EE386 Digital Signal Processing Lab **Jul-Dec 2021**

Time-Frequency Analysis of Music Signals

Group number: G09 *email:* siddharthiyer.191ee153@nitk.edu.in

1 Introduction

In this project we describe and compare the various approaches to analyzing musical instrument sounds. Musical sound is much more complicated than human vocal sound and it occupies a wider band of frequency. Music is a type of sound that has some stable frequencies in a time period. Irrespective of the way it is produced, musical sounds have their fundamental frequency, which is the lowest frequency in the harmonic series and its integer multiples called overtones. In music theory, pitch is the perceived fundamental frequency of a sound, however the actual fundamental frequency may differ from the perceived because of the overtones. The classic approach to the analysis of such signals is using Fourier analysis which identifies the fundamentals and overtones of individual notes, but the time varying nature of music signals renders it insufficient. To produce a time-frequency description of a musical passage, a second method, spectrograms produce a visual representation of the frequency spectrum of a signal, analyzing the changes in fundamental frequencies and their overtones over time as several notes are being played. In order to obtain far more detailed time-frequency descriptions, a third method, Scalograms allows us to zoom in on selected regions of the time-frequency plane in a manner more flexible than Spectrograms. It gives us a visual representation of a wavelet transform and has a natural interpretation in terms of a musical scale. In this project all three techniques are employed in analyzing music. The two time–frequency methods Spectrograms and Scalograms are shown as an extension of the classic Fourier approach. Hence time-frequency portraits that correlate well with our perceptions of sounds are created and their advantages and applications are discussed.

This project is based on [2] which implements time frequency analysis ranging from the classic method of Fourier to the most up-to-date methods of dynamic spectra and wavelets using FAWAV, a software designed by the authors Jeremy Alm and James Walker. We have taken a similar approach in Python. In addition, the following exist as part of the literature.

[3] uses continuous wavelet transforms and Gabor transforms(STFTs or Spectograms) to analyze the rhythmic structure of music and its interaction with the melodic structure, thus finding the hierarchical structure of sound. Initially, the mathematical method of Gabor transform is summarized, followed by spectrograms for visualizing the patterns of time-frequency structures

within some segment of the music. Finally, percussion scalograms are obtained , which are used to analyze percussion passages and rhythm.

[9] tells us about the inadequacy of Fourier mathematics to clearly represent perpetual phenomena of musical signals, thus motivating the theory of Time-Frequency (TF) representations.

[11] gives a detailed account of windows and their applications. Chapters 3 and 5 of this book give a detailed description of some of the important time-domain windows described in this report.

[12] covers the main definitions and properties of wavelet transforms and shows connection among various fields where results have been developed for signal processing applications. It tells us about the limitations of time-frequency representation of signals with regard to their resolution.

Chapter 7 of [10] describes windows and its types and properties.

Chapter 7 of [7] gives a detailed account of Short-Time Fourier Analysis (STFT) covering important topics like time–frequency resolution, the uncertainty principle, reconstruction of a signal among others.

[8] explains in detail the various concepts of Discrete-Time Signal Processing. Chapter 10 of this book talks about the Fourier analysis of signals using DFT, covering important topics like the effects of windowing and spectral sampling among others.

2 The Need for Time-Frequency Analysis

Time-Frequency analysis is a branch of signal processing that attempts to characterize a signal's time and frequency information simultaneously. The need for time-frequency analysis stems from the fact that many real-world signals are not stationary in nature. A non-stationary signal is one whose frequency contents vary with time. In other words, the signal is not a simple linear combination of everlasting exponentials of multiple frequencies, rather the frequencies present in the signal evolve with time. Musical signals are always non-stationary in nature due to the fact that they are essentially successions of different notes in a particular order. Each note generally corresponds to a certain frequency or set of frequencies(fundamental and overtones), and thus, at different instants of time, the signal contains different sets of frequencies. A simple example is shown in Figure 1. The first image depicts a stationary signal, which is simply a sum of 2 sinusoids of different frequencies with both sinusoidal components existing across the entire time range chosen. The second image depicts the time-domain behaviour of a non-stationary signal. Once again, there are 2 sinusoidal components of different frequencies, however, the lower frequency component exists only from $t = 0s$ to $0.1s$ At $t = 1s$, there is an abrupt change and now only the second, higher frequency component is present.

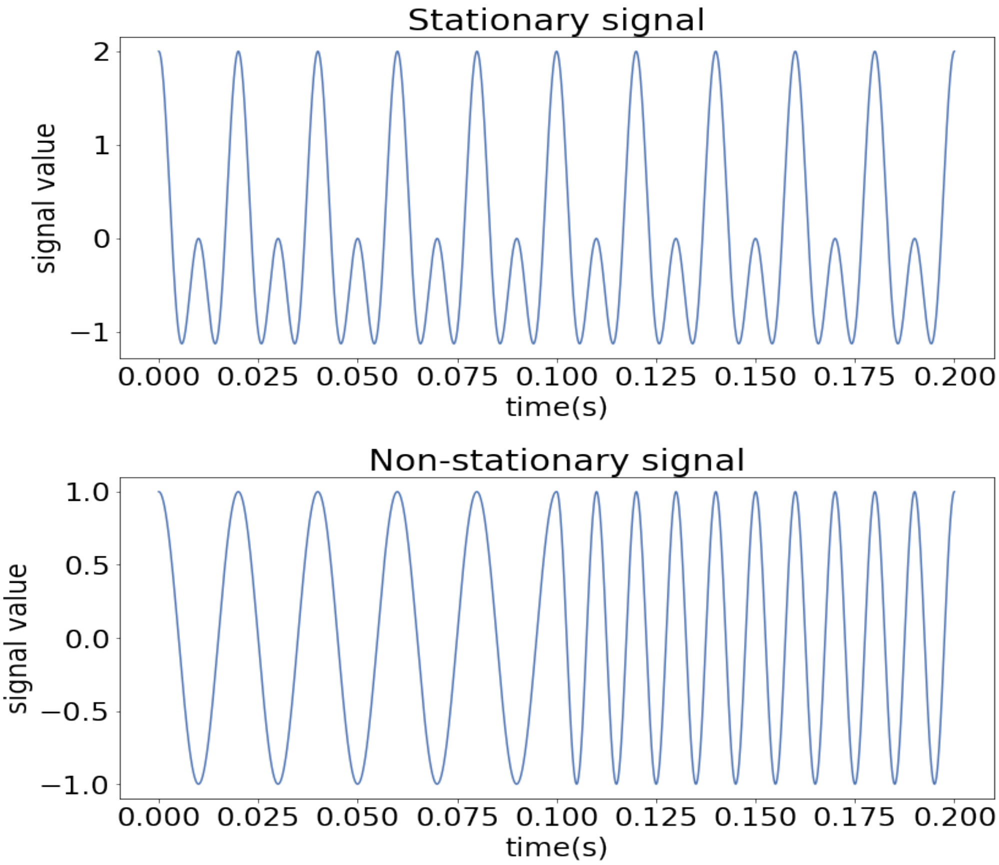

Figure 1: Stationary vs Non-stationary signals

One of the most popular techniques used to examine a signal's frequency content is Fourier analysis. Fourier analysis attempts to express a signal as a combination of simple everlasting exponentials of different frequencies, i.e functions of the form $e^{j\Omega}$, where $\Omega \in \mathbb{R}$. For a continuous-time signal $x(t)$, the Fourier transform analysis equation is given by:

$$X(\Omega) = \int_{-\infty}^{\infty} x(t)e^{-j\Omega t}\, dt \tag{2.1}$$

The signal can be reconstructed from its Fourier transform using the synthesis equation:

$$x(t) = \frac{1}{2\pi}\int_{-\infty}^{\infty} X(\Omega)e^{j\Omega t}\, d\Omega \tag{2.2}$$

Both representations of the signal, $x(t)$ and $X(\Omega)$, contain all the information of the signal, and are simply alternate views of the signal. However, neither of the domain representations directly indicates anything about the other. The time-domain signal tells you exactly how the signal behaves in time, but not what frequency components it contains, while the Fourier transform of $x(t)$ tells you what frequencies are present in the signal, but not when in time they occur. As an example, figure 2 shows the magnitude spectra (evaluated using the DFT) of 2 finite sampled continuous-time signals. By looking at the plots, one would conclude that both signals are identical, as the plots indicate that both signals have exactly the same frequency components, but in fact, both signals are composed of 3 different single-frequency tone components that are played one after the other, rather than all at the same time, i.e the signal is nonstationary, and if one were to listen to the corresponding audio clips, one would clearly observe that the first signal has the frequencies $440Hz$, $600Hz$ and $880Hz$, occurring in that order, while the second signal has the same 3 frequency components occurring in the reverse order. This is similar to a music signal, which has different (sets of) frequencies occurring in a particular order, and the order is of great importance, which is why classical Fourier analysis is not an appropriate tool to analyse these signals. Figure 3 shows the spectrogram(a kind of time-frequency analysis tool) plots of the aforementioned 2 signals. Spectrograms will be discussed in detail in a later section, however, it can be seen clearly from the spectrograms that they do, in fact, indicate exactly which frequencies are present in the signals, as well as the order in which they occur in time.

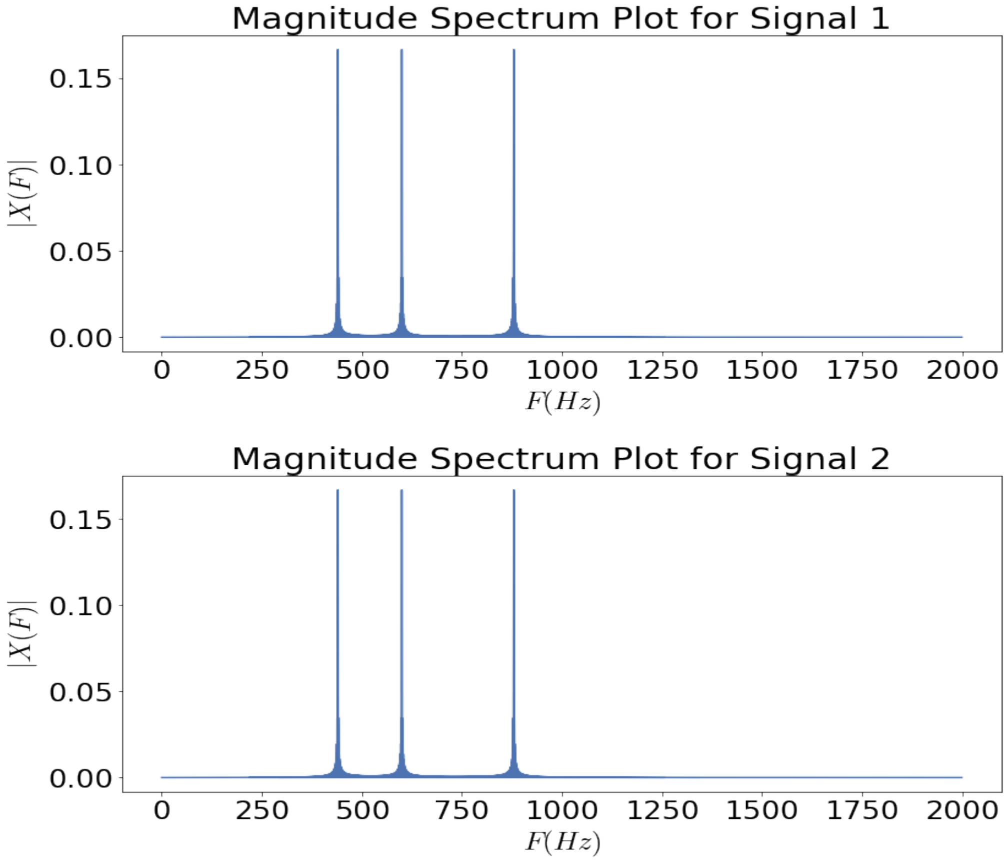

Figure 2: Simple FFT magnitude Plots for the 2 signals

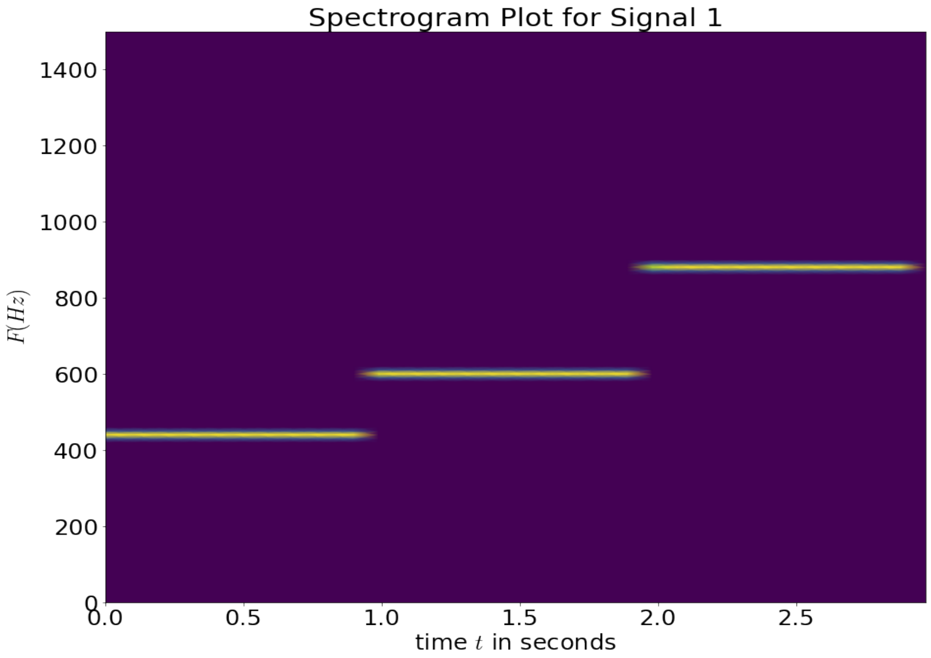

Figure 3: Spectrogram plot of first signal

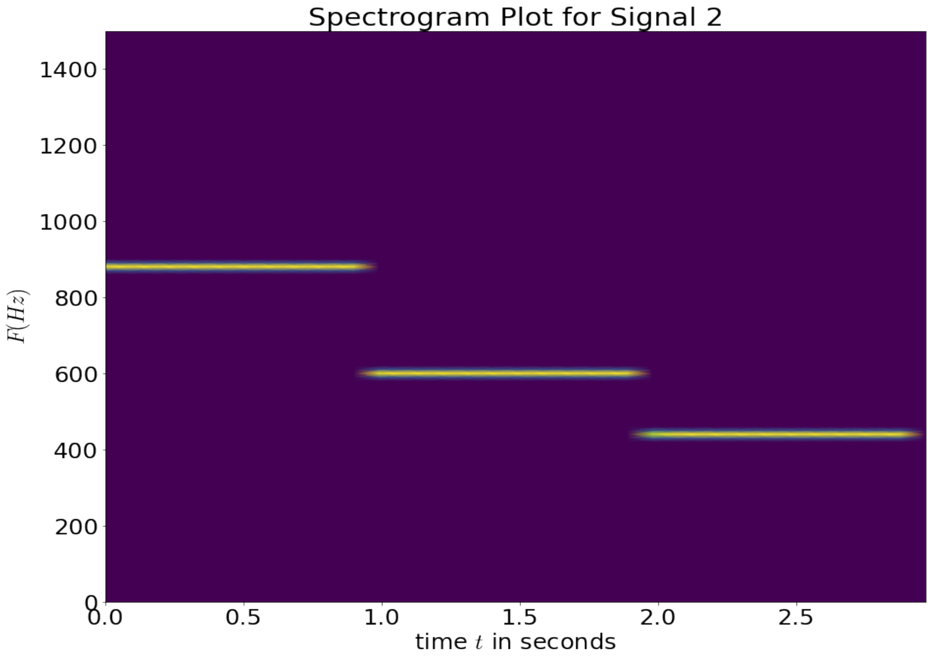

Figure 4: Spectrogram plot of second signal

3 The Short-Time Fourier Transform(STFT)

One of the most important time-frequency analysis tools is the Short-time Fourier transform (STFT), also sometimes called the time-dependent Fourier transform.

