Using Fourier Analysis to Interpret and Model Musical Sound Waves Generated From a Piano

Research Question: How can the fourier analysis be applied to better understand the behavior of, and model the sound wave produced from a piano?

International Baccalaureate Extended Essay
Mathematics

Word Count: 2680
Date of Submission: December 1st, 2023

Table of Contents

Background:

As a former pianist, something that has always intrigued me was how the musical notes played from the instrument would always sound so different to the ears despite coming from the same instrument. I always believed that this must have been the result of some sort of technology or complex machinery that would produce these sounds; however I was surprised to see that the inside of a piano mostly consisted of regular strings and pads. After doing further investigation into why this is the case, I found out that the sound waves generated from vibrations caused within the piano are what resulted in unique frequencies that created distinct sounds to our ears (Yamaha, n.d.). Additionally, the unit for frequencies is Hz, where one hertz represents one full period/cycle of a sound wave moving per second.

Soundwaves can be represented graphically by plotting a periodic signal along a time vs. amplitude graph, where the x-axis represents the time passed in seconds and the y-axis represents the loudness of the sound in decibels (University, n.d.). However, the soundwave of a piano is not exactly a perfect sine wave, because interference from other strings will result in other sine waves with different frequencies being added into the total soundwave. In fact, the specific frequencies responsible for this are exact integer multiples of the original sound's frequency, which is referred to as the fundamental frequency of the soundwave (see figure 1). Thus, the aim of this paper is to determine how mathematics can be applied to better understand the properties of the piano soundwave by decomposing it into its original frequencies, and modeling the soundwave with a mathematical function.

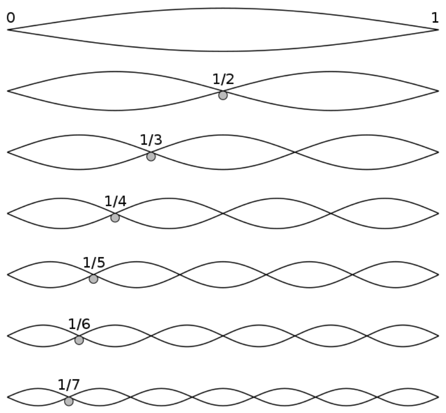


Figure 1 (Qef, 2008)

My approach was as follows: In order to first decompose a sample sound wave into its constituent frequencies, I first applied a fourier transform to it. I then utilized the fourier series representation of the function, with the data obtained from the fourier transform, in order to model the soundwave as a function.

Fourier Series

A fourier series is defined to be an “expansion of a periodic [and continuous] function f(x) in terms of an infinite sum of sines and cosines” (Weisstein, 2023). The general form for a fourier series of a function f(x) with a period of “L” is written as the following:

$f(x) = \frac{1}{2}a_0 + \sum_{n=1}^{\infty} a_n cos(\frac{2\pi n}{L}x) + \sum_{n=1}^{\infty} b_n sin(\frac{2\pi n}{L}x)$, a_0, a_n, and b_n are fourier coefficients:

$$a_0 = \frac{1}{\pi}\int_{-\frac{L}{2}}^{\frac{L}{2}} f(x)\, dx$$

$$a_n = \frac{1}{\pi}\int_{-\frac{L}{2}}^{\frac{L}{2}} f(x)\, cos(\frac{2\pi n}{L}x)\, dx$$

$$b_n = \frac{1}{\pi}\int_{-\frac{L}{2}}^{\frac{L}{2}} f(x)\, sin(\frac{2\pi n}{L}x)\, dx$$

However, the notation can also be expressed in terms of complex exponential coefficients (Svirin, 2023), in order to derive the formula for the fourier transform. Complex numbers are needed to represent both the phase shift and amplitude of a specific inputted frequency as a single output (seen later in the application of the fourier transform).

Substituting Euler’s formulas: $cosx = \frac{e^{ix}+e^{-ix}}{2}$ and $sinx = \frac{e^{ix}-e^{-ix}}{2i}$:

$$f(x) = \frac{1}{2}a_0 + \sum_{n=1}^{\infty} a_n(\frac{e^{i\frac{2\pi n}{L}x}+e^{-i\frac{2\pi n}{L}x}}{2}) + \sum_{n=1}^{\infty} b_n(\frac{e^{i\frac{2\pi n}{L}x}-e^{-i\frac{2\pi n}{L}x}}{2i})$$

$$f(x) = \frac{1}{2}a_0 + \sum_{n=1}^{\infty} (\frac{a_n - ib_n}{2})e^{inx} + \sum_{n=1}^{\infty} (\frac{a_n + ib_n}{2})e^{-inx}$$

Taking the negative of the second summation from -n to -∞ gives two of the same exponentials:

$$f(x) = \frac{1}{2}a_0 + \sum_{n=1}^{\infty} (\frac{a_n - ib_n}{2})e^{inx} + \sum_{n=-1}^{-\infty} (\frac{a_{-n} + ib_{-n}}{2})e^{inx}$$

Recalling the earlier definitions of a_n and b_n:

$$a_n = \frac{1}{\pi}\int_{-\frac{L}{2}}^{\frac{L}{2}} f(x)\, cos(\frac{2\pi n}{L}x)\, dx, \text{ and } b_n = \frac{1}{\pi}\int_{-\frac{L}{2}}^{\frac{L}{2}} f(x)\, sin(\frac{2\pi n}{L}x)\, dx$$

