2013

Applications of Fourier Analysis to Audio Signal Processing: An Investigation of Chord Detection Algorithms

Nathan Lenssen
Claremont McKenna College

CLAREMONT McKENNA COLLEGE

APPLICATIONS OF FOURIER ANALYSIS TO AUDIO SIGNAL PROCESSING: AN INVESTIGATION OF CHORD DETECTION ALGORITHMS

SUBMITTED TO

PROFESSOR DEANNA NEEDELL

AND

DEAN GREGORY HESS

BY

NATHAN LENSSEN

FOR

SENIOR THESIS

SPRING 2013

APRIL 29, 2013

Abstract

The discrete Fourier transform has become an essential tool in the analysis of digital signals. Applications have become widespread since the discovery of the Fast Fourier Transform and the rise of personal computers. The field of digital signal processing is an exciting intersection of mathematics, statistics, and electrical engineering. In this study we aim to gain understanding of the mathematics behind algorithms that can extract chord information from recorded music. We investigate basic music theory, introduce and derive the discrete Fourier transform, and apply Fourier analysis to audio files to extract spectral data.

Contents

1 Introduction

Music is a highly structured system with an exciting potential for analysis. The vast majority of Western music is dictated by specific rules for time, beat, rhythm, pitch, and harmony. These rules and the patterns they create entice mathematicians, statisticians, and engineers to develop algorithms that can quickly analyze and describe elements of songs. In this study we will be addressing the problem of chord detection through attempting to answer the question: "How can we fit an audio file with chord labels?" With the ability to quickly determine the harmonic structure of a song, we can build massive databases which would be prime for statistical analysis. A human preforming such a task must be highly versed in music theory and will likely take hours to complete the annotation of one song, but an average computer can already perform such a task with reasonable accuracy in a matter of seconds [8]. In this study we explore the mathematics underlying such a program and demonstrate how we can use such tools to directly analyze audio files.

In section 2 we provide the reader with a brief introduction to music theory. To understand our question, one must have some basic knowledge of the music framework and vocabulary. We discuss pitches and scales and use them to define what we mean by a chord. Also, we explore nuances of chords that make automated detection such a fascinating and difficult problem.

Continuing with background information, section 3 introduces a general model currently used to detect chords from raw audio clips. An understanding of the analysis process is crucial to the appreciation of the mathematics. We break down the model into signal analysis and chord fitting components and provide a brief background into what each entails.

Section 4 contains an introduction to the mathematics necessary to derive the discrete Fourier transform. The discussion is centered around the difference between the time and frequency domain of a function. We also provide a background on the delta function which serves as a bridge from continuous functions to discrete vectors.

In section 5 we take the mathematics we have developed in section 4 and derive the discrete Fourier transform. We need the discrete version of the Fourier transform because computers cannot process continuous, analogue signals. We must have some mathematical tool that can deal with discrete, sampled ones. The discrete Fourier transform enables us to decompose our input signal into a form that can be handled by the chord fitting portion of our model.

In section 6 we discuss the speed of the discrete Fourier transform and introduce the fast Fourier transform. The fast Fourier transform is then utilized in MATLAB to construct a chromagram or pitch-intensity plot of an audio file.

2 Introduction to Music Theory

Before we delve into the mathematics behind chord recognition, it is important for the reader to have an understanding of what musical structures we are utilizing. In this section we will provide a brief introduction to modern Western musical theory. We build up this theory from a single note or pitch, describe how these pitches relate to each other in scales, and how these scales are summarized by chords. We discuss the difficulties that arise with octaves since the human ear perceives notes that are n-octaves apart as being the same note family or chroma. Through our investigation we will set the stage for the task of extracting chord data from a raw audio wave.

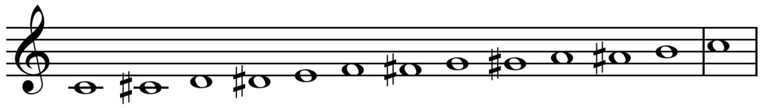

Figure 2.1: A chromatic scale beginning and ending of C. Notice there are 13 notes because C is played at both the top and bottom [10].

2.1 Pitches and Scales

In this study we define a pitch as the human perception of a sound wave at a specific frequency. For instance, the tuning note for a symphony orchestra is A4 which has a standardized frequency of 440Hz. In the notation A4, A indicates the chroma or quality of the note while 4 describes the octave or height. We discuss the uses of the chroma and octave information further in a later section. It is hard to describe anything complex with individual pitches so we define a scale as a sequence of pitches with a specific spacing in frequency. As we follow the pitches of a scale from bottom to top, we start and end on the same note one octave apart (e.g. from C3 to C4). Pitches an octave apart sound similar to the human ear because a a one octave jump corresponds to a doubling in the frequency of the sound wave. Western music defines the chromatic scale as each of the chroma ordered over an octave as 12 notes. These 12 notes are spaced almost perfectly logarithmically over the octave. We can use a recursive sequence to describe the chromatic scale [6]

$$P_i = 2^{1/12} P_{i-1} \tag{2.1}$$

With a pitch at frequency P_i in relation to the preceding note at frequency P_{i-1}. We can visualize the chromatic scale by striking every white and black key in order up an octave on a piano or by the scale depicted in Figure 2.1.

By using the chromatic scale as a tool, we can construct every other scale in Western

Scale "Color"	Defining Steps
Chromatic	RHHHHHHHHHHHHR
Major	RWWHWWWHR
(Natural) Minor	RWHWWHWWR
Diminished	RHWHWHWHWR
Augmented	RAHAHAHR

Table 2.1: Interval construction of the four core scales. Note that the intervals apply for when ascending in pitch only. When determining the descending scale, the order of intervals is reversed.

music through the use of intervals. An interval refers to a change in position along the 12 notes of the chromatic scale. We define the interval of a "half step" or H as a change in one pitch along the chromatic scale. A two half step interval is a "whole step" and denoted by W. We also use interval of three half steps known as an "augmented second" denoted by A. Scales are defined by a sequence of these intervals with the condition that the total sum of steps must equal 12. This guarantees that we start and end on the same chroma known as the root R. In this study we discuss the four most prevalent scales in western music: major, minor, augmented, and diminished. The intervals that describe these scales can be found in Table 2.1. The table can be used by picking any starting note for the root and following the intervals. For instance a C minor (Cm) scale is C,D,E♭,F,G,A,B♭,C.

