Comparison of FFT to Geometric Single-Frequency Identifier Algorithm for Dominant Frequency Estimation Over Varying Noise Levels

At what level of noise does the Fast Fourier Transform (FFT) algorithm outperform a geometric Single-Frequency Identifier algorithm for identifying the dominant frequency of discrete wave data, making it worth using despite its higher time cost?

Computer Science

IB Extended Essay

Word Count: 3985

May 2025 Examination Session

Abstract

Some applications of the FFT (Fast Fourier Transform) exist for the purpose of finding a dominant frequency, or primary signal which may be diluted by noise. This investigation aims to suggest an alternative algorithm which may perform with a lower time-complexity than the FFT and remain accurate in specific circumstances: namely those with a fairly pure signal (with little noise) and a dominant frequency which is significantly less than the sampling rate. It is approximated that a maximum amplitude of noise (relative to the dominant signal's amplitude) is 2% for which the custom algorithm may outperform the FFT; a limit of roughly 7-10% of the sampling rate is also set on the percentage of the dominant frequency for which the SFI maintains that accuracy.

Table of Contents

I. Introduction

It is a frequent question in many fields how to optimize the transformation of wave information from a time-domain to a frequency-domain [1]–[3]. In other words, what is the most effective way to identify and analyze a composite wave's constituent frequencies and amplitudes? In computer science, wave information typically consists of discrete points which have been sampled a constant time interval apart [1], [3]. For sound or other pressure waves, these values represent the displacement of a sensor (such as a microphone) at a given time. For digital waves, they represent voltage.

In some applications, the amplitudes of the constituent frequencies are used to identify the "dominant frequency" of a wave [4]–[7]. The dominant frequency is defined here as the frequency with the highest amplitude. In many instances, *only* the dominant frequency is important and other frequencies can be classified as "noise" [4]-[9]. It is this situation upon which this investigation is based.

The most well-known algorithm for transforming wave data from a time-domain to a frequency-domain is the Fast Fourier Transform, or FFT [1]-[3], [7], [10]. Fig. 1 illustrates this. This algorithm has a theoretical time-complexity of $O(NlogN)$ [11]. It calculates the amplitude of a discrete number of frequency "bins", each representing a small range of frequencies which could be contained in the wave. The dominant frequency is estimated as the bin with the largest amplitude.

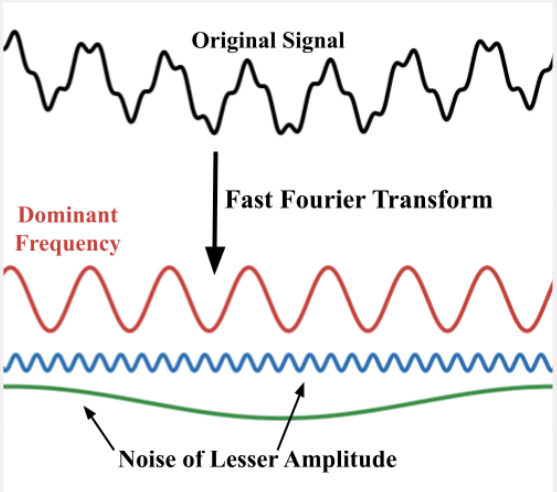

Fig. 1. The signal that must be processed is the combination of the dominant frequency and noise. The FFT "unmixes" them and identifies their respective amplitudes.

Figure created by author.

This raises the question: how much noise must the signal contain to make a transformation necessary for accurate results? After all, the dominant frequency of a signal which contains no noise could be more easily found using 18th-century calculus [12]-[13]. There are many instances of research that utilize this sort of "geometric" approach in the time-domain (requiring no transformation) if the amount of noise is low enough [5]. My investigation proposes a new method, which likewise uses a geometric approach; I will refer to this algorithm as a "Single-Frequency Identifier," or SFI. As the name suggests, this algorithm assumes that only one frequency is present. As the amount of noise present in a wave increases, the accuracy of the predictions of my algorithm should decrease because of this assumption.

Most existing scholarship would suggest that the FFT is optimized enough to be preferable in most, if not all, scenarios. Similarly, several other alternatives are used such as a Discrete Cosine Transform [14], the Stockwell Transform [9], power spectral density [7], the Prony method, and wavelet transforms [7]. Like the FFT, they rely on mathematical transformations, which ultimately provide a lot of redundant data if the vast majority of frequencies have amplitudes near zero. Algorithms which employ a geometric approach are also used, albeit more rarely.

In what instances, then, would it be appropriate to use a geometric approach vs. the FFT? More specifically, **at what level of noise does the Fast Fourier Transform (FFT) algorithm outperform a geometric single-frequency identifier algorithm for identifying the dominant frequency of discrete wave data, making it worth using despite its higher time cost?** A literature review is performed to understand application and theory behind the FFT. The contrasting strategy for the SFI is then designed. Random wave data are generated to test both algorithms. The amplitude of the noise is used as an independent variable to experimentally determine at what point the SFI loses accuracy when compared to the FFT (which should not be

significantly affected by the noise). Results are shown and discussed, with the conclusion that the SFI maintains higher or similar accuracy to the FFT under the condition that the signal contains no more than 2% noise and that the dominant frequency is no more than 7-10% of the sampling rate.

II. Literature Review

A. Background

Fourier Transforms were first published by French mathematician Joseph Fourier in 1822 [15]. The Fast Fourier Transform was first established as a monument in the computer science world in 1965 by James Cooley and John Tukey. Their paper, "An Algorithm for the Machine Calculation of Complex Fourier Series" [11], is the seminal work on the topic. This paper will use the original Cooley-Tukey method as its standard, but it is worth noting that this is hardly representative of the state of the FFT today [16], [17].

The FFT is a crucial element of modern communication systems [10]; scientific computing [2], [3], [10]; diagnostics [18], [19]; tomography, remote sensing, seismology, and artificial intelligence [2]; and the medical field [4]-[6], [8]. In the medical and scientific fields, specifically, I found considerable research suggesting that the FFT was primarily being used to find a dominant frequency [4]-[9], [14], [18]. Geometric alternatives to the FFT were not always considered.

B. Continuous Fourier Transforms

The following explanation is based off of approaches outlined in [1], [3], [15].

Let there exist a composite wave function $x(t)$ that is defined by the combination of two or more sinusoids of unknown frequency and amplitude. A general (continuous) Fourier Transform decomposes this composite function into the magnitudes of its constituent frequencies.