Since the cosine function is even and the only "n" term is within the cosine function, it can be determined that $a_{-n} = a_n$. Likewise, since the sine function is an odd function, it can similarly be deduced that $-b_{-n} = b_n$. Substituting this into our previous equation gives:

$$f(x) = \frac{1}{2}a_0 + \sum_{n=1}^{\infty}\left(\frac{a_n - ib_n}{2}\right)e^{i\frac{2\pi n}{L}x} + \sum_{n=-1}^{-\infty}\left(\frac{a_n - ib_n}{2}\right)e^{i\frac{2\pi n}{L}x}$$

In order to obtain a full summation of all n terms from -∞ to ∞, the only term missing from the two sums is the n = 0th term. The n = 0th term would look like the following:

$$n_0 = \left(\frac{a_0 - ib_0}{2}\right)e^{i(0)x}$$

Substituting 0 into the definition of b_n gives the following:

$$b_0 = \frac{1}{\pi}\int_{-\frac{L}{2}}^{\frac{L}{2}} f(x)\, sin\left(\frac{2\pi(0)}{L}x\right) dx$$

$$b_0 = \frac{1}{\pi}\int_{-\frac{L}{2}}^{\frac{L}{2}} f(x)\, sin(0)\, dx$$

$$b_0 = \frac{1}{\pi}\int_{-\frac{L}{2}}^{\frac{L}{2}} f(x)\, (0)\, dx$$

$b_0 = 0$, thus

$$n_0 = \left(\frac{a_0 - ib_0}{2}\right)e^{i(0)x}$$

$$n_0 = \left(\frac{a_0 - (0)}{2}\right)(1)$$

The final result of the n = 0th term gives $\frac{a_0}{2}$, which is the exact same as the first term of the formula; which shows that $\frac{1}{2}a_0 = \left(\frac{a_0 - ib_0}{2}\right)e^{i(0)x}$, and can be substituted back into our original equation. Thus, the complex exponential fourier series can be written as:

$$f(x) = \left(\frac{a_0 - ib_0}{2}\right)e^{i(0)x} + \sum_{n=1}^{\infty}\left(\frac{a_n - ib_n}{2}\right)e^{i\frac{2\pi n}{L}x} + \sum_{n=-1}^{-\infty}\left(\frac{a_n - ib_n}{2}\right)e^{i\frac{2\pi n}{L}x}$$

$$f(x) = \sum_{n=-\infty}^{\infty}\left(\frac{a_n - ib_n}{2}\right)e^{i\frac{2\pi n}{L}x}$$

Solving the a_n and b_n term in terms of x for further simplification:

$$\frac{a_n - ib_n}{2} = \frac{1}{2}\left(\frac{1}{L}\int_{-\frac{L}{2}}^{\frac{L}{2}} f(x)\, cos(\frac{2\pi n}{L}x)\, dx \ + \frac{1}{L}\int_{-\frac{L}{2}}^{\frac{L}{2}} -\ if(x)\, sin(\frac{2\pi n}{L}x)\, dx\right)$$

$$\frac{a_n - ib_n}{2} = \frac{1}{2}\left(\frac{2}{L}\int_{-\frac{L}{2}}^{\frac{L}{2}} f(x)[cos(\frac{2\pi n}{L}x) \ - \ isin(\frac{2\pi n}{L}x)]dx\right)$$

Substituting Euler's formula: $e^{-ix} = cosx - isinx$:

$$\frac{a_n - ib_n}{2} = \frac{1}{L}\int_{-\frac{L}{2}}^{\frac{L}{2}} f(x)[e^{-i\frac{2\pi n}{L}x}]dx$$

For further simplification, let the fourier coefficient $c_n = \frac{a_n - ib_n}{2} = \frac{1}{L}\int_{-\frac{L}{2}}^{\frac{L}{2}} f(x)e^{-i\frac{2\pi n}{L}x}dx$

Thus, after substituting this back into the original equation, it can be concluded that:

$$f(x) = \sum_{n=-\infty}^{\infty} c_n e^{i\frac{2\pi n}{L}x}$$

With this new equation, the fourier transform can now be derived.

<u>Fourier Transform</u>

The Fourier transform is defined as a "mathematical technique that transforms a function of time, x(t), to a function of frequency, X(w)." (Cheever, 2022). The fourier transform is a special case of the fourier series as the period L → ∞. Since the fourier transform is a function that solves for the amplitude of each distinct frequency for every "n", the fourier coefficient c_n is what determines the amplitude and phase shift of every frequency since it contains the original coefficients a_n and b_n. As L → ∞, the fundamental frequency $\frac{2\pi}{L}$ becomes extremely small and thus the exponent $\frac{2\pi n}{L}$ can take on any real value (since "n" has a range of -∞ to ∞). Thus, re-writing the definition of c_n gives:

$$c_n = \frac{1}{L}\int_{-\frac{L}{2}}^{\frac{L}{2}} f(x)e^{-i\frac{2\pi n}{L}x}dx$$

$$Lc_n = \int_{-\infty}^{\infty} f(x)e^{-i\frac{2\pi n}{L}x}dx$$

Let F(x) represent the fourier transform function, F(x) = Lc_n:

