Journal of Humanistic Mathematics

Volume 4 | Issue 1 January 2014

An Introduction to Fourier Analysis with Applications to Music

Nathan Lenssen
Claremont McKenna College

Deanna Needell
Claremont McKenna College

An Introduction to Fourier Analysis with Applications to Music

Nathan Lenssen

Claremont McKenna College, Claremont, CA
nlenssen@gmail.com

Deanna Needell

Claremont McKenna College, Claremont, CA
dneedell@cmc.edu

Synopsis

In our modern world, we are often faced with problems in which a traditionally analog signal is discretized to enable computer analysis. A fundamental tool used by mathematicians, engineers, and scientists in this context is the discrete Fourier transform (DFT), which allows us to analyze individual frequency components of digital signals. In this paper we develop the discrete Fourier transform from basic calculus, providing the reader with the setup to understand how the DFT can be used to analyze a musical signal for chord structure. By investigating the DFT alongside an application in music processing, we gain an appreciation for the mathematics utilized in digital signal processing.

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 paper we discuss the problem of *chord detection*, where one wishes to identify played chords within a music file. 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 performing such a task must be highly versed in music theory and would 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 [17]. Here 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 and motivate the need for mathematical analysis in chord detection. Section 3 contains an introduction to the mathematics necessary to derive the discrete Fourier transform, which is included in Section 4. We conclude with a description of the Fast Fourier Transform and an example of its use in chord detection in Section 5.

2. Introduction to Music Theory

We begin with some musical terminology and definitions.

2.1. Pitches and Scales

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. A *scale* is 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 one-octave increase corresponds to a doubling in the frequency of the sound wave. Western music uses the chromatic scale in which each of the twelve chroma are ordered over an octave. These twelve notes are spaced almost perfectly logarithmically over the octave. We can use a recursive sequence [13] to describe the chromatic scale:

$$P_i = 2^{1/12} P_{i-1},$$

where P_i denotes the frequency of one pitch, and P_{i-1} the frequency of the previous. We can hear the chromatic scale by striking every white and black key of a piano in order up an octave or visualize it by a scale such as that

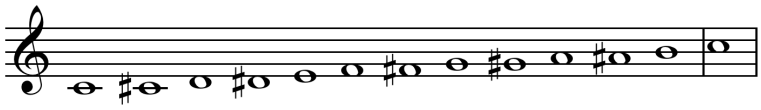

Figure 1 A chromatic scale beginning and ending at C. There are thirteen notes because C is played both at the top and the bottom. Image is from Wikipedia, http://en.wikipedia.org/wiki/File:Chromatische_toonladder.png, accessed on March 10, 2013.

depicted in Figure 1. Note that pitches use the names A through G, along with sharp (♯) and flat (♭) symbols; see [12] for more on musical symbols.

By using the chromatic scale as a tool, we can construct every other scale in Western music through the use of intervals. An *interval* refers to a change in position along the twelve 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. Two half step intervals makes a *whole step* and is 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. There are four prevalent scales in Western music: major, minor, augmented, and diminished. The intervals that describe these scales can be found in Table 1. The table is used by selecting any starting note as the root and then using the intervals to construct the remaining notes. For instance, a C minor (Cm) scale is C, D, E♭, F, G, A, B♭, C.

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

Table 1 Interval construction of the four core scales with the chromatic scale for reference. Note that the intervals apply for when ascending in pitch only. When determining the descending scale, the order of intervals is reversed.

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 below shows the triads for each of the four scales in the key of C. These triads describe the major (maj), minor (min), diminished (dim), and augmented (aug) chords in our model.

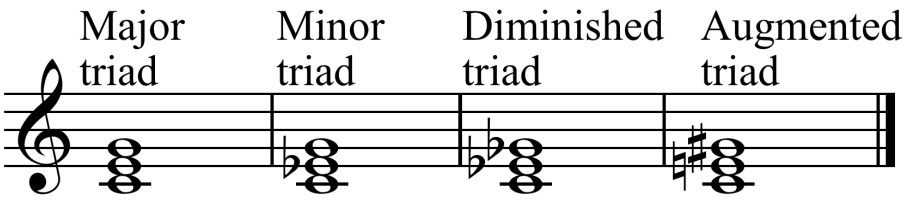

Figure 2 Triads in the key of C. Image from `Wikipedia`, `http://en.wikipedia.org/wiki/File:Type_of_triads-2.png`, accessed on March 12, 2013.

