. John M. Chowning, "The Synthesis of Complex Audio Spectra by Means of Frequency Modulation," *On Foundations of Computer Music*, ed. Curtis Roads and John Strawn (Cambridge: The MIT Press, 1985), 6-7.

created by letting a single trajectory evolve over time and it shows the double-lobed shape of the attractor. These two lobes are the basis for the controlling technique that is applied to this chaotic system.

Before I describe the controlling technique, I should briefly discuss chaotic systems in general. A chaotic system is a deterministic mathematical object that displays sensitive dependence on initial conditions.³¹ In other words, nearby trajectories will separate at an exponential rate and this separation is not due to random input. The behavior of such a system is predictable over a short period of time but unpredictable in terms of long-term behavior. This property is particularly useful for encryption schemes because someone trying to crack the scheme using the same chaotic system and initial conditions would end up with a different trajectory in the long run. Of course, the receiver of the encrypted message would also end up with a different trajectory making the encrypted message indecipherable. This is where the controlling technique becomes important.

The double-scroll attractor can be controlled in such a way that the trajectories around it become periodic. The controlling technique exploits the fact that a chaotic system is predictable over a short period of time. In other words, if we know where a point is at a certain time, we can predict with reasonable accuracy the evolution of that point for a short period of time. On the double-scroll attractor, this amounts to knowing around which lobes a trajectory will loop when the current location of a point is known.

In the picture of the double-scroll attractor, the two lines are the regions where the controls are applied. These lines are divided into many small regions called bins. If the point at

31. Steven H. Strogatz, *Nonlinear Dynamics and Chaos* (New York: Addison-Wesley Publishing Company, 1994), 323-4.

the center of bin is evolved in the system over a short period of time, it is an accurate predictor of where points in its bin will end up after the same amount of time. This information is reduced to knowing on which lobe a trajectory will end after a fixed amount of time when the starting bin is known. If the lobes of the attractor are coded with a 1 for one of the lobes and a 0 for the other, we can predict a precise pattern of ones and zeros that will occur when the center point of each bin is evolved. This is essentially the information that is used when controlling the attractor.

The technique of controlling the attractor to produce certain patterns of bits is actually unimportant to the way in which it is used for producing sounds. In fact, the sound production technique is sort of a byproduct of the communications technique. When a specific pattern of controls, ones and zeros, is applied repetitively to an evolving system, period orbits form around the attractor. The periodic orbits were the inspiration for the musical use of this technique. The initial goal of our project was to see exactly what types of periodic waveforms could be produced with different repeating patterns of controls. Dan and I were to use our musical creativity along with our backgrounds in mathematics and physics, to figure out what could be done musically with these waveforms.

We started by using code in Matlab that would calculate the trajectory from an initial condition and a repeating pattern of controls. The resulting trajectory existed in three dimensions and we extracted one dimension and saved it as a sound file. These sound files were usually shorter than one second and sounded like a beep or a rumble but we were inspired to continue the search and discover different possibilities from different codes.

One of the early problems we ran into was sampling and sample rate. The Matlab code was using the Runge-Kutta fourth order technique to calculate the trajectory and the step size was set very small so that the calculation could be as accurate as possible. As a result, many of

our waveforms were oversampled and sounded like rumbles when we played them back. To solve this problem, we defined a cut variable that would only save every n^{th} point. This way we could keep the accuracy of a small step size while producing waveforms with audible frequencies. After experimentation, we found that a cut value of 10 produced the best results.

After we could successfully produce different pitches using different codes, we began to use several codes in the same input file to create a melody. At this point in the project, we were viewing the codes as separate notes of the same instrument. By sending different codes to the attractor we were hoping to create small musical melodies. This worked but it was very difficult to get the attractor to respond in the expected way. As musicians, we were used to thinking in terms of pitch and duration. When we first made melodies with the attractor, we had to convert our ideas of pitch and duration into patterns of bits and numbers of repetitions. This was a little bit awkward and our input files became very large because we were sending many repetitions of the same code. Some examples of our early melodies can be found on Tracks 22 through 25. They are called Bass, Solo, Longfile and Pop.

