A Novel Method for Removing the Cyclic Frequency Shift due to the Earth's Motion about the Sun from Astronomical Continuous-Wave Data

Curtis Rau[1,2], Colin Gordon[1,2], David Witalka[1,2], and Samuel Akwei-Sekyere[1,3]
Department of Mathematics[1]
Department of Physics and Astronomy[2]
Neuroscience Program[3]
Michigan State University

In this paper we will introduce the Gustafson Algorithm for removing frequency/phase modulation from a signal. It takes as input time series data and outputs the original single tone carrier frequency. This method is powerful because it works in the frequency rather than time domain which more easily allows for filtering of the signal and the desired carrier frequency is close to constant over time. Here we will focus on the Gustafson Algorithm's applications to LIGO and the search for gravity waves, but there are many other possible applications for the algorithm such as radio wave astronomy and communications systems using frequency-modulated signals.

[5] [**?**] [1] [3] [4]

DIVISION OF LABOR

Curtis Rau Curtis was the most familiar with the topic and therefore performed the role of group leader identifying topics and problems of interest to be researched. He was also in charge of writing the algorithm section and the following discussion of the results. The c++ implementation of the algorithm was also done by him. He presented his part in the class presentation.

- organized topics and problems of interest, directed group research
- wrote algorithm and discussion of results
- made figures for presentation and paper
- wrote rough draft and template for paper
- Wrote the C++ implementation

Colin Gordon Colin contributed greatly in his research and solving of the phase moulation problem for astronomical signals cause by rotation of the earth using circular orbit approximation. He wrote the introduction to the paper, advised Sam on his section for noise and error analysis, and helped review and edit the paper as a whole. He presented his work in class.

- researched and solved phase modulation of astronomical signal caused by rotation of earth using circular orbit approximation
- wrote parts of the introduction
- reviewed paper and edited

David Witalka David researched and wrote the sections concerning the Nyquist frequency and Carson's rule. He also added the section pertaining to the speed of the Gustafson Algorithm. David edited and reworded the introduction and helped review and edit the paper as a whole. He also created and organized the slides presented in class.

- created and organized presentation slides
- researched and edited sections on sampling theorem, Nyquist Frequency and aliasing, Carson's rule
- reviewed paper and edited

Samuel Akwei-Sekyere Same researched and wrote the large section on noise and error analysis. He helped greatly in rewording problem sections through out the paper, and greatly expanded the discussion of all topics. Sam placed a major role in reviewing and editing other's sections. He presented his work in class.

- reviewed and expanded paper (all sections)
- researched and wrote sections on error analysis, wrote parts of the introduction

I. INTRODUCTION

General Relativity postulates that information about the spatial distribution of matter in the universe is transmitted via a gravitational field. Information regarding changes in mass distribution of the universe propagate through space-time away from the location of that change at the speed of light. Special events, where the quadrupole moment of the mass distribution changes, emit gravitational radiation. Extremely violent events in the universe can release enough gravitational radiation to be detectable from Earth.

Recently, the Laser Interferometer Gravitational-Wave Observatory (LIGO) obtained the first emperical evidence of gravitational waves. The discovered ripples in space-time by LIGO was a function of the inspiral of two black holes that eventually merged — an event that consumed approximately 3 solar masses [2]. It is believed that gravitational wave sources are common in the universe and sources of continuous monochromatic (single frequency) gravitational radiation are predicted to exist — at least sources with frequencies that persist over many periods []. Neutron stars have also been implicated as one of the prime sources of gravitational waves. These stars are expected to emit single frequency waves over long periods of time. Additionally, it has also been proposed that an appreciable fraction of detectable gravitational wave emitters are binary neutron star systems — two neutron starts orbiting around each other.

A fundamental challenge to efficient detection of gravitational waves is that orbiting binary neutron star systems invoke frequency modulations which reduce the power of the carrier frequency and distributes it to higher and lower harmonics. In essence, the power of the signal is dispersed over a range of side-band frequencies; this phenomenon is due to the Doppler effect arising from the Earth's motion relative to the source of the wave. Since the noise observed in the LIGO data is approximately Gaussian and of low signal-to-noise ratio, the extraction of reliable information about gravitational waves is arduous.

In order to calculate the carrier frequency of the wave, consider the wave as one emitted from a source that is megaparsecs away (potentially in other galaxies). By virtue of its distance, the spherical coordinates will not change significantly and the source will move relative to the sun with nearly constant velocity over the time we collect data. By the same token, the Sun will experience the monochromatic wave produced by the source as truly monochromatic, while its detection on the Earth will be in the form of a frequency modulated signal.