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) 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) chords. 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 scales. 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; we refer the interested reader to [12] for more information.

2.3. Chord Inversions

We have described seven chord families and twenty-one roots, giving a total of 147 different possible chords. However, this assumes that the root is

always the lowest note in the chord, which isn't always the case. Instead, a *chord inversion* may be used. An inversion can be described algorithmically as follows. First, the root is raised an octave. This is known as the first inversion. If we repeat that process again (with the new lowest note), we are left with the second inversion. For a triad, two inversions recover the original chroma arrangement. When chords involve a seventh, one can perform three inversions. See Figure 3 for an example.

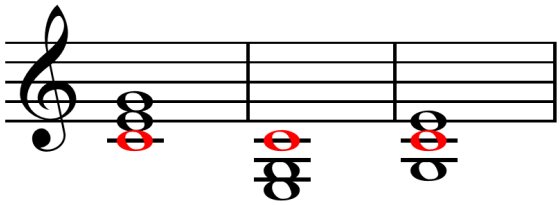

Figure 3 Root position, first inversion, and second inversion triads in the key of C. Image from `Wikipedia`, `http://en.wikipedia.org/wiki/File:Root_position,_first_inversion,_and_second_inversion_C_major_chords.png`, accessed on March 12, 2013.

Including inversions, the total number of chords is:

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

Distinguishing between so many 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.

2.4. Towards Chord Detection

Chord analysis has been widely studied (see e.g. [1, 7, 9, 13, 15, 17] and the references therein). In this article, we will focus on a chord recognition model developed by Sheh and Ellis in 2003 [17]. 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 the most likely chord representations for the music are determined.

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. The FFT is used to determine the fundamental frequencies and therefore pitches that are present in the raw signal. However, the octave of the pitch is generally irrelevant to the chord identity, so one needs 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) [7].

Once the PCP of the input is discovered, it is used to determine the best fitting chords for the music. To do this, a Hidden Markov Model that has been trained by a large set of pre-annotated audio files is used. The optimal chord assignments given a PCP are determined through the expectation maximization (EM) algorithm [8]. The Viterbi alignment algorithm [8] can be used to forcibly align the chord labels with the timing of the music, creating a fully analyzed track.

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.

3. Introduction to the Fourier Transform

3.1. Frequency and Time Domains

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 [2, 3]. One instantly sees the relevance to chord detection since string instruments such as violins or pianos produce sound by amplifying the vibrations of 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 [2, 3]. History aside, the most crucial information from the development of Fourier analysis is that functions can be represented in both the *time domain* as well as the *frequency domain*.

The connection between the two domains is easiest to see in a periodic system such as a vibrating string. One would likely describe the motion of

a string by focusing on the changing position of points on the string over time. More rigorously, one can model such motion using a differential equation where the initial condition is the initial displacement [2]. Solving such differential equations yields a function $f(t)$ (where t is time) which represents the motion of the string. This function $f(t)$ is known as the *time-domain representation* of the motion of the string; it provides information about how the string behaves as time progresses.

Now imagine that we pluck the string in precisely the way that makes it vibrate only at the fundamental harmonic (see Figure 4 below). This pattern is represented in the time domain by a single sinusoid with frequency ν_0. Notice that the motion of the string is completely described by this frequency ν_0 and the amplitude of the oscillation. Thus, the frequency-domain representation $F(\nu)$ 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 helps to visualize the frequency domain, but in real systems there is typically more than one frequency. To account for this, one constructs the frequency-domain representation by 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 can obtain the frequency-domain representation 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.