Soon we were pushing the limits of the original Matlab code. Some of our creations were taking ten to fifteen minutes to generate which was far too long to wait when we were putting the finishing touches on a piece. The interface was also less than ideal for our purposes. It was cumbersome to have to convert durations in seconds to number of repetitions. We needed a faster version of the code that was specific to our needs. We decided to write a program in C that would fit our needs.

Original Synth.c Program

Dan and I had both taken introductory courses in programming so we did not think it would be too difficult to write a program in C to make our sound files. We soon realized that this project was much larger than the weekly assignments in our introductory courses. We decided to divide up the project into two different tasks. I worked on the input mechanism: parsing the input file and creating a structural representation in memory. Dan worked on reading the structure and creating the output file.

Before we started working on our separate parts, we had to decide how all of our musical events would be arranged in the program. Our goals for this were conservation of memory and speed. The representation we designed uses thirty-two bits per event. The first sixteen bits are called the code part of the event. This stores the sequence of ones and zeros used to control the attractor. The next eight bits are the info part of the event. The info bits tell our synthesizer if the current event is a note or a rest. If the event is a note, the info byte holds a number that tells the synthesizer how many bits of the code part should be used. The final byte in our structure holds the duration of either the note or the rest. This structure is quite efficient yet flexible enough so that additions to the program fit within our events. For example, in later versions of the program we added tempo changes and dynamics. These we represented in our structure by a rest event with no duration and data in the code portion of the event.

Once we had decided on the internal structure of the program, I began to work on parsing the input file to create a list of our thirty-two bit events and Dan worked on creating an audio file from the event list. My portion of the code read an ASCII text file one line at a time and tokenized it. If the first token was a sequence of bits, these bits were copied into the code section of the event and a special bit was set in the info part that designated this event as a note. If the

first token was the word `test`, when the special bit in the info part was set to zero. In both cases, the second token was the duration and this value was stored in the final byte of the structure.

After many weeks of programming and debugging, we had a working version of our controlled chaos synthesizer written in C. This software was much faster than the original Matlab version so we could create larger and more complicated songs. We used this code for the pieces on Tracks 26 through 30. These pieces are called, in order, `old_comp`, `comp5`, `metal2`, `metal3` and `melodic`.

Experience with longer pieces of music gave us more ideas about additions and modifications to the program. One of the first problems we came across was that it was very difficult to find a code that would produce a specific pitch we wanted in our melody. Up until this point in the project we had created our melodies from pitches we knew were available. This inspired us to change our approach and treat the codes as different instruments instead of different notes of the same instrument. The waveforms were stored in a buffer and we could resample the buffer to get different frequencies. This modification to the design made many more frequencies possible.

Once we had decided that the codes could be different instruments we naturally wanted to use more than one instrument at a time in our compositions. We started to write a version of the synth program that would allow four instruments to play at the same time. These modifications became Dan's project in the summer of 1999 and before long he had made the changes and composed pieces with multiple voices.

In the fall of 1999, the original program had become quite large and I had lost track of all the modifications that were made over the summer. I decided that it would be a good time to rewrite the program from scratch with design goals that we had learned from making the original

version. One important goal was a simple interface for the input file. I wanted to be able to specify the notes by their octave and pitch class name. For example, 8.00 is middle C and 9.00 is the C above that. This would make composition easier because previous versions of the code used frequency in Hertz to specify the note. Another goal was flexibility. I wanted the program to support dynamics, tempo changes, different tuning systems and an unlimited number of instruments. I also wanted the program to be able to write .wav files directly.