$$F(x) = \int_{-\infty}^{\infty} f(x)e^{-i\frac{2\pi n}{L}x}dx$$

Let the variable "w" be equal to $\frac{2\pi n}{L}$

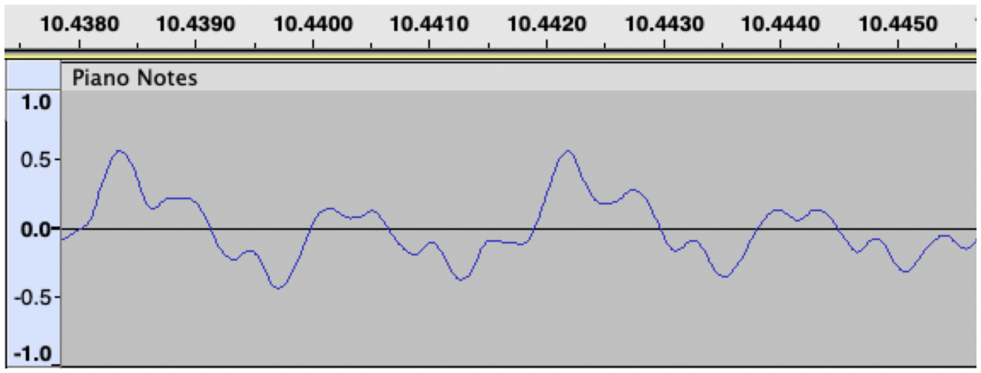
$$F(x) = \int_{-\infty}^{\infty} f(x)e^{-iwx}dx$$

Thus, the fourier transform equation has been obtained.

Methodology:

I first obtained a soundwave by recording notes that I played on my own piano; the note I played was a C_4 which has a fundamental frequency of 261Hz (Suits, 1998). I was able to visualize the software using Audacity, and the pictured soundwave (when stretched in) looks like the following (Figure 2):

Figure 2



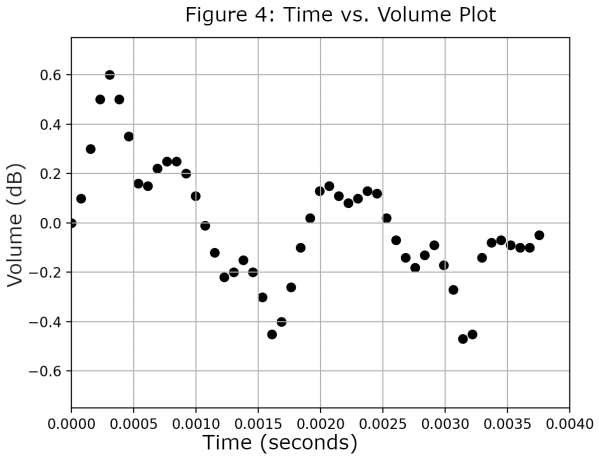
I knew that my soundwave was accurate, as the beginning of the wave was at 10.438 seconds, and it ended at around 10.4418 seconds. Subtracting the two gave a period of 0.0038 seconds, which is approximately the same value as dividing one second by the fundamental frequency of C_4 (261 Hz).However, there was no provided function f(x) for the soundwave provided. Luckily, there exists an alternative form of the fourier transform known as the discrete fourier transform, which makes obtaining the fourier coefficients of a discrete function possible.

Thus, I then resorted to recording the values of 50 equally spaced points along the soundwave in the image with the help of pixel counting software, and recorded the corresponding amplitudes and time intervals at which they were located (Figure 3). Because the total soundwave is 1/261 seconds long, and there are 50 total samples, each sample would be spaced apart 1/(261 * 50) = 1/13050 seconds.

Figure 3: Data collected from the raw soundwave

Time (sec, /13050)	**Volume (dB)**	**Time (sec, /13050)**	**Volume (dB)**	**Time (sec, /13050)**	**Volume (dB)**
0	0.00	17	-0.20	34	-0.07
1	0.10	18	-0.15	35	-0.14
2	0.30	19	-0.20	36	-0.18
3	0.50	20	-0.30	37	-0.13
4	0.60	21	-0.45	38	-0.09
5	0.50	22	-0.40	39	-0.17
6	0.35	23	-0.26	40	-0.27
7	0.16	24	-0.10	41	-0.47
8	0.15	25	0.02	42	-0.45
9	0.22	26	0.13	43	-0.14
10	0.25	27	0.15	44	-0.08
11	0.25	28	0.11	45	-0.07
12	0.20	29	0.08	46	-0.09
13	0.11	30	0.10	47	-0.10
14	-0.01	31	0.13	48	-0.10
15	-0.12	32	0.12	49	-0.05
16	-0.22	33			

After plotting this data on a separate graph, the approximation of the soundwave appears as follows: (see Figure 4 to the right). In comparison to the original soundwave, the points seem to be visually quite similar to what the original soundwave looked like.



<u>Discrete fourier transform</u>

For finding the coefficients of a regular, continuous function, the normal fourier transform for the soundwave would involve taking the integral of the function:

$$F(x) = \int_{-\infty}^{\infty} f(x)e^{-iwx}dx$$

However, because the soundwave I obtained could only be estimated using rough data points, the discrete fourier transform must instead be used.

To derive the discrete fourier transform (Roberts, 2023), first let f(x) instead be a discrete function with a repeating signal, "N" be the number of distinct samples separated by time intervals "t". The N samples will be written as: f[0], f[1], f[2], … , f[N - 1], where each signal f[n] is treated as an impulse with a length of f[n]. Since the integrand only occurs at points that were sampled, the integral can be re-written:

$$F(x) = \int_{0}^{(N-1)t} f(x)e^{-iwx}dx$$

As the discrete Fourier transform is evaluated over a finite interval rather than from -∞ to ∞, the bounds of integration were changed from x = 0 to x = (N - 1)t which represents the x-coordinate of the first to the last sample point in a given period. Because each point is discrete, the integral instead becomes a summation:

$$F(x) = f[0]e^{-i(0)} + f[1]e^{-iwt} + ... + f[n]e^{-iwnt} + ... + f[N-1]e^{-iw(N-1)t}$$