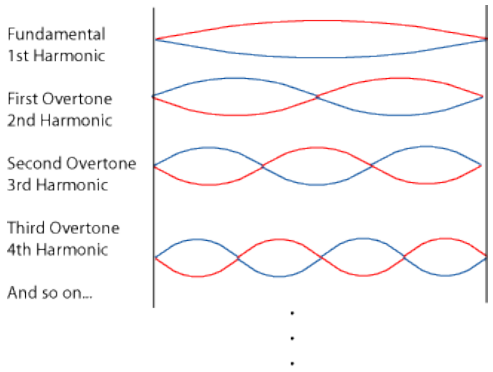

Figure 4 The first four standing waves of a vibrating, fixed string. Image from [16].

a string by focusing on the changing position of points on the string over time. More rigorously, one can model such motion using a differential equation where the initial condition is the initial displacement [2]. Solving such differential equations yields a function $f(t)$ (where t is time) which represents the motion of the string. This function $f(t)$ is known as the *time-domain representation* of the motion of the string; it provides information about how the string behaves as time progresses.

Now imagine that we pluck the string in precisely the way that makes it vibrate only at the fundamental harmonic (see Figure 4 below). This pattern is represented in the time domain by a single sinusoid with frequency ν_0. Notice that the motion of the string is completely described by this frequency ν_0 and the amplitude of the oscillation. Thus, the frequency-domain representation $F(\nu)$ 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 helps to visualize the frequency domain, but in real systems there is typically more than one frequency. To account for this, one constructs the frequency-domain representation by 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 can obtain the frequency-domain representation 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.

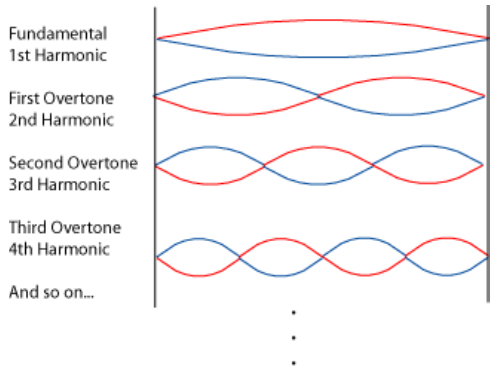

Figure 4 The first four standing waves of a vibrating, fixed string. Image from [16].

Audio signals are recorded and listened to in the time domain; they contain the intensity of the sound as a function of time. However, pitches and therefore chords are represented by frequencies, as shown in Section 2. The Fourier transform is used to convert the time-domain input signal into a frequency-domain 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. We should note that the Fourier transform of real-valued functions can have a complex-valued frequency-domain representation.

3.2. The Continuous Fourier Transform

We first define ω_k as the angular frequency through its relationship with the ordinary frequency ν by:

$$\omega_k \equiv 2\pi k\nu.$$

Then the relationship between the time-domain function f and its corresponding frequency-domain function F is defined by the *Fourier transform*:

$$F(\omega_k) \equiv \int_{-\infty}^{\infty} f(t)\, e^{-2\pi ikt}\, \mathrm{d}t, \qquad k \in (-\infty, \infty). \tag{1}$$

Note the presence of sinusoidal components within the complex exponential; $e^{i\omega t} = \cos(\omega\, t) + i \sin(\omega\, t)$. From (1) one can derive the inverse Fourier transform:

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

Fourier analysis in the continuous case is a rich subject of study. However, digital applications require its discrete counterpart, the discrete Fourier transform (DFT).

3.3. The Discrete Fourier Transform

Motivated by the continuous case, we define the discrete Fourier transform for vector $f \in \mathbb{C}^N$ by:

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

where we write f_n to denote the nth entry of the vector f. Likewise, the inverse DFT has the form:

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

We next study these transforms in more detail, keeping in mind our original goal, chord detection and audio processing.