Another important task was to learn everything that this system of sound synthesis had to offer. With our original program we had picked codes at random and loosely classified them as noisy or clean. Now we wanted to know how many codes would produce periodic behavior and what each code sounded like. In other words, we needed a library of sounds with every periodic code. This, along with the desire to produce a new version of the synth program, prompted me to write the code in C++. Since both programs would need code to calculate the trajectory and apply controls, I could write objects that handled this and use them in both programs.

The Library

I decided to work on the library before I worked on the new synth program because I wanted to know every sound this technique had to offer. First I worked on the code to search for control codes that would produce periodic orbits. This code was divided into two C++ classes: a trajectory class and a control code class. The trajectory class advanced a point to the next point and controlled it if necessary. The control code class took care of the control codes. It would take a string of ones and zeros and convert it to an integer or convert the current control code integer to a string. It would also check each bit sequentially when called by the trajectory object.

After all the bits had been checked, it would loop back to the first bit. These two objects together efficiently calculated the trajectory of the double-scroll attractor.

The code object was also able to increment the current code in integer format to the next code. This was useful when checking all the codes with a certain number of bits. For example, if I were checking all the four bit codes, I would tell the code object that the bit count is four and I would set the first code to 0. The code object would then increment this number after each code was checked until it reached the binary code 1111. The process would then stop and I would have checked all the four bit codes.

It quickly became clear that checking every code in this fashion would take a very long time for codes with high bit counts. For instance, there are 1024 codes with ten bits. It was useless to check them all because many codes would produce the same orbits. In fact, if one code produced periodic behavior then any other code that was a rotation of the original would produce periodic behavior. By rotation, I mean that 101001 is the same as 010011, 100110, 001101, 011010 and 110100. To reduce the number of codes to check, I defined the following equivalence relation on the set of codes: codes are considered equal if they can be rotated so that they match bit by bit.

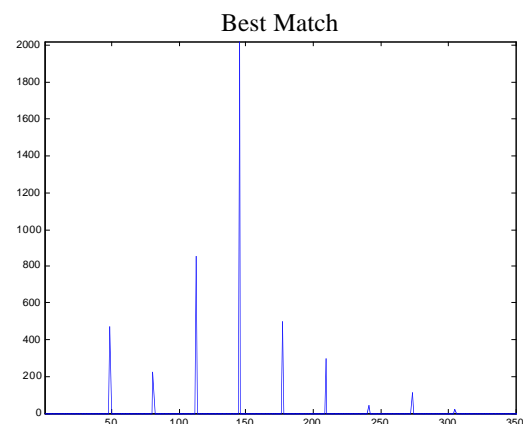
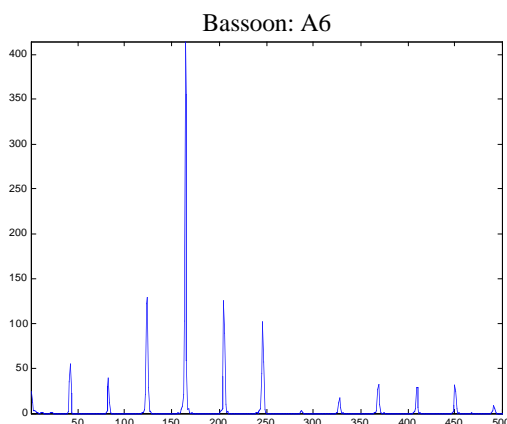
This greatly reduced the number of codes to check. For example, after passing to equivalence classes for 10 bit codes there were only 108 unique codes instead of 1024. For each code that remained, I tested to see if it produced a periodic orbit. This amounted to setting the initial condition in the center of each bin and evolving the system. Every time the trajectory crossed a control line, I checked which bin it was in and which bit we were on in the code. If the trajectory ever returned to the same bin at the same bit, the process was stopped and started in the center of the next bin. If a periodic orbit was found from every bin, then the code was

written to a file of periodic codes. Out of the 108 unique ten bit codes, there were 61 that produced periodic orbits.

