University of Lynchburg
Digital Showcase @ University of Lynchburg

Undergraduate Theses and Capstone Projects

Spring 4-2011

Quantifying Harmony and Dissonance in Piano Intervals and Chords

Michael Blatnik
University of Lynchburg

Follow this and additional works at: https://digitalshowcase.lynchburg.edu/utcp

Part of the Other Physics Commons

Recommended Citation
Blatnik, Michael, "Quantifying Harmony and Dissonance in Piano Intervals and Chords" (2011). *Undergraduate Theses and Capstone Projects*. 137.
https://digitalshowcase.lynchburg.edu/utcp/137

This Thesis is brought to you for free and open access by Digital Showcase @ University of Lynchburg. It has been accepted for inclusion in Undergraduate Theses and Capstone Projects by an authorized administrator of Digital Showcase @ University of Lynchburg. For more information, please contact digitalshowcase@lynchburg.edu.

Quantifying Harmony and Dissonance in Piano Intervals and Chords

Michael Blatnik

Senior Honors Project

Submitted in partial fulfillment of the graduation requirements

of the Westover Honors Program

Westover Honors Program

April, 2011

Dr. John Eric Goff **Committee chair**

Dr. Chinthaka Liyanage

Dr. Nancy Cowden

Abstract

The level of dissonance in piano intervals and chords was quantified using both experimental and computational methods. Intervals and chords were played and recorded on both a Yamaha YPT-400 portable keyboard and a Steinway & Sons grand piano. The recordings were run through spectral analyses, and dissonance values were calculated using a dissonance equation. The result was a ranking of comparative dissonance levels between each chord and interval. Though the goal was to find a universal ranking of chords, it was instead determined that such a ranking cannot exist. The non-universal rankings revealed that the transition from least dissonant to most dissonant was gradual.

I. INTRODUCTION

The primary goal of this work is to explore the concept of intrinsic dissonance within music. The Oxford English Dictionary[1] states the definition of *dissonant* as "disagreeing or discordant in sound, inharmonious; harsh-sounding," and the definition of *consonant* as "musical harmony or agreement of sounds." Musically, dissonant chords are used to give a piece tension, which, classically, is resolved to consonance.[2] Though the musical context of a chord plays a role in how that chord is perceived, we consider only the inherent dissonance within the playing of an interval (two notes played simultaneously) or a chord (three or more notes played simultaneously).

The intervals we considered are within the octave range beginning at middle C (C4, frequency 262.6 Hz). The notes within this octave for a C major scale are shown in Figure 1 in piano notation.

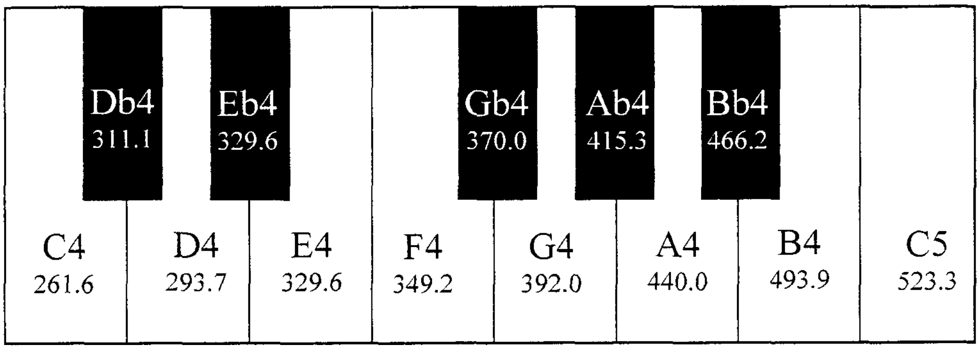
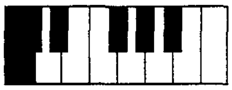
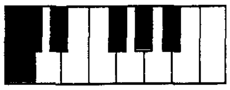
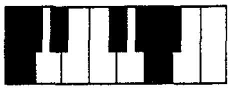
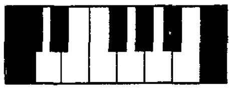


Figure 1 – Piano notation of a C4 major scale with frequencies (Hz).

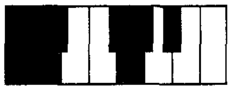
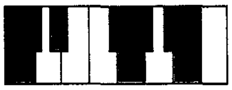
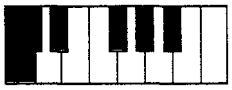
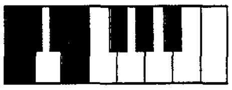
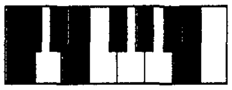
Table 1 shows the intervals with the corresponding notes on a piano along with each interval's Roman numeral notation.

Table 1 – C-intervals.

Interval	Roman numeral notation	Piano notation	Interval	Roman numeral notation	Piano notation
Unison	I		Fifth	V	
Semitone	ii		Minor sixth	Vi	
Second	II		Major sixth	VI	
Minor third	iii		Minor seventh	Vii	
Major third	III		Major seventh	VII	
Fourth	IV		Octave	VIII	
Diminished fifth	v				

We considered 17 C-chords, shown in Table 2, all within the octave beginning at middle C. See Appendix A for a description of the abbreviations in chords.

Table 2 – C-chords.

Chord	Piano Notation	Chord	Piano Notation
C major		C sus 2	
C minor		C sus 4	
C 7		C 6	
C min 7		C minor 6	
C min maj 7		C dim	
C maj 7		C dim 7	
C 7b5		C aug	
C 7#5		C aug 7	
C min 7b5			

We answer the following questions: Can physics explain how a chord is perceived with respect to consonance or dissonance? Can a formula quantify the inherent dissonance within an interval or chord? If so, is a ranking of dissonance for chords universal? Essentially, should the blacks and whites of music that are consonance and dissonance be instead shades of gray?

II. PHYSICS BACKGROUND

Sound waves

Quantifying dissonance in musical sound requires an exploration of the principles of sound. Sound waves are caused by pressure variations in a given medium.[3] Pressure is measured in Pascals (Pa), where 1 Pa = 1 N/m^2. Standard atmospheric pressure in air, 1 atm, is equal to approximately 10^5 Pa. A disturbance in a medium, such as the collision of two objects in air, gives a sudden pressure rise or fall to the air immediately around the two objects. A rarefaction is the reduction of a medium's density, which results in a lower pressure, whereas a compression is in an increase in density, a higher pressure. Rarefactions and compressions in a medium disperse from the source of the disturbance in the form a sound wave.

A sound might be perceived as a click, but when rarefactions and compressions of the air occur at regular time-intervals, they can be perceived as a musical tone. A sound is considered a musical tone if the sequence of regularly-repeated pressure changes has a frequency between approximately 18 and 15,000 vibrations per second. Frequencies of sound are measured in Hertz (Hz), which is vibrations per second. The human ear is capable of detecting frequencies between 20 and 20,000 Hz, but the ear's ability to perceive frequencies varies from person to person.[4]

A simple tone, is a musical tone for which the source of sound produces sound waves sinusoidally at a given fundamental frequency.[3] A complex tone is produced from the addition of multiple simple tones. The wave form of a simple tone and a complex tone are shown in Figure 2.

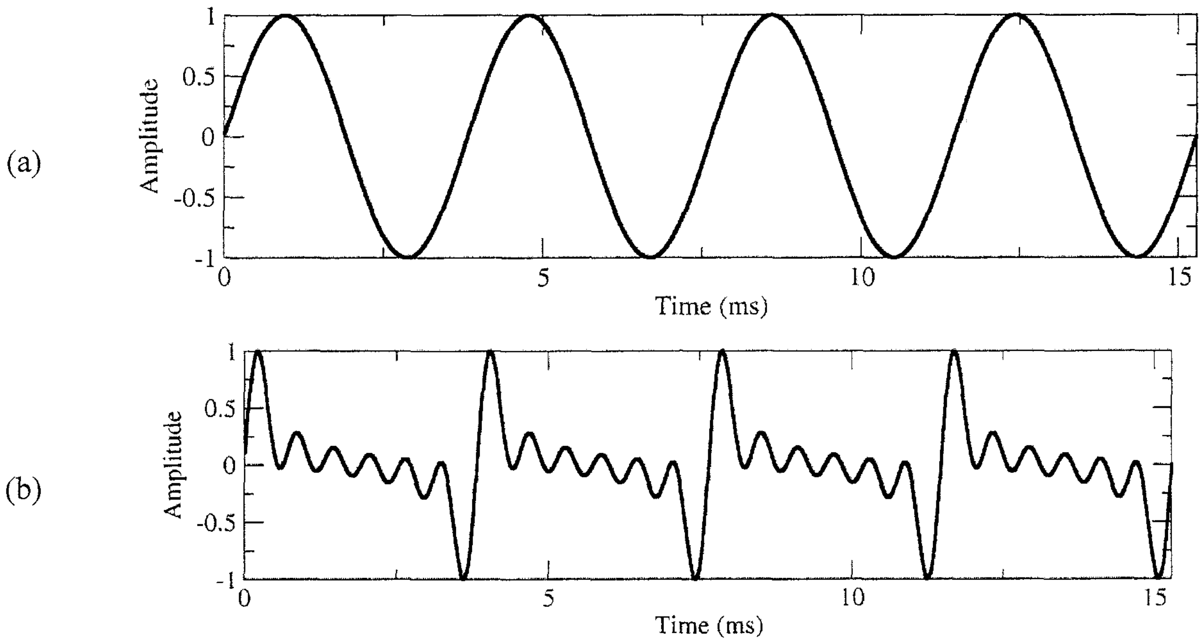
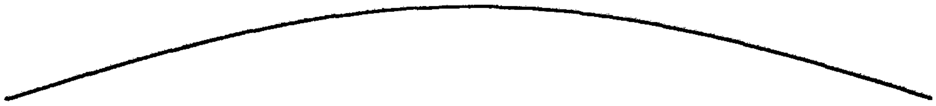


Figure 2 – Wave forms of (a) a simple tone at 261.6 Hz (Middle C or C4), and (b) a complex tone consisting of 6 equally-weighted simple tones: 261.6 Hz, 523.2 Hz, 784.8 Hz, 1046.4 Hz, 1308.0 Hz, and 1569.6 Hz.

Overtones

For a stringed instrument, like a piano, *overtones* arise from the possible standing waves on the string. Each standing wave is a sinusoidal wave fit between the fixed ends of the string. Table 3 illustrates the fundamental frequency f along with five of its overtones.

Table 3 – The overtones on a string with fixed ends.

Overtone	Frequency	Wave
Fundamental	f	
1st	$2f$	
2nd	$3f$	
3rd	$4f$	
4th	$5f$	
5th	$6f$	

For a fundamental frequency f, the overtones frequencies will be $2f, 3f, 4f, 5f, \ldots$ Each instrument has certain characteristics, such as shape and material, which determine the relative intensities of these overtones.

Giordano and Nakanishi[5] explored the computational simulation of a piano string struck by a hammer using the work of Chaigne and Askenfelt.[6] Michael Blatnik repeated that piano simulation in PHYS333 at Lynchburg College in the spring semester of 2010, and it became the original basis of this work. In the simulation, the hammer hits the string at a point of one-eighth the string's total length. Figure 3 shows the string's transverse displacement at the hammer strike location.

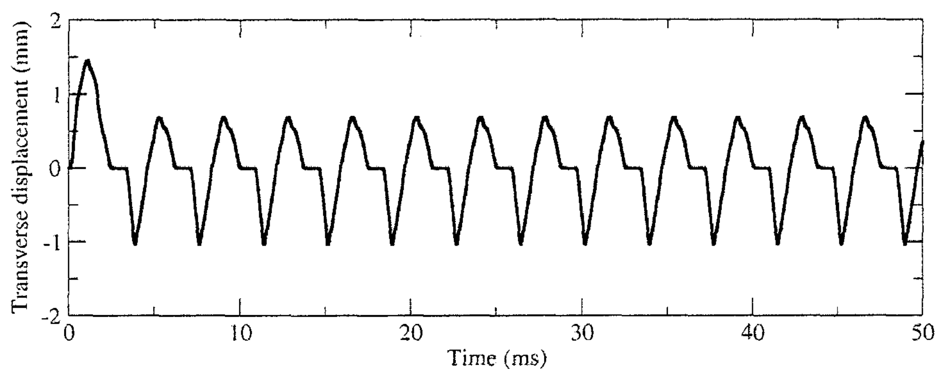


Figure 3 – Piano string's transverse displacement at hammer strike location.

Because the oscillation of the string's displacement in Figure 3 is not sinusoidal, the piano produces a complex tone rather than a simple tone. The Fast Fourier Transform (FFT) is computational method used to output what frequencies are present within a given sound sample (see Appendix B:3 for code). The piano string's transverse displacement from Figure 3 was run through an FFT to output the power spectrum shown in Figure 4.

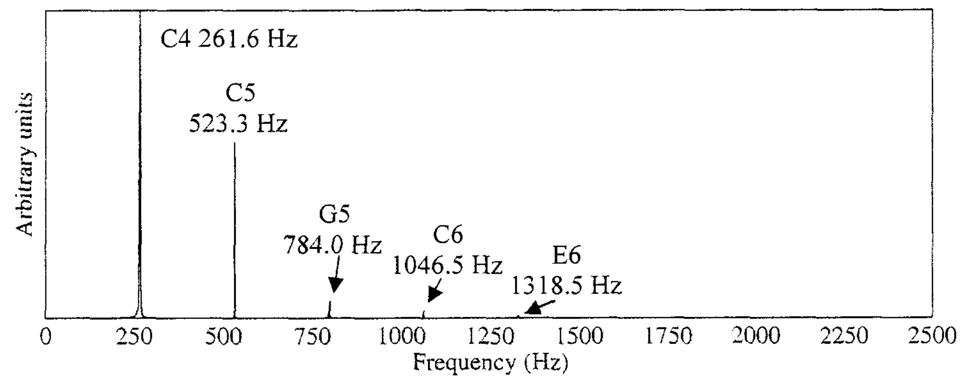


Figure 4 – Power spectrum for amplitude fluctuations in Figure 3.