$$F(x) = \sum_{n=0}^{N-1} f(n)e^{-iwx}$$

As the discrete function is still treated as if it were periodic, the DFT equation is evaluated for its fundamental frequency $\frac{1}{Nt}$ and its harmonics, "w" can be set to 0, $\frac{2\pi}{Nt}(1), \frac{2\pi}{Nt}(2), \ldots, \frac{2\pi}{Nt}(n), \ldots, \frac{2\pi}{Nt}(N-1)$, thus in general:

$$F(x) = \sum_{n=0}^{N-1} f(n)e^{-i\frac{2\pi n}{N}x}$$

Since this is generalizing for all integer multiples of the fundamental frequency, the domain for "x" in the discrete fourier transform is $\{x\epsilon Z,\ 0 \le x \le N - 1\}$. Using this discrete fourier transform equation, the fourier coefficients for all integer multiples of the fundamental frequency can now be determined.

Processing Data

First, consider the 0th coefficient (x = 0), which is a unique case and represents the amplitude of the 0th frequency (0 Hz). Substituting all known values into the equation (x = 0, N = 50):

$$F(0) = \sum_{n=0}^{49} f(n)e^{-i\frac{2\pi n}{50}(0)}$$

$$F(0) = \sum_{n=0}^{49} f(n)e^{(0)}$$

$$F(0) = \sum_{n=0}^{49} f(n)(1)$$

Expanding this out now gives:

$$F(0) = f(0) + f(1) + f(2) + \ldots + f(49)$$

This is simply the sum of the values of all discrete points obtained from my plotting/estimation.

$$F(0) = (0.00) + (0.10) + (0.30) + \ldots + (-0.05)$$

$$F(0) = 0.46$$

Next, consider the case for the 1st coefficient (x = 1):

$$F(1) = \sum_{n=0}^{49} f(n)e^{-i\frac{2\pi n}{50}(1)}$$

$$F(1) = \sum_{n=0}^{49} f(n)e^{-i\frac{\pi n}{25}}$$

Expanding this out gives:

$$F(1) = x_0 e^{-i\frac{\pi(0)}{25}} + x_1 e^{-i\frac{\pi(2)}{25}} + x_2 e^{-i\frac{\pi(2)}{25}} + \ldots + x_{49} e^{-i\frac{\pi(49)}{25}}$$

Substituting euler's formula $e^{-ix} = cos(x) + isin(x)$ gives the following:

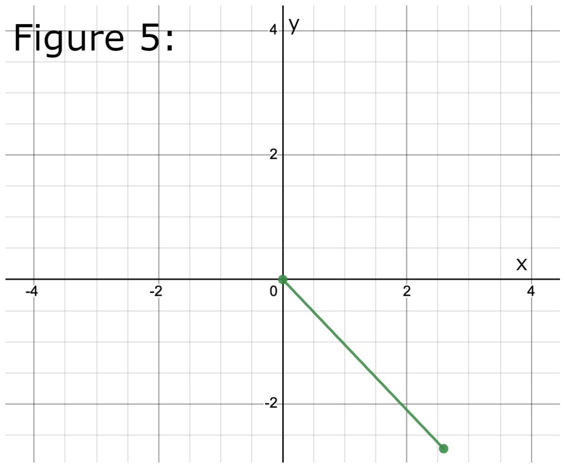
$$F(1) = x_0[cos(\frac{\pi(0)}{25}) + isin(\frac{\pi(0)}{25})] + \ldots + x_{49}[cos(\frac{\pi(49)}{25}) + isin(\frac{\pi(49)}{25})]$$

$$F(1) = (0.00) + (0.10)[0.992 - 0.125i] + \ldots + (-0.05)[0.992 + 0.125i]$$

$$F(1) = 2.59 - 2.72i$$

After plotting this complex number onto a complex plane, where the x-axis is the real coefficient of the complex number, and the y-axis is the complex coefficient, the following graph is derived (Figure 5).

Figure 5:



Using pythagoras' theorem, the amplitude "a" of the soundwave can be determined by taking the magnitude of the resultant vector:

$$a = \sqrt{x^2 + y^2}$$

$$a_1 = \sqrt{(2.59)^2 + (-2.72)^2}$$

$$a_1 = \sqrt{6.7081 + 7.3984}$$

$$a_1 \approx 3.76$$

Additionally, the resulting angle of the vector represents the phase shift of the sinusoidal wave; this angle is based off of a cosine wave (McFee, 2022). Thus, finding the resulting phase shift requires taking the arctangent of the complex coefficient over the real coefficient:

$$\theta = tan^{-1}(\frac{y}{x})$$

$$\theta = tan^{-1}(\frac{-2.72}{2.59})$$

$$\theta \approx 5.47\ radians$$

Thus, the amplitude for the first frequency wave is 3.76, and the phase shift is 5.47 radians. The resulting sinusoidal function is: $f(x) = 3.76cos(261(x + 5.47))$ (note that 261 was included as it was the fundamental frequency of the piano when x = 1).