Defined for continuous situations, $x(t)$ may be expressed as an infinite sum of sine functions with different frequencies. This is done by transforming $x(t)$ onto the complex plane and turning wave values into magnitudes (distances from the origin) of a polar coordinate system to create a

new function *X(f)* where *f* is frequency and *X(f)* is that frequency's magnitude. As the input frequency *f* is changed, *x(t)* itself is "wrapped" around the origin of the complex plane with different frequencies. The integral of the transformed function will reflect the magnitude of a corresponding frequency in the wave (Fig. 2).

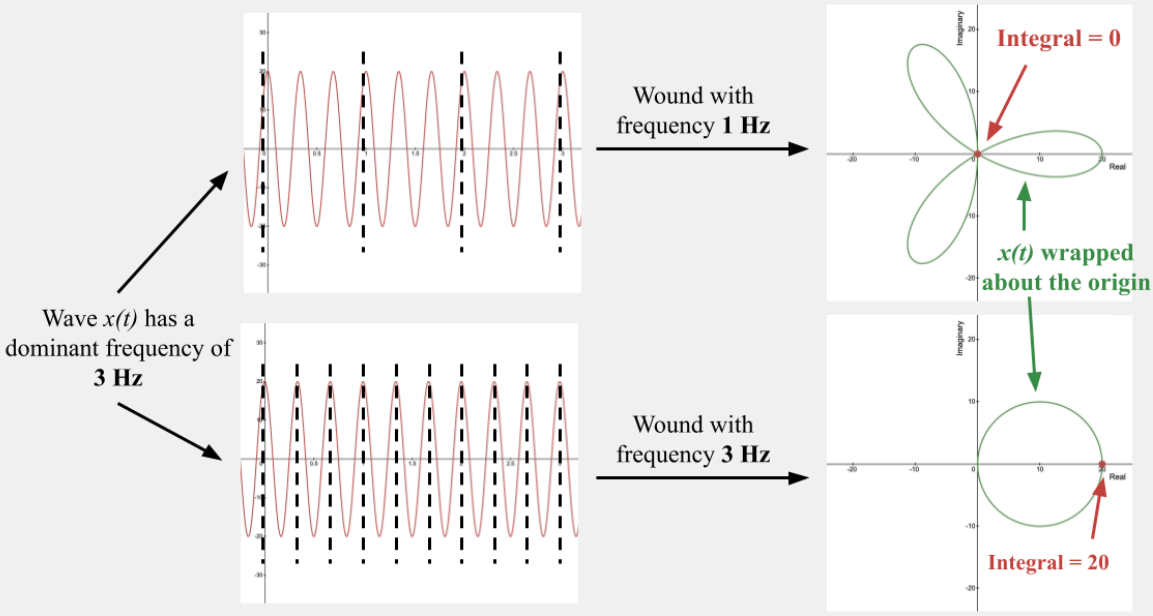

Fig. 2. Most input frequencies will cause the integral to be near zero. When the input frequency is equal to one of the frequencies contained in the wave (in this case 3 Hz), the peaks and valleys align, causing an integral corresponding to the amplitude of that frequency.

Figure created by the author.

If *X(f)* is the amplitude of frequency *f*, then Fourier's transform is illustrated using the following equation:

$$X(f) = \int x(t) e^{-2i\pi f t} dt \qquad (1)$$

Notice that the original function *x(t)* is being multiplied by $e^{i\theta}$ such that θ is a variable term. This is derived from Euler's formula:

$$e^{i\theta} = cos(\theta) + i\, sin(\theta) \qquad (2)$$

In other words, the original function *x(t)* is multiplied by a combination of a real sine and an imaginary cosine, essentially wrapping it around the origin of the complex plane. When the weighted integral of the wrapped function is found, it represents the amplitude of the input frequency *f* (Fig. 3). This will be applied to discrete data in the next section.

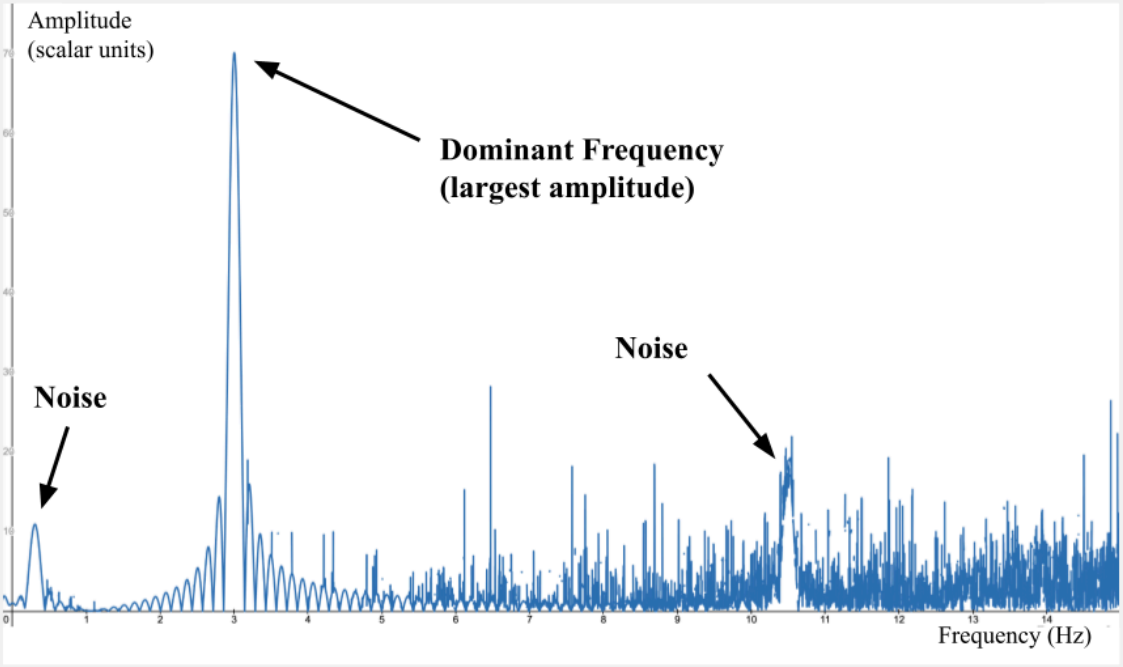

Fig. 3. By putting the wave into a frequency domain, a Fourier Transform makes it trivial to find the dominant frequency.

Figure created by the author.

C. Discrete and Fast Fourier Series

The following explanation is based off of approaches outlined in [1], [3], [10], [11].