3.4. Sinusoids

We define a sinusoid as a function of the form:

$$x(t) = A\sin(2\pi\nu t + \phi).$$

When discussing audio signals, as in [18], we let:

$$\begin{aligned} A &= \text{amplitude;} \\ \nu &= \text{radian frequency (rad/sec);} \\ 2\pi\nu &= \text{frequency (Hz);} \\ t &= \text{time (s);} \\ \phi &= \text{initial phase (radians);} \\ 2\pi\nu t + \phi &= \text{instantaneous phase (radians).} \end{aligned}$$

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

$$e^{i\theta} = \cos(\theta) + i\sin(\theta), \text{ and} \tag{3}$$

$$e^{\pm i2\pi\nu x} = \cos(2\pi\nu x) \pm i\sin(2\pi\nu x), \tag{4}$$

the latter being the form most relevant to audio signal processing.

3.5. The Delta (δ) Function

Although sound is naturally analog, computers must store sound in digital format. For that reason, sound must be *sampled* in an efficient way for

an accurate representation.[1] Given a continuous signal $f(t)$ as an input over time $t \in [0, T]$, the sampler returns a time-series vector that contains values according to some sampling frequency with a uniform gap Δt between samples. This vector can be viewed as a continuous "function" that is a piece-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 [3]. The delta function, or *spike*, is a special mathematical construct with a number of properties that are intuitive and useful to the study of the DFT. We now outline a few of its properties that will be most relevant to us.

First, the delta function is zero at every point except $t = 0$, where it is infinite. That is:

$$\delta(t) = \begin{cases} \infty, & \text{if } t = 0; \\ 0, & \text{if } t \neq 0. \end{cases}$$

This property illuminates why we use the term spike. The delta function is an infinitely thin, infinitely tall peak centered at zero. A delta function centered at $t = a$ can of course be written as $\delta(t - a)$.

Second, the area under the delta function is defined to be 1. That is:

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

This property can be exploited to reveal the useful general sifting property to discretize integrals:

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

The relevance of the general sifting property to sound processing cannot be overstated. For a continuous input signal $f(t)$, the sifting property and $\delta(t - a)$ can be used to obtain an evenly spaced sequence of a values corresponding to the instantaneous values of the original function. While certainly not a sophisticated method, it provides a relevant and simple application that demonstrates the power of the delta function.

[1] The mathematics of sampling is a large field, we refer the reader to modern texts such as [14] for a thorough discussion, and [5, 10] for recent progress in this area.

3.6. Fourier Transforms of Delta Functions

We begin by looking at the Fourier transform of a single spike centered at zero, which will be instrumental in our derivation of the DFT. By the general sifting property we have:

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

This means that a spike in the time domain translates to a flat line in the frequency domain (an even weight among all frequencies). This is a somewhat physically curious result that can be reconciled by understanding that for every frequency ν, $\cos(2\pi\nu \cdot 0) = 1$, 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 the 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 delta function:

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

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:

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

By clever addition of pairs of Inverse Fourier transforms, we can determine what spikes in the frequency domain have inverse Fourier transforms that are sine and cosine functions in the time domain. We notice that to get a cosine 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 have:

$$\begin{aligned} f(t) &= \int_{-\infty}^{\infty} \left[\frac{1}{2}\delta(\nu-\nu_0) + \frac{1}{2}\delta(\nu+\nu_0)\right] e^{i2\pi\nu x}\,\mathrm{d}x \\ &= \frac{1}{2}\int_{-\infty}^{\infty} \delta(\nu-\nu_0)e^{i2\pi\nu x}\,\mathrm{d}x + \frac{1}{2}\int_{-\infty}^{\infty} \delta(\nu+\nu_0)e^{i2\pi\nu x}\,\mathrm{d}x \\ &= \frac{e^{i2\pi\nu_0 x} + e^{-i2\pi\nu_0 x}}{2} \\ &= \cos(2\pi\nu_0 x). \end{aligned} \tag{5}$$

Similarly for the sine function we have the inverse Fourier transform:

$$\begin{aligned} f(t) &= \int_{-\infty}^{\infty} \left[\frac{1}{2i}\delta(\nu-\nu_0) - \frac{1}{2i}\delta(\nu+\nu_0)\right] e^{i2\pi\nu x}\,\mathrm{d}x \\ &= \frac{1}{2i}\int_{-\infty}^{\infty} \delta(f-f_0)e^{i2\pi\nu x}\,\mathrm{d}x - \frac{1}{2i}\int_{-\infty}^{\infty} \delta(\nu+\nu_0)e^{i2\pi\nu x}\,\mathrm{d}x \\ &= \frac{e^{i2\pi\nu_0 x} - e^{-i2\pi\nu_0 x}}{2i} \\ &= \sin(2\pi\nu_0 x). \end{aligned} \tag{6}$$

In other words, the 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 ν 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.