Figure 4 shows the presence of the fundamental frequency and the first four overtones. The fundamental frequency lies just above 250 Hz, whereas the first harmonic lies around 500 Hz, the third around 750 Hz, the fourth around 1000 Hz, and so forth. These arbitrary amplitudes represent amplitudes of the fluctuations of air pressure surrounding the string.

Measuring sound

Pressure differences help quantify sound.[7] Pressure fluctuations associated with sound are small compared to atmospheric pressure. The faintest perceivable sound has a gauge pressure around 2×10^{-5} Pa; the so-called threshold of pain, the limit of useful hearing sensation, has a gauge pressure around 20 Pa. Though extremely loud sounds are five orders of magnitude less than atmospheric pressure, the pain threshold is over six orders of magnitude higher than the pressure of the faintest perceivable sound. To cover such a broad range of sound, a logarithmic scale for the sound pressure is used. The sound pressure level (SPL) is found using the equation

$$\mathrm{SPL} = 20 \cdot \log_{10}\left(\frac{p_{rms}}{p_{ref}}\right) \text{ dB}, \tag{1}$$

where a decibel (dB) is a logarithmic unit for sound. The reference sound pressure level $p_{ref} = 2 \times 10^{-5}$ Pa corresponds to the faintest perceivable sound. The root-mean-square pressure p_{rms} is

$$p_{rms} = \sqrt{\frac{1}{T}\int_0^T p^2(t)dt}\,, \tag{2}$$

where $p(t)$ represents a pressure wave with period T. When $p_{rms} = p_{ref}$, SPL $= 0$ dB; when $p_{rms} = 20$ Pa, SPL $= 120$ dB.

Using equation (1), the difference in the sound pressure levels between two simple tones with respective root-mean-square pressures p_1 and p_2, is

$$\mathrm{SPL}_1 - \mathrm{SPL}_2 = 20 \cdot \log_{10}\left(\frac{p_1}{p_2}\right) \text{dB}. \tag{3}$$

For multiple simple tones where the arbitrary i^{th} tone has $p_{rms} = p_i$, and the tone with the highest root-mean-square pressure has $p_{rms} = p_{max}$, SPL_i is

$$\text{SPL}_i = \text{SPL}_{max} + 20 \cdot \log_{10}\left(\frac{p_i}{p_{max}}\right) \text{ dB.} \tag{4}$$

The ratio of pressures helps determine the sound pressure level, but the sound pressure level does not correspond linearly to how *loud* a tone is perceived.

Humans' perception of *loudness* is subjective. Fletcher and Munson[8] generated "curves of equal loudness" by playing a reference simple tone of 1000 Hz, playing another tone, and having their subjects adjust the second tone's intensity until it had roughly the same perceived loudness as the 1000 Hz reference tone. They discovered that the loudness perception of simple tones with the same SPL depends on the frequency of the tone.

The phon is the unit used to describe the *loudness level* L_N, and is the number of decibels needed to raise a tone at a given frequency to make it have the same perceived loudness as a 1000 Hz tone at a given sound pressure level. The loudness level gives a way to describe how loud a tone is perceived in relation to the reference tone of 1000 Hz, but loudness levels are nonlinear. A sound with $L_N = 100$ phon is more than twice as loud as a sound with $L_N = 50$ phon. The sone, is used to measure *loudness,* N, in a linear way, such that a doubling of the number of sones results in a doubling of the perceived loudness. One sone is defined arbitrarily as the loudness of a 1000 Hz tone with an SPL = 40 db (40 phon at 1000 Hz). For loudness levels above 30 db, the relation is essentially logarithmic and is expressed with the equation

$$N = 2^{\frac{L_N - 40\text{ phon}}{10\text{ phon}}} \text{ sone.} \tag{5}$$

Raising the loudness level by 10 phon will result in a doubling of loudness. Around 250-2000 Hz, every 10 db increase in SPL is approximately a 10 phon increase in loudness level, and thus a doubling of loudness.

III. MUSIC THEORY

Musical scales

Working around 550 B.C.E. Pythagoras was the first to identify consonance in music.[9] He claimed consonance was the result of relatively small whole number ratios, such as 1:1, 2:1, 3:2, and 4:3, between two frequencies. The unison interval corresponds to 1:1, the octave to 2:1, the fifth to 3:2, and the fourth to 4:3. Table 4 shows the scale Pythagoras developed based on the consonant ratios. The notes C, D, E, …, C' in Tables 4 and 5 do not refer to the notes of the modern piano but rather to historical scales.

Table 4 – Pythagorean scale.

Note	C	D	E	F	G	A	B	C'
Ratio	1:1	9:8	81:64	4:3	3:2	27:16	243:128	2:1
Ratio decimal	1.000	1.125	1.266	1.333	1.500	1.688	1.898	2.000
Interval	Unison	Second	Major third	Fourth	Fifth	Sixth	Major seventh	Octave

By the early Renaissance, music had become more harmonic, that is, notes were played simultaneously rather than only in succession.[9] Harmonic intervals showed the apparent dissonance involved in the Pythagorean scale, such as the Pythagorean major third (81:64 or 1.266 ratio), where the E-note had a higher frequency from what was found to be consonant (a 5:4 or 1.25 ratio). To reform music, the *just* scale was developed (see Table 5).

Table 5 – Just scale.

Note	C	D	E	F	G	A	B	C'
Ratio	1:1	9:8	5:4	4:3	3:2	5:3	15:8	2:1
Ratio (decimal)	1.000	1.125	1.250	1.333	1.500	1.667	1.875	2.000
Interval	Unison	Second	Major third	Fourth	Fifth	Sixth	Major seventh	Octave

The just scale maintains maximum perceived consonance within the scale beginning at C. Transposing, or shifting the scale up or down, results in a different set of ratios. With D as a bass note for the transposed scale, the fifth of D in relation to C has a ratio of 9:8 × 3:2 = 27:16 = 1.6875, which is close to but not equal to the C:A ratio of 1.667. Scales based on different notes in the just scale result in a different sound as the ratios change for each key.

By the eighteenth century, composers desired a scale that allowed for transposition from one key to another without changing the sound. The 12-tone equally-tempered scale thus gained popularity, as it kept all the ratios between adjacent semitones fixed, thus allowing for transposition.[9] Originally conceived by Simon Stevin in the 16th century, the equally-tempered scale defined the semitone ratio as $2^{1/12}$. The octave was divided into 12 of these semitones, and the 2:1 ratio on the octave was maintained because $2^{12/12} = 2$. The frequency ratios for any octave are $1, r, r^2, r^3, r^4, r^5, r^6, r^7, r^8, r^9, r^{10}, r^{11}, r^{12}$, where r^{12} represents the octave ratio, and thus $r^{12} = \frac{2}{1} = 2$. The ratio of any two adjacent semitones is equal to the ratio of any other two adjacent semitones: $\frac{r}{1} = \frac{r^2}{r} = \cdots \frac{r^{11}}{r^{10}} = \frac{r^{12}}{r^{11}} = 2^{1/12}$. Table 6 compares the just scale with the equally-tempered scale.

Table 6 – Equally-tempered scale vs. just scale.

Note	C	Db	D	Eb	E	F	Gb	G	Ab	A	Bb	B	C'
Equally-tempered ratio	1	$2^{1/12}$	$2^{2/12}$	$2^{3/12}$	$2^{4/12}$	$2^{5/12}$	$2^{6/12}$	$2^{7/12}$	$2^{8/12}$	$2^{9/12}$	$2^{10/122}$	$2^{11/12}$	2
Decimal	1	1.060	1.123	1.189	1.260	1.335	1.414	1.498	1.587	1.682	1.782	1.888	2
Just scale ratio	1:1		9:8		5:4	4:3		3:2		5:3	7:4	15:8	2:1
Decimal	1.0		1.125		1.250	1.333		1.500		1.667	1.750	1.875	2.000
Interval	I	ii	II	iii	III	IV	v	V	Vi	VI	Vii	VII	VIII

Though equal temperament keeps the octave at a 2:1 ratio, it is only a close approximation for the other intervals. An equally-tempered fifth, for example, is the ratio 1.498, which is approximately 0.11% lower than the ratio of a consonant fifth (3:2). Although the equally-tempered-scale did not keep the intervals at maximum perceived consonance, the equally-tempered scale was standardized and the piano and keyboard we tested are both tuned to this scale.

IV. QUANTIFYING DISSONANCE

Helmholtz and beats

In the late 19th century, Helmholtz[10] theorized that beats resulting from interference between fundamental and overtone frequencies were the source of dissonance within a musical sound. Beats occur as a result of the interference of two sound waves of slightly different frequencies. The difference in frequency is the number of beats per second. Figure 5 shows the addition of simple tones differing by 2 Hz.

(a)

(b)

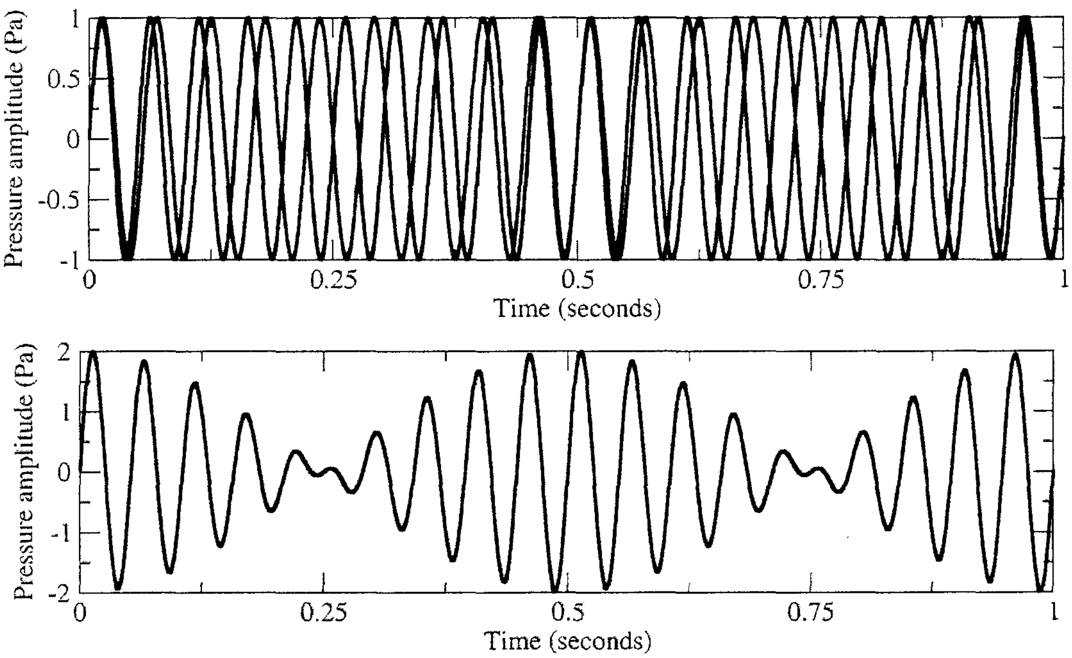


Figure 5 – (a) Two simple tones at 20 Hz (blue) and 18 Hz (red). (b) Beats resulting from the addition of the two simple tones in (a).

Figure 5 shows the regions of large amplitude at $t = 0, 0.5,$ and 1.0 seconds, where the sound waves add up most constructively. Regions of small amplitude, $t = 0.25$ and $t = 0.75$ seconds, are the points where the sound waves add up most destructively. Small amplitudes produce the least loudness; large amplitudes produce the greatest loudness. Fluctuations in loudness result in discernible beats, two beats per second in the case of the simple tones in Figure 5.

Helmholtz found that maximum perceived dissonance occurs when two simple tones differ by about 33 beats per second. He categorized the order of consonant intervals from most consonant to least consonant: 1) Octave, 2) Twelfth, 3) Fifth, 4) Fourth, 5) Major Sixth, 6) Major Third, 7) Minor Third, while the other intervals, were deemed dissonant. Figure 6 illustrates the difference between the power spectrums of a consonant interval, the fifth, and a dissonant interval, the semitone.

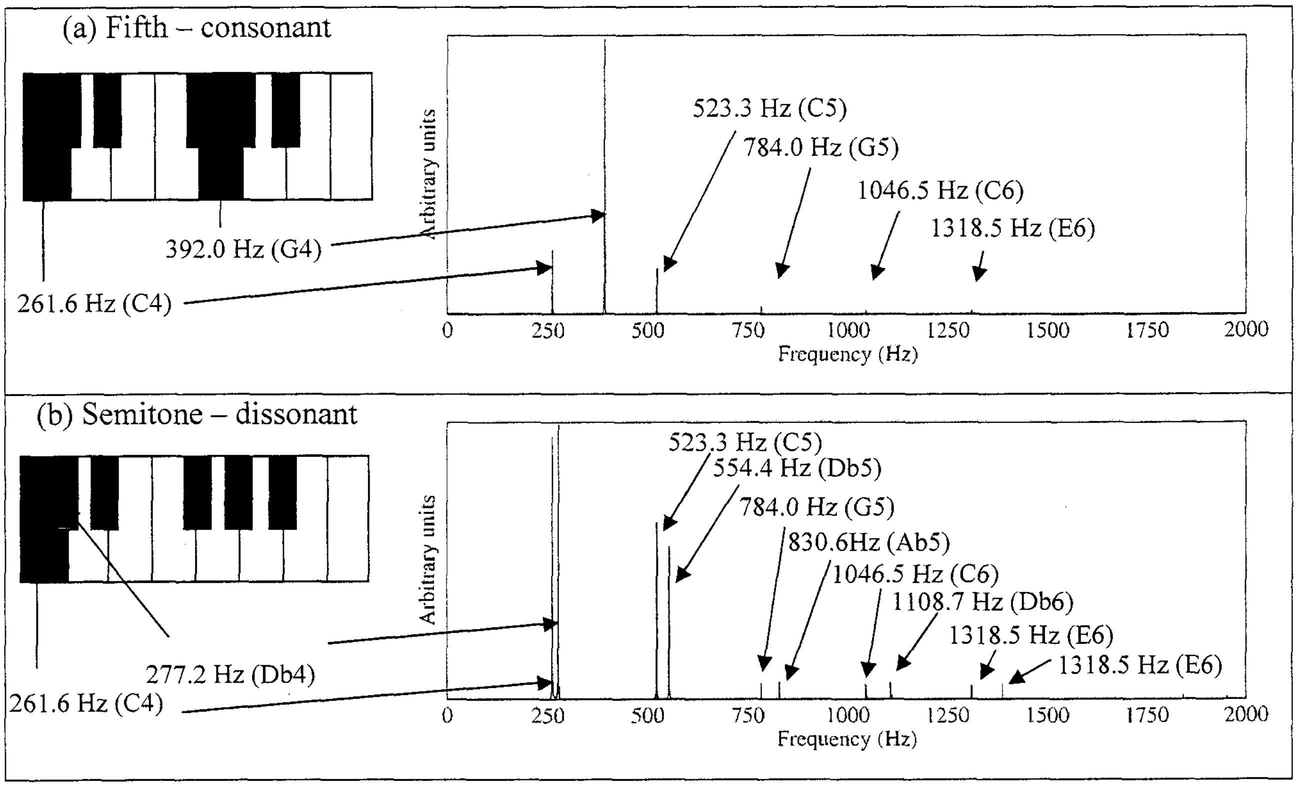


Figure 6 – (a) Fifth and (b) semitone power spectrums from Yamaha YPT-400 portable keyboard.