The discrete definition of a Fourier Transform is based off the continuous version (1). The original data set, designated array x, consists of digital values which were sampled a known Δt apart. For sound or other pressure waves, x_i would be the displacement of a sensor (such as a microphone) at a given time. For an electrical wave, it would be voltage.

The Discrete Fourier Transform (DFT) transforms array x into array X of the same length N:

$$X_k = \sum_{n=0}^{N-1} x_n \cdot e^{-2\pi ikn/N} \tag{3}$$

Each value X_k represents the amplitude of a "bin"—or range—of frequencies. The frequency bins have a constant width of Δf which can be easily calculated from N and Δt.

The second half of the values in array X mirror the first half and are therefore useless. This is because the maximum frequency (called the Nyquist frequency) in Hz which can be deduced from the sample is ½ the number of values sampled in one second. Hence, the precision and range of which the frequencies can be estimated is determined by the sample rate and total sample time. This will become significant as the FFT is later compared to the SFI.

Instead of using two nested "for" loops in order to calculate the series in (3) N times for every value of X_k (so really N^2 times, since there are N values of X_k), the DFT could be thought of as a matrix multiplication in which a "fundamental frequency" (not to be confused with the dominant frequency) is defined as $\omega_N = e^{-2\pi i/N}$. This ω_N can then be raised to the power of kn to get the complex unit vector from (3).

$$e^{-2\pi ikn/N} = [e^{-2\pi i/N}]^{kn} = \omega_N^{kn} \tag{4}$$

This is used to establish a "DFT matrix" of size NxN in which each row is a different integer value of k, each column is a different integer value of n, and all values are expressed in terms of the constant ω_N. When this matrix is right-multiplied by a vector containing the values of x, a new vector containing the values of X is found.

$$\begin{bmatrix} X_0 \\ X_1 \\ X_2 \\ \vdots \\ X_n \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 & \cdot\cdot & 1 \\ 1 & \omega_N & \omega_N^2 & \cdot\cdot & \omega_N^{(N-1)} \\ 1 & \omega_N^2 & \omega_N^4 & \cdot\cdot & \omega_N^{2(N-1)} \\ \vdots & \vdots & \vdots & & \vdots \\ 1 & \omega_N^{(N-1)} & \omega_N^{2(N-1)} & \cdot\cdot & \omega_N^{(N-1)^2} \end{bmatrix} \begin{bmatrix} x_0 \\ x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} \quad (5)$$

Please note that (5) is exactly equivalent to (3) by the rules of matrix multiplication [20].

When computed this way, the DFT is known to have a quadratic time-complexity of $O(N^2)$ because it involves doing N multiplications for each of N rows. This algorithm can be optimized to instead have a time complexity of $O(NlogN)$ by recursively splitting the data set into two smaller ones, and then combining their results using a miniature DFT matrix multiplication [10]. This optimization is known as the Fast Fourier Transform (FFT), and it takes advantage of the symmetries in the DFT matrix (5) to reduce redundant calculations. Section (IV) will clarify the role of the FFT for the scope of this investigation.

D. Single-Frequency Identifier

When I say that I am using a "geometric" approach in my wave analysis, I am suggesting that the wave is being analyzed in the original time-domain (as opposed to being mathematically transformed to another domain). A time-domain approach is by not unique to this investigation. It was used before the invention of the FFT because the unoptimized DFT was so costly in processing time that it wasn't always a viable method of wave analysis [21]-[22]. Geometric approaches are still used to examine wave properties today in certain cases [5], but have declined in popularity with the improvement of transform algorithms such as the FFT. Many papers have

been published in which the authors use one method or another (geometric or transform) to obtain a dominant frequency estimation without having ample empirical data to suggest it was the best decision for their application; by proposing and testing a geometric method, this investigation fundamentally aims to outline the conditions under which one might choose not to use an algorithm involving a wave transformation.

While the theory presented in (III) as my "Single-Frequency Identifier" is a custom solution, it is based off of the fundamental calculus groundwork put in place simultaneously by Gottfried Leibniz [19] and Isaac Newton [18] as well as general wave phenomena introduced by Leonhard Euler throughout his publications.

III. Theory and Calculation

The SFI is a basic algorithm of my design that is intended to identify a dominant frequency through geometric analysis of discrete wave data. The objective is to outperform the FFT in speed by offering a solution which has a linear time-complexity of *O(N)* instead of *O(NlogN)* so that it is slightly less impacted by large data samples.

Assume a continuous wave *x(t)* contains only one frequency *f*. It can now be defined like so:

$$x(t) = A\,sin(2\pi ft + \phi) \tag{6}$$

where A is the amplitude and ϕ is a phase angle. The derivative of this function is defined like so:

$$x'(t) = 2\pi Af\,cos(2\pi ft + \phi) \tag{7}$$

The SFI algorithm identifies all places where *x(t)* = 0. At these locations, $sin(2\pi ft + \phi) = 0$ which implies that $cos(2\pi ft + \phi) = 1$. Hence the slope *m* of *x(t) at an x-intercept* is equal to the following:

$$m = x'(t) = 2\pi Af * 1 \tag{8}$$

Solving for frequency *f*:

$$f = \frac{m}{2\pi A} \tag{9}$$

As section (IV) will demonstrate, the SFI estimates the amplitude of the input wave and the average slope of a crossing in order to find the frequency.

Increasing the number of data points *N* will cause more crossings, which I predict should increase the calculations that need to be performed linearly (on average). This algorithm should therefore have a linear time-complexity of *O(N)*.

IV. Testing Methodology

A. Independent Variable

For every trial, a wave is generated which is fed through both algorithms to compare their results. The independent variable is the amplitude of their noise, which can be thought of as the "impurity" of the wave.

$$\text{"Impurity"} = \frac{A_N}{A_S} = \frac{A_N}{1} = A_N \tag{10}$$

where $A_S=1$ is the amplitude of the dominant frequency and A_N is the sum of the amplitudes of all other frequencies[1].

B. Dependent Variable

For both algorithms, accuracy will be calculated by subtracting the algorithm's estimation of the dominant frequency from the actual dominant frequency and taking the absolute value. The time the algorithm takes to perform the estimation will also be tracked and factored into conclusions.

C. Control Variables and Generation Parameters

For simplicity, all trials will use an amplitude of $A_N = 1.0$ scalar units for the dominant frequency. Wave data will be generated for a 1-second interval with a sampling rate of 512 Hz. The dominant frequency itself will be a uniformly-selected random floating point number between 1 Hz and 40 Hz. Initial observation suggested that frequencies significantly more than 7-10% of the sampling rate were not accurately estimated by my SFI (for reasons hypothesized later). This

[1] The "N" in A_N is referring to "noise", not to N, the sample size.

is why a 40Hz constraint is placed on the testing data; however, this limitation is important and will be acknowledged in the conclusions.