4. A Derivation of the Discrete Fourier Transform

4.1. Spike Trains and the Discrete Fourier Transform

As we have defined a spike as a δ function, a *spike train* is simply a linear combination of δ functions. 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 the spike train over time $h(t)$ as:

$$h(t) = \sum_n f_n \delta(t - t_n), \qquad n = 0, 1, \ldots, N-1.$$

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

$$h(t) = \sum_n f_n \delta(t - t_n) = \sum_n f(t_n)\delta(t - t_n) = \sum_n f(t)\delta(t - t_n).$$

Thus, we can easily construct the spike train of a function by summing a series of evenly spaced delta functions multiplied by the original continuous function.

With the definition of a spike train we can begin to reveal the DFT. Assume we have an evenly spaced spike train of length N on a 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 $t_n = nL/N$ for $n = -N/2 + 1 \ldots N/2$. Again making use of the general sifting property of δ functions we have:

$$\begin{aligned} F(\nu) &= \int_{-\infty}^{\infty} \left(\sum_{n=-\frac{N}{2}+1}^{\frac{N}{2}} f_t \delta(t - t_n) \right) e^{-i2\pi\nu t} \, \mathrm{d}t \\ &= \sum_{n=-\frac{N}{2}+1}^{\frac{N}{2}} f_n \int_{-\infty}^{\infty} \delta(t - t_n) e^{-i2\pi\nu t} \, \mathrm{d}t \\ &= \sum_{n=-\frac{N}{2}+1}^{\frac{N}{2}} f_n e^{-i2\pi\nu x_n}. \end{aligned} \tag{7}$$

Thus the Fourier transform of a spike train is simply the sum of exponentials weighted by their intensities in the spike train [3]. We now have an expression that is very close to the definition of the DFT given in (2). To make (7) match, we need to define and use the reciprocity relations of the discrete Fourier transform.

4.2. 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 in $\mathbb{C}^N$ of length N evenly spaced over a total time of L, centered at $t = 0$ so that the time

interval runs over $[-L/2, L/2]$. If the 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 length Ω, again centered at $\nu = 0$ so that the interval of frequencies is $[-\Omega/2, \Omega/2]$. If the 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 the time domain function. Likewise, we are looking at the 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 the period and we will call this longest period the *fundamental frequency*, which is also the step size in the frequency domain. That is:

$$\Delta\nu = \frac{1}{L}. \tag{8}$$

Thus the frequencies we will recognize will all be multiples of $\Delta\nu$ and have integer periods over the time interval. It is then clear that the length described in the frequency domain is just the number of points multiplied by the frequency step, or $\Omega = N\Delta\nu$. Using this in conjunction with (8) we have the first reciprocity relation:

$$\Omega = N\Delta\nu = \frac{N}{L} \quad \Longrightarrow \quad A\Omega = L. \tag{9}$$

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 (9) by noting that taking temporal data over a longer range of time means that the DFT yields a smaller range of frequencies and vice versa [3].

Recalling from (8) that $\Delta\nu = 1/L$, we can present the second reciprocity relation as:

$$\frac{1}{\Delta\nu} = L = N\Delta t \quad \Longrightarrow \quad \Delta t\Delta\nu = \frac{1}{N}.$$

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.

We now have the tools necessary to derive the precise form of the DFT.