The fifth's power spectrum in Figure 6 shows a dominate frequency of 392.0 Hz (G4). The C4-G4 combination has 130.4 beats per second. The other point of possible dissonant beats on the fifth is between G4 and C5, with 131.3 beats per second. Both of these beat frequencies are far from what Helmholtz found to be most dissonant, 33 beats per second. In comparison, the semitone's significant points of dissonance occur at C5-Db5, 15.6 beats per second, and at C5-Db5, 31.1 beats per second. Both of these beat frequencies are close to Helmholtz' 33 beats per second of maximum dissonance.

To build empirical statistics of how dissonance is perceived, Plomp and Levelt[11] conducted a series of experiments in 1965 in which 380 subjects had to judge simple tone intervals on scales of consonance/dissonance. The subjects listened to simple tone intervals picked at random to prevent interval recognition. There were 44 total intervals, picked from different regions of the musical range with fundamental frequencies of 125, 250, 500, 1000, and 2000 Hz. Plomp and Levelt's findings showed that the majority of subjects found the most dissonant intervals occurred around 20-40 beats per second, depending on the fundamental frequency, thus agreeing with Helmholtz's 33-beats-per-second theory. The lower the fundamental frequency of the interval, the lower was the number of beats where maximum dissonance was perceived.

Sethares[12] found a curve to fit to Plomp and Levelt's dissonance statistics data. Sethares parameterized Plomp and Levelt's statistics with a model of the form

$$d(x) = e^{-ax} - e^{-bx}, \tag{6}$$

where x is the difference between the frequencies of two simple tones, and a and b are constants. Sethares statistically found that $a = 3.5$ and $b = 5.75$. The dissonance function, $d(f_1, f_2, l_1, l_2)$, for two frequencies f_1 and f_2 with respective loudnesses l_1 and l_2, is

$$d(f_1, f_2, l_1, l_2) = \min(l_1, l_2)[e^{-as(f_2-f_1)} - e^{-bs(f_2-f_1)}], \tag{7}$$

where

$$s = d^*/[s_1 \min(f_1, f_2) + s_2], \tag{8}$$

where $d^* = 0.24$, the maximum of equation (6). From a least-square fit, Sethares could ensure that his model closely fit Plomp and Levelt's statistics by altering the values of s_1 and s_2. The ideal values of s_1 and s_2 were found to be $s_1 = 0.021$ and $s_2 = 19$.

Sethares assumed that the total dissonance is the sum of its constituent parts and the total dissonance for any collection of frequencies is given by

$$D_F = \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n} d(f_i, f_j, l_i, l_j). \tag{9}$$

where D_F is the total dissonance generated from playing the frequencies $f_1, f_2, \ldots, f_n$ with respective loudnesses $l_1, l_2, \ldots, l_n$. The frequencies $f_1, f_2, \ldots, f_n$ could be the fundamental and overtone frequencies of a single note, interval, or chord. His model gives comparative dissonance values for any timbre, which is determined by both the number and the prominence of all frequencies within a musical sound.

To calculate loudnesses, Sethares made the assumption that the loudness level was equal to the SPL so that for a loudness l,

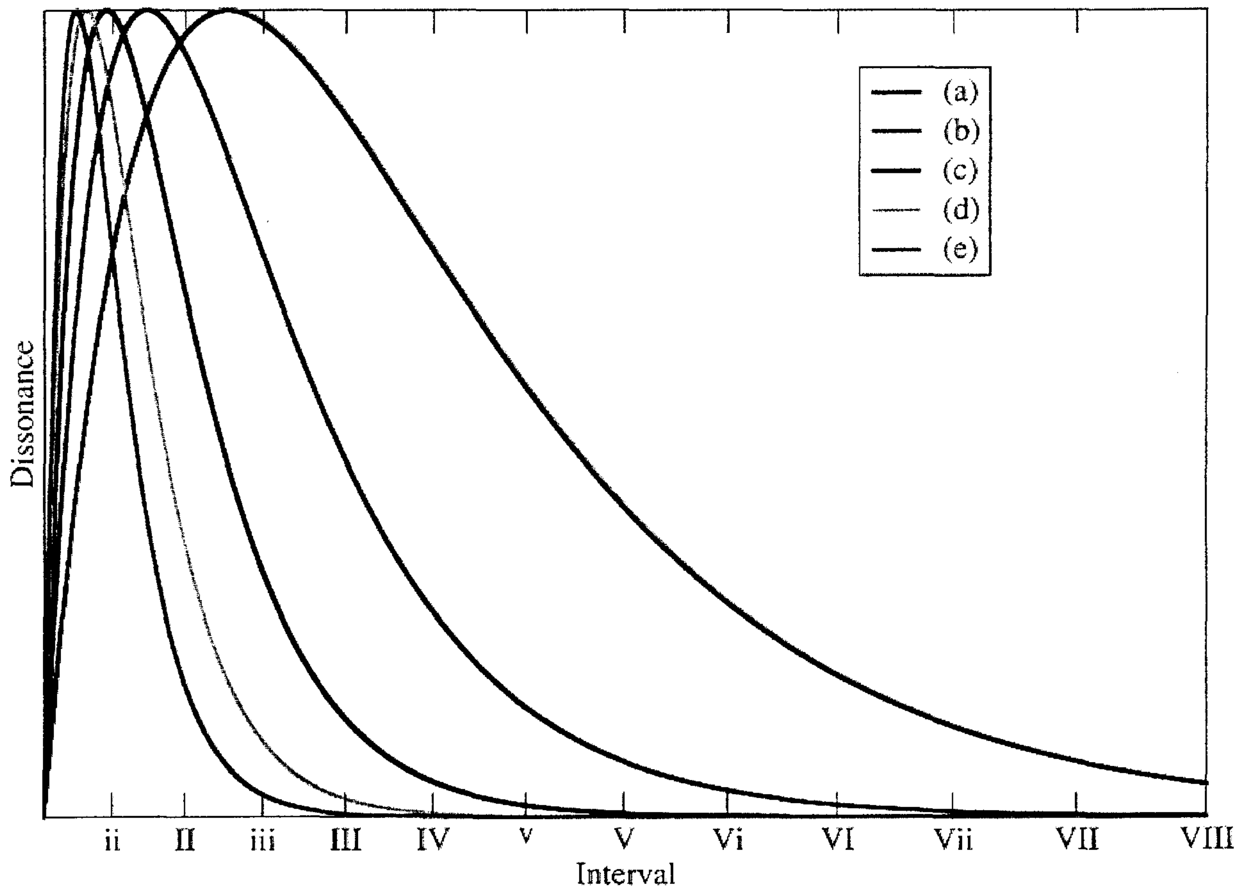
$$l = 2^{\frac{\text{SPL}-40\text{ dB}}{10\text{ dB}}}\text{ sones.} \tag{10}$$

The loudness level is only approximately equal to SPL between about 250 and 2000 Hz. Sethares had already factored in human perception of loudness when he used the Plomp and Levelt curves of perceived dissonance. For this reason, and because we only considered frequencies ranging between about 250 and 2000 Hz, loudness is approximately defined by equation (10).

A replication of Sethares's work using equation (9) is shown in Table 7 for the Plomp and Levelt fundamental frequencies.

Table 7 – Normalized dissonance curves for simple tones as a function of equally-tempered intervals.

Fundamental frequency	(a) 125 Hz	(b) 250 Hz	(c) 500 Hz	(d) 1000 Hz	(e) 2000 Hz
Frequency with maximum dissonance	144.8 Hz	272.2 Hz	526.9 Hz	1036.4 Hz	2055.4 Hz
Beats/sec of maximum dissonance	19.8	22.2	26.9	36.4	55.4



Lower fundamental frequencies require a larger interval to remove dissonance. Because most instruments consist of overtones and not just simple tones,[13] the dissonance curves of complex tones will be more complicated than the curves in Table 7.

To illustrate the dissonance curve for complex tones, Sethares used a timbre where the first overtone had a loudness of 88% the fundamental, and each successive overtone had a loudness of 88% the previous overtone. Table 8 shows the dissonance curves for this timbre.

Table 8 – Dissonance for complex tones with a fundamental frequency of 261.6 Hz. Consonant points (green) and dissonant points (red) have been marked.

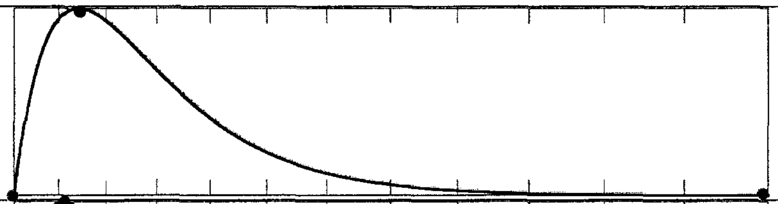
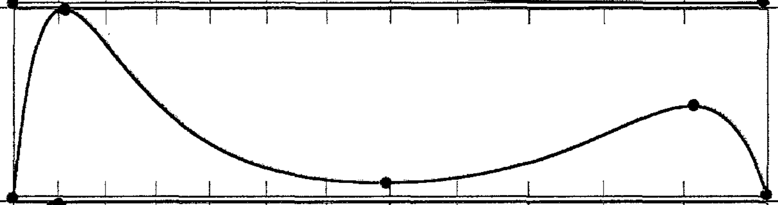
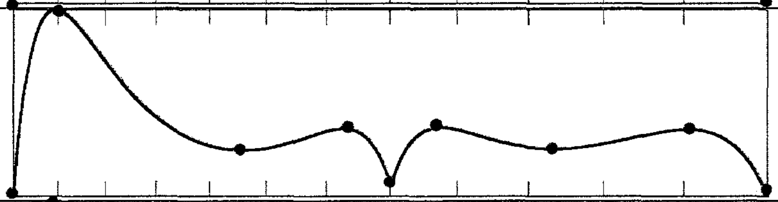
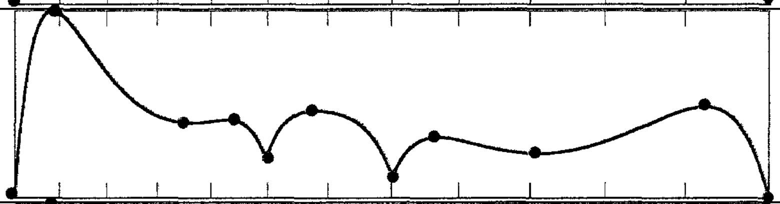
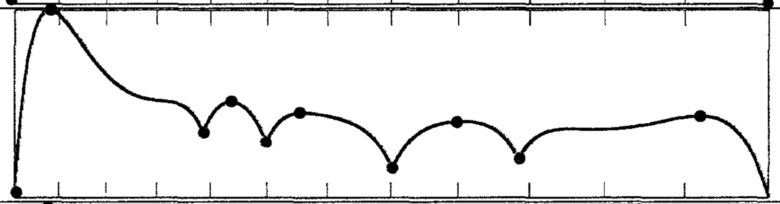
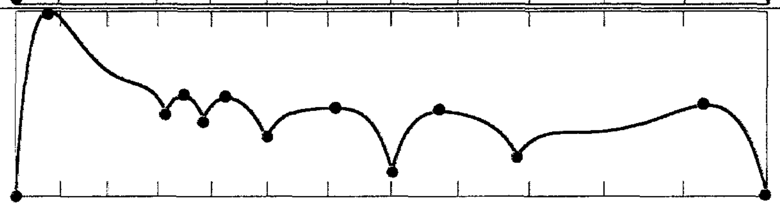
Over-tones	Dissonance	Consonant ratios	Just interval
0		1.00 – 1:1 2.00 – 2:1	Unison Octave
1		1.00 – 1:1 1.48 – 3:2 2.00 – 2:1	Unison Fifth Octave
2		1.00 – 1:1 1.31 – 4:3 1.50 – 3:2 1.72 – 7:4 2.00 – 2:1	Unison Fourth Fifth Harmonic 7th Octave
3		1.00 – 1:1 1.23 – 5:4 1.33 – 4:3 1.50 – 3:2 1.69 – 5:3 2.00 – 2:1	Unison Major 3rd Fourth Fifth Major 6th Octave
4		1.00 – 1:1 1.25 – 5:4 1.33 – 4:3 1.50 – 3:2 1.67 – 5:3 2.00 – 2:1	Unison Major 3rd Fourth Fifth Major 6th Octave
5		1.00 – 1:1 1.20 – 6:5 1.25 – 5:4 1.33 – 4:3 1.50 – 3:2 1.67 – 5:3 2.00 – 2:1	Unison Minor 3rd Major 3rd Fourth Fifth Major 6th Octave

Table 8 reveals that a timbre with more overtones has more points of dissonance and consonance. Though the consonant points do not always align perfectly with the equally-tempered intervals, the consonant points are roughly around both the equally-tempered and just intervals. The final plot in Table 8 contains all five overtones and has points of consonance in all the intervals Helmholtz deemed as consonant.