2.2 Triads and Chords

Multiple pitches played simultaneously are defined as a chord. Chords are essential in music analysis because they compactly describe the entire melodic and harmonic structure of a section of music. That is, a chord indicates what notes and combination of notes should be played at a moment in time. A triad is a specific and simple chord containing the first, third and fifth note of a scale (I III V). Figure 2.2 shows the triads for each of the four scales in the key of C. These triads describe the major

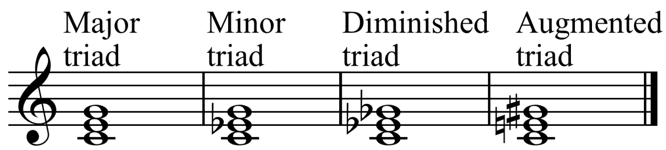

Figure 2.2: Triads in the key of C [12].

(maj), minor (min), diminished (dim), and augmented (aug) chords in our model.

While triads are a useful way to understand the tonal structure of music, four notes are often needed to completely describe tonal character. Adding and possibly altering the seventh note of the scale (VII or 7) creates new and essential chords. The three chords that we need a seventh to describe are the major seventh (maj7), minor seventh (min7), and dominant seventh (dom7). The major and minor seventh chords follow directly from the major and minor scales. They each contain the I III V VII of their respective scale. The dominant seventh chord does not follow one of the scales we have described. With respect to the major scale it contains the I III V VII♭. The theory behind the dominant seventh chord is a consequence of the theory of musical modes, which is more complex than is necessary for the understanding of our study.

We have now described all of the chord families that our model will be detecting. The number of possible chords is determined by combinations with the chromatic scale. These families are sufficient at capturing the majority of western music. In this study we also have 21 possible roots and the possible chord names can be found in Table 2.2 [8].

2.3 Chord Inversions

Currently we have 7 chord families and 21 roots giving us a total of 147 different chords to distinguish between. However this number is assuming that we always have

Chord Families	maj, min, maj7, min7, dom7, aug, dim
Roots	A♭, B♭, C♭, D♭, E♭, F♭, G♭, A, B, C, D, E, F, G, A♯, B♯, C♯, D♯, E♯, F♯, G♯

Table 2.2: The naming parameters for chords in our model. A chord is named by one chord family and one root[8].

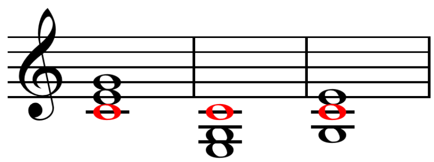

Figure 2.3: The root and inversion triads in the key of C[11].

the root as the lowest note in the chord. In reality chords with root on bottom do not occur all of the time. Instead one of the chords inversions or rearrangements is found. An inversion can algorithmically be described as the chord that has been transformed, by taking the root and raising it an octave. Then the previously second lowest note is now on bottom. This is known as the first inversion. If we apply that process again, we are left with the second inversion and third lowest note on bottom. For a triad we have two inversions before we are back to the original chroma arrangement. When chords involve a seventh, we have three inversions. Thus the total amount of chords we now how to distinguish between is

$$C = (21 \text{ roots})[(4 \text{ triads})(2 \text{ inversions}) + (3\ 7^{th} \text{ chords})(3 \text{ inversions})] = 357 \quad (2.2)$$

Distinguishing between this amount of distinct possible chords is quite a task, particularly since the octave information of a note cannot help us narrow down the chord choice. In the next section we will explore how mathematicians and engineers take raw audio files and determine the chord progressions.

3 The Chord Detection Algorithm

Throughout this study we will be analyzing a chord recognition model developed by Sheh and Ellis in 2003 [8]. While we are discussing one specific model, the general process in current chord analysis is relatively standard [6, 8]. This model can be broken down into two major components: signal analysis and chord fitting. The goal of the signal analysis portion of the model is to break down a raw audio signal into chroma intensities. These intensities are then fed into the chord fitting portion where we determine the most likely chord representations for the music.

The primary mathematical tool used to decompose the raw audio signal is the Fast Fourier Transform (FFT), which is an optimized discrete Fourier transform algorithm. We use the FFT to determine the fundamental frequencies and therefore pitches that are present in our raw signal. However as explained in the previous chapter, the octave of the pitch is generally irrelevant to the chord identity. Thus, we will need to transform the pitches obtained through the FFT into octave-independent chroma. The chroma information of an audio file is known as the Pitch Class Profile (PCP) [8].

Once we have determined the PCP of our input, we use it to determine the best fitting chords to our music. To do this we implement a Hidden Markov Model that has been trained by a large set of pre-annotated audio files. The optimal chord assignments given a PCP are determined through the expectation maximization algorithm[5]. Since the computer has no idea of what a "beat" is, we must align the chord labels that the expectation maximization algorithm determines with specific time points in the original audio track. The Viterbi alignment algorithm can be used to forcibly align the chord labels, creating a fully analyzed track [5, 8].

As stated previously, the focus of this study is on the signal processing portion of the chord detection algorithm. We now explore the FFT by building an understanding of the underlying mathematics. In doing so, we reveal how Fourier analysis is useful in determining the chroma structure of an audio file.

4 Introduction to the Fourier Transform

4.1 Frequency and Time Domains

Before we delve into the math surrounding the Fourier transform, it is helpful to understand what it is doing at an intuitive level. The development of the mathematical construct now referred to as the Fourier transform was motivated by two problems in physics that are very prevalent and observable. These problems are the conduction of heat in solids and the motion of a plucked string whose ends are fixed in place [1, 2]. We instantly see the relevance to our study as string instruments such as a violin or piano produce sound by amplifying the vibrations of such a fixed string. The history of the development of the Fourier series and transform is interesting and rich. A reader interested in its development is recommended to read the introductions of these books [1, 2]. History aside, the most crucial information we take away from the development of Fourier analysis is that functions can be represented in both the time domain as well as the frequency domain.

But what do we mean by the time and frequency domains? The connection between the two is easiest to see in a periodic system such as the vibrating string example. You would likely describe the motion of such a string by talking about how the position of points on the string are changing as time progresses. More rigorously, we can model the motion using a differential equation where the initial condition is the

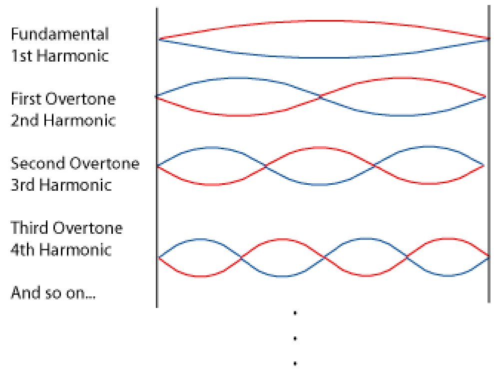

Figure 4.1: The first four standing waves of a vibrating, fixed string [7].

initial displacement. A differential equation can to describe how the elastic forces of the string evolve the position of the string over time [1]. By solving such differential equations, we can represent the motion of the string by a function $f(t)$ where t is time. This function $f(t)$ is known as the time domain representation of the motion of the string; it provides us with information on how the string behaves as time progresses.