4.3. The Discrete Fourier Transform

Recall from (7) that:

$$F(\nu) = \sum_{n=-\frac{N}{2}+1}^{\frac{N}{2}} f_n e^{-i2\pi\nu x_n}. \tag{10}$$

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 $\nu_k = k\Delta\nu = k/L$ for $k = -L/2 + 1 \ldots L/2$. We use this to show that (10) can be rewritten as:

$$F(\nu) = \sum_{n=-\frac{N}{2}+1}^{\frac{N}{2}} f_n e^{-i2\pi\nu_k x_n} = \sum_{n=-\frac{N}{2}+1}^{\frac{N}{2}} f_n e^{-i2\pi nk/N}. \tag{11}$$

The DFT as defined in (2) is now apparent.

We now move on to see how the DFT can be employed to analyze music.

5. The Chromagram

Since the DFT is a sum of indexed values, we can express equation (11) in matrix form as the linear equation:

$$\boldsymbol{F} = \boldsymbol{W}\boldsymbol{f},$$

where $\boldsymbol{f}$ is the vector in the time domain of length N, $\boldsymbol{F}$ is the output in the frequency domain, and $\boldsymbol{W}$ is the nonsingular matrix:

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

We know that $\boldsymbol{W}$ must be nonsingular as one can readily verify that the columns are orthogonal. Since $\boldsymbol{W}$ is nonsingular, we may express the inverse DFT by:

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

Using this method of matrix multiplication, we can calculate the DFT in $O(n^2)$ time where n is the length of the input vector. However, with sampled music, the inputs are extremely high dimensional and we would like to find a method that computes the DFT much faster.

5.1. The Fast Fourier Transform

In 1965, Cooley and Tukey published an algorithm that fundamentally changed the digital signal processing landscape [4]. By exploiting symmetries of the DFT, they were able to reduce the running time of DFTs from $O(n^2)$ to $O(n \log n)$. This algorithm is the first fast Fourier transform (FFT), named for this increase in computational speed. As can been seen in Figure 5 below, this reduction in processing time is quite significant even for an input vector of length 50. In our examples, we use audio that has sampling frequencies of 11025 Hz. Thus in a three minute song, there are about 2,000,000 input points. In this case, the $O(n \log n)$ FFT algorithm provides a frequency representation of our data:

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

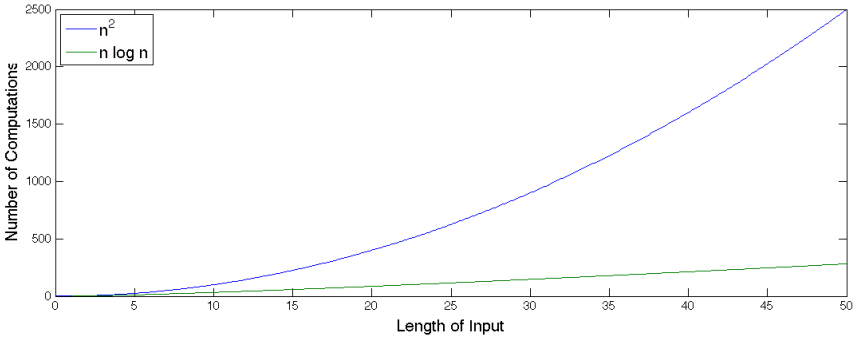

Figure 5 A quick comparison of $O(n^2)$ and $O(n \log_2 n)$ processing speed.

Clearly this boost in speed that the FFT provides is substantial. 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.

5.2. Spectrograms and Chromagrams

Looking at the FFT of overlapping segments of time in an audio file, we can construct the Short-Time Fourier Transform (STFT) [11]. Each of the FFTs in the STFT reveals the frequency-domain representation or spectrum of a small time interval of the input signal. Combining all of these time segments with the proper synthesis, we can construct data on the intensities of frequencies as time progresses [18]. The STFT allows us to add a time dimension to the 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 the signal. The creation of these spectrographs is crucial in determining how chords are changing in music. As stated previously, the octave information of the sound is irrelevant. 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 the Hidden Markov Model to determine the chord represented [17].