The STFT is an integral transform that maps a signal to a complex function of 2 real variables: τ and Ω. The continuous-time STFT of the signal $x(t)$ is given by:

$$X(\tau, \Omega) = \int_{-\infty}^{\infty} x(t)w(t-\tau)e^{-j\Omega t}\,dt \tag{3.1}$$

where Ω is the continuous time frequency (rad/sec) and $w(t)$ is a window function. The window function is simply a signal whose value is 0 outside a particular range (usually symmetric about the origin), and whose value peaks at $t = 0$ and tapers away towards the ends. Thus multiplying the signal $x(t)$ with $w(t)$ can be thought of as selecting a part of the signal near $t = 0$. The term in the integral is however, not $w(t)$, but rather $w(t-\tau)$. This can be thought of as shifting the window to the right by τ seconds before multiplying it by $x(t)$. Thus by choosing the value of τ, we can focus only on the portion of $x(t)$ near any desired time instant and carry out the Fourier analysis of the windowed signal in that range. This implies that we are, in a way, sliding our window function across the time axis, selecting different parts of the time-domain signal and performing Fourier analysis on that part of the signal alone.

Another view of the STFT is based on decomposition of the signal in terms of basis functions. In conventional Fourier analysis, the signal is expressed as a linear combination of simple everlasting complex exponentials of different real frequencies, i.e functions of the form $e^{j\Omega t}$ for all $\Omega \in \mathbb{R}$. Since, in general, we need an infinite number of such exponentials, the linear combination is replaced by multiplication with a function followed by integration (and an extra scaling). This function is nothing but the continuous time-Fourier transform of $x(t)$. In the STFT, we are instead resolving the signal $x(t)$ into a combination of a different set of basis functions. This set is essentially the set of all functions $w(t-a)e^{j\Omega t}$, where $a \in \mathbb{R}$ and $w(t)$ is the window function, i.e the set of all shifted and windowed complex exponentials.

Therefore, by using windowed and shifted complex exponentials as the basis functions, the STFT is able to localize the exponential to a particular location in time, which is why, by expressing the signal as a combination of such components, STFT is able to provide time and frequency data, as each component has a time component (the shift a of the window in the basis function), and a frequency component (the frequency Ω of the complex exponential in the basis function). The STFT is a complex function of a and Ω which indicates the ”strength” of the basis function $w(t-a)e^{j\Omega t}$ in the signal.

The STFT is an invertible transform. The exact inverse will be discussed in detail in the endterm report, however, this tells us that the STFT, like the Fourier transform, does indeed contain all the information that the signal does and is just an alternate view of the signal.

A few properties of the continuous STFT are:

1. Time shifting: If $X(\tau, \Omega)$ is the STFT of $x(t)$, then the STFT of $x(t - t_0)$ is $e^{-jt_0\Omega}X(\tau - t_0, \Omega)$
2. Frequency shifting: If $X(\tau, \Omega)$ is the STFT of $x(t)$, then the STFT of $x(t)e^{j\Omega t}$ is $X(\tau, \Omega - \Omega_0)$
3. Complex Conjugation: If $X(\tau, \Omega)$ is the STFT of $x(t)$, then the STFT of $x^*(t)$ (complex conjugate of $x(t)$) is, (assuming $w(t)$ is real), $X^*(\tau, -\Omega)$

The next section discusses some continuous time windows used in the STFT.

4 Windows

A window function is a mathematical function that is zero-valued outside of some chosen interval, normally symmetric around the middle of the interval. Its value is highest at the center of the interval, i.e $t = 0$, and tapers away towards the edges of the interval. Multiplying a time-domain signal by a window is essentially like viewing only a part of the signal that the windows permits, or in other words, 'viewing the signal through the window'
There are many functions that fit the definition of a window function, that can be used in the STFT.

1. **Rectangular Window**
   The rectangular window is the simplest window, which simply has a value 1 in a particular (symmetric) range. Multiplying by the rectangular window is equivalent to a plain truncation of the signal.
   The (zero-centered) rectangular window may be defined by:
   $$w_R(n)\begin{cases} 1, & -\tau \leq t \leq \tau \\ 0, & \text{otherwise} \end{cases}$$
   where 2τ is the window length. Its Fourier transform $W_r(\Omega) = 2\tau sinc(\Omega\tau)$ The main lobe of $X_r(\Omega)$ has a width $\frac{2\pi}{\tau}$, i.e first zero crossing at $\Omega = \frac{\pi}{\tau}$. This is the narrowest main lobe possible for any window which implies that the rectangular window provides the best frequency resolution of all windows, however, it has the disadvantage that its side lobes are quite large. The normalised first side lobe magnitude is -13dB, i.e only 13dB less than the peak magnitude. This leads to the issue of spectral leakage. In time-frequency analysis, we generally do not know what frequencies the signal we are analysing contains beforehand, and therefore rectangular windows are not used very often to avoid extensive spectral leakage which causes difficulty in interpreting the stft.
2. **Triangular window**
   The Triangular window of width 2τ may be defined by

$$w_{tri}(t)\begin{cases}1-|\frac{t}{\tau}|, & -\tau\le t\le\tau\\ 0, & \text{otherwise}\end{cases}$$

Its Fourier transform is given by $W_{tri}(\Omega) = \tau sinc^2(\frac{\Omega\tau}{2})$ The width of its main lobe is exactly double that of the rectangular window of same width, i.e $\frac{4\pi}{\tau}$ (Poorer frequency resolution). However, the normalized peak magnitude of the first side lobe is -26.5 dB, which is less the rectangular window of same length. Also, the rate at which side lobe peak magnitude decreases as Ω increases is higher in the triangular window (side lobe fall off rate $\frac{1}{\Omega^2}$ as opposed to $\frac{1}{\Omega}$).

3. **Hann and Hamming windows**

   - **Hann**

     The Hann window (or hanning or raised-cosine window) of length 2τ is defined as

     $$w_{hann}(t)\begin{cases}0.5+0.5cos(\frac{\pi t}{\tau}), & -\tau\le t\le\tau\\ 0, & \text{otherwise}\end{cases}$$

     The main lobe width is same as that of the triangular window, however, the normalized peak side lobe magnitude is -31dB, which is less than that of the triangular window of width 2τ. The rate of fall-off of side lobe levels is also higher (at $\frac{1}{\Omega^3}$)

   - **Hamming**

     The Hamming window of width 2τ is defined as

     $$w_{hann}(t)\begin{cases}0.54+0.46cos(\frac{\pi t}{\tau}), & -\tau\le t\le\tau\\ 0, & \text{otherwise}\end{cases}$$

     It was designed to be of the same form as the Hanning window, but with coefficients optimized for lowest possible first side lobe level. Due to this, the first side lobe's peak magnitude is -44dB (normalized), however this comes at the cost of slower side-lobe fall-off. The width of the main lobe is identical to that of a Hann window of same size.

4. **Gaussian window**

   The Gaussian function or ''bell curve' is given by $x(t) = \dfrac{1}{\sigma\sqrt{2\pi}}e^{-t^2/2\sigma^2}$

   Its Fourier transform, $W(\Omega)$, is of the same form, $W(\Omega) = e^{-\Omega^2/2(1/\sigma)^2}$

   This window has certain advantages when it comes to time-frequency resolution. It is in fact, the function with the best possible time-bandwidth product, $\Delta t\Delta f$, reaching the theoretical minimum. This will be discussed more in detail in a later section.

   The actual Gaussian function described above, is an everlasting signal, and therefore to use it as a window, it needs to be truncated. In the original paper by Denis Gabor where he first described the STFT, he made use of the Gaussian window, and therefore the STFT using a Gaussian window is sometimes called the Gabor transform.

5 Discretizing the STFT: Spectrograms

The STFT for continuous time signals, as described above is purely a mathematical construct and cannot be implemented on a computer. The concept of the STFT can, however be extended to discrete-time signals as well. Just as the Discrete-time Fourier transform (DTFT) is the analogue of the Continuous-time Fourier transform (CTFT), for discrete signals, the STFT for a discrete-time signal $x(n)$ (not a finite signal, in general), is given as:

$$X(m, \omega) = \sum_{n=-\infty}^{\infty} x(n)w(n-m)e^{-j\omega n} \tag{5.1}$$

Here, $w(n)$ is a discrete window that is used in the continuous STFT, and it is also a finite length sequence, i.e $w(n)$ is nonzero only in the range $n \in \{0, 1, ...L-1\}$ where L is the window length. The frequency variable ω is continuous. The logic behind this version of the STFT is similar, we select a portion of the discrete-time sequence $x(n)$ using the shifted window $w(n-m)$ and then apply the normal DTFT analysis equation to this windowed portion of the function. In this case, $X(m, \omega)$, for any particular value of m, is the DTFT of the sequence $x(n)w(n-m)$, because of which it is periodic with a period 2π (rad/sample), in ω. The discrete-time STFT can also be written as:

$X(m, \omega) = \sum_{n=-\infty}^{\infty} x(n+m)w(n)e^{-j\omega(n+m)}$

$= e^{-j\omega m} \sum_{n=-\infty}^{\infty} x(n+m)w(n)e^{-j\omega n}$

This is clearly the same as finding the DTFT of $x(n+m)w(n)$ and multiplying it by the phase term $e^{-j\omega m}$. This view of the discrete-time STFT is based on keeping the window fixed and instead translating the signal so that different parts of the signal fall in the range of the window for different values of m.

Discrete-Time Window Functions

We now briefly look at some of the important discrete-time window functions and their properties. All of these functions are finite with length L, and with the first nonzero sample at $n = 0$.

The discrete-time windows used for windowing a data sequence into frames, are all nonzero for $n \in \{0, 1, ...L-1\}$. In essence, they are sampled and shifted versions of the continuous time windows. The properties of the windows described below are all based on the magnitude of the window's Fourier transform (DTFT) $W(\omega)$, alone, and therefore an additional phase term due to time shift does not affect them and so the properties described below for an origin-centered window also hold for the corresponding causal window.

Window name	Main lobe width	Peak side lobe level	Side lobe roll off rate
Rectangular	$\frac{4\pi}{L}$	-13 dB	-6 dB/oct
Triangular	$\frac{8\pi}{L}$	-26.5 dB	-12 dB/oct
Hanning	$\frac{8\pi}{L}$	-31 dB	-18 dB/oct
Hamming	$\frac{8\pi}{L}$	-41 dB	-6 dB/oct

n the above table, the main lobe width is in terms of rad/sample frequency, the peak side lobe level is the normalized dB height of the largest side lobe and the side lobe roll-off rate is in dB/octave.

The below figure shows plots of a few important time-domain windows. These plots were generated using a finite number of samples, and hence, represent discrete-time windows. Each window is of length $L = 100$.

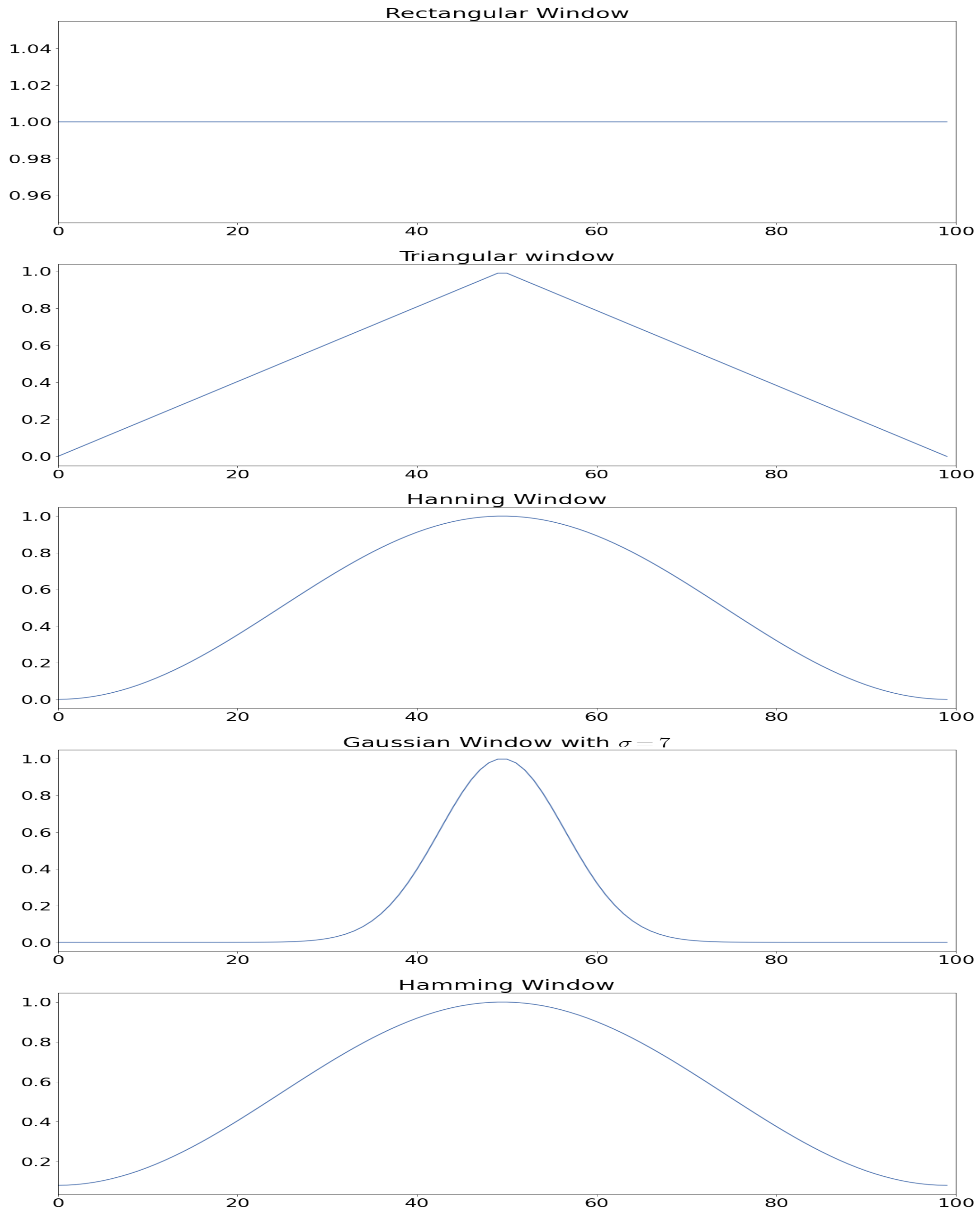

Figure 5: Some time-domain windows(discrete)

The Spectrogram

The discrete-time STFT described above, is still not suitable for computation, as it involves a continuous frequency variable ω. The discrete-time STFT can only be evaluated at a fixed number of sample points. The ω values chosen at which the above expression is evaluated, must be of the form $\omega_k = \frac{2\pi k}{N}, k \in \{0, 1, ..., N-1\}$. This is precisely the kind of frequency domain sampling that is performed to evaluate the DFT of a sequence. In the above expression, if we replace ω by ω_k, we get

$$X(m,k) = \sum_{n=-\infty}^{\infty} x(n)w(n-m)e^{-j\frac{2\pi k}{N}n} \text{ for } k \in \{0, 1, ...N-1\} \tag{5.2}$$

Of course, this introduces the risk of time-domain aliasing, however, the signal $x(n)w(n-m)$ is guaranteed to be timelimited, as $w(n)$ is finite, therefore, if we can ensure that the number of DFT points is at least equal to window length, the above expression will provide accurate results. Also, the time-domain periodicity introduced by frequency sampling allows us to convert the infinite sum into a finite sum. It is more convenient to use the second form of the discrete-time STFT for this, which is given by $X(m,\omega) = e^{-j\omega m}\sum_{n=-\infty}^{\infty} x(n+m)w(n)e^{-j\omega n}$ replacing ω by ω_k, gives

$$X(m,k) = e^{-j\frac{2\pi k}{N}m} \sum_{n=-\infty}^{\infty} x(n+m)w(n)e^{-j\frac{2\pi k}{N}n} \tag{5.3}$$

Comparing this with the DFT, and recognising that $x(n+m)w(n)$ is a finite-length sequence (with a length L, i.e the window length), we can replace this infinite sum by a finite sum, provided we can guarantee the number of DFT points is greater than the length of this finite sequence. Generally, the number of DFT points, N is taken as equal to L, the window length, although in some cases, zero padding is performed on each frame and N is taken as greater than L. This will be discussed more in the endterm report, when we examine reconstruction from the STFT.