For our plucked string example it is easy to visualize the time domain as that is how we perceive its motion with our eyes. The string is vibrating in some repetitive, sinusoidal pattern. Imagine we plucked the string in precisely the way that made the string vibrate only at the fundamental harmonic illustrated in Figure 4.1. We represent this pattern in the time domain by a single sinusoid with frequency ν_0. Notice that we can completely describe the motion of the string by this frequency ν_0 and the amplitude of the oscillation.

Thus, the frequency domain $F(\nu)$ of this specific string has just one spike at $\nu = \nu_0$ with the height of the spike equal to the amplitude of the wave. The example of the fundamental harmonic allows us to visualize what the frequency domain is conveying, but in real systems we never have exactly and only one frequency. We then construct our frequency domain representation through an infinite series of these harmonics weighted in such a way that they represent the motion of the string. This series is known as the Fourier series and provides the basis for the Fourier transform. Through the Fourier transform we are able to obtain the frequency domain of a time domain function. The Fourier transform is invertible with the inverse Fourier transform returning the time domain function from a frequency domain function.

Before we delve into the mathematics behind the Fourier transform, it is useful to understand why it is so crucial to our study. Audio signals are recorded and listened to in the time domain; they are the intensity of the sound as a function of time. However, pitches and therefore chords are represented by single frequencies as shown in section 2. Therefore we use a Fourier transform to convert the time domain input signal into a frequency representation which can be analyzed for intensities of specific pitches. The relationship between the pitches at a given time is then analyzed to determine the chord that is being played. With some intuition of the purpose of the Fourier transform and its role in our study, we proceed to investigating its mathematics. It is important to note that the Fourier transform is a transformation in the complex plane. We will see that even for real-valued inputs such as audio signals, the Fourier transform will return a complex-valued frequency domain representation.

4.2 The Continuous Fourier Transform

As introduced in section 4.1, the Fourier transform is a mathematical tool that can be used to transform a function from the time domain $f(t)$ to the frequency domain $F(\omega_k)$. We define ω_k as the angular frequency through its relationship with the ordinary frequency ν as

$$\omega_k \equiv 2\pi k\nu \tag{4.1}$$

The Fourier transform is an invertible transformation. The inverse Fourier transform takes a frequency domain function and returns its representation in the time domain.

We define the Fourier transformation as

$$F(\omega_k) \equiv \int_{-\infty}^{\infty} f(t)\, e^{-2\pi i k t}\, \mathrm{d}t \qquad k \in (-\infty, \infty) \tag{4.2}$$

The presence of the complex sinusoidal is clear by recalling that $e^{i\omega t} = \cos(\omega\, t) + i\sin(\omega\, t)$. From equation 4.2 we can derive the inverse Fourier transform. Our inverse transform is

$$f(t) = \int_{-\infty}^{\infty} F(\omega_k)\, e^{2\pi i k t}\, \mathrm{d}k \qquad k \in (-\infty, \infty) \tag{4.3}$$

We have presented this section to provide the reader with the basic structure of the Fourier transformation, but will not provide any more detail on the continuous case in this paper. These equations provide the base for our algorithms to find the spectra of waves, but since digital signals are not continuous we need to find a way to use the Fourier Transformation on discrete sequences rather than continuous functions. Thus, we are motivated to define and derive the discrete Fourier transform.

4.3 The Discrete Fourier Transform

When faced with a set of numbers or vector instead of a continuous function, we must use an algorithm that does not involve an integral. As we know, the sum is the discrete analogue to the integral. Thus we define and eventually show that the discrete Fourier transform (DFT) over a complex vector f_n of length N as

$$F_k \equiv \sum_{n=-\frac{N}{2}+1}^{\frac{N}{2}} f_n e^{-i2\pi nk/N} \qquad k = 0 : N-1 \tag{4.4}$$

This should look quite similar in form to the continuous version of the Fourier transform in equation 4.2. We still have the exponential kernel and are now performing a careful sum instead of a integral. Likewise, our inverse DFT has the form

$$f_k = \sum_{n=-\frac{N}{2}+1}^{\frac{N}{2}} F_n e^{i2\pi nk/N} \qquad k = 0 : N-1 \tag{4.5}$$

With these equations in mind we now present the derivation of them from the continuous Fourier transform. We do so in a way that is motivated by the nature of our input sound signals before and after they have been sampled from continuous functions into vectors.

4.4 Sinusoids

Before we dive into our derivation of the DFT, we take note of an interesting property of the sine and cosine functions. Our inclusion of this topic may not make sense at the time to a reader without experience in Fourier analysis, but the usefulness of our discussion will present itself shortly. We define a sinusoid as a function of the form

$$x(t) = A\sin(2\pi ft + \phi) \tag{4.6}$$

When discussing audio signals we let

$$\begin{aligned} A &= \text{Amplitude} \\ 2\pi\nu &= \text{frequency(Hz)} \\ t &= \text{time (s)} \\ \phi &= \text{initial phase (radians)} \\ 2\pi\nu t + \phi &= \text{instantaneous phase (radians) [9]} \end{aligned}$$

Fourier transforms are built on the complex properties of sinusoids which follow from Euler's Identity

$$e^{i\theta} = \cos(\theta) + i\sin(\theta) \tag{4.7}$$

$$e^{\pm i2\pi\nu x} = \cos(2\pi\nu x) \pm i\sin(2\pi\nu x) \tag{4.8}$$

Equation 4.7 is Euler's Identity in its most basic form. We include Equation 4.8 as this is how we will be using the identity in this study. We will be discussing specific modes or wavelengths of sinusoids defined by their frequency.