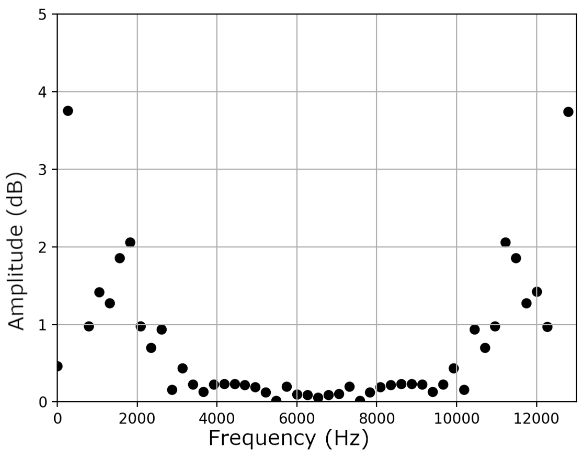
Since I had to compute 50 total terms, I ended up writing a computer program that would automatically calculate all terms, magnitudes, and phase shifts (see appendix). As a result, I obtained the following data:

Figure 6: DFT Data

Nth frequency (Hz, x261)	Resulting Coefficient	Nth frequency (Hz, x261)	Resulting Coefficient	Nth frequency (Hz, x261)	Resulting Coefficient
0	-0.46 + 0.00i	17	0.02 + 0.23i	34	0.12 - 0.20i
1	2.59 - 2.72i	18	-0.20 - 0.10i	35	0.20 - 0.09i
2	1.21 - 6.30i	19	-0.11 - 0.16i	36	0.07 + 0.11i
3	-0.17 + 0.96i	20	0.10 - 0.08i	37	0.11 + 0.20i
4	0.04 - 1.42i	21	0.02 + 0.01i	38	-0.41 + 0.15i
5	-1.26 - 0.21i	22	0.01 + 0.20i	39	0.07 - 0.14i
6	-1.42 - 1.20i	23	0.04 + 0.09i	40	0.06 - 0.93i
7	-2.05 - 0.15i	24	-0.07 + 0.06i	41	0.26 - 0.65i
8	0.97 - 0.08i	25	0.06 + 0.00i	42	0.97 + 0.09i
9	0.26 + 0.65i	26	-0.07 - 0.06i	43	-2.06 + 0.15i
10	0.06 + 0.93i	27	0.04 - 0.09i	44	-1.42 + 1.20i
11	0.07 + 0.14i	28	0.01 - 0.20i	45	-1.26 + 0.20i
12	-0.41 - 0.15i	29	0.02 + 0.00i	46	0.04 + 1.42i
13	0.11 - 0.20i	30	0.09 + 0.08i	47	-0.15 - 0.96i
14	0.07 - 0.11i	31	-0.11 + 0.16i	48	1.18 + 6.30i
15	0.20 + 0.09i	32	-0.20 + 0.09i	49	2.58 + 2.72i
16	0.12 + 0.20i	33	0.02 - 0.23i		

After calculating the amplitudes for each frequency, I then plotted the data onto an amplitude vs. frequency graph to visualize the data collected (Figure 7). Interestingly enough, I found that the graph appeared to be symmetrical, and upon further investigation, this is because the spectrum created by the discrete fourier transform is a double-sided frequency graph (Cerna, 2023).

Figure 7: Amplitude vs. Frequency graph



Additionally, according to the sampling theorem (Bourke, 1993), frequencies above the nyquist frequency (the highest sampled frequency divided by 2; same as the median frequency) cannot actually be sampled. Thus, in order to make this a single-sided frequency plot, the amplitudes of all frequencies above the nyquist frequency (n = 24) will be removed, and the remaining frequencies' amplitudes will be doubled. Furthermore, since 50 total samples were taken, all magnitudes must be divided by the total number of samples taken in order to average it out.

For example, for the n = 1 frequency (261 Hz), the amplitude obtained from earlier was 3.76 dB, and the resulting amplitude after considering the nyquist limit and averaging out amongst the whole data set gives: $3.76 \cdot 2 \div 50 = 0.15dB$. This seems to be a more reasonable amplitude, as the highest maximum point on our original soundwave only reached approximately 0.5dB, and a coefficient of 3.76 would simply be too large.

Thus, the final cosine wave for this frequency is: $f(x) = 0.15cos(261(x + 5.47))$.
I then repeated this calculation for all of the magnitudes up to and including the n = 24th frequency, since all other frequencies past this one could not be sampled.

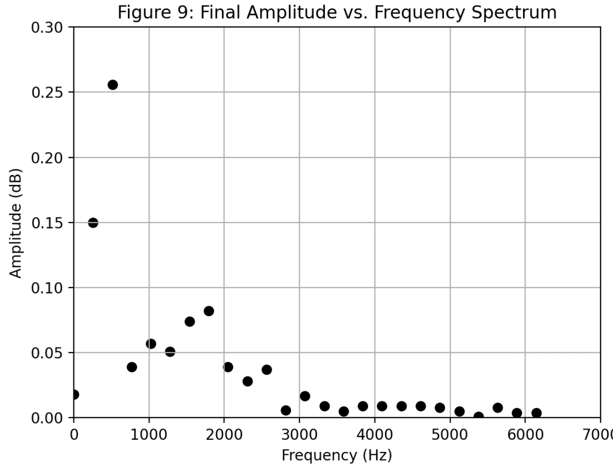
After recording all the resulting amplitudes, the data is shown as follows (Figure 8):

Figure 8: Final Magnitudes of All Sampled Frequencies

Nth frequency (Hz, x261)	Resulting Amplitude (dB)	Nth frequency (Hz, x261)	Resulting Amplitude (dB)	Nth frequency (Hz, x261)	Resulting Amplitude (dB)
0	0.018	9	0.028	18	0.008
1	0.150	10	0.006	19	0.005
2	0.256	11	0.017	20	0.001
3	0.039	12	0.009	21	0.008
4	0.057	13	0.005	22	0.004
5	0.051	14	0.009	23	0.004
6	0.074	15	0.009	24	0.004
7	0.082	16	0.009		
8	0.039	17	0.009		