V. METHODS

Sound was recorded from both a Yamaha YPT-400 portable keyboard and a Steinway & Sons grand piano (serial number: 194426; approximate date of construction: 1917-1918). A Samson Q1U-USB microphone was used for the grand piano and connected to an Acer Aspire 5610 laptop. The keyboard output was connected directly into the microphone input of the laptop so that unwanted background noise would be eliminated.

The electronic keyboard was used because it produces the same sound for each key independent of the force with which the key was struck, thus, providing reproducible sounds. The keyboard was set to the Portable Grand Piano feature with touch sensitivity turned off. The volume was set to 50%, the reverb to 0%, and the chorus to 0%. No sustain pedal was used, and the notes were depressed until the sound faded to inaudible.

For recording the grand piano, the microphone was placed directly underneath the piano, approximately centered with both the piano's length and width. The microphone was placed on a small tripod stand and pointed up vertically, having an effective height of 20 cm. Because the sounding board is located under the piano, the sound was louder underneath the piano than above it. The room used for recording was 3.17 m (length) × 2.21 m (width) × 2.80 m (height) with padded walls to decrease acoustical reflections.

The recording software used for both the piano and the keyboard was Audacity ® 1.3.12-beta (Unicode). A freely available software, Audacity allows for easy exporting of sound recordings as "wav" files. The microphone level for the computer input for both the piano and the keyboard was set to 50%. Each recording took 8 s of sound. Piano keys were held down, allowing the string to vibrate freely until the sound decayed to inaudible. The sustain pedal was not depressed so that the other strings would not resonate sympathetically. The sound took about 8 s to fade to inaudible for the grand piano, whereas the notes tended to fade to inaudible after about 4 s for the keyboard. Generally, about 1 to 1.5 s were allowed to pass between beginning the recording and pressing the piano keys.

The intervals and chords we recorded were within one octave, all with a bass note of C4 (middle C, 261.6 Hz). Tables 1 and 2 show the intervals and chords recorded. Three trials of each interval and of each chord were recorded for each instrument. The sampling frequency used in all instances was 22,050 Hz.

The recordings were exported as wav files, which contain uncompressed sound data.[14] Within a wav file, voltage readings from the microphone are converted to arbitrary voltage units between -1 and 1 through analog-to-digital conversion. After exporting the wav files, the data were extracted using sox, freely available sound software for the Linux operating system. Output data consisted of a two-column array listing recording time and arbitrary voltage readings.

The converted sound data was then read into a FORTRAN code (Appendix B:2). The silence prior to the playing of the note was truncated by only considering data after the wav file voltages were greater than or equal to 0.05. An FFT was used (Appendix B:3) to output the power spectrum of the frequencies present in the sound data. The FFT requires the number of samples to be 2^n, where n is an integer. The sampling rate and the recording time determine the largest n that can be used. We used $n = 15$, which gave 2^{15} samples or about 1.49 s of recorded sound. This number of samples was used to focus on the beginning of the sound sample when the sound was loudest rather than the longer trailing decay of the sound after the keys are struck. The higher frequencies die out quicker than the lower ones, so the overtones will be strongest during the beginning of the sound.[12]

The power spectrum frequency bin is $f_{bin} = 22{,}050/(2^{15}) \approx 0.673$ Hz, meaning that all frequencies ranging from 0-0.673 Hz are labeled as having a frequency of 0.673 Hz on the power spectrum. This frequency resolution is more than adequate as the smallest frequency difference we considered is the semitone interval from C4 to Db4: $261.6 \times 2^{1/12}$ Hz -261.6 Hz $\approx$ 15.6 Hz.

Frequencies outside of the musical range (20-20,000 Hz) were given a power spectrum amplitude (PSA) of 0 to eliminate noise not associated with the piano. The power spectrum amplitudes of all the chords and intervals were divided by the highest power spectrum amplitude (PSA_{max}) found amongst all chord and interval power spectrums. By dividing by PSA_{max}, the power spectrum amplitudes were normalized and ranged from 0 to 1. Only one power spectrum had a PSA $= 1$, the spectrum corresponding to PSA_{max}.

To find significant peaks in the data, a loop was run in which each frequency bin was inspected to see if its PSA was greater than the PSA of the adjacent frequency bins. Because the peaks in the power spectrum are sometimes a collection of small peaks around the highest peak, it was necessary to look at many bins to the left and to the right of the current bin to ensure that the bin the code was inspecting was a significant peak. The code searched 12.11 Hz to the left and to the right of each frequency bin to determine where a significant peak occurred.

The final step to ready the recording samples for the dissonance equation was to convert the power spectrum to loudness. We used equation (4) to compare multiple tones' sound pressure levels using the ratio of pressures. The ratio of PSA_i to PSA_{max} is $\frac{PSA_i}{PSA_{max}} = \frac{p_i}{p_{max}}$. Our expression for finding SPL_i is then

$$\mathrm{SPL}_i = \mathrm{SPL}_{max} + 20 \cdot \log_{10}\left(\frac{PSA_i}{PSA_{max}}\right) \text{ dB.} \tag{11}$$

We assigned the sound pressure level of the loudest frequency present in any of our samples, to a comfortable listening level for music such that $\mathrm{SPL}_{max} = 70$ dB. [15] The lower limit of human hearing for simple tones for the musical ranges in consideration (250-2000 Hz) is around 25 dB, so any sound pressure levels less than 25 dB were set to 0 dB. Loudness was then found using equation (10) where

$$l = 2^{3+2\cdot\log_{10}\left(\frac{\mathrm{PSA}}{\mathrm{PSA}_{max}}\right)} \text{ sones.} \tag{12}$$

Figure 7 shows the conversion process from a power spectrum with arbitrary units to a sound pressure level power spectrum, and then to a loudness power spectrum.

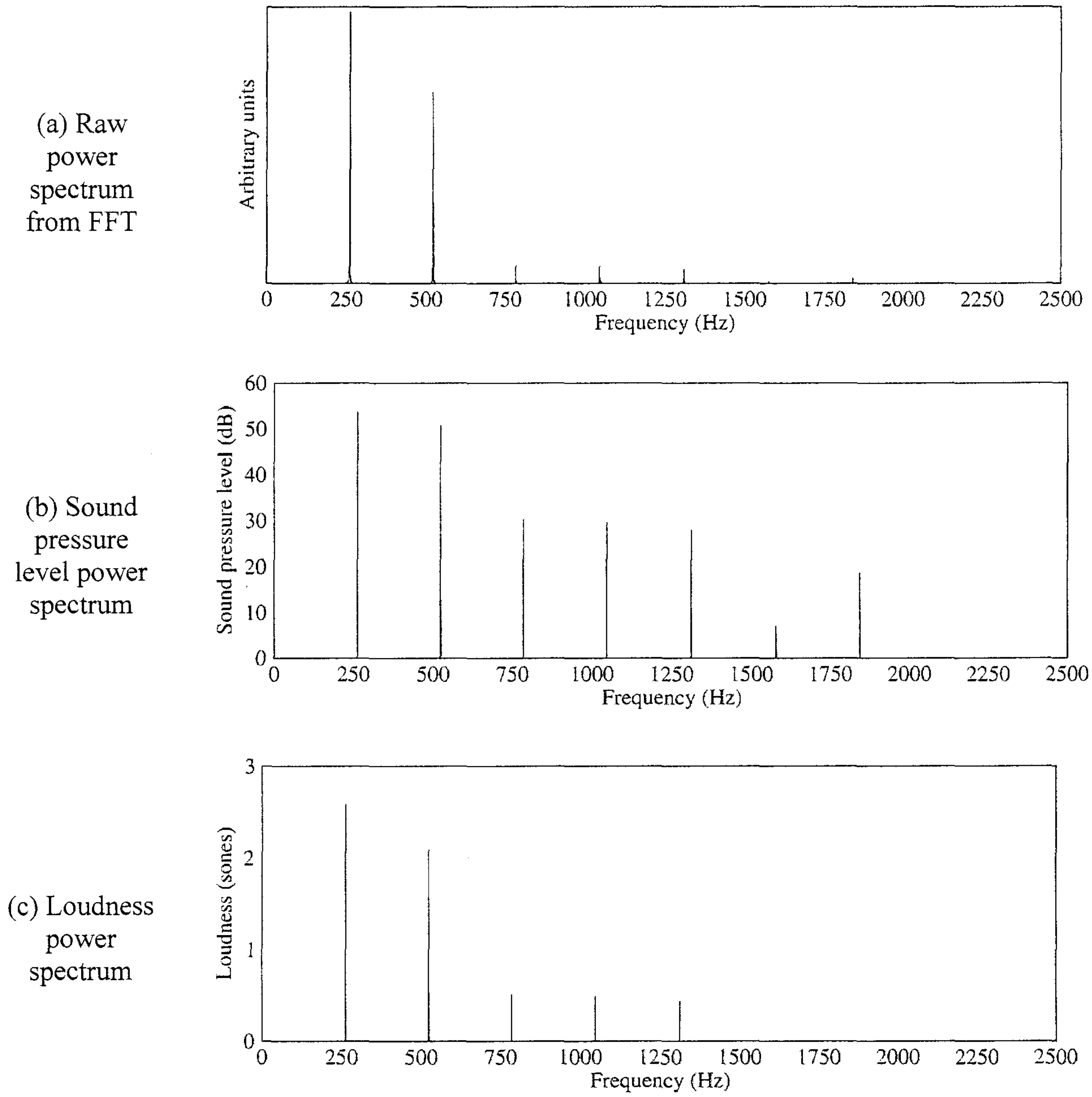


Figure 7 – Converting the raw power spectrum to loudness.

The loudness spectrum in Figure 7 (c) reveals that frequencies that have low relative amplitudes in (a) , such as the peaks around 750, 1000, and 1250 Hz hold higher relative loudness amplitudes. Frequencies with a sound pressure level less than 25 dB were truncated, such as the peaks just above 1500 Hz and above 1750 Hz.

After converting to loudness, the data were input into the dissonance equation, equation (9). Comparing the mean dissonance values provided a ranking of dissonance for chords and intervals.

III. RESULTS

Intervals

The keyboard and grand piano results for mean dissonance values are shown in Figure 8.

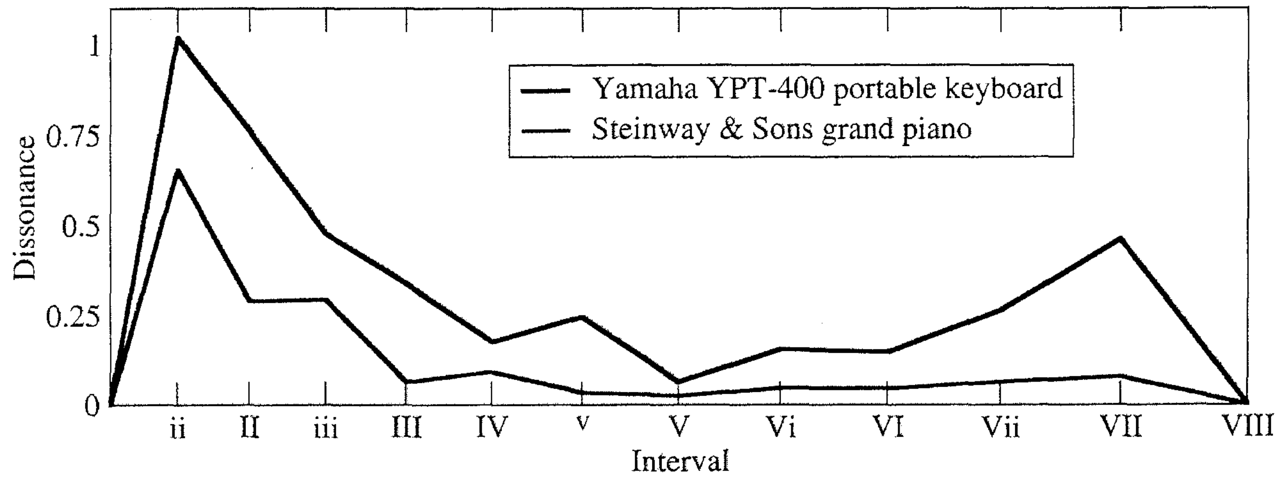
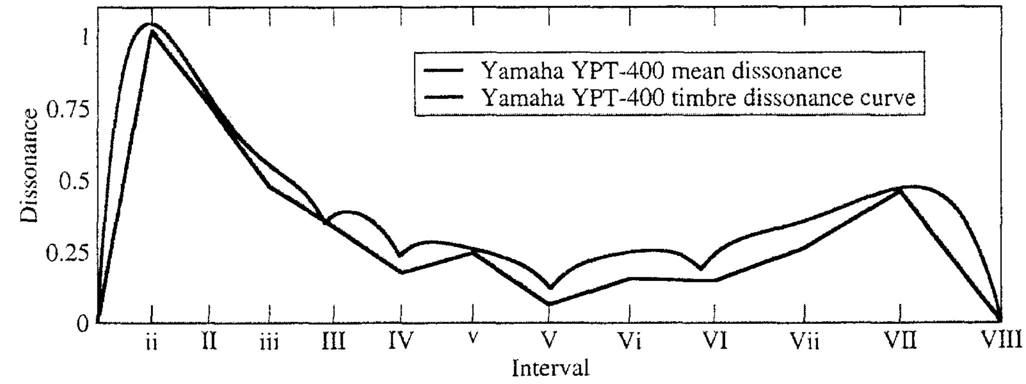


Figure 8 – Mean dissonance value comparison between keyboard and grand piano.