We now aim to prove that sinusoids are orthogonal with respect to their frequency. Before we present the orthogonality property for sums of sinusoids we must define the Kronecker delta and the modular Kronecker delta. The Kronecker delta is a function of two variables m, n that returns

$$\delta_{mn} = \begin{cases} 1 & \text{if } m = n \\ 0 & \text{else} \end{cases}$$

The modular Kronecker delta is slightly different and will be used in the discrete orthogonality theorem. In this case "mod" refers to normal modulo arithmetic

$$\hat{\delta}_N(k) = \begin{cases} 1 & \text{if } k = 0 \bmod N \\ 0 & \text{else} \end{cases}$$

With these definitions in hand we are now prepared to present the orthogonality theorems [2].

4.5 Orthogonality of Sinusoidals (Continuous)

The Fourier transform builds a basis in the frequency domain by taking advantage of the idea that sinusoids of different frequencies are orthogonal. We claim that over an interval 2π with $m, n \in \mathbb{Z}$ that,

$$\int_{-\pi}^{\pi} \cos(mt)\cos(nt)\,\mathrm{d}t = \pi\delta_{mn} \tag{4.9}$$

To show this we must investigate two cases: $m = n$, and $m \neq n$. When $m = n$ we may write

$$\int_{-\pi}^{\pi} \cos(mt)\cos(nt)\,\mathrm{d}t = \int_{-\pi}^{\pi} \cos^2(nt)\,\mathrm{d}t \tag{4.10}$$

By recalling the symmetry of the cosine function we may reduce our integral as

$$\int_{-\pi}^{\pi} \cos^2(nt)\,\mathrm{d}t = 2\int_{0}^{\pi} \cos^2(nt)\,\mathrm{d}t \tag{4.11}$$

Since $2\cos^2(x) = 1 + \cos(2x)$ we have

$$\begin{aligned} 2\int_{0}^{\pi} \cos^2(nt)\,\mathrm{d}t &= \int_{0}^{\pi} 1 + \cos(2nt)\,\mathrm{d}t \\ &= \left(t + \frac{\sin(2nt)}{2n}\right)\Bigg|_{t=0}^{\pi} \\ &= \pi \end{aligned} \tag{4.12}$$

Since n is an integer and all multiples of 2π result in the sine function having zero value. We have shown half of our claim from equation 4.9.

When $m \neq n$ the calculus is a little messier, but can again be simplified by trigonometric identities. We start the same way as above by recalling the symmetry about the origin of the cosine function.

$$\int_{-\pi}^{\pi} \cos(mt)\cos(nt)\,\mathrm{d}t = 2\int_{0}^{\pi} \cos(mt)\cos(nt)\,\mathrm{d}t \tag{4.13}$$

By the property of products of cosines we may write

$$\begin{aligned} 2\int_{0}^{\pi} \cos(mt)\cos(nt)\,\mathrm{d}t &= \int_{0}^{\pi} \cos((m-n)t) + \cos((m+n)t)\,\mathrm{d}t \\ &= \left(\frac{\sin((m-n)t)}{m-n} + \frac{\sin((m+n)t)}{m+n}\right)\Bigg|_{t=0}^{\pi} \\ &= 0 \end{aligned} \tag{4.14}$$

Since $m, n \in \mathbb{Z}$ so $m + n \in \mathbb{Z}$ and $m - n \in \mathbb{Z}$. For the same reasoning as the $m = n$ case, both of the sine functions are zero and the result is zero. We have proven our claim from equation 4.9. In a similar way we can show the orthogonality of sine functions and discover that

$$\int_{-\pi}^{\pi} \sin(mt)\sin(nt)\,\mathrm{d}t = \int_{-\pi}^{\pi} \cos(mt)\cos(nt)\,\mathrm{d}t = \pi\delta_{mn} \tag{4.15}$$

With the continuous orthogonality of sines and cosines, we have a platform to understand and accept sinusoids as a basis for the frequency domain representation of a function. Since they are orthogonal we may use linear combinations of sinusoids to create a spanning set and represent our time domain function in the frequency domain. However, this representation will not be sufficient for the DFT and we need to explore the orthogonality of sinusoidals further.

4.6 Orthogonality of Sinusoidals (Discrete)

In the DFT we work with sums instead of integrals. Thus we have to provide a new orthogonality property for this discrete case. We can be clever and use the exponential function to prove directly that the kernels of our Fourier series will be orthogonal.

Theorem 4.1. Orthogonality of Sinusoids: *Let j and k be integers and let N be a positive integer. Then*

$$\sum_{n=0}^{N-1} e^{i2\pi nj/N} e^{i2\pi nk/N} = N\hat{\delta}_N(j-k) \tag{4.16}$$

Proof. First consider the N complex numbers $e^{i2\pi k/N}$ for $k = 0 : N-1$. These are known as the Nth roots of unity as they satisfy the equation

$$\left(e^{i2\pi k/N}\right)^N = e^{i2\pi k} = 1 \tag{4.17}$$

We know that the roots of unity are zeros of the complex polynomial $z^N - 1$. This polynomial may be factored as

$$\begin{aligned} z^N - 1 &= (z-1)(z^{N-1} + z^{N-2} + \ldots + z + 1) \\ &= (z-1)\sum_{n=0}^{N-1} z^n \end{aligned} \tag{4.18}$$

Now let $z = e^{i2\pi(j-k)/N}$. We now have two cases: when $(j-k) = 0 \mod N$, and when $(j-k) \neq 0 \mod N$. Since $e^{i2\pi k/N} - 1 = 0$ we have in the first case,

$$\sum_{n=0}^{N-1} z^n = \sum_{n=0}^{N-1} e^{i2\pi(j-k)/N} = \sum_{n=0}^{N-1} 1 = N \tag{4.19}$$

In the second case when $(j-k) \neq 0 \mod N$ we must have

$$\sum_{n=0}^{N-1} z^n = \sum_{n=0}^{N-1} e^{i2\pi(j-k)/N} = 0 \tag{4.20}$$

By 4.19 and 4.20 we have shown the discrete orthogonality of sinusoids [2]. □

Again, orthogonality is significant because in enables us to form an orthogonal basis of sinusoids that build our frequency domain. The coefficients on linear combinations of sinusoids obtained by the DFT are the values in the frequency domain that describe fundamental frequencies embedded in our input signal.