Now, we also generally evaluate the above sum only for certain values of m. In theory, the signal $x(n)$ can be everlasting, however, in practice, all signals are finite in length. This means we only have to consider a finite number of values of the shift m. We also do not consider every possible value of m, but maintain a gap of some samples between 2 consecutive values of m for which we evaluate the sum. This gap is called the 'Hop' length, H, and m can be written as rH, where r is the frame number. Taking all these points into account, the above expression is rewritten as:

$$X(r,k) = e^{-j\frac{2\pi k}{N}rH} \sum_{n=0}^{N-1} x(n+rH)w(n)e^{-j\frac{2\pi k}{N}n} \tag{5.4}$$

generally, the term $e^{-j\frac{2\pi k}{N}rH}$ is ignored, since we are mainly interested in the magnitude of the STFT, and therefore, for practical computations, we can use the below form of the discrete STFT:

$$X(r,k) = \sum_{n=0}^{N-1} x(n+rH)w(n)e^{-j\frac{2\pi k}{N}n} \tag{5.5}$$

In this equation, H is the hop length (essentially window length minus the number of overlapping samples between consecutive windows), r is the frame number, N is the number of DFT points, generally equal to window length, L. This is clearly equivalent to computing the

N-point DFT of the r^{th} windowed frame of $x(n)$, i.e for the r^{th} frame, we begin from $n = rH$, take L samples of $x(n)$, multiply this elementwise with the chosen length-L window, and then compute the DFT of this length-L signal. This is exactly the approach that has been adopted in the implementation and results section of this project.
Since we are focused only on the magnitude of the STFT, we often, instead of $|X(r,k)|$, compute, $|X(r,k)|^2$, and generate either a 3-d plot of $|X(r,k)|^2$, plotted with r and k as the independent variables, or we could also use a suitable 2-d representation, like a heat map. This is essentially the spectrogram of the signal.

6 Limitations and the Uncertainty Principle

A fundamental problem associated with all time-frequency representations of a signal, is the time-frequency uncertainty, which prevents us from achieving high time and frequency resolution simultaneously. For this discussion, we return to the theoretical continuous-time STFT. Before stating the principle, we must define what is meant by resolution. Roughly speaking, resolution in time is the ability to differentiate between 2 pulses that are certain distance apart, or the minimum distance between pulses for which such discrimination is possible with the given window function. Frequency resolution similarly describes the minimum frequency between 2 sinusoids that allows to to distinguish between them, with the given window function. In other words, time resolution measures the window's ability to separate 2 pulses in the time domain and frequency resolution measures the ability of the chosen window function's Fourier transform to separate 2 sinusoidal components in the frequency domain. The formal definitions are based on a statistical interpretation of the signal:
Uncertainty in time,

$$\Delta t = \frac{\int_{t=-\infty}^{\infty} t^2 |w(t)|^2 \, dt}{\int_{t=-\infty}^{\infty} |w(t)|^2 \, dt} \tag{6.1}$$

And if the Fourier transform of $w(t)$ in terms of cyclic frequency is, is $W(f)$, then the uncertainty in frequency,

$$\Delta f = \frac{\int_{f=-\infty}^{\infty} f^2 |W(f)|^2 \, df}{\int_{f=-\infty}^{\infty} |W(f)|^2 \, df} \tag{6.2}$$

Ideally, we would like to make both Δt and Δf as small as possible so that our STFT with the window $w(t)$ gives us arbitrarily good time resolution and frequency resolution. However, a fundamental limit is imposed on the resolutions, by the Uncertainty principle, which says that

$$\Delta t \Delta f \geq \frac{1}{4\pi} \tag{6.3}$$

This means, even by choosing the best possible window, the best possible product of time and frequency resolution we can get is $\frac{1}{4\pi}$. If we attempt to increase the time resolution, by making the window narrower, the frequency resolution will decrease, and vice versa. Another way of looking at this is that increasing the length of a window leads to a reduction in the size of its main lobe and vice versa. The width of the main lobe defines the ability to differentiate sharply between 2 nearby frequencies, and the width of the window in the time domain defines the ability to differentiate between 2 pulses in the time domain. A narrower window means better time resolution, but a poorer frequency resolution due to widening of the main lobe, and a wider window leads to narrowing of the main lobe, i.e better frequency resolution, but a poorer time resolution. A demonstration of this tradeoff is provided in the results section.

Another aspect of the STFT when it comes to resolution, is that its resolution is fixed, which the fundamental limitation of the STFT. The time and frequency resolution of the STFT are fixed, because they are both based on the nature and width of the window function. The window function we use remains the same irrespective of the frequency in the STFT. Howwever, there are cases when we would like to have a better frequency resolution (at the cost of poorer time resolution) at some frequencies, and a better time resolution(at the cost of poorer frequency resolution) at some other frequencies, however, the STFT does not give us such flexibility as it does not enable us to vary the width of the window function as per our requirements. Normally, we would like to use narrower windows at higher frequencies, because higher frequencies lead to more rapid/sudden changes in time, and we would like to capture those changes more accurately. At lower frequencies, we would like to have a better frequency resolution, and we do not mind sacrificing some time resolution to achieve this.
Additionally, spectrograms plot the frequencies on a linear scale, but most musical scales are in fact logarithmic. For example, going to the next highest similar note, starting from a particular note, is called traversing an octave, which implies that the higher note has twice the frequency of the lower note, and despite the nonlinear relation, the listener's perception of these two notes is similar. This is exactly what the wavelet transform attempts to achieve, by using shifted as well as scaled versions of a mother wavelet as the basis functions. The scalogram, derived from the wavelet transform, then attempts to plot the frequencies on a logarithmic scale to improve the picture.

7 Testing and Results

The above concepts have been implemented and tested on python. Though scipy contains an inbuilt function to generate the spectrogram for a signal, all spectrogram plots in this section have been obtained using a user-defined spectrogram-plotting code that was written from scratch. This code permits the usage of any window function available in scipy.signal.windows, and also allows the user to set the number of points for the DFT evaluation in the spectrogram (i.e window length) and the number of samples of overlap between consecutive windows. First, a simple digitally-generated music signal with the classic ”Do-re-mi-fa” progression of notes was tested. The simple FFT magnitude spectrum plot of this signal is shown below:(magnitude has been normalized)

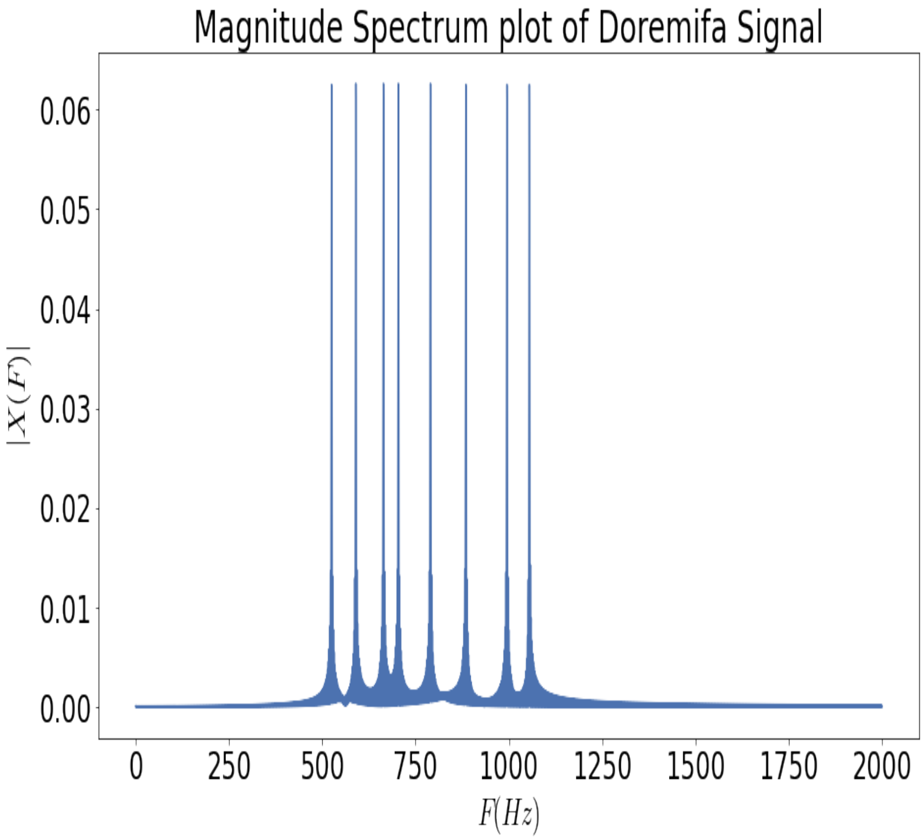

Figure 6: FFt magnitude spectrum of Doremifa signal

It is seen that a simple FFT analysis, followed by peak finding would tell us exactly which frequencies are contained in the signal, but nothing about when in time they occur. On the other, hand, if we look at the spectrogram plot, shown below

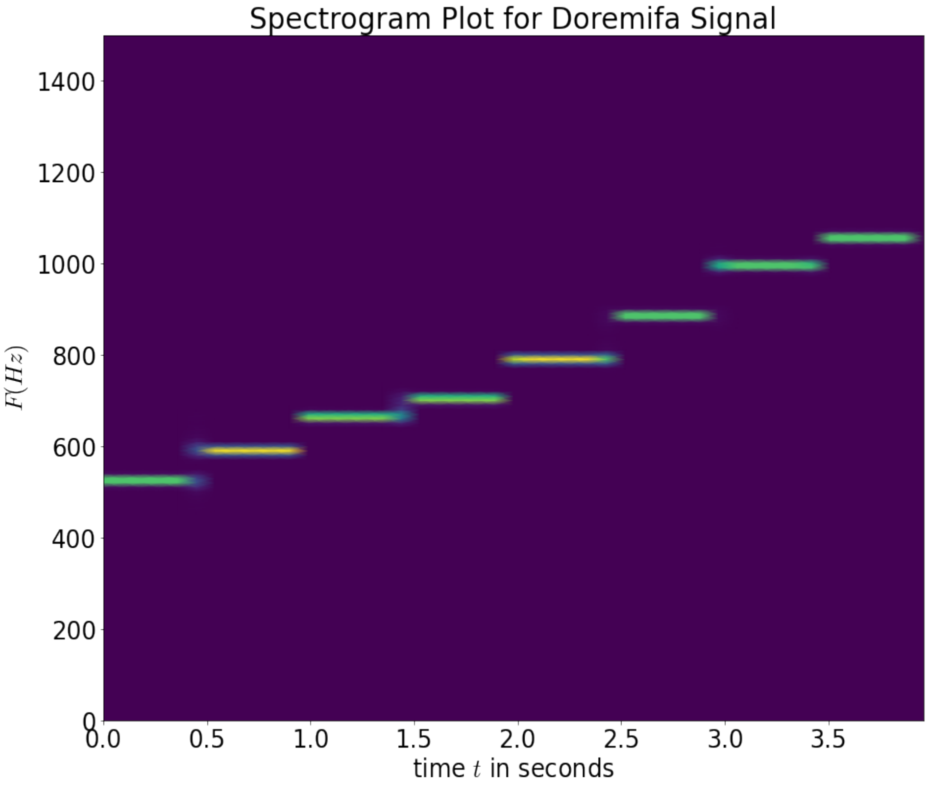

Figure 7: Spectrogram for Doremifa signal

we can make out clearly, the progression of notes from low to high as time progresses. In all the spectrogram plots, similar to the one shown above, the x axis quantity, rather than the frame number, r, has been renormalized to time in seconds., in the expression for the discrete STFT, and the Y axis has been renormalized to the continuous-time frequency $F(Hz)$. The code generates the spectra for $k \in \{0, 1, ...N-1\}$, where N is the number of DFT points (taken as window length itself, in this case), however, by controlling the plotting limits, we can leave out half the spectrum, as each windowed time-domain signal is still real and since we are interested only in magnitude, the second half of the spectrum will be exactly identical to the first half. The quantity indicated by the color intensity is in fact, the squared magnitude of the discrete-STFT value returned for that combination of k and m.