The keyboard and grand piano are similar, but certain peaks of dissonance for the keyboard, such as the diminished fifth (v), are not dissonance peaks on the grand piano. The grand piano holds little distinction between dissonance values from the intervals between the diminished fifth and the major seventh, but the keyboard has clearly-defined differences in these intervals.

The continuous timbre dissonance curve was generated from the loudness power spectrum of the note C4. The dissonance timbre curve shows what the dissonance curve would look like if every infinitesimally small frequency from C4 to C5 had the same timbre. Figure 9 compares the experimentally-found mean dissonance curves with the timbre dissonance curves.

(a) Keyboard dissonance curves



(b) Grand piano dissonance curves

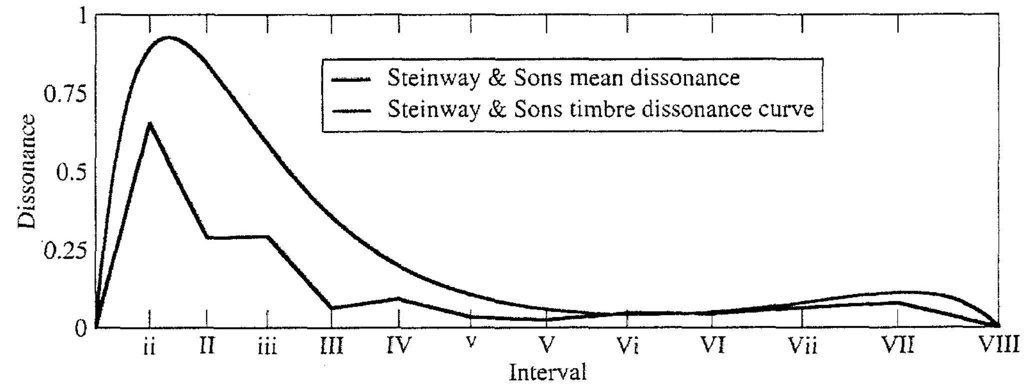


Figure 9 – Mean dissonance curve vs. timbre curve for the grand piano and keyboard.

The keyboard's timbre curve and experimentally-found curve are similar. This shows that the keyboard dissonance can be approximated with the timbre dissonance curve. The grand piano timbre curve differs from the experimentally-found curve for intervals up to the minor sixth. The grand piano dissonance cannot be approximated with the timbre curve.

The quantitative ranking of dissonance values for the intervals in Figure 8 is compared with Helmholtz' ranking in Table 9.

Table 9 – Interval dissonance ranking.

Rank	Keyboard		% of most dissonant interval	Grand piano		% of most dissonant interval	Helmholtz	Helmholtz ranking agrees with keyboard, grand piano, ranking for the following intervals: Key.	Piano
1	1.019	ii	100.00%	0.654	ii	100.00%	-		
2	0.762	II	74.80%	0.292	iii	44.69%	-		
3	0.477	iii	46.78%	0.289	II	44.22%	-		
4	0.461	VII	45.20%	0.092	IV	14.01%	-		
5	0.338	III	33.18%	0.078	VII	11.97%	-		
6	0.262	Vii	25.76%	0.063	III	9.69%	-		
7	0.245	v	24.07%	0.063	Vii	9.67%	iii	X	X
8	0.176	IV	17.28%	0.046	Vi	7.09%	III	X	
9	0.155	Vi	15.20%	0.045	VI	6.85%	VI		
10	0.147	VI	14.47%	0.034	v	5.14%	IV		
11	0.064	V	6.25%	0.024	V	3.70%	V	X	X
12	0.003	VIII	0.32%	0.001	VIII	0.08%	VIII	X	X

There is no universal ranking between the grand piano and the keyboard, which is supposed to simulate a grand piano. Neither instrument holds the same ranking of consonance as Helmholtz' speculated, although the keyboard differs only with the order of the fourth and the major sixth.

Note that the grand piano ranking in Table 9 shows its diminished fifth only slightly more dissonant than the fifth. As the diminished fifth is not considered a consonant interval, it is difficult to believe that the diminished fifth could be the third most consonant interval. To investigate the cause of this discrepancy, the loudness spectrum of both the fifth and the diminished fifth are analyzed in Table 10.

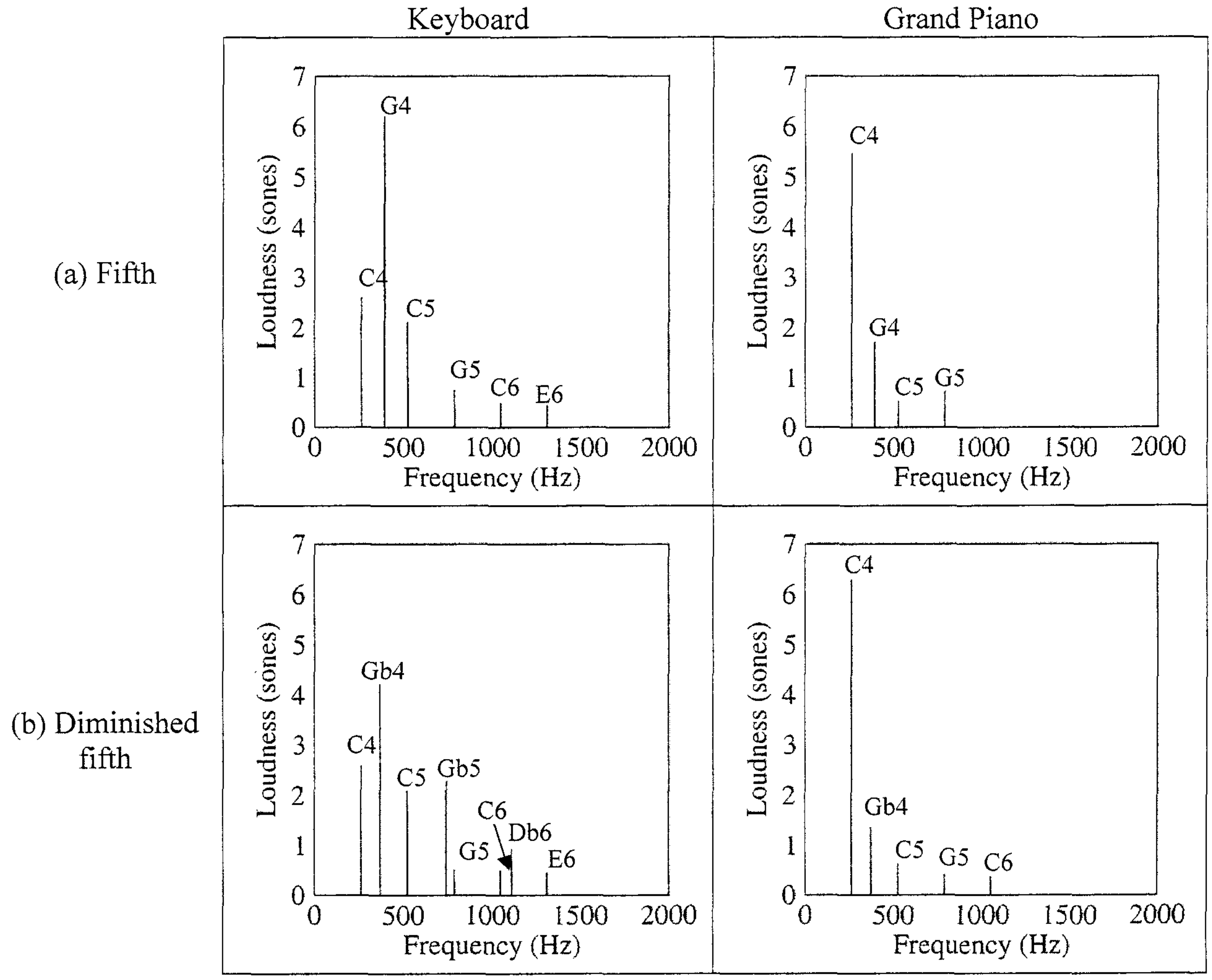


Figure 10 – Comparing the loudnesses of the fifth and the diminished fifth.

In Figure 10 (a), for the fifth interval, both the keyboard and the piano have a significant number of loudness peaks (six and four, respectively). The grand piano's fundamental (C4) dominates whereas the fifth (G4) dominates for the keyboard. Table 9 shows that the dissonance values for the fifth interval in both instruments are similar and that the fifth is the second most consonant interval for both instruments.

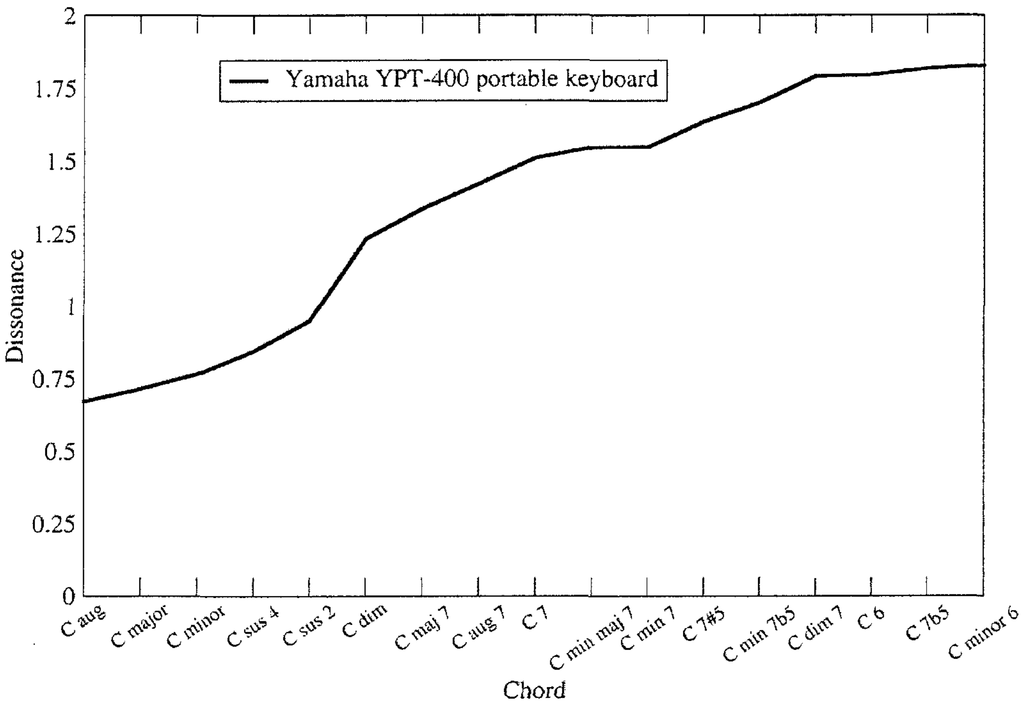
The discrepancy between the grand piano's diminished fifth and the keyboard's diminished fifth are shown Table 10 (b). There are eight total frequencies present in the keyboard's diminished fifth whereas the piano has only five. Of more importance than the number of loud frequencies is the presence of dissonant frequency combinations. Two such combinations for the keyboard are Gb5-G5 and C6-Db6. Both of these combinations are a semitone apart, the most dissonant interval. No apparent frequency combinations appear to be close enough to cause too much dissonance in the grand piano's diminished fifth spectrum. The closest combination of loud peaks for the piano is C4-Gb4 with a beat frequency of 108.4 Hz, which is too large to be dissonant.

The grand piano did not produce as many overtones as the keyboard. The keyboard programming includes these overtones regardless of recording conditions, but the grand piano's output will be different with every key strike. Recording underneath the piano as opposed to above the piano had an effect on the frequencies detected by the microphone. Above the piano the overtones were more prominent, whereas below, the fundamental frequency dominated. These are possible explanations as to the differences between the power spectrum of the grand piano and that of the electronic keyboard. The keyboard and the grand piano's interval ranking differed, and this difference becomes more strongly apparent with the chords that involve more piano keys.

Chords

The chord dissonance ranking for the keyboard and the grand piano reveals an incongruity in order similar to the interval dissonance ranking. The chords' dissonance levels are plotted and ordered from most consonant to most dissonant in Figure 10.

(a)
Keyboard
dissonance
curves



(b)
Grand piano
dissonance
curves

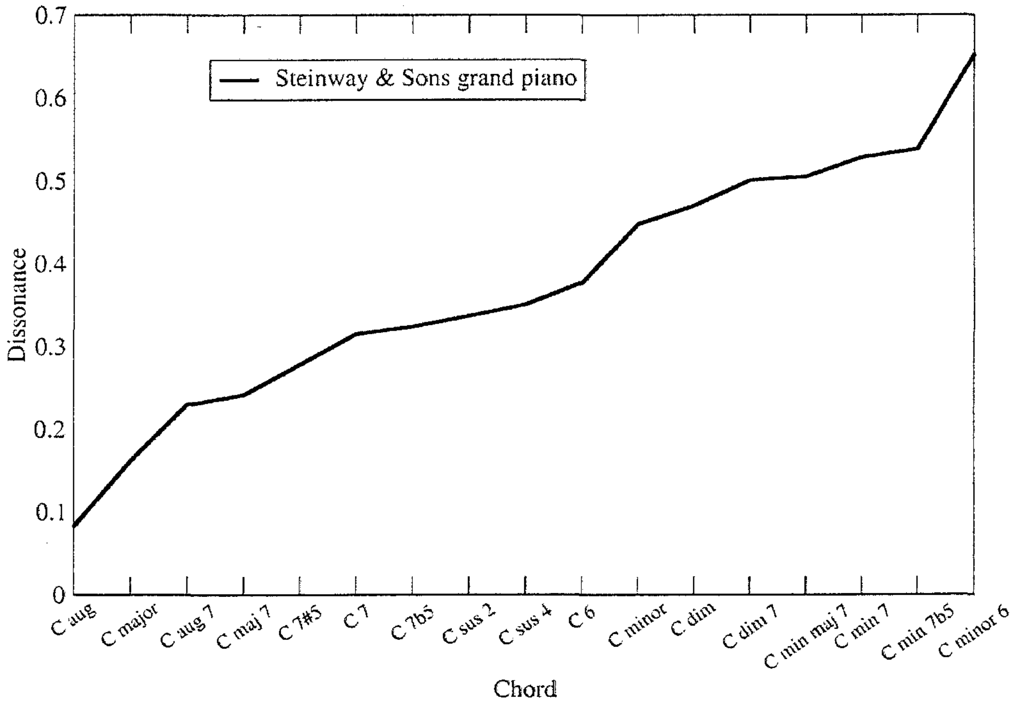


Figure 11 – Chord dissonance for (a) the keyboard and (b) the grand piano.