4.7 The Delta (δ) Function

Due to the nature of computers we are cornered into using DFTs for music and signal processing. When an analogue sound wave is stored in a computer, it must be sampled. The mathematics of sampling is a large field and we will leave the discussion of sampling algorithms and analogue-to-digital converters for future work. In this study all that needs to be understood is that we have some sampler that takes a continuous signal $f(t)$ as an input over time $t \in [0, T]$. The sampler then returns a time-series vector that contains values according to some sampling frequency with a uniform gap Δt between samples. We can view this vector as a continuous "function" that is a peace-wise collection of Dirac delta functions δ, each centered at some multiple of our sampling period $n\Delta t$. We say "function" because δ is not a function by the rigorous definition, but rather a distribution or generalized function[2]. The delta function, or spike as it will often be referred to in this study, is a special mathematical construct with a number of properties that are intuitive and useful to our study of DFTs. We now outline a few of its most relevant properties to our study, understanding that this list is by no means exhaustive.

First, the δ function is non-zero at every point but 0 where it is infinite. That is,

$$\delta(t) = \begin{cases} \infty & \text{if } t = 0 \\ 0 & \text{if } t \neq 0 \end{cases} \tag{4.21}$$

This property illuminates why we use the term spike. The δ function is an infinitely thin, infinity tall peak centered around zero. When we want our delta function to occur at a value of $t \neq 0$, we may shift as we would any other function. Thus a δ function centered at $t = a$ can be written as $\delta(t - a)$.

Second, the area under the delta function is 1. That is,

$$\int_{-\infty}^{\infty} \delta(t)\,\mathrm{d}t = 1 \tag{4.22}$$

This property can be exploited to reveal the useful general sifting property in which we can use δ functions to discretize integrals.

$$\int_{-\infty}^{\infty} f(t)\delta(t - a)\,\mathrm{d}t = f(a) \tag{4.23}$$

The relevance of the general sifting property to our study cannot be overstated. In addition to being essential to our forthcoming derivation of the DFT from the Fourier transform, we can immediately notice its potential as a crude sampling tool. With a continuous input signal $f(t)$, we can use the general sifting property and $\delta(t - a)$ for an evenly spaced sequence of a values to get the instantaneous values of our original function. While certainly not a sophisticated method, it provides a relevant and simple application that demonstrates the power of the δ function.

With the δ function defined, we can now move on to investigate what its Fourier transform looks like. We present its Fourier transform and inverse Fourier transform that will serve as tools to define a spike train and eventually the DFT.

4.8 Fourier Transforms of δ Functions

While the two examples presented in this section are interesting, their usefulness in our investigation will not immediately be apparent. However these simple applications of the Fourier transform and its inverse will be instrumental in our derivation of the DFT. We begin by looking at the Fourier transform of a single spike centered at zero. By the general sifting property,

$$F(t) = \int_{-\infty}^{\infty} \delta(t)\, e^{-i2\pi\nu t}\, \mathrm{d}t \;=\; e^0 \;=\; 1 \tag{4.24}$$

This means that a spike in the time domain translates to an flat line in the frequency domain or a even weight among all frequencies. This is a somewhat physically curious result that can be reconciled by understanding that for every frequency, $\cos(2\pi\nu \cdot 0) = 1$ and at every other value of t, the sinusoids take on every value between 0 and 1.

We also see an interesting result when we have a spike at the origin in the frequency domain. Intuitively this spike symbolizes that our time dependent function is made up of a single sinusoid with frequency zero, or a straight line. We can confirm this by looking at the inverse Fourier transform of a δ function

$$f(t) = \int_{-\infty}^{\infty} \delta(t)\, e^{i2\pi\nu t}\, \mathrm{d}t \;=\; e^0 \;=\; 1 \tag{4.25}$$

These results are useful and interesting, but do not provide us with the connection between a spike and a sinusoid necessary to build the DFT. It appears that a spike in the frequency domain at any location besides the origin will correspond to some sort of sinusoid in the time domain. We can see this is true by taking the inverse Fourier transform of a δ function shifted by some frequency ν_0 which yields

$$f(t) = \int_{-\infty}^{\infty} \delta(\nu - \nu_0) e^{i2\pi\nu x}\,\mathrm{d}x \;=\; e^{i2\pi\nu_0 x}$$
$$= \cos(2\pi\nu_0 x) + i\sin(2\pi\nu_0 x) \qquad (4.26)$$

By clever addition of pairs of Inverse Fourier transforms, we can determine what spikes are necessary in the frequency domain such that their inverse Fourier transform are real-valued sine and cosine functions. We notice that to get a sine function in the time domain we must construct spikes in such a way that their sine components cancel. Recalling the definition of cosine from Euler's formula we get,

$$\begin{aligned} f(t) &= \int_{-\infty}^{\infty} \left[\frac{1}{2}\delta(\nu - \nu_0) + \frac{1}{2}\delta(\nu + \nu_0)\right] e^{i2\pi\nu x}\,\mathrm{d}x \\ &= \frac{1}{2}\int_{-\infty}^{\infty} \delta(\nu - \nu_0) e^{i2\pi\nu x}\,\mathrm{d}x + \frac{1}{2}\int_{-\infty}^{\infty} \delta(\nu + \nu_0) e^{i2\pi\nu x}\,\mathrm{d}x \\ &= \frac{e^{i2\pi\nu_0 x} + e^{-i2\pi\nu_0 x}}{2} \\ &= \cos(2\pi\nu_0 x) \end{aligned} \qquad (4.27)$$

Similarly for our sine function we have the inverse Fourier transform

$$\begin{aligned} f(t) &= \int_{-\infty}^{\infty} \left[\frac{1}{2i}\delta(\nu - \nu_0) - \frac{1}{2i}\delta(\nu + \nu_0)\right] e^{i2\pi\nu x}\,\mathrm{d}x \\ &= \frac{1}{2i}\int_{-\infty}^{\infty} \delta(f - f_0) e^{i2\pi\nu x}\,\mathrm{d}x - \frac{1}{2i}\int_{-\infty}^{\infty} \delta(\nu + \nu_0) e^{i2\pi\nu x}\,\mathrm{d}x \\ &= \frac{e^{i2\pi\nu_0 x} - e^{-i2\pi\nu_0 x}}{2i} \\ &= \sin(2\pi\nu_0 x) \end{aligned} \qquad (4.28)$$

We summarize these results by describing how the cosine and sine functions respond to a Fourier transform. Our cosine function of frequency ν has a Fourier transform of two positive real-valued spikes at $\pm\nu$ in the frequency domain. A sine function of frequency f has a Fourier transform that lies purely in the imaginary frequency domain with a negative spike at $+\nu$ and a positive spike at $-\nu$. The sine function serves as a demonstration of the necessity of complex numbers in the Fourier transform. A real valued sine wave is described by a completely imaginary frequency representation.