After I finished the library of codes, I used the same objects to build a new version of the synth program. This version supported multiple instruments, dynamics, time signatures, tempo markings and different tunings systems. In December of 1999, I used this new version of the program to synthesize the first thirty measures of Bach's Toccata and Fugue. This piece can be found on Track 31 on the CD.

Instrument Modeling

In the spring of 2000, I began to research synthesis of instrument tones so that the technique of controlled chaos could be used for this as well. One of first steps I took was to look at the spectrums of notes over the entire range of a particular instrument. When I compared the spectrums of instrument tones to those of our waveforms, I was surprised to find that many of our waveforms had similar structure. For example, compare the spectrum of the bassoon note below with the spectrum of the waveform generated by code 0001. The patterns of amplitudes of the first six harmonics are very similar in both spectrums.



Since our waveforms seemed to have structure similar to instrument tones, the natural direction to take was to write a program that would find the best match waveform in our library for any natural instrument tone. The program essentially consisted of two parts: a way of coding our waveforms and a way of matching instrument sounds to them.

Since we were matching the waveforms based on the amplitudes of the harmonics, the most sensible way to represent the waveforms would be by a vector of these amplitudes. In the current version of the library, all of the codes were periodic over 1024 points. This made the FFTs very clean and all of the peaks were in the center of the frequency bins. I took the first sixty-four harmonics and that became my vector of amplitudes. If any of the amplitudes in my vector were less than 0.5% of the maximum, I set the value of that harmonic in the vector to zero.

I then made each sixty-four dimensional vector have unit length. This was so they could be compared to vectors from instrument tones using the formula below.

$$a \bullet b = |a| |b| \cos \theta$$

Since the vectors were scaled to have unit length, the dot product on the left is equal to the cosine of the angle between the vectors. If the angle between the vectors is zero, then the dot product is equal to one. This is the maximum value that the dot product can have, and it means that the vectors a and b are identical. If the angle between the vectors is close to zero, then the dot product is close to one. The best match was chosen to be the vector that produced the greatest dot product with the instrument vector.

As mentioned above, the waveforms were all periodic over 1024 points. This meant that they all had a fundamental frequency of $44100/1024$ Hz, approximately 43 Hz, when the sample rate was 44100 samples per second. To find the values for each position in the vector, I looked

for the amplitudes of each harmonics. For example, the first harmonic is around 43 Hz so I defined a small window from about 41 to 45 Hz and picked the largest amplitude in that window. The second harmonic is around 43×2 Hz, so I looked in a window around that frequency and picked the largest amplitude. I repeated this process for the first 64 harmonics and stored them as the vector.

Now I had to create vectors that represented an instrument sound. Since instrument sounds vary over the duration of the note, I decided to divide each note up into sections and find the vector for each section. At this point, it was important to know the fundamental frequency of the note I was analyzing because the FFTs of the instrument sounds were not as clean as those of the library waveforms. There were often frequencies in the instrument recordings that I did not want to end up in my vector of peaks. For example, in the recordings of the piano notes, the sound of the hammer striking the string shows up as strong low frequencies. Although that sound is important to the attack, it does not affect the amplitudes of the sustained harmonics. To facilitate the process, I encoded the fundamental frequency in the audio file name. With knowledge of the fundamental of the note being analyzed, I was able to only look at frequencies within a window around each of the harmonics. All of the other frequencies were ignored. The amplitudes of the first sixty-four harmonics were stored as a vector.

The instrument vectors were also normalized to have unit length and I took separate dot products with all of vectors in the library. The dot product with the value closest to one was the best match. This method found reasonable matches but there was still one improvement that I wanted to make.