To add noise to the wave, 1-4 additional frequencies are chosen in the same range of 1-40 Hz. These frequencies are each assigned an amplitude so that they sum to the trial's target noise amplitude A_N. A_N starts at 0 and increases in increments of 0.001 until it reaches a maximum of 0.1. In the results, A_N will be expressed as a percentage. Thirty trials, with different randomly generated waves, are repeated for each value of A_N.

D. Iterative Testing

While being far from the fastest method of computation, python is used for this experiment because it is mainstream and allows for easy manipulation and display of the data. The time data presented in this paper and in the appendix are obtained from the testing of these algorithms on a 3GHz 6-Core Intel Core i5 processor.

In order to collect my data, I have written a script which systematically iterates values for the independent variable (noise amplitude or A_N) and generates waves to test both algorithms. For each A_N, thirty trials are run with different randomly-generated waves; the same waves are used for both algorithms.

The frequency estimates given by the FFT and SFI are subtracted from the actual dominant frequency to determine an error for that trial. The data for each trial are saved to a spreadsheet and later visualized. The results will be displayed here in graph format; code and numerical values can be accessed from the appendix.

F. Single-Frequency Identifier Algorithm

The basic strategy for the SFI is as follows:

```
# method takes list x
def sfi(x):
       amplitude = (max of x - min of x) / 2

       sum = 0                    # sum of slopes of crossings
       crosses = 0                # number of crossings
       length = length of x

       for all x_i except the last:

              # check for crossing with x_{i+1}
              if x_i * x_{i+1} < 0:
                     m = |x_{i+1} - x_i|
                     sum += m + (50*m^3) # see commentary below
                     crosses += 1

       average_slope = sum / crosses
       frequency = average_slope / (amplitude * 2π)

       return frequency
```

In order to later calculate an average, the SFI is adding the magnitude of the slope of each crossing to a variable called "sum". To each slope I also add the term $50 * |m|^3$. This is in place for situations where the frequency is a significant portion of the sampling rate. When calculating the slope between points, the SFI would underestimate the slope at the crossing of the x-axis because the points are actually far from the crossing (Fig. 4). This term artificially inflates it accordingly.

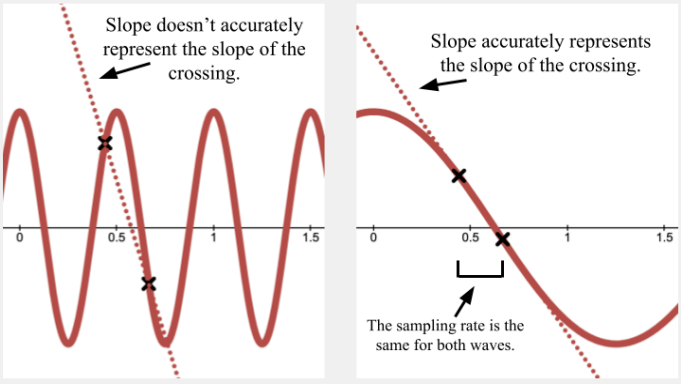

Fig. 4. When estimating the slope of a crossing, a wave with a high frequency will require artificial inflation because the discrete data points will generally be further from the x-axis.

Figure created by the author.

E. Fast Fourier Transform Algorithm

The basic strategy for the FFT is as follows [1], [10], [11]:

```
# fft method takes in list x
def fft(x):
      N = length of x

      # recursive base case
      if N <= 1: return x

      # split computation in half and call recursively
      evens = fft([x_0, x_2, x_4, ..., x_N-1])
      odds = fft([x_1, x_3, x_5, ..., x_N-2])

      # combine results using a miniature-DFT
      X = [(odds_k * e^-2iπk/N) for all k]
      X = [evens_k + X_k for all k] + [evens_k - X_k for all k]

      # ^^ here, k = 0, 1, 2, ..., N/2

      return X
```

Notice how the DFT is being performed on a list (odds) which was previously run through the same DFT. Rather than performing the full transformation for every value of X_i (and having some overlap in computed expressions), each individual expression is computed once and combined later.

V. Results

A. Observation

Initial experimentation suggested that higher frequencies caused the SFI to underestimate the frequency of a discrete sinusoid. In (IV), I hypothesized the reason for this and described my solution. Even with that solution implemented, I observe that the SFI dramatically loses accuracy once the dominant frequency exceeds 7-10% of the sampling rate.

B. Experimental Results

Following the procedure outlined in (IV), both algorithms were tested over the same data. Their error was then calculated by finding the difference between their estimation and the actual dominant frequency. As the noise was increased, results were plotted to produce Fig. 5.

The FFT is seen to have a constant maximum error of ~0.5 Hz while the signal has between 0% and 10% noise. It also appears to be fairly evenly distributed between having an error of 0 Hz

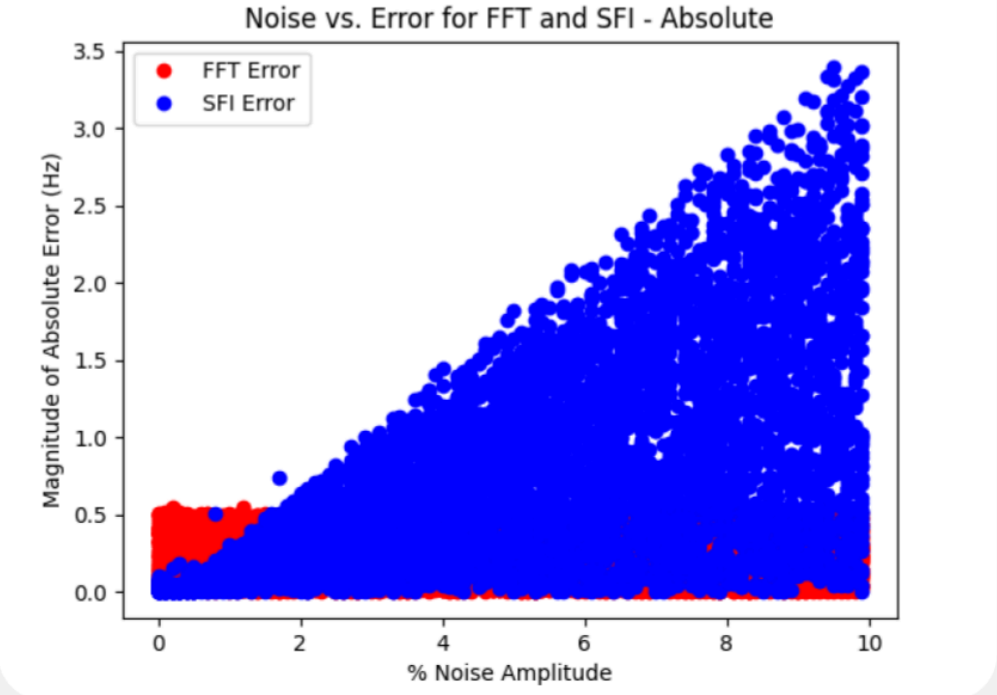

Fig. 5. Absolute error of both algorithms for all trials.