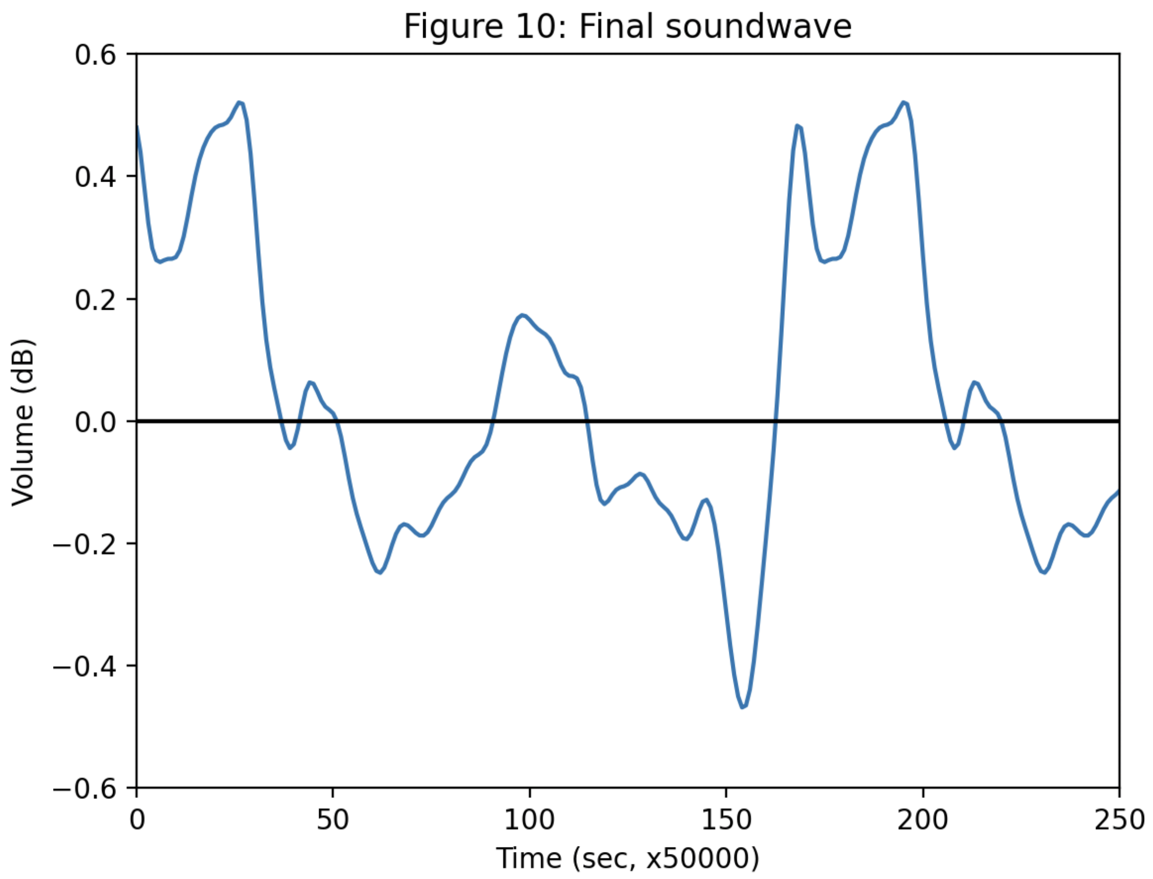
After plotting this data onto a new amplitude vs. frequency spectrum, the following graph can be obtained (Figure 9). When observing this graph, I noticed that as the frequency approached 4000 Hz, the volume almost approached zero decibels (no noise). This makes sense, because the highest frequency note on the piano is 4186 Hz (Ellinger, 2012) which is equivalent to

Figure 9: Final Amplitude vs. Frequency Spectrum



C_8. Since there should (ideally) only be interference from the piano itself, it should be expected that frequencies higher than the highest note on the piano would have a volume of zero. This is likely not guaranteed to be the case, since there were other background noises and objects in the room causing interference (further discussed in the conclusion).

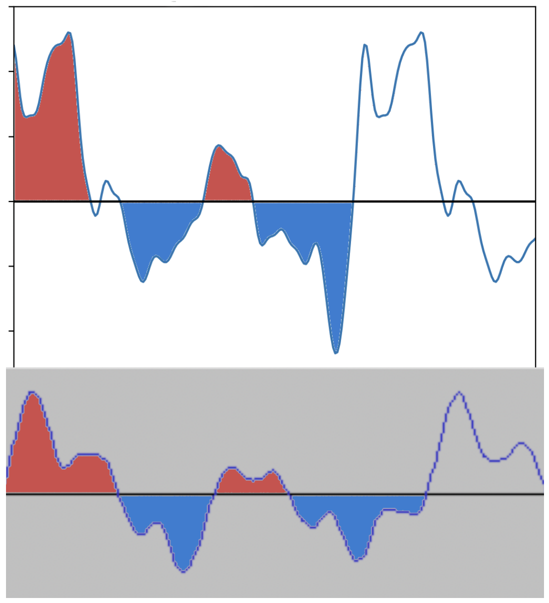
Lastly, after adding all the resulting cosine functions together, the final soundwave that I was able to model looked something like this (Figure 10):



Therefore, the final soundwave function is: f(x) = 0.018 + 0.15cos(261[x + -0.81]) + 0.256cos(522[x - 1.382]) + 0.039cos(783[x - 1.401]) + 0.057cos(1044[x - 1.544]) + 0.051cos(1305[x + 0.164]) + 0.074cos(1566[x + 0.703]) + 0.082cos(1827[x + 0.075]) + 0.039cos(2088[x - 0.079]) + 0.028cos(2349[x + 1.197]) + 0.037cos(2610[x + 1.507]) + 0.006cos(2871[x + 1.111]) + 0.017cos(3132[x + 0.364]) + 0.009cos(3393[x - 1.069]) + 0.005cos(3654[x - 1.024]) + 0.009cos(3915[x + 0.412]) + 0.009cos(4176[x + 1.03]) + 0.009cos(4437[x + 1.48]) + 0.009cos(4698[x + 0.454]) + 0.008cos(4959[x + 0.953]) + 0.005cos(5220[x - 0.711]) + 0.001cos(5481[x + 0.3]) + 0.008cos(5742[x + 1.512]) + 0.004cos(6003[x + 1.149]) + 0.004cos(6264[x - 0.65]).