Figure 11 reveals a gradual transition from the most consonant to most dissonant chord for both the keyboard and the grand piano. Both instruments agree on the most consonant chord (C augmented), the second most consonant chord (C major), and the most dissonant chord (C minor 6). Apart from these similarities, the rankings differ in most respects. Table 10 shows the quantitative dissonance values for the chords.

Table 10 – Chord dissonance ranking.

Rank of Dissonance	Keyboard			Grand piano		
	Value	Chord	% of most dissonant chord	Value	Chord	% of most dissonant chord
1	1.828	C minor 6	100.00%	0.652	C minor 6	100.00%
2	1.818	C 7b5	99.47%	0.538	C min 7b5	82.49%
3	1.796	C 6	98.27%	0.528	C min 7	80.99%
4	1.792	C dim 7	98.03%	0.505	C min maj 7	77.45%
5	1.701	C min 7b5	93.04%	0.501	C dim 7	76.80%
6	1.636	C 7#5	89.51%	0.470	C dim	72.09%
7	1.548	C min 7	84.69%	0.448	C minor	68.68%
8	1.547	C min maj 7	84.64%	0.377	C 6	57.88%
9	1.511	C 7	82.65%	0.351	C sus 4	53.77%
10	1.423	C aug 7	77.83%	0.337	C sus 2	51.68%
11	1.337	C maj 7	73.14%	0.324	C 7b5	49.70%
12	1.233	C dim	67.47%	0.315	C 7	48.32%
13	0.950	C sus 2	52.00%	0.278	C 7#5	42.63%
14	0.845	C sus 4	46.23%	0.241	C maj 7	37.01%
15	0.766	C minor	41.93%	0.230	C aug 7	35.19%
16	0.715	C major	39.11%	0.161	C major	24.71%
17	0.672	C aug	36.74%	0.083	C aug	12.68%
Mean	1.360	-	74.4%	0.373	-	57.2%

Table 10 shows that there is no universal ranking for the chords. The keyboard holds a higher percentage mean than the grand piano, suggesting that, the keyboard's chords are more dissonant than the piano's. Each step is relatively small, which shows that the dissonance scale is a grayscale rather than a black and white scale.

The Sethares dissonance equation gives higher dissonance values for louder timbres. Figure 10 shows that the keyboard has a louder timbre than the grand piano. So is a quiet dissonant chord such as the grand piano's C min 7b5 (dissonance value: 0.538) more consonant than the louder keyboard's consonant C major (dissonance value: 0.715)? The percentages of the most dissonant chord suggest the answer to this question is "no" because the grand piano's C min 7b5 (dissonance percent: 82.49%) is more than twice the percentage of the keyboard's C major (dissonance percent: 39.11%). The dissonance values are then, as Sethares notes,[12] arbitrary units that cannot be compared between different instruments. Because the dissonance percentages are normalized to the most dissonant chord, the percentages provide a means for comparing dissonance between two instruments.

IV. CONCUSION

To answer the questions posed in the Introduction, physics can explain sound and beats, but the human ear determines dissonance perception. Plomp and Levelt's studies determined for simple tone intervals that the number of beats resulting in the most perceived dissonance depends on an interval's fundamental frequency. This information alone is enough to determine that a ranking of dissonance of intervals or chords cannot be universal. Even for a single instrument the broad frequency range will keep a ranking of chords or intervals from being consistent throughout the instrument. A major third might be consonant in the upper registers of the piano, but when played in the lower bass notes, a major third can be the most dissonant interval. Dissonance rankings are then a function of timbre as well as frequency.

Interpreting dissonance data can be a speculative task because assumptions have been made to get the dissonance values. For starters, the Fletcher and Munson curves of equal loudness were not used. As was previously stated, for frequencies around 1000 Hz, the effects of the equal loudness curves are negligible, so this assumption does not greatly affect the results. Another assumption was that the frequency with the greatest amplitude out of all the power spectrums had essentially an SPL = 70 dB. Equation (12) shows that assigning SPL_{max} = 70 dB scales every loudness value by the same amount. If instead, we had assigned SPL_{max} = 80 dB, each loudness would be multiplied by 2, which would have resulted in higher dissonance values, but dissonance rankings stay the same regardless of loudness scaling.

The choice of a 70 dB maximum sound pressure level assignment does, however, affect which power spectrum peaks were truncated because sound pressure level peaks less than 25 dB were removed. After all spectrums were normalized to the highest peak found in all the spectrums, a 70 dB SPL assignment cuts off all normalized peaks under 0.56% of the highest peak of all the power spectrums. An 80 dB assignment cuts off peaks 0.17% of the highest peak of all the power spectrums. A 10 dB increase SPL_{max} assignment results in considering peaks of about a third the power spectrum amplitude of the peaks cut off without the increase. Choosing 70 dB has the potential of cutting off frequencies that would alter the dissonance values. Although 70 dB is a comfortable music listening level, the sound produced by the maximum peak may have been louder than a comfortable music listening level.

Microphone placement influences what sounds are heard and what acoustical reflections are recorded. In the case of the grand piano, microphone placement explains why the overtones were not prominent in the power spectrum. Further tests placing the microphone above the piano would better represent how sound is heard since the listener hears sound above the piano rather than below.

Only one octave, between C4 and C5, was considered for all of these recordings, which leads to a limited window of available data. A more intensive and complete study could classify the chords and intervals based on every note on the piano, a total of 88 notes.

An alternate method for creating the dissonance curves would be to record each note within an octave individually and add the wave forms or power spectrums together for a combination of notes. This technique may not be as accurate as recording the entire sound because it is unclear whether or not the resulting power spectrum would be the same for both processes. What this approach does provide is a means for simulation of any interval or chord. Average dissonance rankings from all the intervals or chords possible on the instrument could be found with a computer program, thus eliminating the extensive recording time.

This work was conclusive in ranking chords and intervals, but the ranking was not definitive. Because dissonance depends on frequency, loudness, as well as human perception, no universal ranking could be determined. This work does show that there are shades of gray in the scale from consonant to dissonant.

APPENDIX A – Chord abbreviation descriptions.

min = minor, contains a minor interval.

maj = major, contains a major interval.

b = flat, semitone lower.

= sharp, semitone higher.

dim = diminished, higher note of interval is a semitone lower.

aug = augmented, higher note of interval is a semitone higher.

sus = suspension, musical term referring to chords that are nonharmonic or unresolved.

b5 = flat fifth, contains a diminished fifth interval.

#5 = sharp fifth, contains an augmented fifth (minor sixth) interval.

7 = seventh, contains a seventh interval,

6 = contains a sixth interval.

APPENDIX B – FORTRAN codes used for generating figures, tables, and dissonance rankings.

1. Dissonance.f

```
***    Dissonance.f
```

```
***   Michael Blatnik
***   Senior Thesis
***   4/21/11
***
***   The dissonance curves from Figure 9 and Table 8 were generated
***   with this program.

      Program Dissonance

      implicit none

      integer i,n,j,k,t,alpha,nn

      parameter(t=13800)
      parameter(n=6)
      parameter(nn=12)

      double precision freq(0:m),amp(0:m),g(0:m),ratio(0:t),freq2(0:t)
      double precision d(0:t),h,s,c1,c2,a1,a2,s1,s2,li,lj,lij,ee,dstar
      double precision fdif,fmin,arg1,arg2,exp1,exp2,dnew,fund,fundamp
      double precision upper,beats,sqrttwo,Pref,big,zero,maxdissfreq
      double precision maxdissratio

      fund=261.626D+00 !Used for Table 5.
*      fund=125.0D+00*(2.0D+00)**(4) !Used for Figure 10.

      fundamp=10.0D+00          !Loudness of 10 sones for fundamental.
      ee=0.88D+00               !88%
      h=12000.0D+00             !Number of divisions per octave.
      upper=2.3D+00*h/2.0D+00   !Upper limit number at ratio of 2.3.
      sqrttwo=dsqrt(2.0D+00)    !Sqrt(2)
      zero=0.0D+00              !Zero
      big=zero                  !Used for finding maximum dissonance.

***   Constants used for dissonance equation. Used in equations 5 and 6.
      dstar=0.24D+00            !d*
      s1=0.0207D+00             !s1
      s2=18.96D+00              !s2
      a1=-3.51D+00              !-a
      a2=-5.75D+00              !-b
      Pref=20.0D-06             !P_ref, reference pressure 20*10^-6 Pa

***   Loudness amplitudes used for adding in up to five harmonics.
***   Each successive harmonic has 88% the loudness of the previous.
      amp(1)=fundamp
      amp(2)=amp(1)*ee
      amp(3)=amp(2)*ee**2
      amp(4)=amp(3)*ee**3
      amp(5)=amp(4)*ee**4
      amp(6)=amp(5)*ee**5

***   Defines the frequencies of the five harmonics.
      do i=1,n,1
         freq(i)=fund*i
```

```
      enddo

***   Output files.
      open(unit=10,file='dissonance.dat')
      open(unit=20,file='equaltempdis.dat')

***   This larger loop figures out the ratio values needed for the
***   independent variable of the dissonance curve as well as
***   creating dissonance values for the curve.
      do alpha=0,upper,1
         d(alpha)=0.0D+00 !Zeros the initial dissonance for each alpha.

***   ratio(alpha) defines the octave in terms of the equal-tempered
***   scale. A semitone would be defined as alpha=1000, a second as
***   alpha = 2000, etc...
         ratio(alpha)=(2.0D+00)**(dble(alpha)/h)

***   Defines secondary fundamental frequency with 5 harmonics for the
***   ratio value ratio(alpha).
         freq2(alpha)=ratio(alpha)*fund !Secondary fundamental.
         do k=1,n,1
            g(k)=freq(k)*ratio(alpha) !6 frequencies to be used.
         enddo

***   Runs the dissonance curve by looking at every possible combination
***   of both the bass fundamental frequency at 261.626 Hz, it's
***   harmonics, and the fundmental and harmonics of the secondary
***   frequency being considered.
         do i=1,n,1
            do j=1,n,1
               li=amp(i)        !Loudness of first frequency input.
               lj=amp(j)        !Loudness of second frequency input.
               lij=min(li,lj)   !Minimum loudness.
               fmin=min(g(j),freq(i)) !Minimum frequency.
               s=dstar/(s1*fmin+s2) !s from equation 6.
               fdif=dabs(g(j)-freq(i)) !Frequency difference.
               arg1=a1*s*fdif !Argument in exponent 1.
               arg2=a2*s*fdif !Argument in exponent 2.
               exp1=dexp(arg1) !Exponent 1 from equation 5.
               exp2=dexp(arg2) !Exponent 2 from equation 5.
               dnew=lij*(exp1-exp2) !Calculation of added dissonance.
               d(alpha)=d(alpha)+dnew !Adds all dissonances together.
            enddo
         enddo
      enddo

      do alpha=0,upper,1

***   Calculates and prints where the points of maximum consonance are.
         if (d(alpha-1).gt.d(alpha).and.d(alpha+1).gt.d(alpha))then
            Print *, 'Max Consonance  at ', freq2(alpha),', with ratio '
     &           ,ratio(alpha),'.'
         endif
```

```
***    Calculates ratio of maximum dissonance.
          if(d(alpha).gt.big)then
             maxdissratio=ratio(alpha)
             big=d(alpha)
          endif
       enddo

***    Writes dissonance curve to file: ratio vs. dissonance.
       do alpha=0,upper,1
          write(10,*) ratio(alpha),' ',d(alpha)
       enddo
       maxdissfreq=maxdissratio*fund !Frequency of maximum dissonance.
       beats=dabs(fund-maxdissfreq) !Beats between max freq and fund.
       print *,'Maximum dissonance at ratio ',maxdissratio,', ',
     &     'frequency of',maxdissfreq,' Hz, and ',beats,' beats.'

***    Finds equal-tempered interval ratios and writes the
***    ratios and dissonance values to a file.
       do alpha=0,upper,1
          do i=0,h,1
             if(ratio(alpha).eq.2**(dble(i)/nn))then
                write(20,*)ratio(alpha),' ',d(alpha)
             endif
          enddo
       enddo
       end
```