and 0.5 Hz. The SFI is seen to have a maximum error that starts at ~0.2 Hz with 0% noise and increases linearly to be ~3.5 Hz at 10% noise. The maximum error of the SFI intersects that of the FFT just before the signal contains 2% noise.

Fig. 6 shows the average error of the thirty trials for each value of A_N.

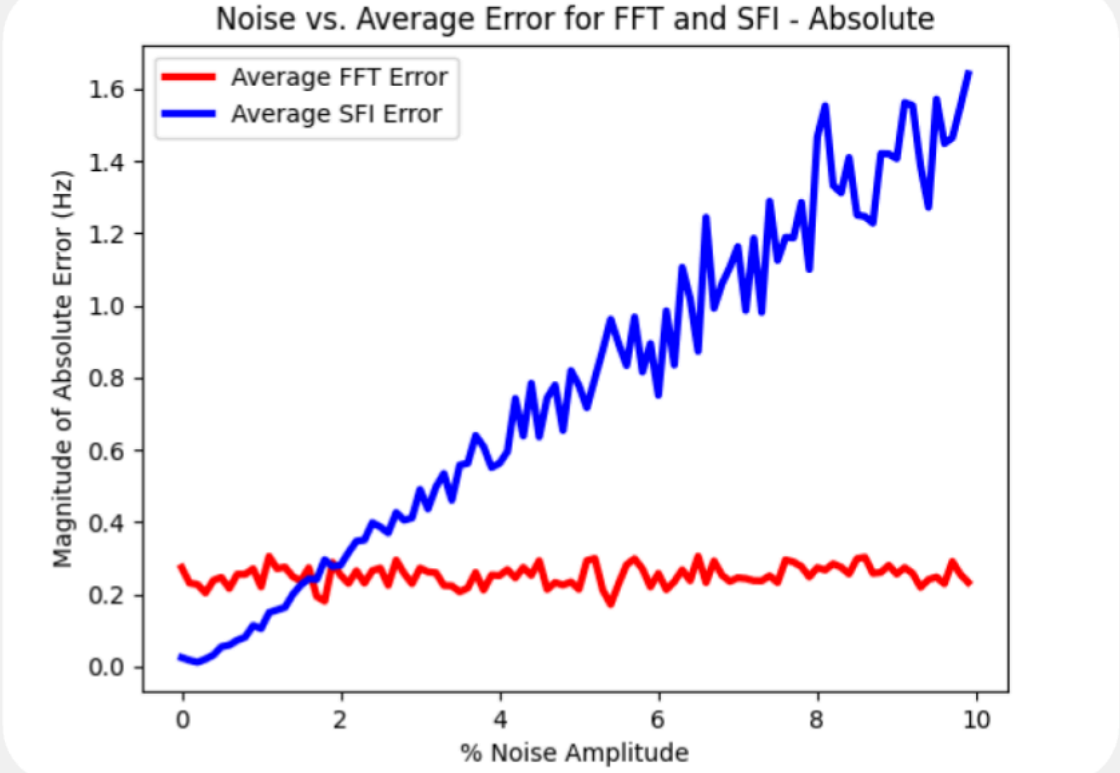

Fig. 6. Average absolute error of both algorithms.

The FFT is seen to have a constant average error of ~0.25 Hz. The SFI appears to be linearly proportional to the percentage of noise (A_N), starting with an average of ~0.0 Hz at 0% noise and increasing to 1.6 Hz at 10% noise. The average error of the SFI is less than that of the FFT until just before $A_N \approx 2\%$.

The computation time of both algorithms was recorded for all trials, producing Fig. 7.

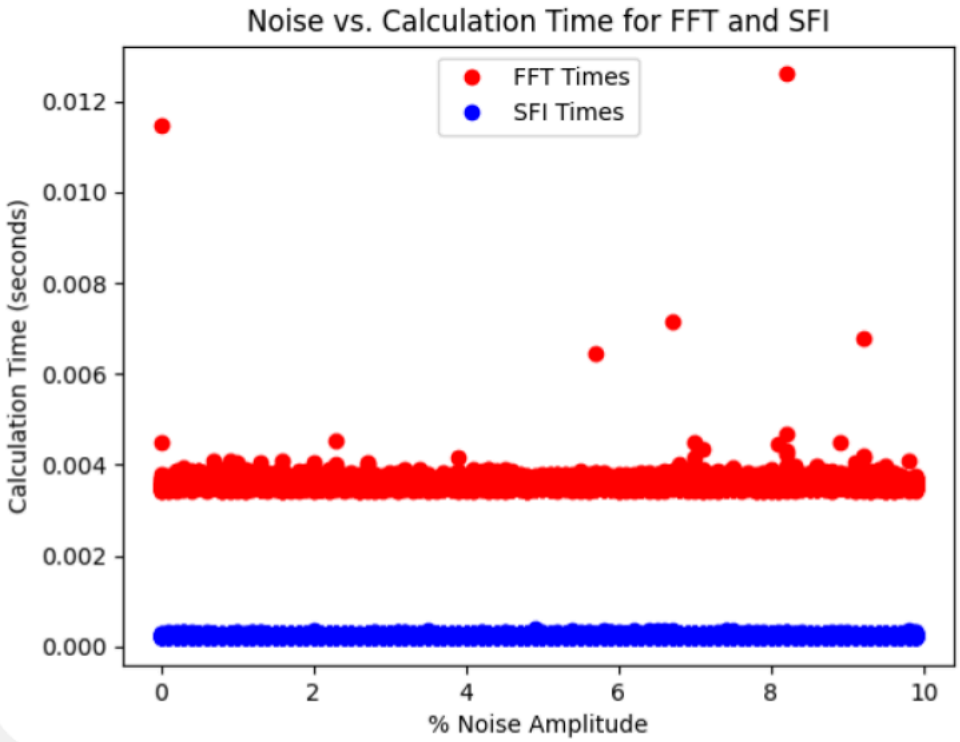

Fig. 7. Computation time for all trials.