Conclusion

Figure 11: Comparing the two soundwaves



Whilst the final soundwave I obtained from the discrete fourier transform was not a perfect match of my original soundwave, the model is still fairly accurate as the general behavior of the model function mostly matches the original soundwave (Figure 11). For example, both functions have 2 positive and 2 negative regions, with the first positive region being much larger than the second, and both sounds peak at approximately 0.5 dB.

Additionally, I wrote code to play the sound back of the soundwave I created, and when compared to the original audio I sampled, it sounded pretty close. Overall I would say that this project was successful as I was able to create a somewhat accurate model for a piano soundwave using mathematics that sounded pretty close. However, that is not to say that there are no limitations, as there definitely were a few.

Firstly, the discrete fourier transform is only able to sample integer multiples of the fundamental frequency; I assumed that the only interference from the soundwave was from the harmonics (the other notes on the piano that were integer multiples of the fundamental frequency), although this is realistically not the case. As mentioned before, there were other noises coming from my background, such as my washing machine or interference from other objects in the room. To fix this, I could record the sound wave in a sound-proof room that only has a piano in it, although this is highly impractical.

Secondly, the sampling size also played a role in the accuracy of my results, as taking a larger sampling size of the soundwave would increase the accuracy of the DFT since it better represents the original soundwave as a whole. However, this is also not very practical since it would take more time to both find and compute the additional samples.

I am quite pleased with how the results turned out, and I have learned a lot about doing a fourier analysis, and using fourier transforms to obtain information about a function. I also appreciated how the DFT, and how math as a whole has very unique applications in real-life, as the DFT can be used to manipulate audio (similar to what I did here), which is something that I did not initially expect. Furthermore, I am glad to have been able to connect technology with this topic by coding software that helped me process data.

I would like to thank and acknowledge my mentor for guiding me throughout this process, and I would like to credit all of the sources I used to research for this project.

Appendix:

This is the following code I wrote to perform the DFT, graph the data, and create the audio file containing the newly modeled soundwave. It should be noted that I used the SciPy, matplotlib, and numpy python modules to create this code, and must be imported for the code to work:

```
# Date: August 21st, 2023
# Purpose: compute discrete fourier series given data points

# Importing libraries
import matplotlib.pyplot as plt
import numpy as np
import math
from math import sin
from math import cos
from numpy import arctan
from scipy.io.wavfile import write

# Initializing all the variables
samples = [0.00, 0.10, 0.30, 0.50, 0.60, 0.50, 0.35, 0.16, 0.15, 0.22, 0.25, \
        0.25, 0.20, 0.11, -0.01, -0.12, -0.22, -0.20, -0.15, -0.20, -0.30, \
        -0.45, -0.40, -0.26, -0.10, 0.02, 0.13, 0.15, 0.11, 0.08, 0.10, \
        0.13, 0.12, 0.02, -0.07, -0.14, -0.18, -0.13, -0.09, -0.17, -0.27, \
        -0.47, -0.45, -0.14, -0.08, -0.07, -0.09, -0.10, -0.10, -0.05]

point = 0
sample_rate = 261
duration = 5
while point < 50:
    plt.scatter(point / (50 * 261), samples[point], color = "black")
    point += 1
plt.xlim(0, 0.004)
plt.ylim(-0.75, 0.75)
plt.grid()
plt.show()
k = 0
n = 0
N = 50
# "a" represents sum of real number values
a = 0
# "b" represents sum of imaginary number values
b = 0
# "m" represents magnitude of sample size
m = 0
# "deg" represents phase shift of the sample
deg = 0
```

```
# Computing/calculating all values
while k <= (N - 1):
    while n <= (N - 1):
        a += (samples[n] * cos(-2 * 3.1415 * k * n / 50))
        b += (samples[n] * sin(-2 * 3.1415 * k * n / 50))
        n += 1

    # Calculating phase shift values
    if a != 0:
        deg = arctan(b/a)
    else:
        deg = 0
    m = math.sqrt(a**2 + b**2)
    a = round(a, 3)
    b = round(b, 3)
    m = round(m, 3)
    deg = round(deg, 3)
    magnitudes.append(m)
    phase_shifts.append(deg)
    print("Index: " + str(k) + ", Real: " + str(a) + \
          ", Imaginary: " + str(b) + ", Magnitude: " + str(m) + \
          ", Phase shift: " + str(deg))
    print(format(a, ".2f") + " + " + format(b, ".2f") + "i" + "\n")
    n = 0
    a = 0
    b = 0
    m = 0
    deg = 0
    k += 1

print("Magnitudes: " + str(magnitudes) + "\n")
print("Phase Shifts: " + str(phase_shifts) + "\n")

# Graphing initial amplitude vs. frequency graph
i = 0
while i <= len(magnitudes) - 1:
    plt.scatter(i * 261, magnitudes[i], color = "black")
    i += 1
plt.xlim(0, 13000)
plt.ylim(0, 5)
plt.grid()
plt.show()
sample_rate = 44100
duration = 5
```

```
def generate_cosine_wave(freq, sample_rate, duration, shift):
    x = np.linspace(0, duration, sample_rate * duration, endpoint = False)
    frequencies = x * freq
    print(frequencies)
    y = np.cos(frequencies * (2 * np.pi) + shift)
    return x, y

phase_shifts = [-0.0, -0.81, -1.382, -1.401, -1.544, 0.164, 0.703, 0.075, \
                -0.079, 1.197, 1.507, 1.111, 0.364, -1.069, -1.024, 0.412, \
                1.03, 1.48, 0.454, 0.953, -0.711, 0.3, 1.512, 1.149, -0.65, \
                -0.0, 0.656, -1.144, -1.508, -0.26, 0.72, -0.948, -0.448, \
                -1.473, -1.023, -0.405, 1.035, 1.077, -0.358, -1.108, -1.502, \
                -1.191, 0.088, -0.072, -0.7, -0.161, 1.545, 1.411, 1.385, 0.812]

single_side_magnitudes = [0.018, 0.15, 0.256, 0.039, 0.057, 0.051, 0.074, \
                          0.082, 0.039, 0.028, 0.037, 0.006, 0.017, 0.009, \
                          0.005, 0.009, 0.009, 0.009, 0.009, 0.008, 0.005, \
                          0.001, 0.008, 0.004, 0.004]

_, sound = generate_cosine_wave(0, sample_rate, duration, phase_shifts[0])
_, add_sound = generate_cosine_wave(0, sample_rate, duration, phase_shifts[0])
sound = sound * single_side_magnitudes[0]
i = 1
while i < 10: #len(single_side_magnitudes) - 1:
    _, add_sound = generate_cosine_wave(261 * i, sample_rate, duration, phase_shifts[i])
    add_sound = add_sound * single_side_magnitudes[i]
    sound += add_sound
    i += 1

print(len(single_side_magnitudes))

plt.plot(sound)
plt.axhline(y=0, color = "black", linestyle = '-')
plt.title("Figure 10: Final soundwave")
plt.xlabel("Time (sec, x50000)")
plt.ylabel("Volume (dB)")
plt.xlim(0, 250)
plt.ylim(-0.6, 0.6)
plt.show()

tone = np.int16((sound / sound.max()) * 32767)
plt.plot(tone[:1000])
plt.show()
write("soundwave.wav", sample_rate, tone)
```