5 A Derivation of the Discrete Fourier Transform

5.1 Spike Trains and the Discrete Fourier Transform

As we have defined a spike as a δ function, a spike train is simply a collection of δ functions. In section 4.7 we mentioned how sampling can be interpreted as collecting the amplitudes of evenly spaced delta functions. That is nothing more than the construction of a spike train from a continuous function. For application to the DFT, we fix the spacing of these spikes in time as Δt. Thus the n^{th} spike is located at $t_n = n\Delta t$. We define our spike train over time $h(t)$ as

$$h(t) = \sum_n f_n \delta(t - t_n) \tag{5.1}$$

In this form the vector f_n represents the intensity or magnitude of the spikes. By recalling the general sifting property of delta function, we can rewrite a spike train as,

$$\begin{aligned} h(t) = \sum_n f_n \delta(t - t_n) &= \sum_n f(t_n)\delta(t - t_n) \\ &= \sum_n f(t)\delta(t - t_n) \end{aligned} \tag{5.2}$$

Thus, we can easily construct the spike train of a function by summing a series of evenly space delta functions multiplied by our original continuous function. Moreover, this view of the spike train is easily translated to a programming language such as MATLAB for simple and intuitive testing of our DFT algorithms.

With the definition of a spike train we can begin to reveal the DFT by investigating its Fourier transform. Assume we have a evenly spike train of length N on an time interval of length L. We center the function at zero so that the interval covered is $[-L/2, L/2]$. Since the points are evenly space in time, we know that the n^{th} point is located at $t_n = nL/N$ with $n = -N/2 + 1 : N/2$. Now we will see what happens when we take the continuous Fourier transform of this spike train. Again making use of the general sifting property of δ functions we have,

$$\begin{aligned} F(\nu) &= \int_{-\infty}^{\infty} \left(\sum_{n=-\frac{N}{2}+1}^{\frac{N}{2}} f_t \delta(t - t_n) \right) e^{-i2\pi\nu t} \, \mathrm{d}t \\ &= \sum_{n=-\frac{N}{2}+1}^{\frac{N}{2}} f_n \int_{-\infty}^{\infty} \delta(t - t_n) e^{-i2\pi\nu t} \, \mathrm{d}t \\ &= \sum_{n=-\frac{N}{2}+1}^{\frac{N}{2}} f_n e^{-i2\pi\nu x_n} \end{aligned} \tag{5.3}$$

Thus the Fourier transform of a spike train is simply the sum of exponentials weighted by their intensities in the spike train [2]. We now have an expression that is very close to our definition of the DFT given in Equation 4.4. To make Equation 5.3 match, we need to define and use the reciprocity relations of the discrete Fourier transforms.

5.2 Reciprocity Relations of the Discrete Fourier Transform

To uncover the reciprocity relations we will use the same sequence in the time domain as used in the previous section: a vector of length N evenly spaced over a total time of L. Again we are centered at $t = 0$ so the time interval runs over $[-L/2, L/2]$. If our temporal spacing is Δt then the points are defined as $t_n = n\Delta t$. We will use a similar method in discretizing the frequency domain. We again use a vector of length N evenly spaced points over a length Ω. We are centered at $\nu = 0$ so our interval of frequencies is $[-\Omega/2, \Omega/2]$. If our spacing of frequencies is $\Delta\nu$ then the points are defined as $\nu_k = k\Delta\nu$. We search for the reciprocity relations that relate the time and frequency domains using the parameters Δt and L for the time domain and $\Delta\nu$ and Ω for the frequency domain.

Since we only have a discrete number of points in the frequency domain, we are limited in the number of sinusoids we can use to represent our time domain function. Likewise, we are looking at our time domain function over a finite and well-defined time interval L. Clearly the longest sinusoid we can resolve in this amount of time is one with L as its period or the fundamental harmonic. Frequency is the reciprocal of period and we will call this longest period our fundamental frequency. The fundamental frequency is also the step size in our frequency domain. That is,

$$\Delta\nu = \frac{1}{L} \tag{5.4}$$

Thus the frequencies we will recognize will all be multiples of $\Delta\nu$ and have integer periods over our time interval. It is then clear that the length described in our frequency domain is just the number of points multiplied by the frequency step or $\Omega = N\Delta\nu$. Using this in conjunction with equation 5.4 we have our first reciprocity relation,

$$\Omega = N\Delta\nu = \frac{N}{L} \quad \Longrightarrow \quad \boxed{A\Omega = L} \tag{5.5}$$

This relation shows that the lengths of the temporal and frequency domains are inversely proportional and jointly fixed with respect to the input vector length N. In practice, we interpret Equation 5.5 by noting that taking temporal data over a longer range of time means that the DFT yields a smaller range of frequencies and visa versa [2].

With the first reciprocity relation in Equation 5.5 and the constraints we have placed on our time and frequency domains, we can easily find the second reciprocity relation. Recall that in the temporal domain we have N points evenly spaced by Δt over the interval $[-L/2, L/2]$. Therefore we must have $N\Delta t = L$. Also recalling from equation 5.4 that $\Delta\nu = 1/L$ we can present our second reciprocity relation as,

$$\frac{1}{\Delta\nu} = L = N\Delta t \quad \Longrightarrow \quad \boxed{\Delta t\Delta\nu = \frac{1}{N}} \tag{5.6}$$

This relation reveals very similar information to the first. The spacings of the temporal and frequency domain are inversely proportional and fixed by the number of points or the length of the vector.

Now that we have defined the reciprocity relations that govern the properties of the DFT when moving between the temporal and frequency domains, we can go back to equation 5.3 and reveal the exact form of the DFT presented in equation 4.4. Keep in mind that the usefulness of the reciprocity relations extends far beyond the derivation of the DFT. They are fundamental in dictating the behavior of DFTs when moving between the temporal and frequency domains.

5.3 The Discrete Fourier Transform

We begin with a reminder of where we left off in the derivation of the DFT in section 5.1. Equation 5.3 left us with

$$F(\nu) = \sum_{n=-\frac{N}{2}+1}^{\frac{N}{2}} f_n e^{-i2\pi\nu x_n} \tag{5.7}$$

Using the assumption that we have N points over a time length of L we know that the step spacing in the frequency domain must be $\nu_k = k\Delta\nu = k/L$ in which $k = -L/2+1 : L/2$. We use this to show that 5.7 can be rewritten as,

$$\begin{aligned} F(\nu) &= \sum_{n=-\frac{N}{2}+1}^{\frac{N}{2}} f_n e^{-i2\pi\nu_k x_n} \\ &= \boxed{\sum_{n=-\frac{N}{2}+1}^{\frac{N}{2}} f_n e^{-i2\pi nk/N}} \end{aligned} \tag{5.8}$$

Where this is the DFT as we have defined it previously in Equation 4.4. Thus we have shown that we can view the discrete Fourier transform as simply the Fourier transform of a spike train. We proceed by investigating how we employ the DFT and apply it to music signal analysis.