The SFI is shown to have a fixed calculation time of significantly less than 1 millisecond. The FFT has a wider range of calculation times, usually falling at just less than 4 milliseconds. Fig. 7 demonstrates that the SFI is more time-effective than the FFT under the generation parameters established in (IV); it also reinforces an implied prediction that the algorithms' runtime is independent of the dominant frequency.

C. Time-Complexity

Theory suggests that the SFI should barely outperform the FFT in terms of time-complexity. An experiment was performed to determine if empirical data matches expectations.

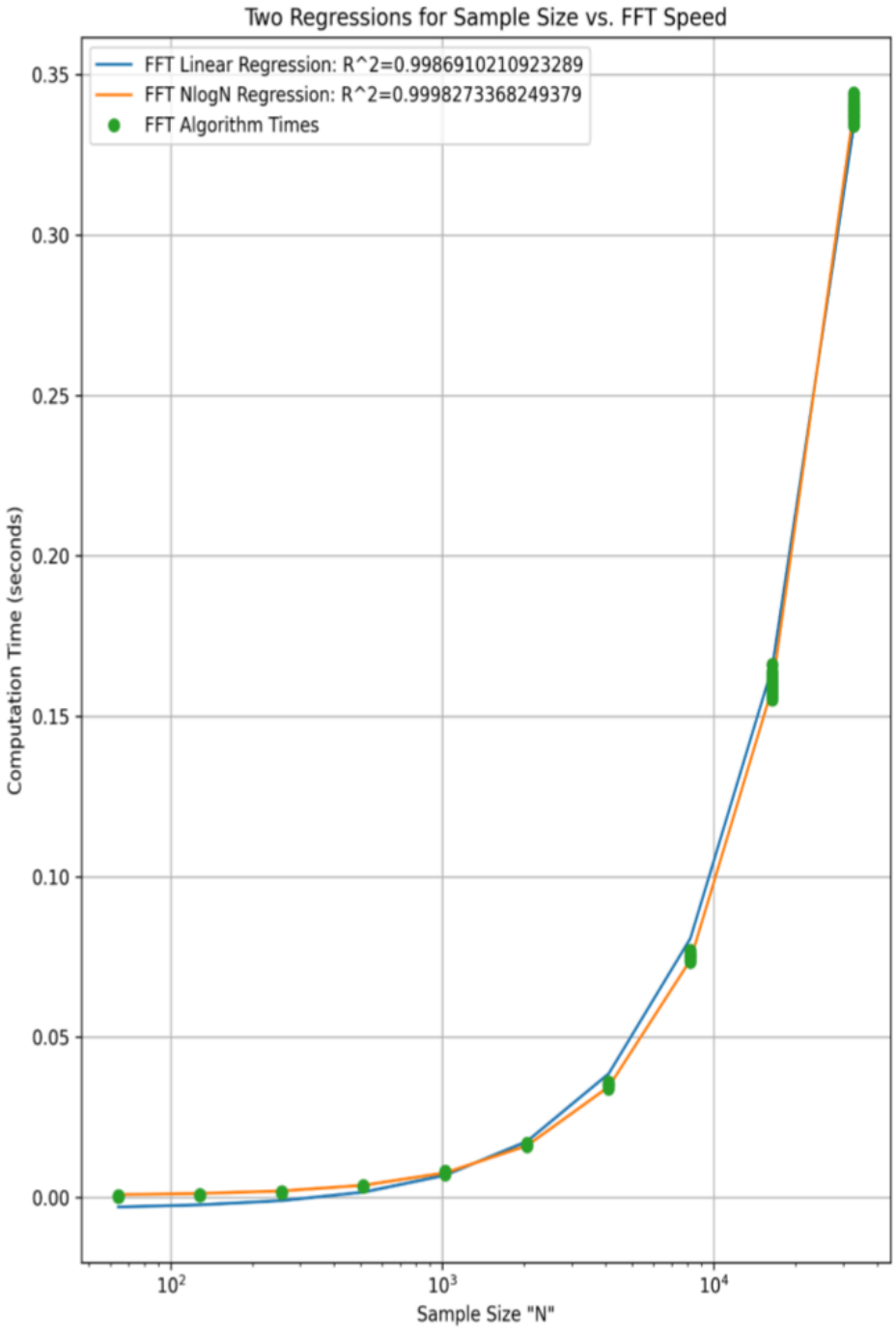

Fig. 8. An NlogN regression fits the speed of the FFT, as expected. (The regression is almost linear; it appears exponential because a logarithmic scale has been used on the x-axis.)