Let us look at an example. We will use the MATLAB code by Ellis [6]. With this code, we can create a spectrograph of a $\sim$ 12-second audio clip of a piano playing an ascending chromatic scale. The results of our analysis are demonstrated in Figure 6 on the next page. The spectrogram of the sound is depicted on top, from which it is clear when each note is struck as well as the general upward trend in frequency as the scale ascends. Through an algorithm developed by Ellis we are able to extract the chromagram from the spectrogram [6].

In the bottom plot of Figure 6, we can see when each key is struck as well as an overall upward trend. However, we observe that when reaching the 12^{th} chroma, the major intensity block jumps back down to the first. 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 to C is a jump from 12 to 1 in the chroma space.

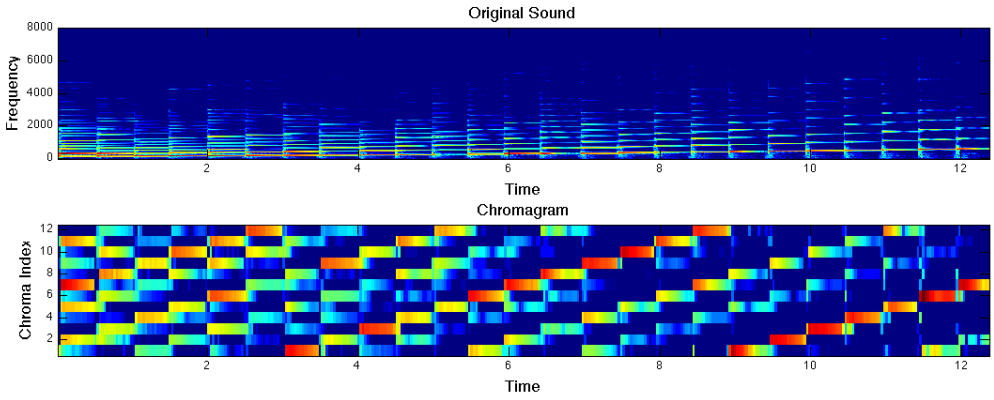

Figure 6 The spectrogram (top) and chromagram (bottom) of an ascending scale.

Clearly, the chromagram is accurately describing the pitch characteristics of a chromatic scale and we have the first step in our chord detection model.

6. Conclusion

In this paper we have laid a foundation for understanding the mathematics behind a chord recognition model. We have provided the reader with general knowledge of music and a description of a contemporary method 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 through examples how the FFT of a function can be used to create a chromagram from an audio file.

Acknowledgments

We would like to thank Professor Dan Ellis of Columbia University for his very helpful guidance and suggestions. In addition, we thank Professor Mark Huber of Claremont McKenna College for directing this discussion to the venue of the *Journal of Humanistic Mathematics*.

References

[1] Mark A. Bartsch and Gregory H. Wakefield, "To catch a chorus: Using chroma-based representations for audio thumbnailing," in *Proceedings of the IEEE Workshop on the Applications of Signal Processing to Audio and Acoustics*, Institute of Electrical and Electronics Engineers, New Platz NY, 2001, pages 15–18.

[2] Rajendra Bhatia, *Fourier Series*, Mathematical Association of America, Washington DC, 2005.

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

[4] James W. Cooley and John Tukey, *An algorithm for the machine calculation of complex Fourier series*, Mathematics of Computation, Bell Telephone Laboratories, New York, 1965.

[5] "Compressive Sensing Resources," online resource, available at http://www.dsp.ece.rice.edu/cs/, accessed on January 8, 2014.

[6] Daniel P. W. Ellis, "Chroma feature analysis and synthesis," available online at http://www.ee.columbia.edu/~dpwe/resources/matlab/chroma-ansyn/, accessed on February 22, 2013.

[7] Takuya Fujishima, "Realtime chord recognition of musical sound: a system using common lisp music," in *Proceedings of the International Computer Music Association*, MPublishing, University of Michigan Library, Ann Arbor MI, 1999, pages 464–467.