6 The Chromagram

Since the DFT is a sum across of indexed values in the exponential kernel, we can express it as the linear equation

$$\boldsymbol{F} = \boldsymbol{W}\boldsymbol{f} \tag{6.1}$$

Where $\boldsymbol{f}$ is the input data in the time domain, $\boldsymbol{F}$ is the output in the frequency domain, and $\boldsymbol{W}$ is the all of the exponentials for $k = 0 : N-1$ from equation 5.8 in a square, nonsingular matrix

$$\boldsymbol{W} = \begin{pmatrix} e^{-i2\pi(0)/N} & e^{-i2\pi(0)/N} & e^{-i2\pi(0)/N} & \cdots & e^{-i2\pi(0)/N} \\ e^{-i2\pi(0)/N} & e^{-i2\pi(1)/N} & e^{-i2\pi(2)/N} & \cdots & e^{-i2\pi(N-1)/N} \\ e^{-i2\pi(0)/N} & e^{-i2\pi(2)/N} & e^{-i2\pi(4)/N} & \cdots & e^{-i2\pi(2(N-1))/N} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ e^{-i2\pi(0)/N} & e^{-i2\pi(N-1)/N} & e^{-i2\pi(2(N-1))/N} & \cdots & e^{-i2\pi((N-1)(N-1))/N} \end{pmatrix} \tag{6.2}$$

Since, $\boldsymbol{W}$ is nonsingular, we may express the inverse DFT as

$$\boldsymbol{f} = \boldsymbol{W}^{-1}\boldsymbol{F} \tag{6.3}$$

Through using this method of matrix multiplication, we can calculate the DFT in $O(n^2)$ time where n is the length of our input vector. However with sampled music, our inputs will be enormous and we would like to find a method that allows us to compute the DFT in a faster amount of time.

6.1 The Fast Fourier Transform

In 1965, James Cooley and John Tukey published an algorithm that fundamentally changed the digital signal processing landscape [3]. By exploiting symmetries of the

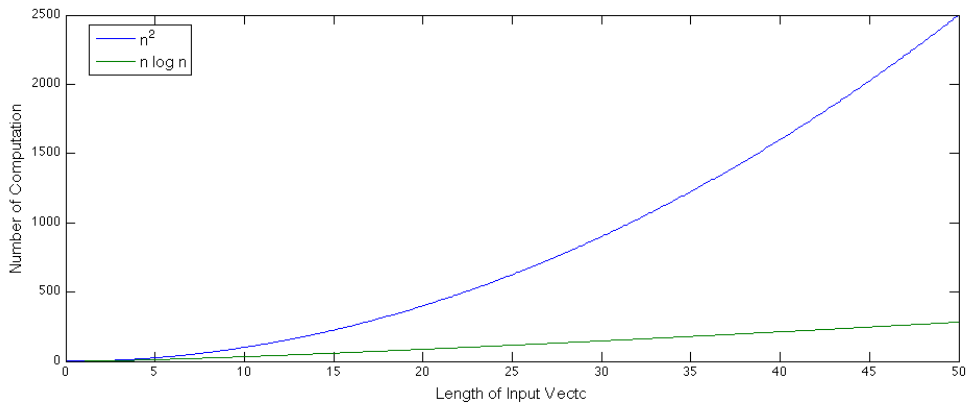

Figure 6.1: A quick comparison of $O(n^2)$ and $O(n \log_2 n)$ processing speed.

DFT, they were able to reduce the running time of DFTs from $O(n^2)$ to $O(n \log_2 n)$. This algorithm is the first fast Fourier transform (FFT) named for this increase in computational speed. As can been seen in Figure 6.1 this cut in processing time is quite significant for a input vector of length 50. In this study we are using audio that have sampling frequencies of 11025 Hz. Thus in a three minute song, we have about 2,000,000 input points. In this case, our $O(n \log_2 n)$ FFT algorithm provides us with a frequency representation of our data

$$\frac{n^2}{n \log_2 n} = \frac{(2 \cdot 10^6)^2}{(2 \cdot 10^6) \log_2 (2 \cdot 10^6)} \approx 100,000 \text{ times faster.}$$

Clearly this boost in speed that the FFT provides is huge. We can calculate a FFT in MATLAB in fractions of a second when a full DFT would take hours. Thus the FFT facilitates the spectral analysis or frequency domain analysis of large data such as audio files.

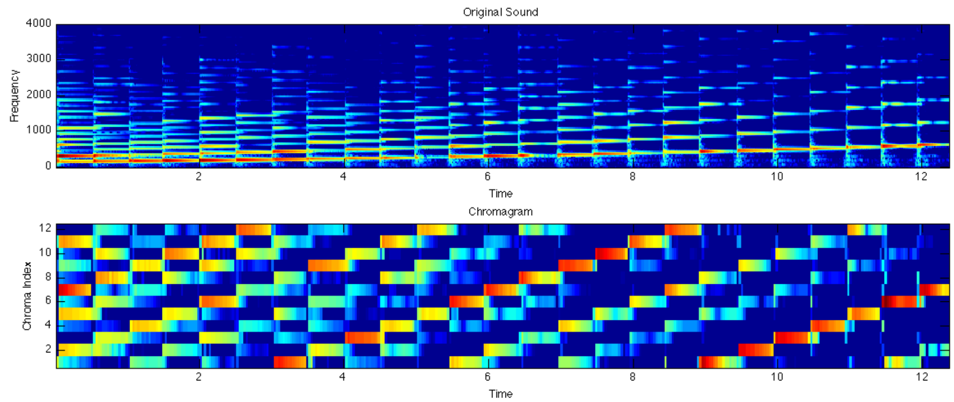

Figure 6.2: The spectrogram (top) and chromagram (bottom) of an ascending scale.