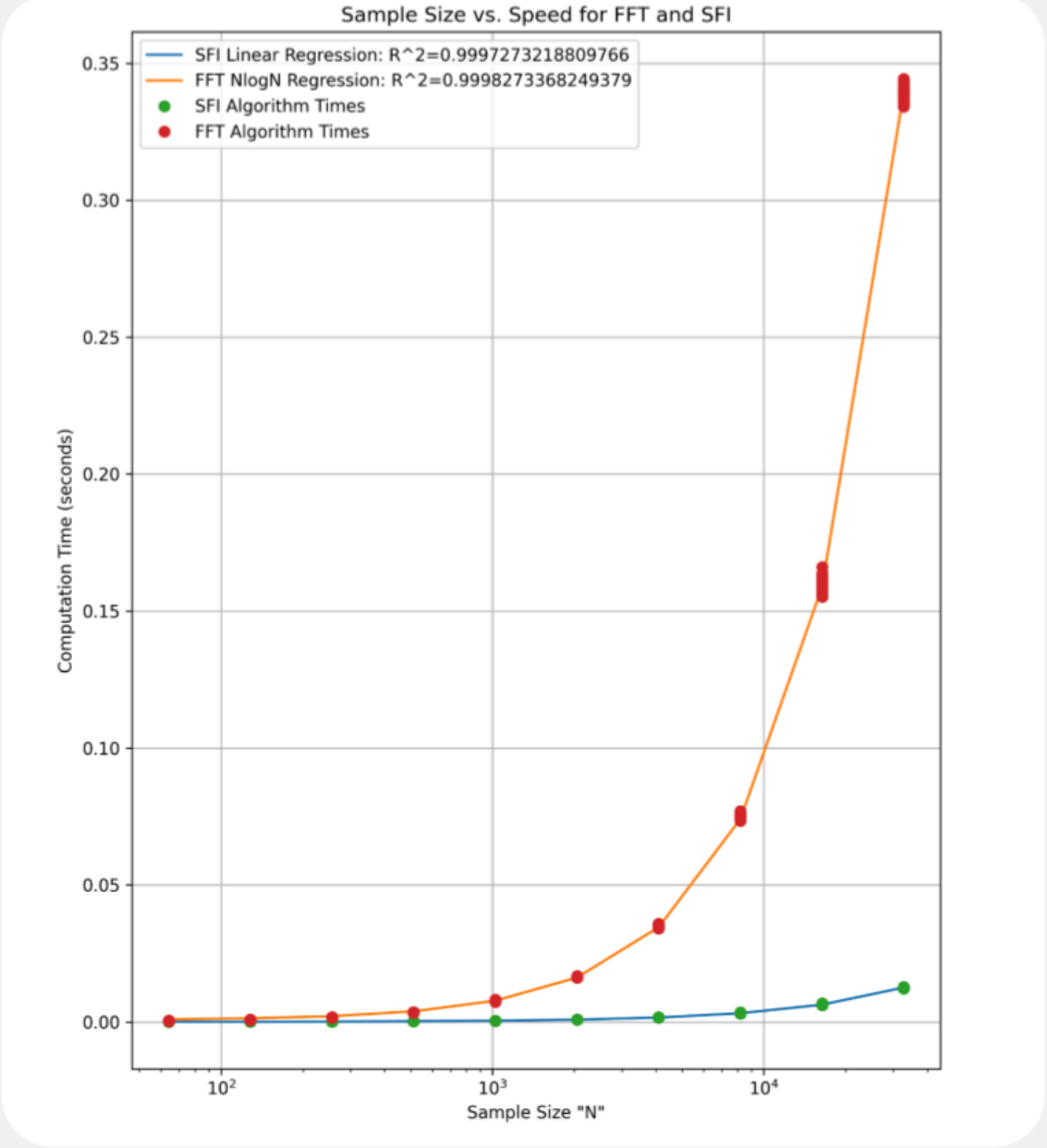

Fig. 9. The speed of the SFI is compared to the FFT as the number of data points is increased. (The SFI regression is linear; both appear exponential because a logarithmic scale has been used on the x-axis.)

As expected, the impact of changing the data size *N* on the time for the FFT to run is modeled best by an *NlogN* regression: the R^2 values in suggest the *O(NlogN)* regression is more

representative than the *O(N)* regression for the FFT (Fig. 8). As hypothesized, a linear regression fits the SFI very well. Despite being highly optimized in comparison to the DFT, the FFT is more costly than the SFI in terms of processing time and calculations as *N* increases (Fig. 9).

D. Analysis and Discussion

1) Fast Fourier Transform: Analyzing the accuracy of the FFT algorithm is trivial. Fig. 5 suggests that the algorithm consistently estimated the dominant frequency with an uncertainty of ±0.5 Hz. Fig. 6 suggests that the FFT's uncertainty is, on average, constant regardless of noise.

Both of these observations are supported by expectations. The output of the Fourier Transform is a list of bins, 1 Hz wide, each of which corresponds to a frequency with an integer value. In other words, the FFT's estimate for the a dominant frequency, when correctly calculated, will always be rounded to the nearest integer. Because the randomly generated value for the dominant frequency is a float, it can be up to 0.5 Hz away from the nearest integer. The magnitude of the error of the FFT could therefore be expected to be evenly distributed within 0 Hz - 0.5 Hz, with an average magnitude of error of 0.25 Hz. Fig. 5 and Fig. 6 support this to a high extent.

Unsurprisingly, the FFT is unaffected by noise. The transform estimates the relative amplitudes of all frequencies in the wave, then selects the one which is largest. No other frequency should be larger than the dominant frequency when the amplitude of all other frequencies totals to only 10% of that of the dominant frequency.

2) Single-Frequency Identifier: Inspection of Fig. 5 and Fig. 6 reveal that the absolute error of the SFI is related to A_N with a strong, linear, positive correlation. Both figures suggest that the SFI becomes comparable in accuracy to the FFT around $A_N \approx 2\%$.

Unlike the FFT, the SFI is not limited to the precision of a single frequency bin. Instead, its output is a single floating point number. Because of this, when noise is negligible, the SFI can obtain higher levels of precision than the FFT, resulting in generally lower error.

As expected, adding noise to the wave causes the SFI to decrease in accuracy. This is because it assumes only one frequency is present; when other frequencies become present in substantial amounts, they inevitably amplify or cancel with the original frequency at the crossings and cause estimations of the slope to become inaccurate. In addition to causing inaccurate measurements for the slopes of crossings, other frequencies likely artificially inflate the wave's amplitude, as it's measured by my algorithm using the maximum and minimum points.

E. Extension and Future Work

When considering improvements for the SFI, it is crucial to consider two elements: its estimation of the average slope of a crossing, and its estimation of the amplitude of the wave.

I have described the challenges with accurately estimating the slope of a crossing, specifically with higher dominant frequencies (which reduce the period) and with a significant amount of noise. Fixes for this would require a fair amount of testing and reexamination of the theory. However, something that could be improved quite easily is the SFI's method of estimating the amplitude of the wave. I currently use the absolute maximum and minimum values; unfortunately, these will be augmented when noise is added. A better approach might be to average relative minimum and maximum values.

Ultimately, even though it requires very specific conditions to perform more optimally than the FFT, my custom algorithm does still have some uses. In potential applications of dominant

frequency analysis, such as biomedical examination, both noise and the target frequency are very low—meaning the SFI could be both more precise and faster than current FFT methods.

VI. Conclusions

The Single-Frequency Identifier outperforms the Fast Fourier Transform in speed, and the difference increases more dramatically as the size of a data set increases. The FFT has a complexity of *O(NlogN)*, as expected. The SFI has a complexity of *O(N)*.

The Single-Frequency Identifier outperforms the Fast Fourier Transform in accuracy and precision of estimating a dominant frequency until noise exceeds ~2%, at which point the FFT begins to operate more effectively.

The Single-Frequency Identifier dramatically loses accuracy if the dominant frequency is greater than 7-10% of the sampling rate; as expected, the Fast Fourier Transform remains accurate until the dominant frequency exceeds the Nyquist limit of 50% of the sampling rate.