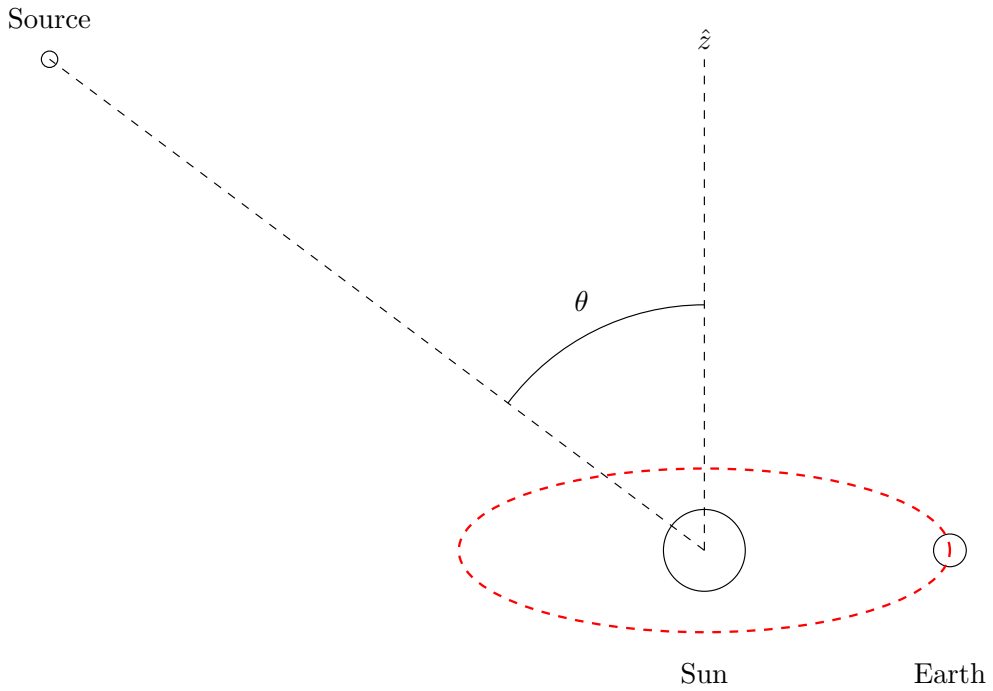


FIG. 1

By assuming that the wave travels at the speed of light, we can capture it as a Lorentz invariant 4-vector with the carrier frequency being the first entry and the second through fourth entries is the wave number. In order to transform it in the Earth's frame we apply an instantaneous Lorentz boost on the 4-vector; the instantaneous boost takes the direction of the motion of the Earth and the velocity of the the Earth in orbit into consideration. In our implementation, the Earth's orbit is captured in the xy plane. The 4-vector is written in the perspective

that assumes the wave is a plane wave which travels towards the Sun on a trajectory that is on the xz plane. The matrix with the trigonometric functions projects that 4-vector on a new basis and effectively forcing the second component of the 4-vector to be in the direction of motion of Earth. The matrix with the Betas and gammas is a Lorentz transformation that applies on 1st and 2nd component. now the 4-vector is in the frame of the Earth. After all of this, we could apply rotations or whatever to change the spatial coordinate system but we only care about the frequency which is invariant under any of those changes.

$$\mathbf{K} = \begin{pmatrix} \gamma & -\gamma\beta & 0 & 0 \\ -\gamma\beta & \gamma & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & -\sin(\omega_e t + \phi_e) & \cos(\omega_e t + \phi_e) & 0 \\ 0 & -\cos(\omega_e t + \phi_e) & -\sin(\omega_e t + \phi_e) & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} \omega_0/c \\ -k\sin\theta \\ 0 \\ -k\cos\theta \end{pmatrix} \tag{1}$$

$$= \begin{pmatrix} \gamma\left(\omega_0/c - \beta k \sin\theta \sin(\omega_e t + \phi_e)\right) \\ \gamma\left(-\beta\omega_0/c + k\sin\theta\sin(\omega_e t + \phi_e)\right) \\ \gamma k \sin\theta \cos(\omega_e t + \phi_e) \\ -k\cos\theta \end{pmatrix}, \tag{2}$$

where θ is the polar angle from the axis perpendicular to the orbit of Earth, γ is the Lorentz factor from special relativity, ω_e is the angular frequency of the Earth's rotation about the Sun, c is the speed of light and β is the velocity of Earth's orbit divided by the speed of light.

From this formulation we can infer that the instantaneous frequency in the Earth's frame can be given by:

$$\omega(t) = \omega_0\gamma\left(1 - \beta\sin\theta\sin(\omega_e t + \phi_e)\right). \tag{3}$$

Since the phase refers to the time integral of the frequency, the expression for the wave being sought after is the real part of $h(t)$:

$$h(t) = h_0 e^{i\left(\omega' t + \Gamma\cos(\omega_e t + \phi_e)\right)} \tag{4}$$

$$\omega' = \gamma\omega_0 \tag{5}$$

$$\Gamma = \frac{\gamma\beta\omega_0\sin\theta}{\omega_e} \tag{6}$$

where h_0 is the amplitude of the wave, Γ is the modulation index, ω_0 is the frequency of the wave to be detected, ϕ_0 is the phase of the wave to be detected at $t = 0$, ϕ_e is the relative phase difference between the Earth's rotation about the Sun and the wave to be detected.