Before proceeding, the above digitally generated signal will provide a good test case to observe the time-frequency uncertainty principle. The spectrogram of the same signal is plotted, but with a larger window size. We expect frequency reolution to improve and time resolution to worsen. The plot obtained for $L = 2000$ samples, rather than the previous 1000 samples, and the same number of overlapping samples, is shown below:

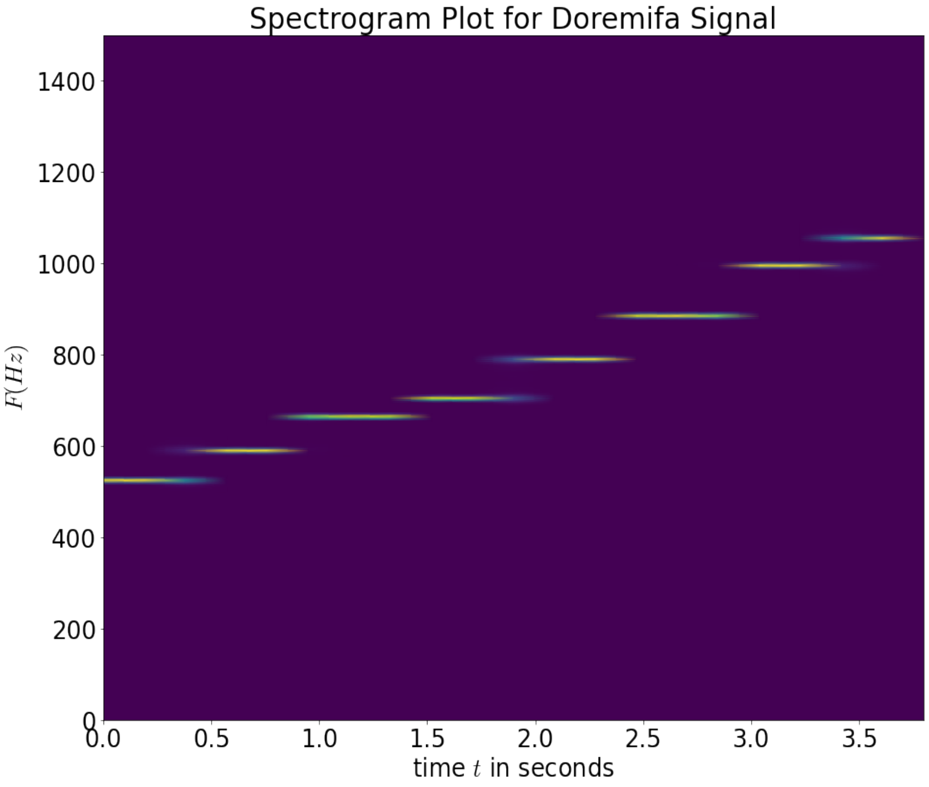

Figure 8: Spectrogram for Doremifa signal with larger window

The plot shows us clearly that, while this spectrogram does a much better job of pinpointing the frequencies of each note, its ability to indicate sudden changes in time is poor, as it has a poorer ability to pinpoint when in time the change between 2 frequencies takes place. Similarly, if we make the window narrower(say $L = 400$ samples, with the same number of overlapping samples), we obtain the following plot:

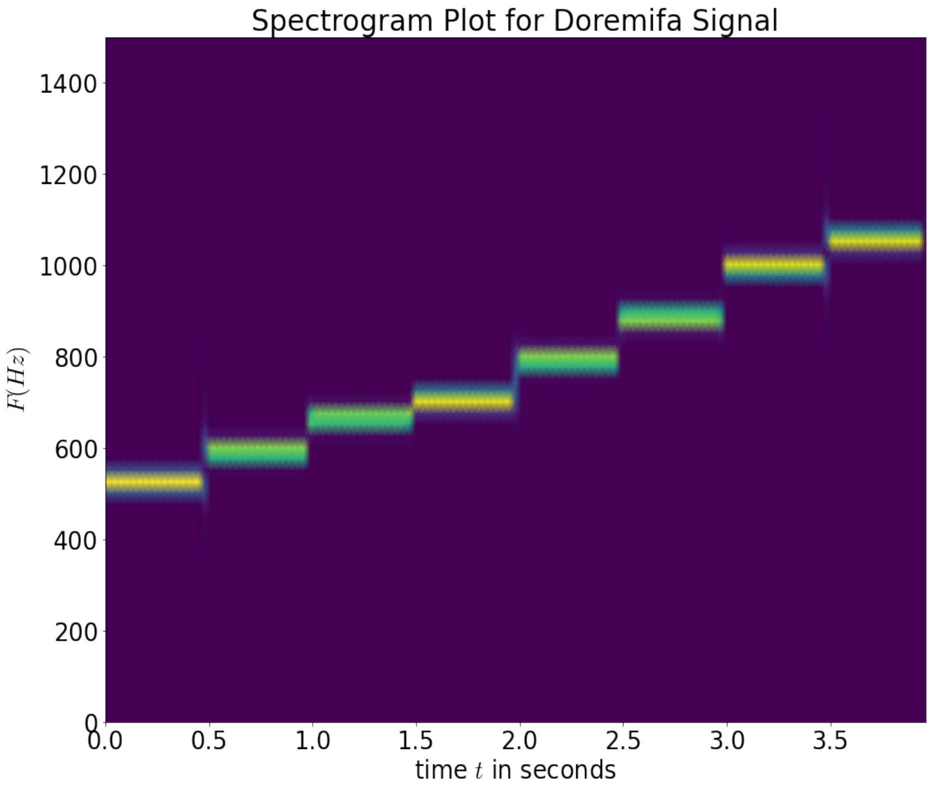

Figure 9: Spectrogram for Doremifa signal with narrower window

In this plot we can see clearly that while this spectrogram is able to indicate with more precision when in time a change between 2 frequencies occurs, the frequency band for each note has increased in size, implying reduced precision in indicating the frequency of a component. Next we apply the spectrogram technique to some real-world musical signals.

We show the spectrograms for arpeggios [6] played on piano, flute and guitar. Figure 10 illustrates the magnitude spectrums of the three signals as a whole and Figures 11 12 13 are the spectrograms of the piano, flute and guitar signals respectively. The vertical scale on these figures is a frequency scale(in Hz) and the horizontal scale is a time scale(in sec). The larger values of these spectra are displayed in darker shades and the plain regions correspond to frequency values that are near zero in magnitude. Thus it provides a description of the sound signals in the *time-frequency plane* A comprehensive investigation into the frequency components will be presented along with scalograms in the latter part of the project after mid-term. For now we investigate the groups of line segments for the individual notes dividing the spectra corresponding to the fundamentals and overtones in each note.

It is interesting to compare these spectra on the basis of ”attack” and ”decay” of the spectral line segments for the notes played by different instruments. For guitar there is a clear separation of the spectra for individual notes and each note slightly tapers as time progresses.

This is expected because the strings are plucked with a plectrum [1] and has a very prominent attack. We can expect the same for piano due to the striking of the piano hammer on its strings, however, foot pedals given to the player control this intensity and we observe less aggressive attack in the spectrogram. The flute signals rather flow into each other with similar width indicating gentler attack. There is also a longer decay for the piano notes due to slow damping down of the piano string vibrations. This is evident in the overlapping of the time intervals underlying individual note segments. For guitar there is least overlapping and quick decay for individual notes in comparison. For the flute where notes arise from a standing wave within the flute created by a gentle breath of the player, decay in tandem with the player's breath as the standing wave collapses and hence do not undergo significant overlapping.

This comparison illustrates how significant features of musical notes can quantitatively be captured in the time-frequency portraits provided by spectrograms. It is well known that in addition to the harmonic structure of fundamentals and overtones, precise features as attack and decay in notes are important factors in the perception of musical quality by humans.

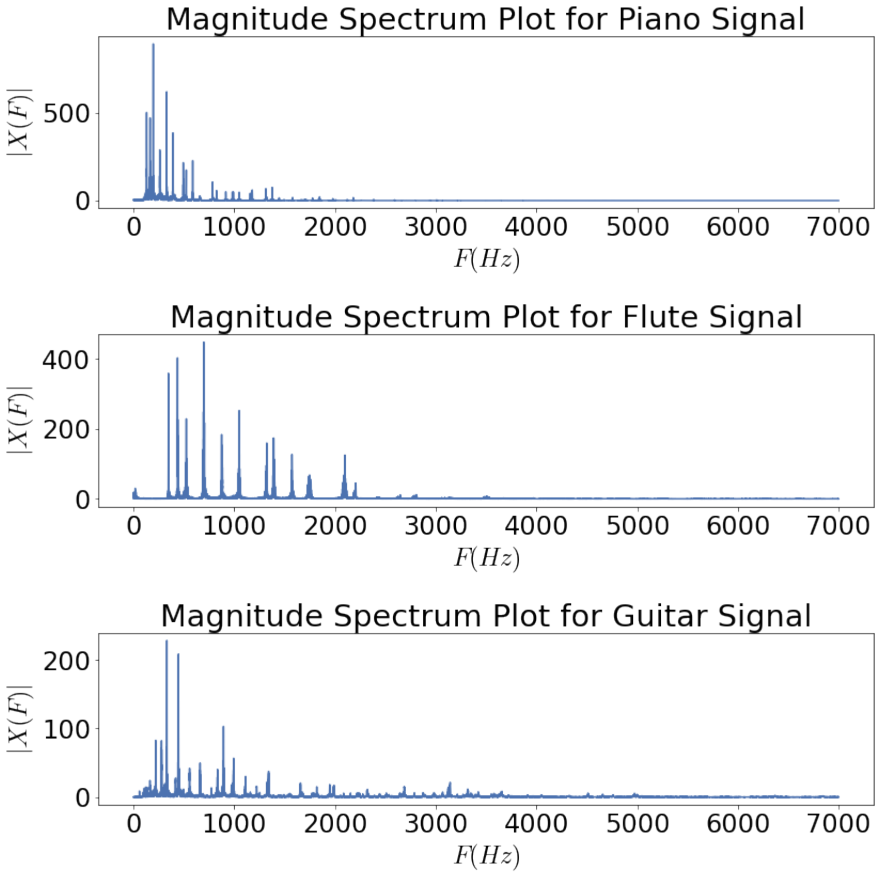

Figure 10: FFT of piano, flute and guitar signals

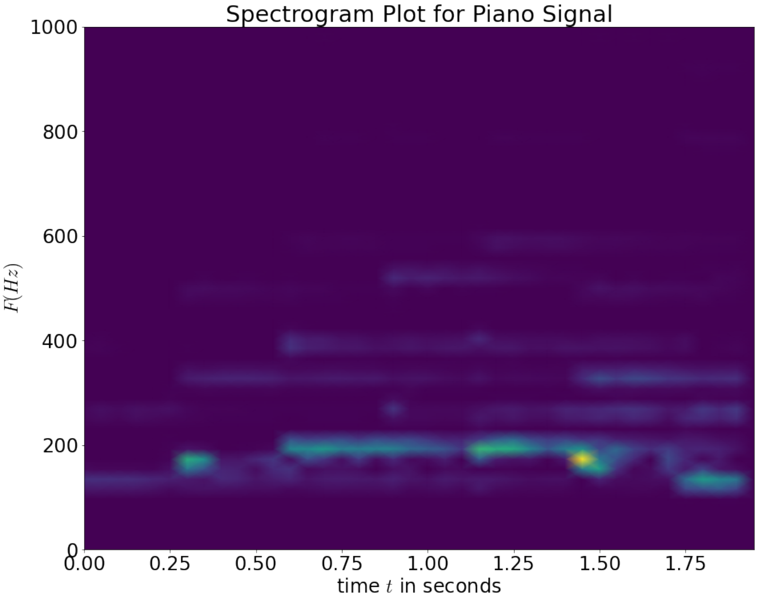

Figure 11: Spectrogram of Piano

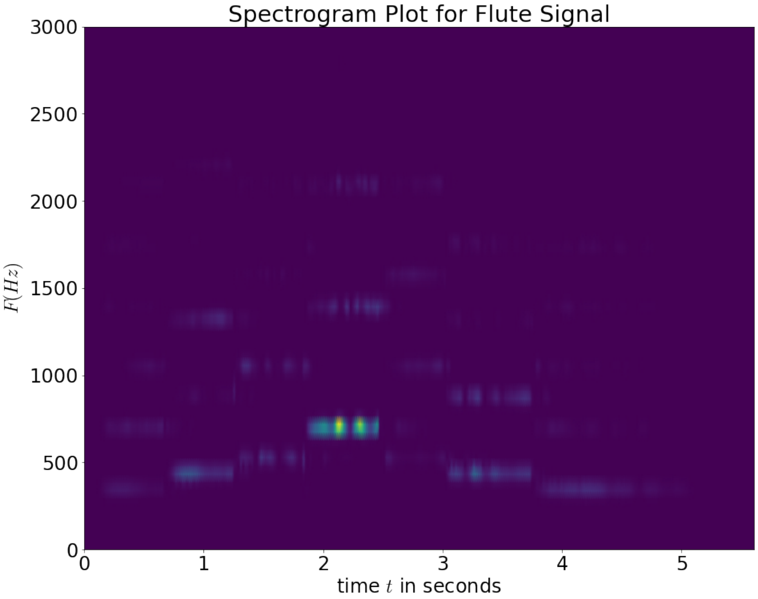

Figure 12: Spectrogram of Flute

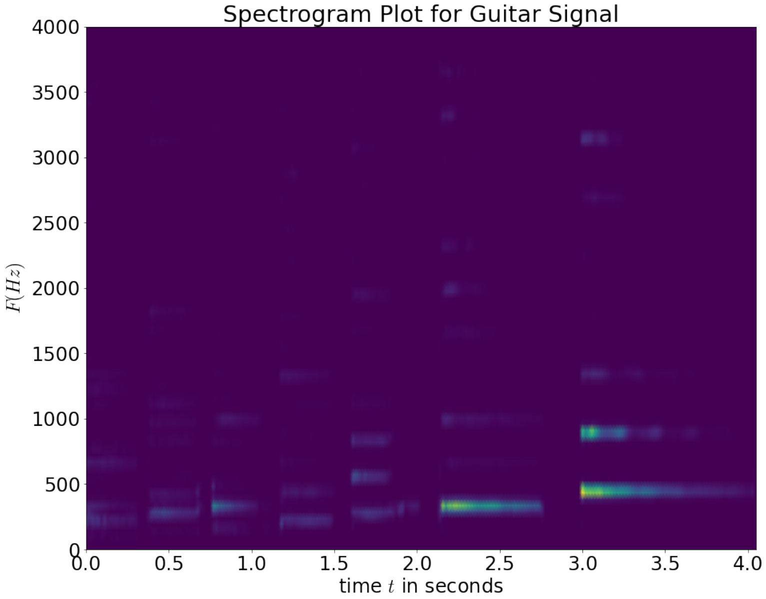

Figure 13: Spectrogram of Guitar

8 Summary and Plan for the coming month

In this report, we have demonstrated how conventional Fourier analysis fails to provide a meaningful picture of the time-frequency contents of a non-stationary signal. We looked at how music signals, being composed of different sets of frequencies occurring in a certain order, are non-stationary signals and hence require time-frequency analysis techniques to properly characterize them. We then looked at one of the simplest time-frequency analysis techniques: the Short-Time Fourier Transform, in detail, and how it can be used to generate a spectrogram plot that gives us a meaningful description of the signal. The uncertainty rule, a fundamental limit imposed on time-frequency analysis techniques, was looked into and then the shortcomings of the STFT/Spectrogram technique were discussed. Finally, the spectrogram was implemented practically and tested out on real musical signals to obtain their time-frequency pictures as well as additional information about the characteristics of the instrument that generated the signal.

Going forward, we would like to first briefly describe the conditions and processes involved in spectrogram inversion: the inverse problem of returning to the original time-domain signal, given its spectrogram. After this, we plan to look into the wavelet transform and its application in time-frequency analysis, how it overcomes the pitfalls of the simple STFT/Spectrogram technique, and how it can be used to generate a scalogram: a much more meaningful time-frequency representation of a musical signal. Wherever possible, we plan to implement the above concepts in python. Any additional relevant topics found will also be addressed in the post-midterm report.

Post-Midterm Report

9 Inversion of Spectrograms

We demonstrated above that a spectrogram provides a useful method of analysis, i.e., splitting of the signal into parts and processing these parts, the parts being the sub-signals $\{x(n)w(n-m)\}$ that are processed via FFTs. Complementing this analysis, there is also synthesis, whereby we reconstruct the signal from its constituent parts. This synthesis is an inversion process for spectrograms: recovering the original sound signal $x(n)$ from its spectrogram of FFTs $\{X(m,k)\}$, $m = 1, 2, ..., M$. One important application of this inversion process is to the compression of recorded music and speech. In order to perform inversion, it is sufficient that the succession of windows $\{w(n-m)\}$ satisfies

$$A \leq \sum_{m=1}^{M} w(n-m) \leq B \tag{9.1}$$

for some positive constants A and B. The inequalities in (9.1) are a generalization of

$$\sum_{m=1}^{M} w(n-m) = 1 \tag{9.2}$$

In practice, it has been found that some windows w (such as Hamming or Hanning windows), which satisfy (9.1), perform better for frequency identification than windows that are required to satisfy the more stringent condition (9.2).