The SFI generally has very limited application, but with improvements I believe it could prove a valuable alternative to more complex transformations in specific scenarios. Ultimately, this investigation has achieved its goal of suggesting cases when researchers should consider a geometric approach over of a transformation for estimating a dominant frequency.

References

[1] C. S. Burrus, *Fast Fourier Transforms*. 2012.

[2] P. Alexey, P. Olga, and S. Natalia, “Evolution of One-Dimensional and Two-Dimensional Discrete Fourier Transform,” 2022 24th International Conference on Digital Signal Processing and its Applications (DSPA). IEEE, Mar. 30, 2022. doi: 10.1109/dspa53304.2022.9790768.

[3] M.-A. Delsuc and P. O’Connor, “The Fourier transform in analytical science,” Nature Reviews Methods Primers, vol. 4, no. 1, Jul. 2024, doi: 10.1038/s43586-024-00326-2.

[4] A. Cabasson, O. Meste, S. Zeemering, U. Schotten, and P. Bonizzi, “Is the Dominant Frequency Accurate Enough for Atrial Fibrillation Signals ?,” Computing in Cardiology Conference (CinC). Computing in Cardiology, Dec. 31, 2022. doi: 10.22489/cinc.2022.418.

[5] R. Dalvi, A. Suszko, and V. S. Chauhan, “Identification and annotation of multiple periodic pulse trains using dominant frequency and graph search: Applications in atrial fibrillation rotor detection,” 2016 38th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC). IEEE, Aug. 2016. doi: 10.1109/embc.2016.7591500.

[6] R. Gowtham and R. Chandrasekaran, “Analysis of Gastric Signal After Consumption of the Pre-Processed Food by Using FFT Transform,” 2023 International Conference on Sustainable Communication Networks and Application (ICSCNA). IEEE, Nov. 15, 2023. doi: 10.1109/icscna58489.2023.10370311.

[7] E. N. Tanriseven, H. Igel, and H. A. Bilgin, “Comparison of Dominant Frequency Analysis Methods for Blast Induced Ground Vibrations,” 2018 14th IEEE International Conference on Signal Processing (ICSP). IEEE, Aug. 2018. doi: 10.1109/icsp.2018.8652429.

[8] M. Shamas et al., “Estimating the dominant frequency of High Frequency Oscillations in depth-EEG signals,” 2017 Fourth International Conference on Advances in Biomedical Engineering (ICABME). IEEE, Oct. 2017. doi: 10.1109/icabme.2017.8167561.

[9] R. A. Padilla, A. P. Parrilla, A. R. Gomez, and F. J. M. Gutierrez, “Sensitivity Analysis of Dominant Frequency of Transient Recovery Voltage using Stockwell Transform,” 2021 IEEE 15th International Conference on Compatibility, Power Electronics and Power Engineering (CPE-POWERENG). IEEE, Jul. 14, 2021. doi: 10.1109/cpe-powereng50821.2021.9501225.

[10] S. Brunton. “The Fast Fourier Transform (FFT).” YouTube.com. https://www.youtube.com/watch?v=E8HeD-MUrjY (accessed October 25, 2024)

[11] J. W. Cooley and J. W. Tukey, “An Algorithm for the Machine Calculation of Complex Fourier Series,” Mathematics of Computation, vol. 19, no. 90, pp. 297–301, 1965, doi: 10.1090/s0025-5718-1965-0178586-1.

[12] I. Newton, *The Method of Fluxions and Infinite Series*. 1736.

[13] G. W. Leibniz, *Nova Methodus pro Maximis et Minimis,* 1684.

[14] J. Luan, S. Lee, and P. H. Chou, “Fast Compressive Sensing Based on Dominant Frequency Estimation,” 2015 IEEE 12th International Conference on Mobile Ad Hoc and Sensor Systems. IEEE, Oct. 2015. doi: 10.1109/mass.2015.45.

[15] J.-B.-J. Fourier, *Théorie Analytique de la Chaleur*. 1822.

[16] P. Alexey, P. Olga, and S. Natalia, “Fast Parametric Fourier Transform,” 2022 International Conference on Dynamics and Vibroacoustics of Machines (DVM). IEEE, Sep. 21, 2022. doi: 10.1109/dvm55487.2022.9930933.

[17] M. Frigo and S. G. Johnson, “The Design and Implementation of FFTW3,” Proceedings of the IEEE, vol. 93, no. 2, pp. 216–231, Feb. 2005, doi: 10.1109/jproc.2004.840301.

[18] X. Zhang, S. Fan, H. Duan, and B. Xu, “Analysis of transient dominant frequency signal for single-phase earthed fault based on prony algorithm,” 2008 China International Conference on Electricity Distribution. IEEE, Dec. 2008. doi: 10.1109/ciced.2008.5211810.

[19] J. P. P. Guerrero, J. J. Saucedo-Dorantes, R. A. Osornio-Rios, J. A. Antonino-Daviu, and V. Biot-Monterde, “Unbalance and misalignment detection in induction motors under oscillating loads using current and Fast Fourier Transform (FFT),” 2023 IEEE Energy Conversion Congress and Exposition (ECCE). IEEE, Oct. 29, 2023. doi: 10.1109/ecce53617.2023.10362339.

[20] A. Cayley, "On the Theory of Matrices," Philosophical Transactions of the Royal Society of London, vol. 148, pp. 17–37, 1858. doi: 10.1098/rstl.1858.0002.

[21] Nyquist, H. “Certain Topics in Telegraph Transmission Theory.” Transactions of the American Institute of Electrical Engineers, vol. 47, no. 2, 1928, pp. 617–44. doi: https://doi.org/10.1109/t-aiee.1928.5055024.

[22] Pol, Balth. van der, and J. van der Mark. 1928. “LXXII. The Heartbeat Considered as a Relaxation Oscillation, and an Electrical Model of the Heart.” The London,

Edinburgh, and Dublin Philosophical Magazine and Journal of Science 6 (38): 763–75. doi:10.1080/14786441108564652.

Appendix

All of the code and data obtained by this experiment are open source, hosted on GitHub.

Access the full python code used to produce the results:

https://github.com/Luca-Skyline/Fourier-Research-Paper/tree/49b3494c46cca8f0e4b7edee82c90a32f6f66b7c/time_complexity_analysis

Access the spreadsheet that contains the numerical data obtained by the experiment:

https://github.com/Luca-Skyline/Fourier-Research-Paper/blob/49b3494c46cca8f0e4b7edee82c90a32f6f66b7c/time_complexity_analysis/data.csv