The spectrums of natural instrument almost never have sixty-four strong harmonics; sometimes there are only five to ten. This observation led me to make a change to the program

that found the best match. I had noticed, when looking at the spectrums of the controlled-chaos waveforms, that often a desirable pattern of harmonics would occur starting at the second or third harmonic. My program would not pick this up as a good match because I always matched the instrument peaks starting at first. The new version is now able to find matches within the vectors of controlled-chaos waveforms. For example, if my instrument vector had five prominent peaks, I would try to match this against peaks one through five, two through six, three through seven, and so on for all of the controlled-chaos vectors. The best match out of any of these is recorded along with the starting peak.

This method of synthesis offers several advantages over other frequency domain techniques. The first advantage is that this method can achieve dynamic frequency spectrums with minimal information. It captures the changes in the amplitudes of the harmonics by finding multiple best matches for the attack, sustain and release portions of the note. Although similar results could be achieved with additive synthesis, the amount of information required for this would be immense. Additive synthesis would need to store multiple sixty-four bit floating-point values for every harmonic to capture the changing amplitudes. This new technique can store all of the necessary information for a decent match in only a few bits: up to sixteen for the control code and some extras that keep track of the best match.

The second advantage of this synthesis technique is that it uses the actual instrument data in the synthesis process. The nonlinear techniques, such as FM synthesis, could easily synthesize dynamic spectrums but could not easily match the pattern of harmonics from natural instrument sounds. This is because the sidebands were symmetric on either side of the carrier frequency; a phenomena that does not occur in natural sounds. Creating instrument-like sounds

with these methods required trial and error, a time consuming process. By using the instrument waveforms to pick the best match, we have an efficient method that produces decent results.

CHAPTER 4

CONCLUSIONS AND FUTURE DIRECTIONS

There are still many options to explore concerning sound synthesis by controlled chaos. For example, several of the waveforms from the library could be used together at once to match instrument sounds more closely. One method could be to match the lower harmonics with one code and the upper harmonics with another. This way, we could get good matches in both regions, which would probably, produce even better results. I am also looking into using the waveforms with many harmonics in combination with filter banks to try subtractive synthesis techniques. Some of the codes in the library produce up to 128 harmonics and these rich waveforms might create realistic results when properly filtered.

I also plan to research other techniques for analyzing instrument sounds. Currently, I am relying on large and small Fourier Transforms to give me information about the frequency components of the notes. One of the limitations of the Fourier Transform is that in order to obtain good frequency resolution you have to give up time resolution. Unfortunately, both resolutions are important to instrument sounds. A good demonstration of this fact is the analysis of a piano note. For notes in the middle of the piano, there are three strings per note. These strings are tuned to the same frequency but are never exactly the same. Consequentially, the small differences in frequency between the three strings cause beats that add warmth and depth to the sound. In order to separate these frequencies into different bins, a very large number of data points would be needed. But in piano notes, the upper harmonics decay faster than the lower harmonics and this decay would be lost because the Fourier Transform of a large number of points gives the average strength of each harmonic in that window. Therefore, it is difficult to

capture high frequency resolution and time resolution with the Fourier transform. Several methods, called time-frequency methods, have been created to deal with this problem. One such method is wavelet analysis and it has been applied to music signals with success. I plan to look into this method further to learn how it can help the synthesis technique.

Although many promising directions have yet to be explored, the synthesis technique described in Chapter 3 is already producing remarkable results. The library of periodic waveforms is quite diverse and decent matches have been found for many common instruments. But as I learned from researching the classic synthesis techniques, there is more to an instrument's sound than the relative strength of its harmonics. There are attacks, decays and microvariations in the waveform that seem unimportant in the physical analysis of the sound, but are essential to its perceptual effect. This is where synthesis by controlled chaos has a built-in advantage. Chaotic systems have a natural attack when the trajectory approaches the attractor and the microvariations in the chaotic system itself cause the long-term unpredictability. Controls to the trajectory are only applied in two spots along the attractor leaving most of the waveform untouched. A look at any of these waveforms in the time-domain confirms that they are inherently more complex than the sum of a few sine waves. This is something our ears had told us from the beginning; the sound of these waveforms seemed to have depth. Now that we have cataloged all of the possible waveforms from this system we need a way of combining them into even more complicated waveforms. Using many waveforms together to create a single sound is an exciting and promising direction for further work.