Assuming that (9.1) holds, the largest possible constant A and the smallest possible constant B are called the frame bounds for the window w. Assuming A and B are frame bounds, we now prove that (9.1) suffices for inversion. Provided B is not too large, performing the spectrogram analysis

$$\{x(n)\} \mapsto \{x(n)w(n-m)\}_{m=1}^{M} \mapsto \{X(m,k)\} \tag{9.3}$$

will be numerically stable. Furthermore, performing inverse FFTs on each of the subsignal FFTs in (9.3) and adding the results together produces

$$\sum_{m=1}^{M} x(n)w(n-m) = x(n)\sum_{m=1}^{M} w(n-m) \tag{9.4}$$

Because of the inequality $A \leq \sum w(n-m)$, we have $1/\sum w(n-m) \leq A^{-1}$. Therefore, we can divide out the factor $\sum w(n-m)$ in (9.4) to obtain the discrete values $\{x(n)\}$. The constant

A^{-1}, provided it is not too large, ensures the numerical stability of this division. Thus, (9.1) implies that spectrograms obtained from discrete values of a sound signal are invertible. When an analog sound signal $x(t)$ is band-limited, then this signal $x(t)$ can be recovered from the discrete values $\{x(n)\}$ as well. This last step is often unnecessary for digitally recorded sound because the only data available are the discrete values $\{x(n)\}$. Moreover, if the reconstructed values are rounded to integers—because the original data in the computer sound file were integers—then the error between the original sound data and the reconstructed values will be zero. One application of spectrogram inversion is to the compression of audio signals. After discarding (setting to zero) all the values in a spectrogram with magnitudes below a threshold value, the inversion procedure creates an approximation of the original signal that uses significantly less data than the original signal. That is because the thresholded spectrogram can be greatly compressed by removing the large amount of redundancy created by all of the zero values arising from thresholding. In figures 12, 13 and 11 for example, all of the plain regions shown in the spectrograms stand for zero values and that implies considerable redundancy for those spectrograms. Some of the best results in audio compression are based on sophisticated generalizations of this spectrogram technique; these techniques are called *lapped orthogonal transform coding* or *local cosine coding* [4]. Such audio compression techniques underlie the real-time audio players and audio download sites available on the Internet.

10 Wavelets and the Continuous Wavelet Transform

We have seen how the STFT as a time-frequency analysis tool has certain limitations. The main issue is its fixed resolution. Since we use the same window function, with the same fixed length throughout the analysis (for all values of τ and Ω), we do not have the freedom to vary the resolution with frequency or time, it is a characteristic of the window chosen. We often wish to have a higher frequency resolution at lower frequencies, and a higher time resolution for higher frequencies, as explained in section 6. This concept is referred to as multiresolution analysis, as it provides the ability to vary the resolution as the frequency of the analysing function varies. A tool that naturally achieves multiresolution time-frequency analysis is the wavelet transform.

Before defining the wavelet transform, we shall examine the mathematical framework for signal transforms in brief. In conventional Fourier analysis, we attempt to decompose a signal into a linear combination of a particular set of functions, in this case, the family of functions of the form $e^{j\Omega t}, \Omega \in \mathbb{R}$. However, since Ω varies continuously, the linear combination is replaced by an integral, which is the CTFT analysis equation. This can be viewed in a different way. If we consider the signals to be vectors, we can define an inner product between 2 signals. The conventional inner product between 2 signals, $f(t)$ and $g(t)$, is defined as:

$$\langle f(t), g(t) \rangle = \int_{-\infty}^{\infty} f(t) g^*(t)\, dt \tag{10.1}$$

where $*$ represents complex conjugation. If we look at the standard Fourier transform analysis equation, $F(\Omega) = \int_{-\infty}^{\infty} f(t) e^{-j\Omega t}\, dt$, we can see that it is nothing but $\langle f(t), e^{j\Omega t}\rangle$. It must be noted that the inner product used above also induces a corresponding norm. A norm is a function that assigns a positive-valued scalar representing the 'length' or 'size' of the vector to every vector in the space under consideration. In this case, the norm induced by the inner product defined as above, is $\sqrt{\langle f(t), f(t)\rangle}$. The norm of a signal $f(t)$ is represented as $||f(t)||$, and with this inner product, $||f(t)|| = \sqrt{\int_{-\infty}^{\infty} f(t) f^*(t)\, dt} = \sqrt{\int_{-\infty}^{\infty} |f(t)|^2\, dt}$. This is clearly nothing but the square root of the signal energy, using the conventional definition of energy $E = \int_{-\infty}^{\infty} |f(t)|^2\, dt$.
To measure the 'strength' of different frequency components of the form $e^{j\Omega t}$ in the signal $f(t)$, we must first assume that $f(t)$ can be written in terms of the continuum of complex exponentials, as $\frac{1}{2\pi}\int_{-\infty}^{\infty} F(\Omega) e^{j\Omega t}\, d\Omega$ (the $\frac{1}{2\pi}$ will help simplify the expression later) . Now, let us take the inner product of $f(t)$ with the complex exponential $e^{j\Omega_o t}$, i.e a complex exponential of particular frequency. We get $\langle f(t), e^{j\Omega_o t}\rangle = \frac{1}{2\pi}\int_{-\infty}^{\infty}\int_{-\infty}^{\infty} F(\Omega) e^{jt(\Omega-\Omega_o)}\, d\Omega\, dt$. The integrals can be interchanged to get $\frac{1}{2\pi}\int_{-\infty}^{\infty} F(\Omega)\int_{-\infty}^{\infty} e^{jt(\Omega-\Omega_o)}\, dt\, d\Omega$. For the inner integral, in fact $\int_{-\infty}^{\infty} e^{jt(\Omega-\Omega_o)}\, dt = 2\pi\delta(\Omega-\Omega_o)$. (Source: item 8 in additional resources section). Thus, $\langle f(t), e^{j\Omega_o t}\rangle = \frac{1}{2\pi}\int_{-\infty}^{\infty} F(\Omega) 2\pi\delta(\Omega-\Omega_o)\, d\Omega = F(\Omega_o)$. Thus finally we see that taking the inner product between the function being analysed ($f(t)$) and the analysing function of a particular frequency $e^{j\Omega_o t}$, we can in fact find the 'strength' of that component in the signal $f(t)$, which is nothing but $F(\Omega_o)$.

To understand the wavelet transform, next we define the 'distance' between 2 signals using the standard inner product, $d_{fg} = ||f(t) - g(t)||$. The distance between 2 signals, in a way, measures their closeness. If 2 signals are not very similar, the distance d_{fg} is small, and vice versa. Now, suppose the simple complex exponentials in the Fourier transform are replaced by a certain family of functions, called wavelets (which will be described shortly), and suppose that each member of this family of wavelets has 2 parameters, a and b, and is represented as $\psi_{a,b}(t)$. The, the continuous wavelet transform of $f(t)$ is essentially the inner product $\langle f(t), \psi_{a,b}(t)\rangle$, with an extra scaling factor. Before the exact equation is given, this can be interpreted intuitively based on inner products and distances. As we saw in the case of the Fourier Transform, taking the inner product with a particular wavelet $\psi_{a,b}(t)$, measures the 'strength' of that particular wavelet in the signal $f(t)$. Alternatively, in terms of the distance, the distance $d_{f,\psi_{a,b}} = ||f(t) - \psi_{a,b}||$, can be written (using the linearity of the inner product), as: $||f(t)||^2 + ||\psi_{a,b}(t)||^s - 2Re\{\langle f(t), \psi_{a,b}(t)\rangle\}$, where the last term is clearly twice the real part of the cwt of $f(t)$ (scaled). Thus, the cwt, in a sense, measures the distance between $f(t)$ and $\psi_{a,b}(t)$. The higher its magnitude is, the smaller the 'distance' between the signal being analysed and the wavelet $\psi_{a,b}(t)$ is, and vice versa. In other words, the cwt measures how close the signal is to a particular wavelet. We shall see that each wavelet possesses time localisation as well as a particular frequency of oscillations, and thus the cwt, by 'comparing' the signal with different wavelets, is able to indicate the 'strength' of different frequencies in the signal in the neighborhoods of particular time instants, and thus is acts as a time frequency representation. The exact form of the cwt is as follows:

$$F(a,b) = \frac{1}{\sqrt{a}} \int_{-\infty}^{\infty} f(t)\psi_{a,b}^{*}(t)\,dt \tag{10.2}$$

And the function $\psi_{a,b}(t)$ is derived from a single function $\psi(t)$, the so-called mother wavelet. $\psi_{a,b}(t)$ is essentially a shifted and scaled version of the mother wavelet, where b and a are the shifts and scale in time, respectively, or

$$\psi_{a,b}(t) = \psi(\frac{t-b}{a}) \tag{10.3}$$

Therefore, $\psi_{a,b}(t)$ contains the set of all wavelets derived by applying shifting by any real value, and scaling by generally any positive real value, on the mother wavelet. These wavelets are thus called the child wavelets. In the expression for the CWT, we see the scaling term $\frac{1}{\sqrt{a}}$, which is present to ensure all the child wavelets used have the same L_2 norm, or energy as the mother wavelet. To derive this, the energy of the child wavelet is $\int_{t=-\infty}^{\infty} |\psi(\frac{t-b}{a})|^2\,dt = \int_{t1=-\infty}^{\infty} |\psi(t_1 - \frac{b}{a})|^2\,adt_1$ with the substitution $t = at_1$. It can be seen that the final expression is nothing but a times the energy of a shifted version of the mother wavelet, or a times the energy of the mother wavelet itself, thus to equalize the energies of the child and mother wavelets, the child wavelet's magnitude must be divided by $\sqrt{a}$. We can absorb the $\frac{1}{\sqrt{a}}$ into the expression for the child wavelet and say $\psi_{a,b}(t) = \frac{1}{\sqrt{a}}\psi(\frac{t-b}{a})$. Thus the cwt of $f(t)$ is nothing but the inner product of $f(t)$ with this modified $\psi_{a,b}(t)$.

We must next examine what features a wavelet must possess. An exponential of the form $e^{j\Omega t}$ could be called a wave, or at least, its real and imaginary parts appear as waves: they are everlasting sinusoids that oscillate at a certain frequency, Their magnitude remains the same over the entire range on which they are defined, i.e, for all $t \in \mathbb{R}$. Because of this, they provide no time localization, which is precisely why using them as analysing functions (i.e, performing a Fourier transform), gives us a representation of the signal that lacks any time information. Wavelets, however, possess a time localization and a frequency associated with them. In this project, we make use of a certain class of wavelets, called Gabor, or Morlet wavelets. For these wavelets, only within a certain time range is the amplitude of the wave appreciably high. For an origin-centered wavelet, the amplitude function that multiplies the sinusoid or complex exponential will be maximum at $t = 0$ and will fall to 0 as we move away from the origin on either side. By providing a shift (b) in this wavelet, we shift it over to some other point $t = b$, in the neighborhood of which the wavelet has an appreciable magnitude. Additionally, we also scale the wavelets by the factor a, and this affects both the oscillation frequency of the wavelet as well as the effective width of the wavelet (can be expressed with standard deviation/variance), which is nothing but the effectiveness of the time localization. As the value of a increases, the oscillation frequency decreases, i.e we are trying to analyse lower frequency components, and at the same time, the wavelet tends to get stretched out or elongated, which in effect means a reduced time resolution and improved frequency resolution. On the other hand a lower value of scale means the wavelet's oscillation frequency is higher and the wavelet is narrower/more compressed. Thus the scaling operation naturally causes the frequency resolution to improve as the frequency of the wavelet being used for analysis decreases, or, as we vary the frequency of the wavelet using the scaling parameter in order to analyse different frequencies in the signal, the scaling naturally adjusts the resolution so that a better frequency resolution is obtained at lower frequencies and vice versa, which is exactly what was desired. This is not dependent on the time shift parameter b, and thus for any time b, we have the ability to analyse multiple frequencies with varying resolution, each frequency corresponding to a particular scale value. The mathematical expression for the mother Gabor wavelet is given by:

$$\psi(t) = \frac{1}{w} e^{-\pi \frac{t}{w}^2} e^{j2\pi\eta \frac{t}{w}} \tag{10.4}$$

We can clearly see that the mother wavelet has an oscillatory component $e^{j2\pi\eta\frac{t}{w}}$, which is modulated by a Gaussian function $e^{-\pi\frac{t}{w}^2}$. The complex exponential is a pure phase term, so if we plot the magnitude of the mother wavelet, it will be purely a Gaussian function, which forms the envelope for the oscillation. The real and imaginary parts of the mother wavelet are pure cosine and sine terms, respectively, which are modulated by the Gaussian envelope, and if we plot the real or imaginary part of the mother wavelet alone against time, we will be able to observe the oscillation as well. In this case, the Gaussian function is providing the time localization, as its amplitude falls off as we move away from the center, and the complex exponential is providing the frequency characterization. Here, the mother wavelet has 2 parameters, w and η, which we can set before beginning the analysis. w is called the width parameter, and it is a measure of how dilated or compressed the mother wavelet is. From the above expression, we can clearly see that a larger value of w implies a larger standard deviation for the Gaussian term, and thus a more spread-out wavelet. This must not be confused with the scaling operation, which is done subsequently on the mother wavelet to obtain either dilated or compressed versions of the mother wavelet. The parameter η, along with w, sets the frequency of oscillation of the mother wavelet, because clearly from the expression above, the oscillation frequency in Hz is $\frac{\eta}{w}$. When we perform shifting and scaling on the mother wavelet to obtain the child wavelet $\psi_{a,b}(t)$, the new oscillation frequency becomes $\frac{\eta}{aw}$, and thus varying the scale parameter a varies the frequency of the wavelet with which the signal is being compared, which basically means as we change the scale, we are moving along the frequency axis, or rather, we are moving parallel to the frequency axis in the time-frequency plane for the analysis. However, it is not a linear relationship, the scale is inversely proportional to the frequency of the wavelet, and they are related by $F = \frac{F_o}{s}$, where s is the value of scale that was used for analysis and F is the frequency coordinate of the point in the time-frequency plane. F_o is the Hz frequency of the mother wavelet.