2. IntervalChordDissonance.f

```
***    IntervalChordDissonance.f
***    Michael Blatnik
***    Senior Thesis
***    4/21/11
***
***    This is the main code that uses converted wav files to create
***    all the power spectrums, calculate the dissonances, and export
***    the data needed to make most of the figures in the thesis.

       program IntervalChordDissonance

       implicit none

       integer i,j,n,k,m,l,numpeaks,count,start

       parameter(n=2**15)!Number for FFT (power of 2)

***    Number of input files, 42 for intervals, 51 for chords.
       parameter(m=42)
*** Starting number for input files (4 for intervals because of 3 trials
*** of silence, 1 for chords).
      parameter(start=4)

       double precision buffer(1:2*n),freq(1:2*n,0:m),freq1,dt,blank
       double precision P(1:2*n,0:m),V(1:2*n),big,fund,limit,array(0:n)
```

```
      double precision bigfreq,cutoff,sum,onep,sig(1:2*n,0:m)
      double precision power(1:2*n,0:m),beats,f1,f2,l1,l2,d(1:m),l12
      double precision a,b,s1,s2,dstar,s,highcutoff,increase,onefive
      double precision A1,A2,sqrttwo,Pe1,Pe2,Pref,arg1,arg2,fdif
      double precision numlines(0:n),sigfreq(1:2*n,0:m),peaks(0:n)
      double precision ratio(0:m),samplingfreq,SPL,loud(1:2*n,0:m)
      double precision medbig,meanbig,dmed(0:m),davg(0:m),highestpeak
      double precision biggestpeak


      character*80 info, inputfile, outputfile,time,powerfile,sigpeak
      character*80 testfile,newpower,meandissonance,mediandissonance
      character*80 newsigpeak,loudpower

***   Constants
      samplingfreq=22050.0D+00 !Sampling frequency of 22,050 Hz.
      onep=0.01D+00 !One percent.
      onefive=0.005D+00
      fund=261.6D+00 !Fundamental Frequency 261.626Hz C4
      cutoff=0.005D+00 !One half of a percentage point
      limit=20.0D+00
      sqrttwo=dsqrt(2.0D+00) !Sqrt(2)
      k=0
      dt=dble(1)/samplingfreq !Time step.
***   Highest FFT peak found by running all four renditions of program.
***   This is used to scale down all the FFTs.
      highestpeak=2699194.66D+00

***   Files to write.
      open(unit=70,file="mediandissonanceratio.dat")
      open(unit=71,file="mediandissonance.dat")
      open(unit=75,file="meandissonanceratio.dat")
      open(unit=76,file="meandissonance.dat")

***   Sets up an array freq(i) of double values for the real and
***   imaginary components of the FFT. Size of array: 2*n.
      do j=1,m,1
         do i=1,2*n-1,2
            freq1=dble(i-1)/(dt*dble(n)*2.0D+00)
            if (i.lt.n+1)then
               freq(i,j)=freq1
               freq(i+1,j)=freq(i,j)
            endif
            if (i.eq.n+1)then
               freq(i,j)=freq1
               freq(i+1,j)=-freq(i,j)
            endif
            if (i.gt.n+1)then
               k=k+4
               freq(i,j)=-freq(i-k,j)
               freq(i+1,j)=freq(i,j)
            endif
         enddo
      enddo
```

```
***   Main loop that runs through each recording sample at a time.
***   Each converted .wav file is read and output to a cleaned up data
***   files. The FFT.f program is then called and a power spectrum is
***   output. The power spectrum is cleaned-up by zeroing all values
***   that are less than 0.5% the value of the largest peak in each
***   spectrum, all values below 99% of the fundamental frequency, and
***   above 6000 Hz.
      do j=start,m,1

***   Creates a file for each input sound file and output file for each
***   j.
         write(inputfile,'(a,i2.2,a)') "input_",j,".dat"
         write(outputfile,'(a,i2.2,a)') "output_",j,".dat"
         write(powerfile,'(a,i2.2,a)') "power_",j,".dat"
         open(unit=10, FILE=inputfile)
         open(unit=20, FILE=outputfile)
         open(unit=60,file=powerfile)

***   Reads first two lines of code which contain texts to ignore them.
         do i=1,2,1
            read(10,*) info
         enddo

***   Reads through the normalized sound sample file and once the sound
***   sample reaches one percent sends the program to 100 to begin
***   actual reading in of data.
         do i=3,n,1
            read(10,*) blank,array(i)
            if (dabs(array(i)).gt.onefive)then
               goto 100
            endif
         enddo

***   Reads in the time and normalized voltage signal from the sound
***   file.
100     do i=1,n,1
            read(10,*) time,V(i)

***   Creates a buffer array size 2n for input into the FFT routine
***   (size 2n), using the normalized voltages for odd entries and 0
***   for even entries.
            buffer(2*i-1)=V(i)
            buffer(2*i)=0.0D+00

***   Writes new data file with length n, time vs. normalized voltage.
            write(20,*) real(i)*dt,' ', V(i)
         enddo

***   Calls FFT subroutine and inputs buffer array, n, and 1.
         call FFT(buffer,n,1)

***   P(i) is the power spectrum array, which is derived by squaring
***   the real and imaginary parts and adding together.
```

```
          do i=1,2*n,2
             P(i,j)=buffer(i)**2+buffer(i+1)**2
          enddo

***    Creates data file power_##.dat for the power spectrum viewing.
          do i=1,n,2
             write(60,*)freq(i,j),' ',P(i,j)
          enddo

***    Finds highest peak in the current power spectrum.
          big=0.0D+00
          do i=3,n,2
             if (P(i,j).gt.big) then
                big=P(i,j)
                bigfreq=freq(i,j)
             endif
          enddo

***    Power spectrum clean-up.
          do i=1,n,2
             if (freq(i,j).lt.limit) then
                P(i,j)=0.0D+00
             endif

***    After normalizing to the highest peak of all the power spectrums,
***    the power spectrums essentially become pressure power spectrums,
***    measured in Pa. The highest peak of 1 has an SPL of 70 dB.
             P(i,j)=P(i,j)/highestpeak
          enddo
       enddo

       do j=start+1,m,1
          if (bigj(j).gt.bigj(j-1))then
             biggestpeak=bigj(j)
          endif
       enddo
       print *,biggestpeak

       do j=start,m,1
          do i=1,n,2
             P(i,j)=P(i,j)/highestpeak
          enddo

          write(sigpeak,'(a,i2.2,a)') "sigpeak_",j,".dat"
          write(newpower,'(a,i2.2,a)')"newpower_",j,".dat"
          write(loudpower,'(a,i2.2,a)')"loudpower_",j,".dat"
          write(SPLpower,'(a,i2.2,a)')"SPLpower_",j,".dat"
          open(unit=30,file=newpower)
          open(unit=40,file=SPLpower)
          open(unit=80,file=sigpeak)
          open(unit=90,file=loudpower)

          numpeaks=0
          peaks(j)=0.0D+00
```

```
***   Tedious method used to ensure that peaks surrounding the
***   significant peaks in the pressure power spectrum were not
***   considered significant. As the power spectrum array consists
***   of values of 0 for even i and a power for odd values, searching
***   for significant peaks had to look only at the odd values.
***   Essentially, this condition will only label a peak as a
***   significant peak if it is higher than all of the peaks
***   12.11 Hz to the right and to the left of the current P(i).
         do i=37,n,2
            if (P(i,j).gt.P(i-2,j).and.P(i,j).gt.P(i-4,j).and.P(i,j)
     &          .gt.P(i-6,j).and.P(i,j).gt.P(i-8,j).and.P(i,j)
     &          .gt.P(i-10,j).and.P(i,j).gt.P(i-12,j).and.P(i,j)
     &          .gt.P(i-14,j).and.P(i,j).gt.P(i-16,j).and.P(i,j)
     &          .gt.P(i-18,j).and.P(i,j).gt.P(i-20,j).and.P(i,j)
     &          .gt.P(i-22,j).and.P(i,j).gt.P(i-24,j).and.P(i,j)
     &          .gt.P(i-26,j).and.P(i,j).gt.P(i-28,j).and.P(i,j)
     &          .gt.P(i-30,j).and.P(i,j).gt.P(i-32,j).and.P(i,j)
     &          .gt.P(i-34,j).and.P(i,j).gt.P(i-36,j).and.P(i,j)
     &
     &          .gt.P(i+2,j).and.P(i,j).gt.P(i+4,j).and.P(i,j)
     &          .gt.P(i+6,j).and.P(i,j).gt.P(i+8,j).and.P(i,j)
     &          .gt.P(i+10,j).and.P(i,j).gt.P(i+12,j).and.P(i,j)
     &          .gt.P(i+14,j).and.P(i,j).gt.P(i+16,j).and.P(i,j)
     &          .gt.P(i+18,j).and.P(i,j).gt.P(i+20,j).and.P(i,j)
     &          .gt.P(i+22,j).and.P(i,j).gt.P(i+24,j).and.P(i,j)
     &          .gt.P(i+26,j).and.P(i,j).gt.P(i+28,j).and.P(i,j)
     &          .gt.P(i+30,j).and.P(i,j).gt.P(i+32,j).and.P(i,j)
     &          .gt.P(i+34,j).and.P(i,j).gt.P(i+36,j)
     &          )then

               SPL=70.0D+00+20.0D+00*dlog10(P(i,j))

               if (SPL.ge.25.0D+00)then
                  numpeaks=numpeaks+1
                  peaks(j)=peaks(j)+1.0D+00
                  loud(numpeaks,j)=(2.0D+00)**((SPL-40.0D+00)/10.0D+00)
                  sig(numpeaks,j)=P(i,j)
                  sigfreq(numpeaks,j)=freq(i,j)
                  write(90,*) freq(i,j),' ',loud(numpeaks,j)
                  write(30,*) freq(i,j),' ',P(i,j)
                  write(40,*) freq(i,j),' ',SPL
                  write(80,*) freq(i,j),' ',P(i,j)
                endif
            else
               P(i,j)=0.0D+00
               write(30,*) freq(i,j),' ',P(i,j)
               write(40,*) freq(i,j),' ',P(i,j)
               write(90,*) freq(i,j),' ',P(i,j)
            endif
         enddo
      enddo

***   This loop runs all the loudness power spectrums through the
```

```
***   dissonance equation. Essentially, this loop of the code is
***   the loop contained within the Dissonance.f code when solving
***   that uses the dissonance equation. Refer to Dissonance.f for
***   comments of this loops.
      do j=start,m,1
         d(j)=0.0D+00
         write(newsigpeak,'(a,i2.2,a)') "newsigpeak_",j,".dat"
         open(unit=95,file=newsigpeak)
         do i=1,peaks(j),1
            do l=1,peaks(j),1
               if(i.ne.l)then

                  write(95,*) sigfreq(i,j),' ',sigfreq(l,j),' ',
     &                  loud(i,j),' ',loud(l,j)

                  a=-3.51D+00
                  b=-5.75D+00
                  s1=0.0207D+00
                  s2=18.96D+00
                  dstar=0.24D+00

                  Pref=2.0D-05
                  f1=sigfreq(i,j)
                  f2=sigfreq(l,j)
                  l1=loud(i,j)
                  l2=loud(l,j)
                  fdif=dabs(f2-f1)

                  l12=min(l1,l2)
                  s=dstar/(s1*min(f1,f2)+s2)
                  arg1=a*s*fdif
                  arg2=b*s*fdif
                  increase=l12*(dexp(arg1)-dexp(arg2))
                  d(j)=d(j)+increase
               endif
            enddo
         enddo
         d(j)=d(j)*0.5D+00
      enddo

***   Finds the median of dissonance data for each set of 3 samples for
***   each interval/chord. For the chords code, j was set to 1.
      count=0
      do j=start,m,3
         if (d(j).lt.d(j+2).and.d(j+2).lt.d(j+1))then !1 3 2
            dmed(j)=d(j+2)
         elseif(d(j).lt.d(j+1).and.d(j+1).lt.d(j+2))then !1 2 3
            dmed(j)=d(j+1)
         elseif(d(j+1).lt.d(j).and.d(j).lt.d(j+2))then !2 1 3
            dmed(j)=d(j)
         elseif(d(j+2).lt.d(j).and.d(j).lt.d(j+1))then !2 3 1
            dmed(j)=d(j)
         elseif(d(j+1).lt.d(j+2).and.d(j+2).lt.d(j))then !3 1 2
            dmed(j)=d(j+2)
```

```
          elseif(d(j+2).lt.d(j+1).and.d(j+1).lt.d(j))then !3 2 1
             dmed(j)=d(j+1)
          endif

***   Generates the equally-tempered interval ratios.
          ratio(j)=(2.0D+00)**(dble(count)/dble(12))
          count=count+1
      enddo

      count=0

***   Finds the average of dissonance data for each set of 3 samples for
***   each interval/chord.
      do j=1,m,3
         davg(j)=(d(j)+d(j+1)+d(j+2))/(3.0D+00)
         count=count+1
      enddo
      do j=start,m,3
         write(70,*)ratio(j),' ',dmed(j) !Used only for intervals
         write(75,*)ratio(j),' ',davg(j) !Used only for intervals
         write(71,*)count,' ',dmed(j)
         write(76,*)count,' ',davg(j)
         count=count+1
      enddo

      close(unit=10)
      close(unit=20)
      close(unit=30)
      close(unit=40)
      close(unit=60)
      close(unit=70)
      close(unit=71)
      close(unit=75)
      close(unit=76)
      close(unit=80)
      close(unit=90)
      close(unit=95)
      end
```

3. FFT.f

```
***   FFT.f
***   Michael Blatnik
***   Senior Thesis
***   4/9/11
***
***   This subroutine uses the Fast Fourier Transform. It is referred to
***   in the program Power.f and IntervalChordDissonance.f. The FFT code
***   comes from Numerical recipes: the art of scientific computing, and
***   the program:
***   "Replaces [ddata] by its discrete Fourier transform, if [isign] is
***   input 1; or replaces [ddate] by NN times its inverse discrete
***   Fourier transform, if [isign] is input as -1. [ddata] is a complex
***   array of length NN or equivalently, a real array of length 2*NN. NN
```

```
***   MUST be an integer of 2."16

      subroutine four1(ddata,nn,isign)

      implicit none

      integer NN,j,N1,i,m,mmax,isign,istep

      double precision wr, wi, wpr, wpi, wtemp, theta
      double precision tempr, tempi
      double precision ddata(1:2*nn),pi2

      pi2=8.0D+00*datan(1.0D+00)

      N1=2*NN
      j=1

      do i=1,N1,2
         if(j.gt.i)then
            tempr=ddata(j)
            tempi=ddata(j+1)
            ddata(j)=ddata(i)
            ddata(j+1)=ddata(i+1)
            ddata(i)=tempr
            ddata(i+1)=tempi
         endif

         m=N1/2

 1       if((m.ge.2).and.(j.gt.m))then
            j=j-m
            m=m/2
            goto 1
         endif
         j=j+m
      enddo

      mmax=2

 2    If (n1.gt.mmax)then
         istep=2*mmax
         theta=pi2/dble(isign*mmax)
         wpr=-2.0D+00*(dsin(0.5D+00*theta))**(2.0D+00)
         wpi=dsin(theta)
         wr=1.0D+00
         wi=0.0D+00

         do m=1,mmax,2
            do i=m,n1,istep
               j=i+mmax
               tempr=dble(wr)*ddata(j)-dble(wi)*ddata(j+1)
               tempi=dble(wr)*ddata(j+1)+dble(wi)*ddata(j)
               ddata(j)=ddata(i)-tempr
               ddata(j+1)=ddata(i+1)-tempi
```

```
            ddata(i)=ddata(i)+tempr
            ddata(i+1)=ddata(i+1)+tempi
         enddo

         wtemp=wr
         wr=wr*wpr-wi*wpi+wr
         wi=wi*wpr+wtemp*wpi+wi
      enddo
      mmax=istep
      goto 2
   endif
   return
   end
```