APPENDIX I: CD TRACK LIST

- Track 1: Violin middle C
- Track 2: Cymbal crash
- Track 3: Sine wave A 110, 3rd 5th 8th, 2nd 3rd 4th, sine wave A 110
- Track 4: Decreasing amplitudes, Increasing Amplitudes, All harmonics, Odd harmonics
- Track 5: Oboe Bb7 and Clarinet Bb7
- Track 6: Square waves A 110, E 165 and A 220
- Track 7: Synthesized and Real Bassoon at 3 pitches
- Track 8: Concert D by different instruments without attacks
- Track 9: Three sawtooth waves with different phase components at 3 pitches

Additive Synthesis

- Track 10: Clarinet real, clarinet synthesized
- Track 11: Trumpet real, trumpet synthesized
- Track 12: Trumpet with many small FFT windows

Subtractive Synthesis

- Track 13: Original and Subtractive synthesis bassoon
- Track 14: Scale using additive synthesis
- Track 15: Scale using subtractive synthesis
- Track 16: Scale made from recordings

Amplitude Modulation:

- Track 17: Simple AM
- Track 18: Modulator with two poles
- Track 19: Harmonic AM

Frequency Modulation:

- Track 20: Simple FM: vibrato
- Track 21: Trumpet, bassoon, clarinet, bell

Chaos

- Track 22: Bass
- Track 23: Solo
- Track 24: Longfile
- Track 25: Pop
- Track 26: old_comp
- Track 27: comp5
- Track 28: metal2
- Track 29: metal3
- Track 30: melodic
- Track 31: bach

WORKS CITED

- Chowning, John M. The Synthesis of Complex Audio Spectra by Means of Frequency Modulation. In *Foundations of Computer Music*, ed. Curtis Roads and John Strawn, 6-29. Cambridge: The MIT Press, 1985.
- Fischman, Rajmil. A Survey of Classic Synthesis Techniques in Csound. In *The Csound Book*, ed. Richard Boulanger, 223-60. Cambridge: The MIT Press, 2000.
- Halliday, David, Robert Resnick, and Jearl Walker. *Fundamentals of Physics*. New York: John Wiley & Sons, Inc., 1997.
- Howe, Hubert S., Jr. *Electronic Music Synthesis*. New York: W. W. Norton & Company Inc., 1975.
- Matthews, Max. Introduction to Timbre. In *Music, Cognition, and Computerized Sound*, ed. Perry R. Cook, 79-87. Cambridge: The MIT Press, 1999.
- Pierce, John. Introduction to Pitch Perception. In *Music, Cognition, and Computerized Sound*, ed. Perry R. Cook, 57-70. Cambridge: The MIT Press, 1999.
- _____. Sound Waves and Sine Waves. In *Music, Cognition, and Computerized Sound*, ed. Perry R. Cook, 37-56. Cambridge: The MIT Press, 1999.
- Pinkston, Russell. FM Synthesis in Csound. In *The Csound Book*, ed. Richard Boulanger, 261-79. Cambridge: The MIT Press, 2000.
- Shepard, Roger. Pitch Perception and Measurement. In *Music, Cognition, and Computerized Sound*, ed. Perry R. Cook, 149-65. Cambridge: The MIT Press, 1999.
- Slawson, Wayne. *Sound Color*. Los Angeles: University of California Press, 1985.
- Spjut, Erik. An Introduction to Signal Processing with Csound. In *The Csound Book*, ed. Richard Boulanger, 389-482. Cambridge: The MIT Press, 2000.
- Strogatz, Steven H. *Nonlinear Dynamics and Chaos*. New York: Addison-Wesley Publishing Company, 1994.