Works Cited

Bourke, P. (1993, June). DFT (Discrete Fourier Transform) FFT (Fast Fourier Transform). Fast Fourier Transform. https://paulbourke.net/miscellaneous/dft/#:~:text=fs%2F2%20is%20called%20the,when%20digitising%20a%20continuous%20signal.&amp;text=Normally%20the%20signal%20to%20be,to%20remove%20higher%20frequency%20components.

Cerna, M., &amp; Harvey, A. F. (2023). The fundamentals of FFT-based signal analysis and measurement. San Jose State University. https://www.sjsu.edu/people/burford.furman/docs/me120/FFT_tutorial_NI.pdf

Cheever, E. (2022). Introduction to the Fourier transform. Linear Physical Systems - Erik Cheever. https://lpsa.swarthmore.edu/Fourier/Xforms/FXformIntro.html

Ellinger, J. (2012, Spring). Unit 1 Sound Basics - Reading Assignment. MUSC 101 Unit 1 Sound Basics. https://people.carleton.edu/~jellinge/m101s12/Pages/01/01SoundBasics.html

McFee, B. (2022). 5.5. the discrete fourier transform. Digital Signals Theory. https://brianmcfee.net/dstbook-site/content/ch05-fourier/DFT.html

Qef. (2008). Vibration and standing waves in a string, The fundamental and the first six overtones. Harmonic partials on strings - Fundamental Frequency. Wikipedia. Retrieved August 23, 2023, from https://commons.wikimedia.org/wiki/File:Harmonic_partials_on_strings.svg.

Roberts, S. (2023, January 27). Lecture 7 -the discrete fourier transform - university of Oxford. Information Engineering. https://www.robots.ox.ac.uk/~sjrob/Teaching/SP/l7.pdf

Suits, B. H. (1998). Tuning. Frequencies of Musical Notes, A4 = 440 Hz. https://pages.mtu.edu/~suits/notefreqs.html

Svirin, A. (2023). Calculus. Complex Form of Fourier Series. https://math24.net/complex-form-fourier-series.html

University of Manitoba Press. (n.d.). Sound Waves. Sound waves. https://home.cc.umanitoba.ca/~krussll/138/sec4/acoust1.htm

Weisstein, E. (2023). Fourier series. Wolfram Mathworld. https://mathworld.wolfram.com/FourierSeries.html#:~:text=A%20Fourier%20series%20is%20an,the%20sine%20and%20cosine%20functions.

Wolfe, J. (2010, October). Strings, standing waves and harmonics. https://newt.phys.unsw.edu.au/jw/strings.html

Yamaha Corporation. (n.d.). The structure of the piano the sound-producing mechanism. The Structure of the Piano:The Sound-Producing Mechanism - Musical Instrument Guide - Yamaha Corporation. https://www.yamaha.com/en/musical_instrument_guide/piano/mechanism/mechanism002.html#:~:text=One%20end%20of%20the%20strings,soundboard%2C%20vibrates%20to%20produce%20sound