4. Tone.f

```
***   Tone.f
***   Michael Blatnik
***   Senior Thesis
***   4/9/11
***
***   This code was used to generate Figures 1 and 2.

      Program Tone

      implicit none

      integer i,n

      parameter(n=100000)

      double precision puretone(1:n),complextone(1:n),pi,ee,fund,bigpure
      double precision bigcomplex,t,P,Pc,harmonic(1:n)

      pi=4.0D+00*datan(1.0D+00) !Pi
      fund=261.626D+00 !Fundamental frequency (C4).
      ee=0.88D+00 !88%
      bigpure=0.0D+00
      bigcomplex=0.0D+00
      open(unit=10,file='puretone.dat')
      open(unit=20,file='complextone.dat')
      open(unit=30,file='purepower.dat')
      open(unit=40,file='complexpower.dat')
      open(unit=50,file='harmonic.dat')

      do i=1,n,1
         t=dble(i)/dble(n) !Divides the sine wave into n pieces.

***   Pure tone at the fundamental frequency (C4).
         puretone(i)=dsin(2*pi*t*fund)

***   Used to express the harmonics in Figure 2, this value changed from 1
to 6.
         harmonic(i)=dsin(6*pi*t)
```

```
***   Complex tone consisting of 5 harmonics at 88% the amplitude of the
***   previous.
         complextone(i)=dsin(2*pi*t*fund)+ee*dsin(2*pi*t*fund*2)+
     &         ee**2*sin(2*pi*t*fund*3)+ee**3*dsin(2*pi*t*fund*4)
     &         +ee**4*dsin(2*pi*t*fund*5)+ee**5*dsin(2*pi*t*fund*6)

***   Normalizes to 1.
         if(puretone(i).gt.bigpure)then
            bigpure=puretone(i)
         endif
         if(complextone(i).gt.bigcomplex)then
            bigcomplex=complextone(i)
         endif
      enddo

      do i=1,n,1
         t=dble(i)/dble(n)
         write(10,*) t,' ',puretone(i)/bigpure
         write(20,*) t,' ',complextone(i)/bigcomplex
         write(50,*) t,' ',harmonic(i)
      enddo
      end
```

5. Beats.f

```
***   Beats.f
***   Michael Blatnik
***   Senior Thesis
***   4/9/11
***
***   This brief program is used to create Figure 7, which shows beats
***   between two sine waves of 40 and 42 Hz.

      Program Beats

      implicit none
      integer i,n
      parameter (n=10000)
      double precision pi,t,s(0:n),f,j

      pi=4.0D+00*datan(1.0D+00)
      f=2.0D+00*pi/dble(n)
      open(unit=10,file='beats.dat')
***   Generates the comibination of sine waves at 40 and 42 Hz.
      do i=1,n,1
         s(i)=sin(40.0D+00*f*i)+sin(42.0D+00*f*i)
         j=dble(i/n)
         write(10,*) i,' ',s(i)
      enddo
      end
```

6. PianoHit.f

```
***   PianoHit.f
***   Michael Blatnik
***   Senior Thesis
***   4/9/11
***
***   Code from Computational Physics, PHYS333, Spring 2010. The
***   equations used come from Giordano and Nakanishi's Computational
***   Physics.5
***   This code simulates a piano string being struck by a hammer of
***   mass 3.3 grams at different initial velocities. Resulting graphs
***   include the force of the hammer, and the string displacement where
***   the string is struck (at L/8).

      program PianoHit

      implicit none

      integer i,n,imax,ihit,nmax

      parameter(imax=8*30)
      parameter(nmax=2**16)

      double precision L,T,f,K,mh,p
      double precision dt,dx,mu,Fh(0:nmax),c,q,Lh
      double precision zf,tmax,time,ah(0:nmax),r
      double precision y(0:imax,0:nmax),zh(0:nmax),vh(0:nmax)
      double precision yy(1:nmax),nn,ntmax,ntmax2,tmax2

***** Initial conditions to reproduce left figure in Figure 11.6
      L=0.62D+00 !m
      T=650.0D+00 !Tension (650 N)
      p=3.0D+00
      K=1.0D+11 !Stiffness constant of hammer, C4 N/m^(1/3)
      c=330.0D+00 !Speed for C4
      mh=(3.3D-03) !of hammer, in kilograms
      ihit=int(dble(imax)/8.0D+00)
      tmax=50.0D-03 !50 milliseconds (viewing window).
      tmax2=5.0D-03 !5 milliseconds.
      f=c/(2.0D+00*L)

      dx=L/dble(imax) !String step.
      mu=T/(c*c)  !Mass per unit length of a flexible string
      Lh=L/8.0D+00 !Point of hammer strike (L/8).

      dt=dx/c !Time step.
      r=c*dt/dx
      q=dt*dt/(mu*dx)
      ntmax=tmax/dt
      ntmax2=tmax2/dt


***Sets string to 0 at all points at time=0, including the end points.
      do i=0,imax,1
         y(i,0)=0.0D+00
```

```
         y(i,1)=0.0D+00
      enddo

      zh(0)=0.0D+00
      vh(0)=3.0D+00               !m/s, initial velocity of hammer
      ah(0)=0.0D+00

      open(unit=10,file='zh.dat')
      open(unit=20,file='Fh.dat')
      open(unit=30,file='yC4.dat')
      open(unit=31,file='yC4transfer.dat')
      open(unit=33,file='dt.dat')
      open(unit=40,file='test1.dat')
      open(unit=50,file='vh.dat')
      open(unit=60,file='string.dat')

***   This loop uses the wave equation to calculate the position of the
***   the string at a given time. The hammer force is determined by the
***   equation given by Giordano and Nakanishi.[5]
      do n=1,nmax,1
         zf=zh(n-1)-y(ihit,n-1)
         Fh(n)=K*dabs(zf)**p

         if (zf.lt.dble(0))then
            Fh(n)=0.0D+00
         endif
         do i=1,imax-1,1
            if (i.eq.ihit-1)then
               y(i,n+1)=(2.0D+00)*(1.0D+00-r*r)*y(i,n)-y(i,n-1)
     &              +r*r*(y(i+1,n)+y(i-1,n))+q*Fh(n)*0.25D+00
               goto 10
            endif
            if (i.eq.ihit)then
               y(i,n+1)=(2.0D+00)*(1.0D+00-r*r)*y(i,n)-y(i,n-1)
     &              +r*r*(y(i+1,n)+y(i-1,n))+q*Fh(n)*0.5D+00
               goto 10
            endif
            if (i.eq.ihit+1)then
               y(i,n+1)=(2.0D+00)*(1.0D+00-r*r)*y(i,n)-y(i,n-1)
     &              +r*r*(y(i+1,n)+y(i-1,n))+q*Fh(n)*0.25D+00
               goto 10
            endif
            y(i,n+1)=(2.0D+00)*(1.0D+00-r*r)*y(i,n)-y(i,n-1)
     &           +r*r*(y(i+1,n)+y(i-1,n))
 10      enddo

         ah(n)=-Fh(n-1)/mh
         zh(n)=zh(n-1)+vh(n-1)*dt
         vh(n)=vh(n-1)+ah(n-1)*dt
         yy(n)=y(ihit,n)*1000.0D+00
         nn=n*dt*1000.0D+00
         write(31,*)y(ihit,n)    !Used to transfer data to other file
      enddo
```

```
      do n=1,int(ntmax),1        !Generates string displacement for 50 ms
         nn=n*dt*1000.0D+00
         write(30,*)nn,' ',yy(n)
      enddo

      do i=0,imax,1
         write(60,*)i,' ',y(i,int(ntmax))*1000.0D+00
      enddo

      do n=1,int(ntmax2),1
         nn=n*dt*1000.0D+00
         write(50,*)nn,' ',vh(n)
         write(10,*)nn,' ',zh(n)*1000.0D+00
         write(20,*)nn,' ',Fh(n)
      enddo
      end
```

7. Power.f

```
***   Power.f
***   Michael Blatnik
***   Senior Thesis
***   4/9/11
***
***   Code from Computational Physics, PHYS333, Spring 2010.
***   This program's primary goal is to create the power spectrum over a
***   range of frequencies to determine the effect of the harmonics. It
***   uses the signal generated in the program PianoHit. It also uses the
***   FFT.f subroutine to generate the power spectrum.

      program Power

      integer n,i,k,j

      parameter(n=2**16)

      double precision f,y(0:n-1),pi,buffer(1:2*n)
      double precision freq(1:2*n),P(1:2*n),nn,dt,fmax,fstep,freq1
      double precision q(1:5),ratio(0:30),a,ii,big

      pi=4.0D+00*datan(1.0D+00) !Pi
      k=0

      open(unit=10,file='sine.dat')
      open(unit=20,file='real.dat')
      open(unit=30,file='imaginary.dat')
      open(unit=40,file='test.dat')
      open(unit=50,file='power.dat')
      open(unit=60,file='ratio.dat')

      open(unit=31,file='yC4transfer.dat')
      open(unit=32,file='newYC4.dat')
      open(unit=33,file='dt.dat')
```

```
open(unit=70,file='string.dat')

do i=0,n-1,1
   read(31,*)y(i)
enddo

read(33,*)dt
fstep=dble(1)/(dt*dble(n)*2.0D+00) !Frequency step.
fmax=1.5D+03  !Maximum frequency for viewing window of 1500 Hz.
nfmax=fmax/fstep

do i=0,n-1,1
   nn=i*dt*1000.0D+00
   write(32,*)nn,' ',y(i)
enddo

do i=0,n-1,1

     buffer(2*i+1)=y(i+1)
     buffer(2*i+2)=0.0D+00

enddo

do i=1,2*n-1,2

   freq1=dble(i-1)/(dt*dble(n)*2.0D+00)

   if (i.lt.n+1)then
      freq(i)=freq1
      freq(i+1)=freq(i)
   endif

   if (i.eq.n+1)then
      freq(i)=freq1
      freq(i+1)=-freq(i)
   endif

   if (i.gt.n+1)then
      k=k+4
      freq(i)=-freq(i-k)
      freq(i+1)=freq(i)
   endif
   write(40,*)i,' ',freq(i)
   write(40,*)i,' ',freq(i+1)
enddo

call FFT(buffer,n,1)

do i=1,2*n,2
   write(20,*)freq(i),' ',buffer(i)
   write(30,*)freq(i+1),' ',buffer(i+1)
enddo

do i=1,2*n,2
```

```
   P(i)=buffer(i)**2+buffer(i+1)**2
enddo

big=0.0D+00

do i=1,nfmax,2
   if (P(i).gt.big) then
      big=P(i)
   endif
enddo

do i=1,nfmax,2
   write(50,*)freq(i),' ',P(i)
enddo
end
```

References

[1] The Oxford English Dictionary, edited by James A. H. Murray, Henry Bradley; W. A. Craigie, C. T. Onions, and , R. W. Burchfield (Clarendon Press, Oxford, 1989).

[2] Joseph Machlis, The Enjoyment of Music, (W. W. Norton & Company Inc., New York, 1963).

[3] The New Grove Dictionary of Music and Musicians, edited by Sadie Stanley (Grove Dictionaries Inc., Taunton, 2001).

[4] Barry R. Parker, Good vibrations, (Johns Hopkins University Press, Baltimore, 2009).

[5] Nicholas J. Giordano and Hisao Nakanishi, *Computational Physics* (Prentice Hall, New Jersey, 2006), 2nd ed.

[6] Chaigne, Antoin and Anders Askenfelt. "Numerical simulations of piano strings. I. A physical model for a struck string using finite difference methods." Journal of the Acoustical Society of America. 95.2 (1994): 1112.

[7]Heinrich Kuttruff, Acoustics: An introduction. (Taylor & Francis, New York, 2007).

[8] Harvey Fletcher and W.A. Munson, "Loudness, Its Definition, Measurement, and Calculation." Journal of the Acoustical Society of America. **5**. 82-108 (1933).

[9] Music and Mathematics: From Pythagoras to Fractals, edited by John Fauvel, Raymond Flood, and Robin Wilson (Oxford University Press Inc., New York, 2003).

[10] Herman von Helmholtz, On the Sensations of Tone as a Physiological Basis for the Theory of Music, 2nd English Ed. Translated by Alexander J. Ellis (Longmans, Green, and Co., London, 1885).

[11] R. Plomp and W. J. M. Levelt, "Tonal Consonance and Critical Bandwidth," Journal of the Acoustical Society of America. **38**, 548-560 (1965).

---

[12] William A. Sethares, Tuning, timbre, spectrum, scale, 2nd Ed. (Springer, London, 2005).

[13] John S. Ridgen, Physics and the Sound of Music, (John Wiley & Sons, Inc., New York, New 1977).

[14] <http://billposer.org/Linguistics/Computation/LectureNotes/AudioData.html>

[15] McGraw-Hill Encyclopedia of Science & Technology, (McGraw-Hill Book Company, New York, 1977).

[16] William H. Press, Saul A. Teukolsky, William T. Vetterling, and Brian P. Flannery, Numerical Recipes: The Art of Scientific Computing, 3rd Ed. (Cambridge University Press, New York, 2007).