The figures below demonstrate the effects of scaling on a sample wavelet. These are all derived from a mother Gabor wavelet with width parameter $w = 0.25$ and $\eta = 2$. $(F_o = 2/0.2 = 10Hz)$. Only the real part of the wavelet signals are plotted against time

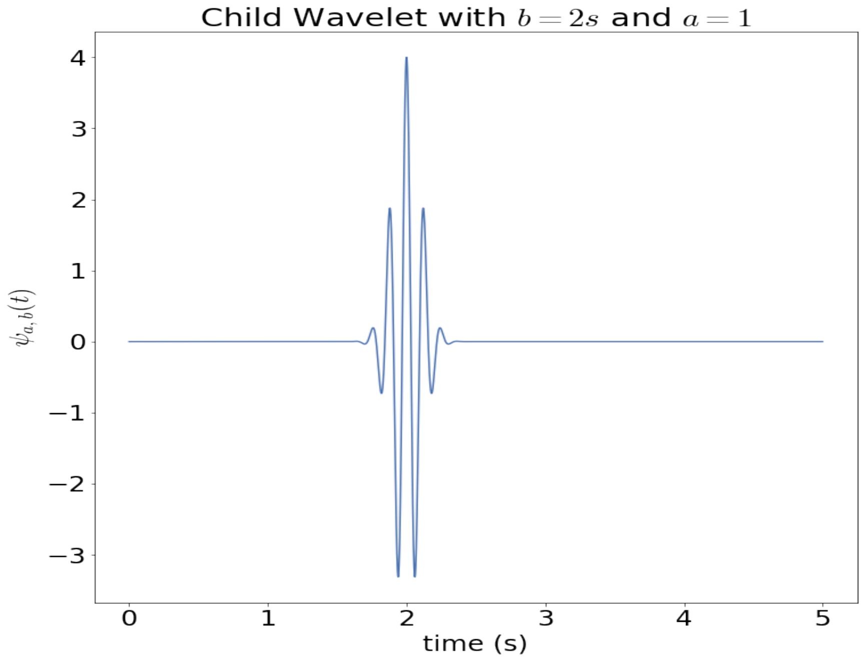

Figure 14: Child wavelet with $b = 2s$, $a = 1$ (Reference)

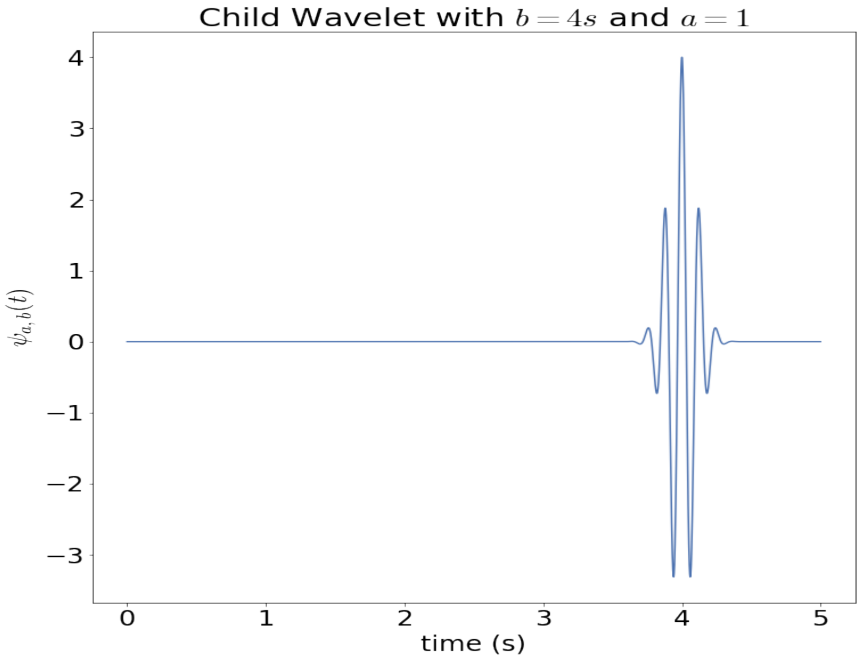

Figure 15: Child wavelet with $b = 4s$, $a = 1$, shifted relative to reference child

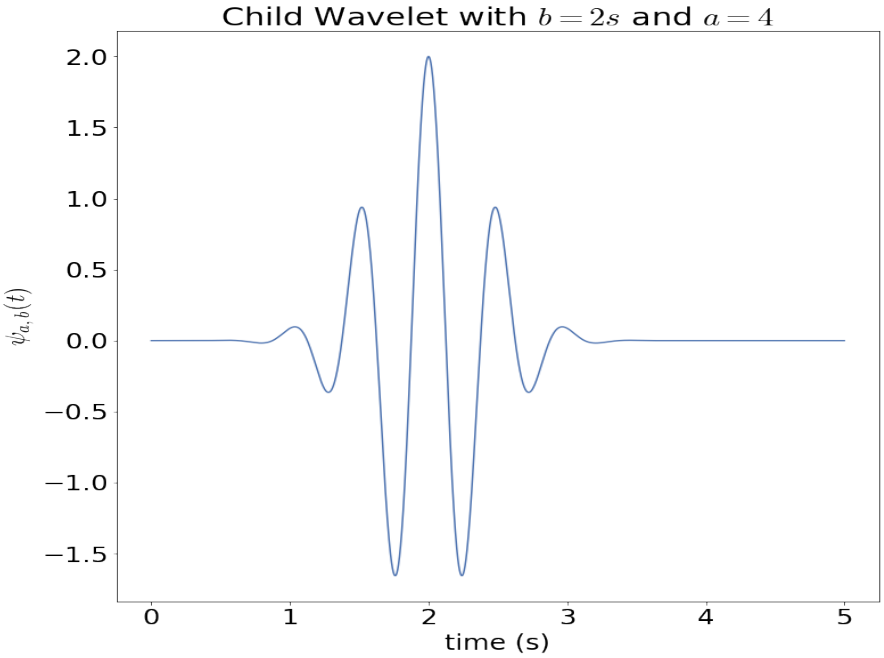

Figure 16: Child wavelet with $b = 2s$, $a = 4$, scaled with respect to reference child

The figures clearly demonstrate the Gaussian envelope and oscillatory component of each child. It can also clearly be seen that a shift alone produces no change in thee width/spread or the oscillation frequency, but only centers the same wavelet at a different point in time. Further, a scaling operation alone does 2 things: it widens/compresses the Gaussian envelope and it reduces/increases the oscillation frequency of the sinusoidal component.

Gabor wavelets have a special advantage over other wavelets. They achieve the theoretical minimum possible time-frequency uncertainty product. In other words, the product of standard deviations of $\psi(t)$ and $F_\psi(f)$(CTFT of $\psi(t)$, where f is Hz frequency), is $\frac{1}{4\pi}$. Considering the Gaussian envelope alone, in the time domain, the standard deviation of the Gaussian function is $\frac{w}{\sqrt{2\pi}}$, and since a Gaussian in the time domain translates to a Gaussian in the frequency domain, $F_\psi(F)$ will also have a standard deviation Δf that can be found similarly by comparing the Fourier transform of the Gaussian in time domain to thhe standard expression for a Gaussian, and it turns out that $\Delta f \Delta t$ is exactly the theoretical minimum, $\frac{1}{4\pi}$.

In the mathematical sense, there are certain conditions a wavelet must fulfill to be of use in wavelet analysis. Once the wavelet analysis has been carried out using equation 10.2, we also wish to be able to reconstruct the time-domain signal from the wavelet transform information. The condition on the wavelet used under which the signal can be perfectly reconstructed from its cwt is called the admissibility condition, which we state here without proof. If $X_\psi(\Omega)$ is the CTFT of the mother wavelet, then the admissibility condition is:

$$C_\psi = \int_{-\infty}^{\infty} \frac{|X_\psi(\Omega)|^2}{|\Omega^2|}\, d\omega \leq \infty \tag{10.5}$$

Since the integration is being done over all $\mathbb{R}$, the integration range also includes $\Omega = 0$, which means the function $\frac{|X_\psi(\Omega)|^2}{|\Omega^2|}$ must exist at $\Omega = 0$, or $X_\psi(0)$ must be 0. We know that $X_\psi(0) = \int_{-\infty}^{\infty} \psi(t)e^{-j0t}\, dt = \int_{-\infty}^{\infty} \psi(t)\, dt$, which is just the average or mean value of the wavelet in the tome domain. Thus, the wavelet must have a zero average value.

If the wavelet satisfies the admissibility condition, then given $F(a,b)$, the cwt of $f(t)$, we must be able to return to the signal $x(t)$. This is done through the inverse continuous wavelet transform:

$$\frac{1}{C_\psi} \int_{-\infty}^{\infty} \int_{0}^{\infty} F(a,b)\psi_{a,b}(t) \frac{1}{a^2}\, da\, db \tag{10.6}$$

where the limits on the inner integral have been restricted to 0 to ∞ because the scale parameter is generally never set as positive, i.e the child wavelets having positive scale are only used in the analysis.

10.1 Some Major Types of Wavelets

We have discussesd the Gabor wavelet family in detail. Here, we talk about a few other types of wavelets also often used in the CWT:

10.1.1 Haar Wavelets

Mathematically defined, the Haar wavelet is a sequence of rescaled square-shaped functions which together form a wavelet family or basis.

As a special case of the Daubechies wavelet, the Haar wavelet is also known as Db1.

The Haar wavelet is the simplest possible wavelet. Its technical disadvantage is that it is not continuous, and therefore not differentiable. This property can, however, be an advantage for the analysis of signals with sudden transitions (discrete signals), such as monitoring of tool failure in machines. These wavelets have very good time localization but poor frequency localization.

The Haar wavelet's mother wavelet function $\psi(t)$ can be described as

$$\psi(t) \begin{cases} 1, & 0 \leq t \leq \frac{1}{2} \\ -1, & \frac{1}{2} \leq t \leq 1 \\ 0, & \text{otherwise} \end{cases} \tag{10.7}$$

10.1.2 Shannon Wavelets

The complex Shannon wavelet is a product of complex sinusoidal function and sinc functions. This wavelet is symmetric and used in complex continuous transform. It is defined by a bandwidth parameter f_b, a wavelet center frequency, f_c. Shannon wavelet is expressed as:

$$\psi(t) = \sqrt{f_b} sinc(f_b t) e^{2\pi f_c t} \tag{10.8}$$

These wavelets have very good frequency localization but poor time localization. They are indeed time-scale mirror images of Haar wavelets.

10.1.3 Mexican Hat Wavelets

Mathematically defined, the Mexican hat wavelet, also called the Ricker wavelet is the negative normalized second derivative of a Gaussian function. It is expressed as shown below:

$$\psi(t) = \frac{2}{\sqrt{3\sigma}\pi^{1/4}}(1 - (\frac{t}{\sigma})^2)e^{-\frac{t^2}{2\sigma^2}} \tag{10.9}$$

11 Discretization of the CWT and Scalograms

Now that we have seen how the value of the CWT at any combination a, b represents the 'strength' of the wavelet $\psi_{a,b}(t)$ in the signal, we must move on to practical implementation. Naturally, the CWT is unfit for implementation on digital computer because both the signal being analysed as well as the wavelet are continuous-time signals and moreover, even the scale and shift parameters vary continuously. In practice, the signal being analysed is a discrete-time (generally finite) signal obtained by sampling a real-world signal, and the wavelet constructed

for use in a wavelet transform program is also discrete-time. Additionally, rather than computing the CWT value for all real values of shift b and all real and positive values of scale a, it is computed instead for a finite number of appropriate shift and scale values. To obtain an expression suitable for implementation, we must discretize the CWT analysis integral. This is done in the normal way, using a Reimann-sum approximation of the integral. Firstly, assuming the signal being analysed, $f(t)$, is nonzero only in the time interval 0 to t_o, the CWT analysis equation becomes simply
$F(a,b) = \int_0^{t_o} f(t)\psi^*_{a,b}(t)\,dt$, as $f(t) = 0$ if t is not in the specified range. Now to discretize the signal $f(t)$ and the wavelet $\psi_{a,b}(t)$, let us assume that the interval $[0, t_o$ is partitioned into N parts of uniform size. The start time of the m^{th} interval, where m is an index going from 0 to $N-1$, is given by $t_m = m\Delta t$, where $\Delta t = \frac{t_o}{N}$ is the width of each segment. The value of the signal $f(t)$ assigned to the m^{th} interval is the value at the beginning of the interval, or $f_d(m) = f(t_m) = f(m\Delta t)$, where $f_d(m)$ is the value assigned for the m^{th} interval. This is in fact nothing but the regular uniform sampling process that is generally applied to signals, i.e a continuous time signal $x_a(t)$ is converted into a discrete-time signal $x(n)$ through uniform sampling at an interval of T seconds (sampling period), or $x(n) = x_a(nT)$. In this case we are using index m rather than n and denoting the sampled signal with the subscript d.
Now, approximating the integral as a Reimann sum with the aforementioned partition, we get $F(a,b) = \int_{t=0}^{t_o} f(t)\psi^*_{a,b}(t)\,dt \approx \Delta t \sum_{m=0}^{N-1} x(m\Delta t)\psi^*_{a,b}(m\Delta t)$. Thus we have discretized the signal being analysed and the wavelet. We must next discretize the parameters a and b so as to be evaluate the CWT at a finite number of points (a,b), which can be done practically.
Now, we know that $\psi_{a,b}(m\Delta t)$ is nothing but $\psi(\frac{m\Delta t - b}{a})$. Now, the shift parameter b represents the shift in time of the wavelet being compared with the signal, and it can also thus be discretized in the same manner as t, taking $b = k\Delta t$. (We use the same step size here for simplicity). However, for the scale parameter a, we will not use a linear set of scale values for the discretization, but rather, we will choose the scale values to be logarithmically related. This is especially useful in the analysis of musical signals, because most well-tempered musical scales are logarithmic (eg: an octave of increase in the perceived pitch is an exact doubling of frequency, hence the name octave). While choosing a range of values for a, we define 2 important terms: the number of octaves and the number of voices per octave. Each octave, as stated before represents a doubling in the frequency. In this case, since we are talking about scale and not frequency, the traversal of an octave means a halving of the scale. In other words, if we are presently at a particular value of scale, moving one octave up corresponds to moving to the note with double the current frequency, or half the current value of the scale. Further, each octave can be subdivided into a number of notes that are traversed before reaching the next octave. These are called the voices of an octave, and the number of voices in an octave determines how finely each octave is divided. A relatable example is the 7 notes: Do, re, mi, fa, so, lah, and ti, which are traversed before reaching the next Do (7 notes in the octave). The next Do has double the frequency of the current Do, or it is the same note of the next octave. The same

concept is applied to scale, rather than frequency. Keeping these things in mind, suppose for the analysis, we choose J voices per octave and I octaves. Then define the index p to lie in the range $\{0, 1, ...I * J\}$. The scale values are computed as $a = 2^{-p/J}$, and we thus get a value of scale for each value of p. For example, suppose we choose $I = 6$ octaves and $J = 12$ voices per octave, we will get $p \in \{0, 1, ...72\}$ and $a = 2^{-p/J} = \{2^0, 2^{-1/12}, ..., 2^{-11/12}, 2^{-1}, ..., 2^{-6}\}$. Thus we traverse 6 octaves, as the scale gets halved 6 times as it goes from 2^0 to 2^{-6}, and further, we take 'exponential steps' of size $2^{-1/12}$ between 2 values of a.

Now that we have the values of a and b for which we would like to evaluate the CWT, we perform the summation:

$$F(a_p, k\Delta t) = \sum_{m=0}^{N-1} f(m\Delta t)\psi^*(\frac{(m-k)\Delta t}{a_p}) \tag{11.1}$$

for each value of a_p and k. Here, a_p is simply used to denote the set, or array $\{2^{-p/J}\}$ and k is the set, or array of values of shift for which we want to evaluate the CWT. The choice of I and J is important, and it must be done in accordance with the kind of scale present in the musical signal.

Once this summation has been evaluated for a number of values of shift and scale, the scalogram can be plotted./ The scalogram is essentially a heat-map or color-map plot of the CWT magnitude (not magnitude squared) against the shift values on the X axis and the scale values on the Y axis. To make it more understandable, the Y axis can be brought back to Hz frequency through the relation $F(Hz) = \frac{F_o}{s}$, where F_o is the frequency of the mother wavelet, as stated before.