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

[9] Ben Gold, Nelson Morgan, and Daniel P. W. Ellis, *Speech and Audio Signal Processing: Processing and Perception of Speech and Music*, 2nd Edition, Wiley-Interscience, Oxford, 2011.

[10] Anthony Griffin, Toni Hirvonen, Christos Tzagkarakis, Athanasios Mouchtaris, and Panagiotis Tsakalides, "Single-channel and multi-channel sinusoidal audio coding using compressed sensing," *IEEE Transactions on Audio, Speech, and Language Processing*, Volume **19** Issue 5 (2011), pages 1382–1395.

[11] Eric Jacobsen and Richard Lyons, "The sliding DFT," *IEEE Signal Processing Magazine*, Volume **20** Issue 2 (2003), pages 74–80.

[12] Steven G. Laitz, *The Complete Musician: An Integrated Approach to Tonal, Theory, Analysis, and Listening*, Oxford University Press, New York, 2011.

[13] Kyogu Lee, "Automatic chord recognition from audio using enhanced pitch class profile," in *Proceedings of the International Computer Music Conference*, MPublishing, University of Michigan Library, Ann Arbor MI, 2006, pages 306–313.

[14] Matt Pharr and Greg Humphreys, *Physically Based Rendering: From Theory to Implementation*, Elsevier / Morgan Kaufmann, Amsterdam, 2010.

[15] Christopher Raphael, "Automatic transcription of piano music," in *Proceedings of The International Society for Music Information Retrieval*, 2002, pages 15–19. Also available online at http://ismir2002.ismir.net/proceedings/02-FP01-2.pdf, accessed on January 8, 2014.

[16] Catherine Schmidt-Jones, "Standing waves and musical instruments," online resource, available at http://cnx.org/content/m12413/latest/, accessed on March 28, 2013.

[17] Alexander Sheh and Daniel P. W. Ellis, "Chord segmentation and recognition using EM-trained hidden Markov models," in *Proceedings of the International Symposium on Music Information Retrieval 2003*, pages 185–191, 2003.

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

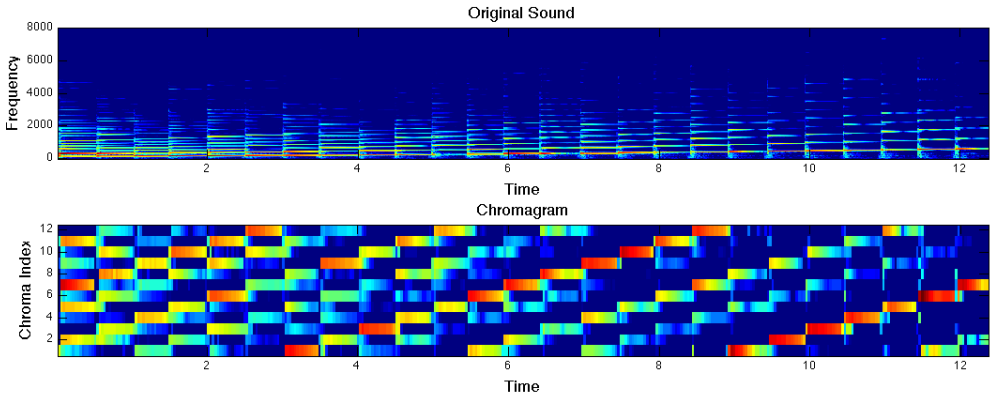

Figure 6 The spectrogram (top) and chromagram (bottom) of an ascending scale.

Clearly, the chromagram is accurately describing the pitch characteristics of a chromatic scale and we have the first step in our chord detection model.

6. Conclusion

In this paper we have laid a foundation for understanding the mathematics behind a chord recognition model. We have provided the reader with general knowledge of music and a description of a contemporary method 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 through examples how the FFT of a function can be used to create a chromagram from an audio file.

Acknowledgments

We would like to thank Professor Dan Ellis of Columbia University for his very helpful guidance and suggestions. In addition, we thank Professor Mark Huber of Claremont McKenna College for directing this discussion to the venue of the *Journal of Humanistic Mathematics*.