6.2 Spectrograms and Chromagrams

By looking at the FFT of overlapping segments of time in our audio file, we can construct the Short-Time Fourier Transform (STFT). Each of the FFTs in the STFT reveal the frequency domain representation or spectrum of a small time interval of our input signal. When combining all of these time segments with the proper synthesis, we can construct data on the intensities of frequencies as time progresses [9]. The STFT allows us to add a time dimension to our DFT which enables us to observe how the frequency domain changes over time. Since frequencies are representative of pitches, we can use the spectrograms to determine the chroma played at a moment of time in our signal. The creation of these spectrographs is crucial in determining how chords are changing in music. As stated previously, we do not care about the octave information of the sound. We determine the chroma intensity by collecting all intensities of a note regardless of its octave. The algorithm of collecting these chroma is referred to as a chromagram, and it contains the information that we pass to our Hidden Markov Model to determine the chord represented.

In this study we will be using MATLAB code written by Professor Dan Ellis at Columbia University [4]. With his code, we were able to create a spectrograph of a

~ 12 second audio clip of a piano playing an ascending chromatic scale. The results of our analysis are demonstrated in Figure 6.2. On top we have the spectrogram of the sound. From this plot it is clear when each note was struck as well as the general upward trend in frequency as the scale ascends. Through an algorithm developed by Professor Ellis we were able to extract the chromagram from the spectrogram [4]. For information of the MATLAB code used in this algorithm please see Appendix A.

In the bottom plot in Figure 6.2 we can see when each key was struck and we can see some sort of upward trend. However, we observe a behavior that when reaching the 12^{th} chroma, the major intensity block jumps back down to the 1^{st}. This jump is clear around $t = 9$ and is explained by our disregard for octave information when discussing chroma. Since we are only describing the note name, a jump from B$\sharp$ to C is a jump from 12 to 1 in the chroma space. Clearly, our chromagram is accurately describing the pitch characteristics of a chromatic scale and we have the first step in our chord detection model.

7 Conclusion

In this study we have laid a foundation for understanding the mathematics behind a chord recognition model. We have provided the reader with a general knowledge of music and a description of the current methods for chord recognition. In addition, we derived the DFT from the Fourier transform and gained an appreciation for the applications of Fourier analysis in signal processing. Finally, we have shown an understanding of MATLAB and demonstrated how its FFT function can be used to create a chromagram from an audio file. Despite this progress our original question "How can we fit an audio file with chord labels?" remains considerably unanswered. While we now can deconstruct a raw time domain signal into its chroma, we have not developed any method for attaching chord labels. As we proceed further with this study, we will investigate how a hidden Markov model can be used to do such fitting and complete our understanding of the model.

8 Acknowledgments

I would like to thank Professor Deanna Needell for her guidance and support as my advisor on this project. I would also like to thank the CMC mathematics department for providing me with a wonderful college experience in a major I never thought I would pursue. A special thanks to Professor Dan Ellis of Columbia for answering my emails so throughly when I became exited in the topic of music analysis. Many thanks to Andrew Langdon for the aid in developing such a fascinating topic. A shout-out to my parents and brothers for all of their support and love over the years.

A MATLAB Code for Spectrogram

```
% Chroma Feature Analysis and Synthesis
% AUTHOR: Dan Ellis [dpwe@ee.columbia.edu]
% DATE CREATED: 2007/04/21 14:03:14
% EDITED BY: Nathan Lenssen [nlenssen13@cmc.edu]
% DATE EDITED: 2013/03/05
%
% This code reads in a .wav file and constructs a spectrogram and
% Chromagram. It requires functions that can be obtained from
% http://www.ee.columbia.edu/~dpwe/resources/matlab/chroma-ansyn/
% chroma-ansyn.tgz

% Read an audio waveform
[d,sr] = wavread('piano-chrom.wav');
% Calculate the chroma matrix.  Use a long FFT to discriminate
% spectral lines as well as possible (2048 is the default value)
cfftlen=2048;
C = chromagram_IF(d,sr,cfftlen);
% The frame advance is always one quarter of the FFT length.  Thus,
% the columns  of C are at timebase of fftlen/4/sr
tt = [1:size(C,2)]*cfftlen/4/sr;
% Plot spectrogram using a shorter window
subplot(211)
sfftlen = 512;
specgram(d,sfftlen,sr);
% Always use a 60 dB colormap range
caxis(max(caxis)+[-60 0])
title('Original Sound')
% Now the chromagram, also on a dB magnitude scale
subplot(212)
imagesc(tt,[1:12],20*log10(C+eps));
axis xy
caxis(max(caxis)+[-60 0])
title('Chromagram')
```

References

[1] Rajendra Bhatia. *Fourier Series*. The Mathematical Association of America (Incorporated), 2005.

[2] William L. Briggs and Van Emden Henson. *The DFT, An Owner's Manual for the Discrete Fourier Transform*. The Society for Industrial and Applied Mathematics, 1995.

[3] James Cooley and John Tukey. *An algorithm for the machine calculation of complex Fourier series*. Mathematics of Computation, 1965.

[4] Daniel P.W. Ellis. Chroma feature analysis and synthesis. `http://www.ee.columbia.edu/~dpwe/resources/matlab/chroma-ansyn/`, [Online; accessed 22-February-2013].

[5] Ben Gold, Nelson Morgan, and Daniel P.W. Ellis. *Speech and Audio Signal Processing: Processing and Perception of Speech and Music*. Wiley and Sons Inc, 1999.

[6] Kyogu Lee. Automatic chord recognition from audio using enhanced pitch class profile. *in Proceedings of International Computer Music Conference*, 2006.

[7] Catherine Schmidt-Jones. Standing waves and musical instruments. `http://cnx.org/content/m12413/latest/StandingWaveHarmonics.png`, [Online; accessed 28-March-2013].

[8] Alexander Sheh and Daniel P.W. Ellis. Chord segmentation and recognition using em-trained hidden markov models. *John Hopkins University*, 2003.

[9] Julius O. Smith. *Mathematics of the Discrete Fourier Transform (DFT)*. W3K Publishing, `http://www.w3k.org/books/`, 2007.

[10] Wikipedia. File:chromatische toonladder.png. `http://en.wikipedia.org/wiki/File:Chromatische_toonladder.png`, [Online; accessed 10-March-2013].

[11] Wikipedia. File:root position, first inversion, and second inversion c major chords.png. `http://en.wikipedia.org/wiki/File:Root_position,_first_inversion,_and_second_inversion_C_major_chords.png`, [Online; accessed 12-March-2013].

[12] Wikipedia. File:type of triads-2.png. `http://en.wikipedia.org/wiki/File:Type_of_triads-2.png`, [Online; accessed 12-March-2013].