12 Results

In this section we apply some of the aforementioned concepts practically. First we demonstrate the application of inversion process to compression of audio signals. Figure 17 shows the Spectrogram of a 2 V V_{rms}, 50 Hz test signal sampled at $f_s = 1024$ Hz and corrupted by $0.001\frac{f_s}{2}$ Hz of white noise.

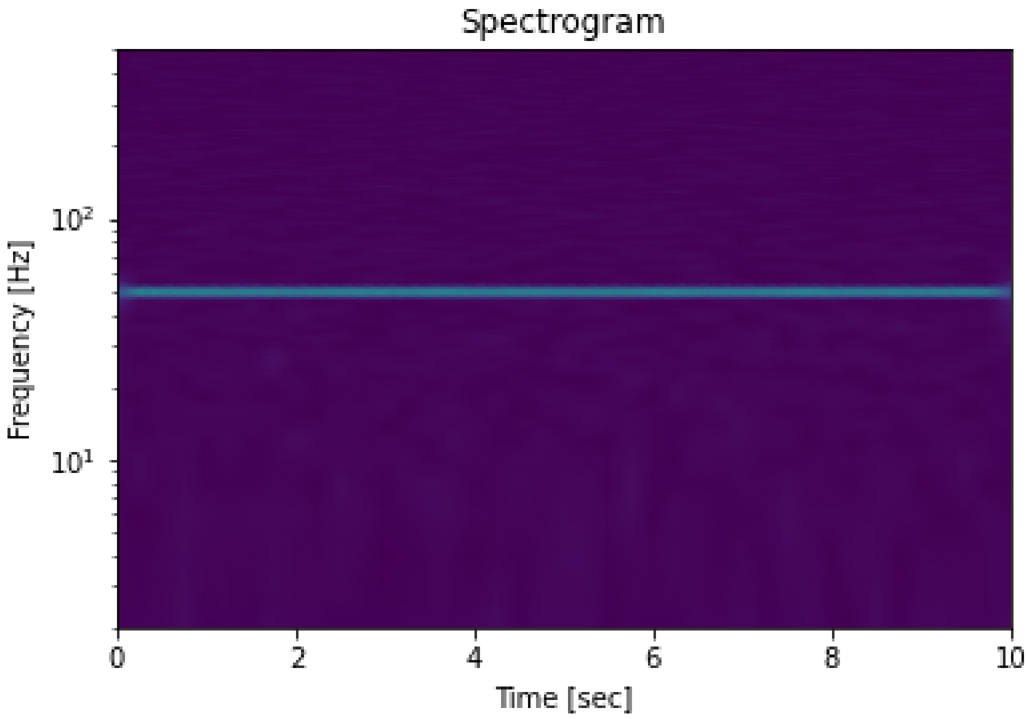

Figure 17: Spectrogram of Audio Signal

The Spectrogram is computed using Hanning window with a segment length of 512 samples and overlap 50%. All of the plain regions shown in the spectrograms stand for near zero values and that implies considerable redundancy for the spectrogram. We discard (set to 0) all the values in the spectrogram with magnitudes below 10% of the carrier magnitude as a threshold. This is done because the spectrogram can be greatly compressed by removing large amount of redundancy created by all of the near zero values. The inversion procedure now creates an approximation of the original signal that uses significantly less data than the original signal. However, in order to enable inversion of STFT with `istft`, the signal windowing must obey the constraint of "nonzero overlap add" (NOLA):

$$\sum_t w^2(n - tH) \neq 0 \tag{12.1}$$

This ensures that the normalisation factors that appear in the denominator of the overlap-add reconstruction equation

$$x(n) = \frac{\sum_t x_t(n)w(n - tH)}{\sum_t w^2(n - tH)} \tag{12.2}$$

are not zero. The NOLA constraint can be checked with the `check_NOLA` function and is `True` for our window specifications mentioned above. An STFT which has been modified is not guaranteed to correspond to exactly realizable signal. `istft` implements the iSTFT via the least-squares estimation algorithm detailed in [5], which produces a signal that minimizes the mean squared error between the STFT of the returned signal and the modified STFT.

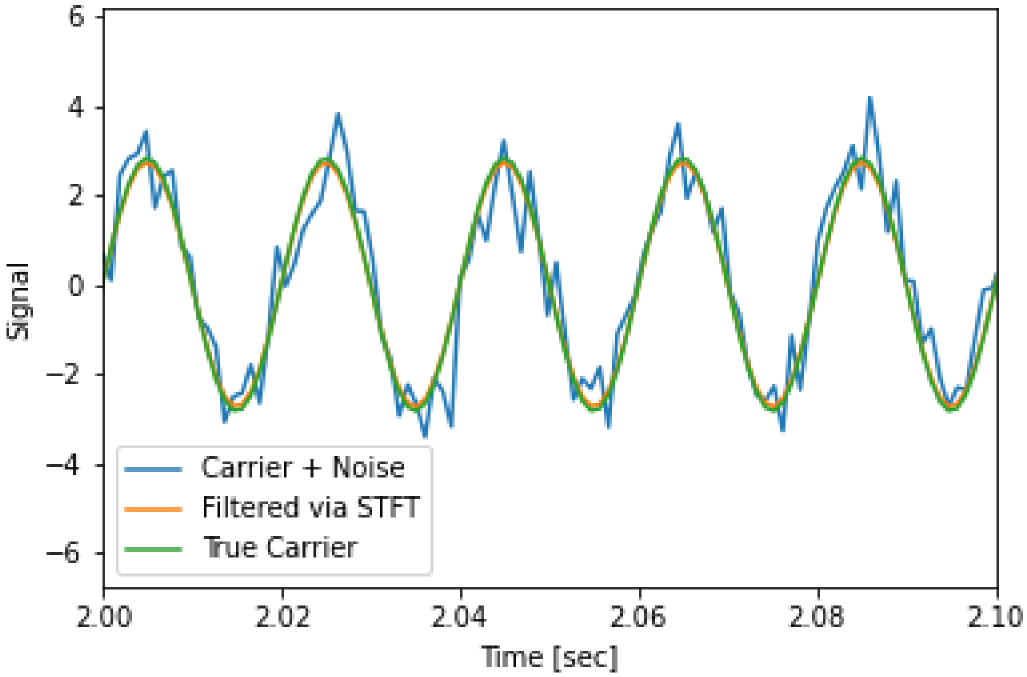

Figure 18: Plot of Original and Cleaned signals

Figure 18 shows the plot of the cleaned signal obtained from inversion and the original signal. The true carrier is also given for reference. We observe that approximated signal closely resembles the true carrier indicating the removal of white noise. However, we note that the cleaned signal does not start as abruptly as the original as can be seen in Figure 19 which presents a zoomed in view along the time axis. This is because some of the coefficients of the transient have also been removed.

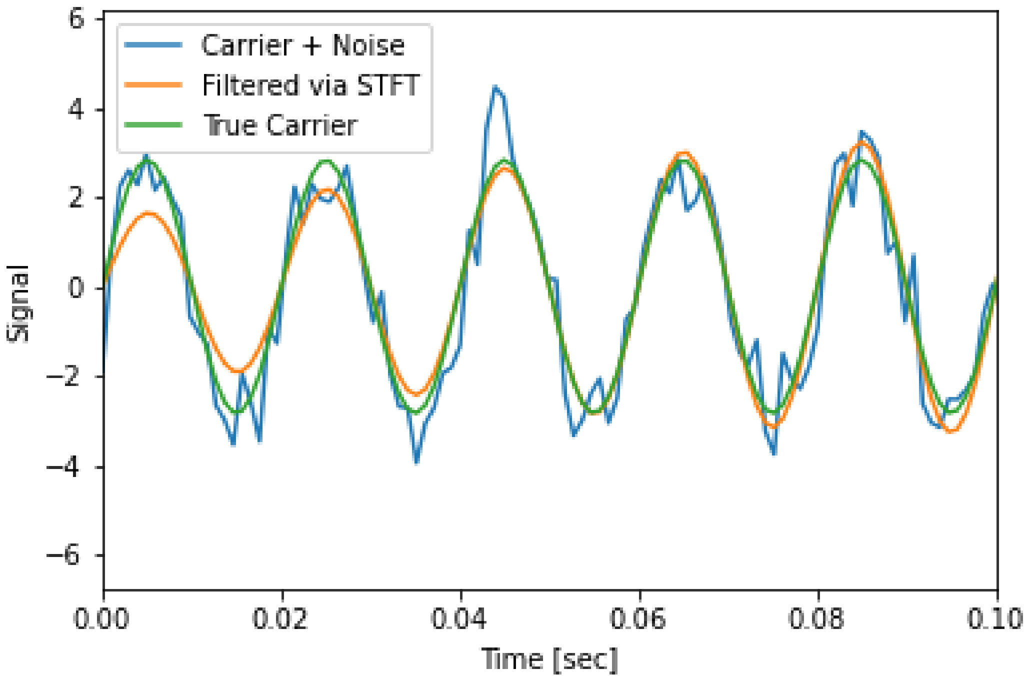

Figure 19: Zoomed in view of Original and Cleaned signals

Such audio compression techniques are the basis for the sophisticated generalisations that underlie the real-time audio players and audio download sites available on the Internet.

Next, we demonstrate the usage of wavelets and scalograms to achieve multiresolution analysis of musical signals.

First, a manually-written code that implements the discretized version of the CWT was tested on a test signal. The test signal was generated as a concatenation of single-frequency sinusoidal signals with frequencies $10, 20, 30, 40, 50$ and 60 Hz respectively. The sampling frequency used is 1000Hz and the overall duration is 1.5 seconds. The reason why the sampling rate and the duration were chosen to be relatively low, is that the code developed is quite rudimentary and not very efficient and thus using a high sampling rate or a longer duration signal would have made the program extremely slow. A user-defined function was written in python to define the mother wavelet. This function takes the width parameter and η value of the mother wavelet(which stay fixed during the analysis) as well as a value of scale(a) and shift(b), and finally, the time range over which the wavelet needs to be defined, as a time vector. It returns an array of complex values representing the samples of the child wavelet $\psi_{a,b}(t)$. This is the same code that was used to obtain the pictures of the wavelets shown in section 10. In the main program, the signal $x(n)$ is read from a .wav file that it has already been written to. Its length N and sampling frequency F_s are recorded. A time vector $t = \{0, 1, ...\frac{N}{F_s}\}$ is defined for the signal. The scale values were chosen using $I = 3$ octaves and $J = 200$ voices per octave, and an array of scale values is generated as $2^{-p/J}$, p is the array $\{0, 1, ...IJ\}$. The shift values are chosen as the array $k = \{-\frac{N}{2}, ...\frac{3N}{2} - \frac{1}{F_s}\}$, with the same step size ($T_s = \frac{1}{F_s}$) as the original signal. The range of shifts is extended beyond the actual signal's time range to get a better picture. A matrix of size $IJ+1$ by $2N$, where $2N$ is the length of vector k, is generated to hold the CWT values. In a nested for loop, where the outer loop iterates through the values of scale and the inner loop through the values of shift, the summation of equation 11.1 is implemented for each value of shift and scale and taking $w = 0.25$ and $\eta = 2$ for the mother wavelet, and the complex CWT value is stored in the aforementioned matrix. The scalogram is plotted as a 2-D color map, against frequency, rather than scale, on the Y axis, and shift (in seconds) on the X axis. For the test signal, the plot obtained is shown below:

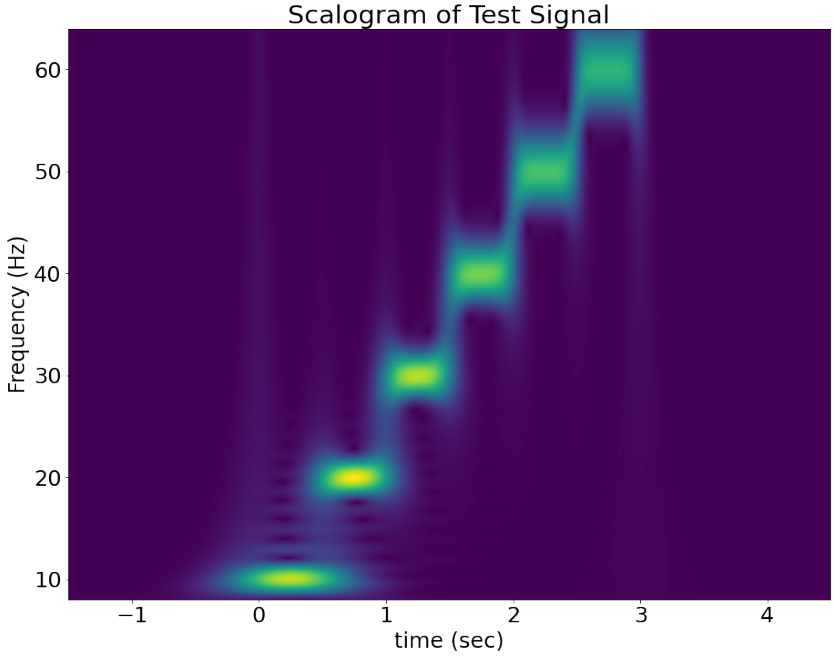

Figure 20: Scalogram of test Signal

The reason for the poor quality (poor time and frequency resolution) is due the constraints mentioned above. However, even this rudimentary code shows the main feature of the CWT: its multiresolution capability. Clearly each line in the scalogram corresponds to one of the sinusoidal subsignals, and it is evident that the CWT acts as a time-frequency tool as it is correctly able to track the change in frequencies with time. Further, if we look at the lines corresponding to lower frequencies, they are much narrower in width along the frequency axis, than the high-frequency lines, i.e they have a sharper or better frequency resolution, but a poor time resolution, as it is not easy to make out the exact instant of transition from one frequency to another. However, as time progresses, the frequencies present in the signal also increase. At higher frequencies, we can see sharper transitions and it is easier to make out the exact instant of the transition, however, the frequency resolution is now very poor, as the lines are much wider (along the Y axis). Though we are still bound by the uncertainty principle at any value of scale, we now have the ability to switch from better time resolution to better frequency resolution

We now demonstrate another merit of wavelet analysis: its ability to allow us to 'zoom in' on certain frequencies. Often, we have a number of frequency lines/components which are very close to each other/compactly spaced. In this case, we would like to have the ability to view these lines with sufficient resolution so as to be able to tell them apart, however, the STFT doesn't allow us to do this. Since the STFT is plotted with a linear frequency scale, there is no way to obtain a better resolution of lower frequencies. In the case of the CWT, if we plot either the inverse of the scale($= \frac{1}{s}$), or the frequency in Hz ($= \frac{F_o}{s}$), for better clarity, on the frequency axis, then due to the way in which the scale values were chosen, the frequency axis is naturally logarithmic with base 2. This not only correlates well with out perception of frequency (i.e pitch), but also allows us to look at the low frequency lines with greater clarity. Often in musical signals, this is extremely useful, for example: in a guitar signal, we may end up with a number of frequency lines even below the fundamental, which may arise due to resonance in the guitar cavity. These lines are often very densely spaced, and the simple Fourier transform and the STFT do not allow us to mathematically 'zoom in' on these frequencies. Here, by 'zoom in', we mean that we wish to be able to view these densely-spaced lines with better resolution. We first demonstrate this using a test signal, having 2 stationary frequency components of 50Hz and 51Hz respectively, a duration of 1s, and a sampling frequency of 500 samples/sec. Its STFT is generated and plotted using the code mentioned in section 7, to obtain the following spectrogram, using hamming window of length 200 samples, with 190 overlapping samples between consecutive frames:

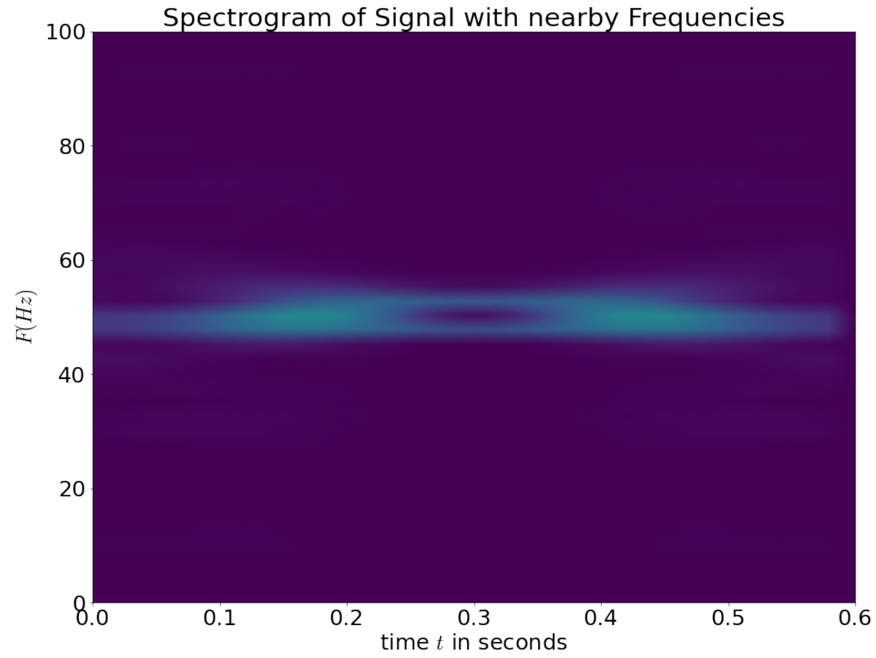

Figure 21: Spectrogram of signal with 2 nearby frequencies

We can see that though we can make out that there are 2 separate lines, the resolution is not very sharp, and it will be difficult to estimate the frequencies of the 2 frequency components using this image. Now, if we plot the scalogram of the same signal, using only 1 octave and 500 voices per octave, The base frequency (of the mother wavelet) is chosen as $\frac{50}{\sqrt{2}}Hz$ using $\eta = \frac{50}{\sqrt{2}}$ and $w = 1$. The reason for choosing this particular value of η is simply to center the frequency lines in the plot, since the frequency scale is logarithmic with base 2. It must be noted here that the y acxis quantity is just $\frac{1}{a}$, and not the frequency $\frac{F_o}{a}$, as we are just trying to get an idea of the frequencies present.

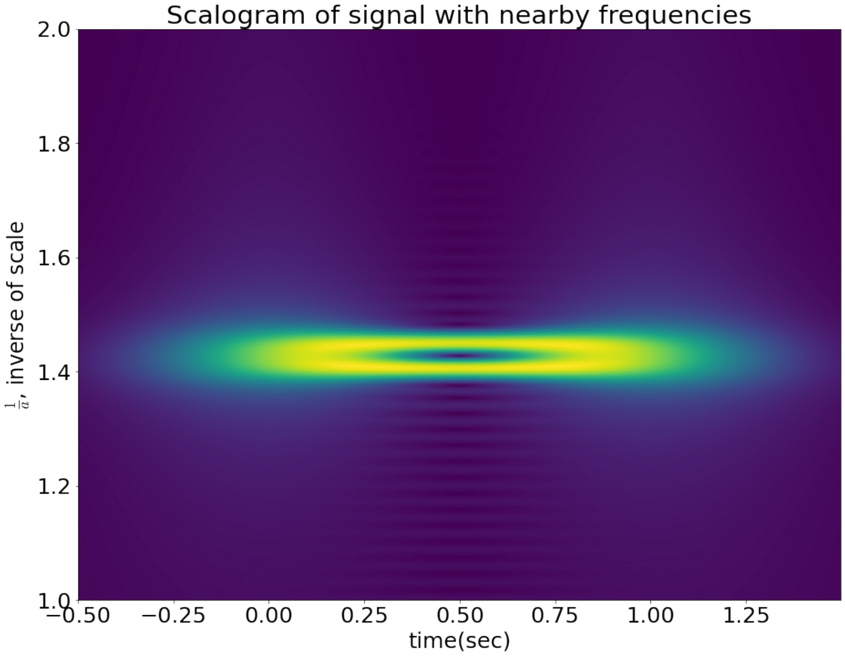

Figure 22: Scalogram of signal with 2 nearby frequencies

It can be seen that the CWT is also able to separate out the 2 components. The improvement here is in terms of the frequency resolution. Since we were given the freedom to choose η, w, J and I, we were able to choose them in such a way as to be able to zoom in on the region around $50Hz$. In the STFT, there is no such mechanism to mathematically zoom in on, or to get a better view of any one section of the frequency axis. This is because in the STFT, we must work with windows of fixed length and thus cannot adjust the frequency resolution to be higher for a particular set of frequencies. The best we can do is mechanically zoom in on the plot, which does not help us in any way with the frequency resolution. With the CWT, by tweaking the parameters, we can achieve high frequency resolution for particular frequencies in some range that we are trying to analyze, This has applications in a number of fields, for example, we could use the same principle to separate and estimate 2 very nearby spectral lines in spectroscopic analysis, which is of great importance in this field, to able to correctly identify which elements are present in a given sample.

We now demonstrate the above concepts with an actual musical signal. The flute signal whose spectrogram was plotted is now subjected to the CWT and its scalogram is obtained, on MATLAB, using the cwt() function (as the signal is quite long and has a high sampling rate, the user-defined code could not be used).

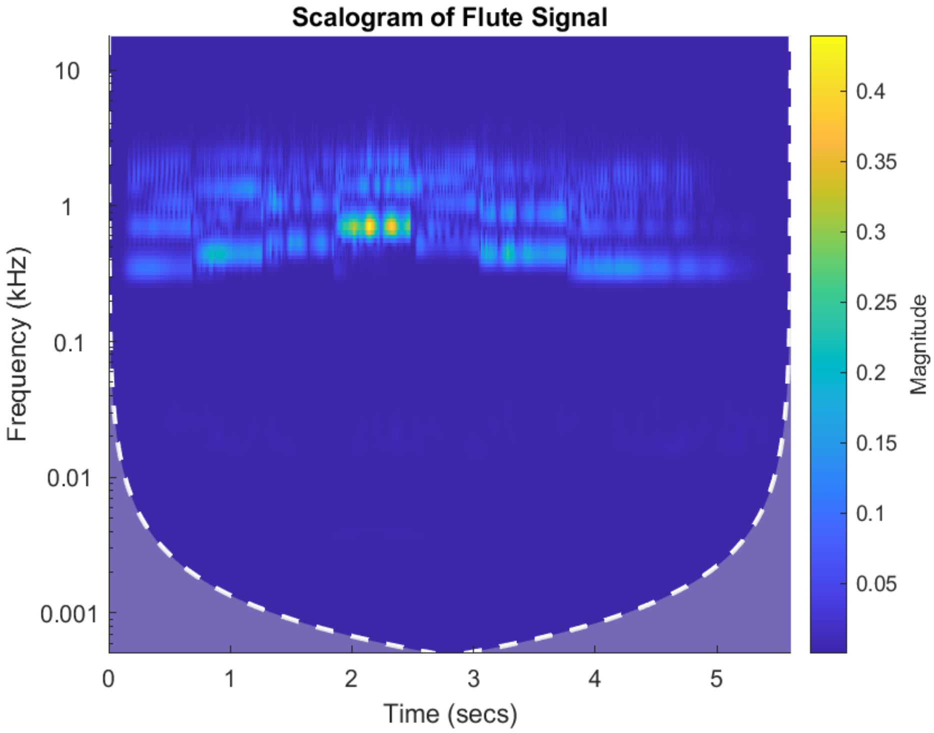

Figure 23: Scalogram of Flute Signal

The plot can be compared with the spectrogram plot previously obtained, which shows that the scalogram also gives an equally good overall picture of the signal's time-frequency portrait. The real advantage of the CWT is its ability to analyze any frequency range in detail. For example, suppose we wished to analyse in detail only the range $200-400$Hz in the first 500 or so samples of the clip, we could choose $\eta = 200$, $w = 1$, and use a single octave with 250 voices per octave, and we get the following zoomed-in scalogram.

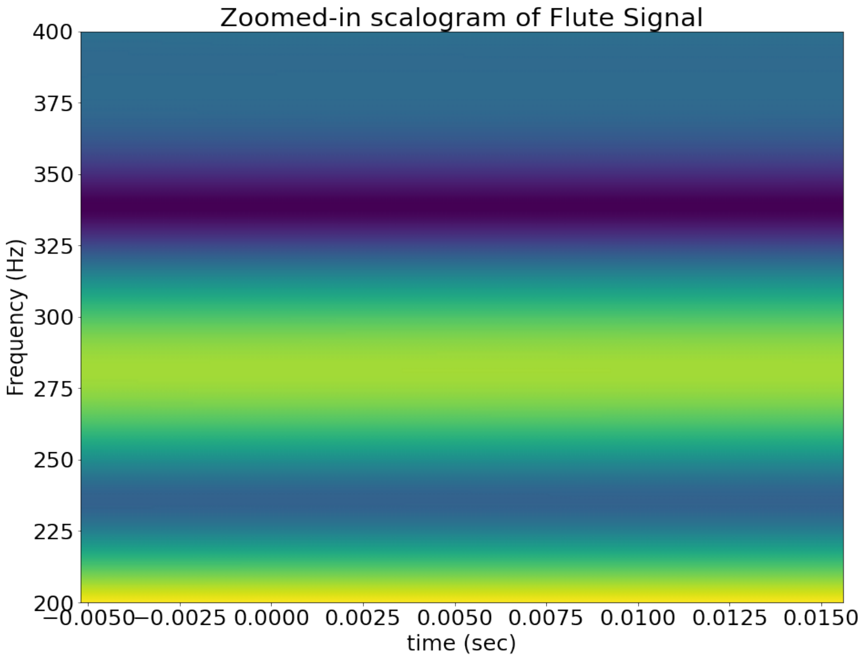

Figure 24: Flute Signal zoomed in on 200-400Hz

Achieving such high resolution in such a narrow range with the STFT would have involved having to increase the window length for the entire analysis and then manually zooming in on this range by adjusting the display limits, while the CWT achieves this in a much more elegant and faster manner.

13 Conclusion

In the course of this project, we have examined two of the most popular time-frequency analysis tools and their use in the analysis of music signals. We saw how a music signal, being composed of progressions of notes, are nonstationary and are thus simple Fourier analysis fails to provide the full picture. We then examined the theory behind the Short Time Fourier Transform, a simple yet powerful time-frequency tool, and also looked into the uncertainty principle which sets a fundamental limit on any form of time-frequency analysis. We then discretized and implemented the STFT to obtain spectrograms, which were then analyzed to gain several insights into the features of a few sample music signals. We further also looked into and demonstrated the conditions for and practicality of being able to invert the STFT. However, we also saw that the STFT was limited in its capability by its fixed resolution. To overcome this, we moved to the Continuous Wavelet Transform, a multi-resolution tool. We examined the theory behind wavelets and the rationale for using such functions as the analysing functions. The CWT was then discretized and implemented to obtain scalograms, which provide flexibilty in terms of frequency resolution and allow us to analyze portions of the signal's spectrum at any given instant with much greater detail. It was concluded that scalograms overcome the limitations of spectrograms, and provide the best possible way of analysing musical signals in the time-frequency plane.

14 Contributions of individual members

1. Siddharth R Iyer - Need for time-frequency analysis, Python coding for spectrograms, wavelets and scalograms, Theory on STFT (continuous and discrete-time), Theory and results pertaining to CWT.
2. P Vishal - Introduction, Literature Review and Additional Wavelets
3. Yogananda B A - Continuous and discrete-time window functions and properties
4. Hruthwik C Vijaya Kumar - Limitations and The Uncertainty Principle
5. Prasad Fidelis D'sa - Introduction, Arranging instrument tracks, Practical inferences and Inversion of Spectrograms in Result section, Python coding and theory of Inversion of Spectrograms

15 Additional Resources

1. `http://www.its.caltech.edu/~matilde/GaborLocalization.pdf`

2. `https://web.iitd.ac.in/~sumeet/WaveletTutorial.pdf`

3. `https://www.engr.colostate.edu/ECE513/SP09/lectures/wavelets.pdf`

4. `https://web.ece.ucsb.edu/~yoga/courses/Adapt/P1%20Time-Frequency%20representation.pdf`

5. `https://nptel.ac.in/content/storage2/courses/117105145/pdf/Week_5_Lecture_Material.pdf`

6. `https://nicholasdwork.com/teaching/1706ee102a/lectures/lecture14.pdf`

7. `http://www.ee.ic.ac.uk/pcheung/teaching/de2_ee/Lecture%205%20-%20DFT%20&%20Windowing%20(x2).pdf`

8. `https://math.stackexchange.com/questions/3410183/ejwt-is-an-ortho-normal-basis-proof`

References

[1] Plectrum, May 2021. URL: `https://en.wikipedia.org/wiki/Plectrum`.

[2] Jeremy Alm and James Walker. Time-frequency analysis of musical instruments. *Society for Industrial and Applied Mathematics*, 44:457–476, 09 2002. `doi:10.1137/S00361445003822`.

[3] Xiaowen Cheng, Jarod V Hart, and James S Walker. Time-frequency analysis of musical rhythm. *Notices of the AMS*, 56(3):256–372, 2009.

[4] R.L. de Queiroz, T.Q. Nguyen, and K.R. Rao. The genlot: generalized linear-phase lapped orthogonal transform. *IEEE Transactions on Signal Processing*, 44(3):497–507, 1996. `doi:10.1109/78.489023`.

[5] D. Griffin and Jae Lim. Signal estimation from modified short-time fourier transform. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 32(2):236–243, 1984. `doi:10.1109/TASSP.1984.1164317`.

[6] MasterClass. Arpeggios explained: What is an arpeggio in music? - 2021, Jun 2021. URL: `https://www.masterclass.com/articles/arpeggio-definition`.

[7] Alfred Mertins and Dr Alfred Mertins. *Signal analysis: wavelets, filter banks, time-frequency transforms and applications.* John Wiley & Sons, Inc., 1999.

[8] Alan V Oppenheim and Ronald W Schafer. Digital signal processing(book). *Research supported by the Massachusetts Institute of Technology, Bell Telephone Laboratories, and Guggenheim Foundation. Englewood Cliffs, N. J., Prentice-Hall, Inc., 1975. 598 p*, 1975.

[9] William J Pielemeier, Gregory H Wakefield, and Mary H Simoni. Time-frequency analysis of musical signals. *Proceedings of the IEEE*, 84(9):1216–1230, 1996.

[10] Alexander D Poularikas. *Handbook of formulas and tables for signal processing.* CRC press, 2018.

[11] KM Muraleedhara Prabhu. *Window functions and their applications in signal processing.* Taylor & Francis, 2014.

[12] Olivier Rioul and Martin Vetterli. Wavelets and signal processing. *IEEE signal processing magazine*, 8(4):14–38, 1991.