An important note is that Γ is a function of only the azimuthal angle of the source with respect to the normal of the gallactic plane. This algorithm will also work perfectly well with any frequency modulated wave produced similar to the above. One likely case scenario is when the compact stars are in a binary system so one or both objects produce waves but the stars rotate around each other causing an analogues phase modulation. In this way there could be applications for the Gustafson algorithm in areas outside of gravitational waves in areas such as radio wave astronomy and communications using frequency modulated signals. We attempt to keep the discussion of the algorithm and the results as general as possible for this reason.

II. THE GUSTAFSON ALGORITHM

The goal of this paper is to propose a search algorithm that can efficiently detect the presence of frequency modulated waves, while being robust against noise. A brute force approach to accomplish this is to take the inner product of

$$h(t) = h_0 e^{i(\omega_0 t + \phi_0 + \Gamma\cos(\omega_1 t + \phi_1))} \tag{7}$$

with the data for different values of ω_0, ω_1, ϕ_0, ϕ_1, and Γ. This is effectively a Fourier Transform as a function of the the four search parameters:

$$\hat{f}(\omega_0, \omega_1, \phi_1, \Gamma) = \int f(t) e^{-i(\omega_0 t + \phi_0 + \Gamma\cos(\omega_1 t + \phi_1))} dt. \tag{8}$$

It is expected that this approach results in a something like a delta function, or at least a Dirchlet kernel because of discretization. The amount of time it will take to perform this computation is $\mathcal{O}(N_t N_{\omega_0} N_{\omega_1} N_{\phi_1} N_\Gamma)$, where N_t denotes the number of data points, and the other N_x is the number of steps in x that the search will take. Typically for a Discrete Fourier Transform (DFT) the number of steps in frequency is equal to the number of data points: $N_t = N_{\omega_0} = N_{\omega_1}$ []. Physically, this infers that the frequency resolution of the DFT is proportional to the duration of the time series over which the DFT is taken. Thus the computation time for the brute force method goes as $\mathcal{O}(N_t^3 N_{\phi_1} N_\Gamma)$.

The case when the signal is buried deep within non-coherent noise is of particular interest. A conventional method to improve the signal to noise ratio (SNR) is by increasing the number of data points; this assumes the signal is more periodic than noise. The problem with taking more data however is that computational time of the brute force method is proportional to N_t^3. We will now introduce the Gustafson Algorithm which is roughly a factor of N_t faster than the brute force method.

To derive the Gustafson Algorithm we start by assuming the data $f(t)$ takes on the form of a complex frequency modulated wave. Although the recorded data is real, it will be shown later that starting with the complex wave is perfectly valid. First, we demodulate the wave and take the Fourier transform of both sides:

$$h_0 e^{i(\omega_0 t + \phi_0 + \Gamma\cos(\omega_1 t + \phi_1))} = f(t) \tag{9}$$

$$h_0 e^{i\phi_0} e^{i\omega_0 t} = f(t) e^{-i\Gamma\cos(\omega_1 t + \phi_1)} \tag{10}$$

$$h_0 e^{i\phi_0} \mathcal{F}_t\left[e^{i\omega_0 t}\right] = \mathcal{F}_t\left[f(t) e^{-i\Gamma\cos(\omega_1 t + \phi_1)}\right]. \tag{11}$$

Define $\mathcal{F}_\sqcup[f(t)](\omega) = \hat{f}(\omega)$. By invoking the convolution theorem we yield,

$$\frac{h_0 e^{i\phi_0}}{\sqrt{2\pi}} \delta(\omega_c - \omega) = \hat{f}(\omega) \star \mathcal{F}_\sqcup\left[e^{-i\Gamma\cos(\Omega t + \phi)}\right]. \tag{12}$$

The Jacobi-Anger Expansion, which comes up often in this paper, has the form []:

$$e^{iz\cos(\theta)} = \sum_{n=-\infty}^{\infty} e^{in\pi/2} J_n(z) e^{in\theta} \tag{13}$$

By invoking the previously mentioned expansion, equation 12 becomes

$$\frac{h_0 e^{i\phi_0}}{\sqrt{2\pi}} \delta(\omega - \omega_0) = \hat{f}(\omega) \star \mathcal{F}_t\left[\sum_{n=-\infty}^{\infty} e^{in\pi/2} J_n(-\Gamma) e^{in(\omega_1 t + \phi_1)}\right]. \tag{14}$$

It is known that even order Bessel functions are even, and odd order Bessel functions are odd []. Concretely,

$$J_n(-z) = (-1)^n J_n(z). \tag{15}$$

By employing this fact we can observe that

$$\mathcal{F}_t\left[\sum_{n=-\infty}^{\infty} e^{in\pi/2} J_n(-\Gamma) e^{in(\omega_1 t + \phi_1)}\right] = \sum_{n=-\infty}^{\infty} (-1)^n e^{in\pi/2} e^{in\phi_1} J_n(\Gamma) \mathcal{F}_t\left[e^{in\omega_1 t}\right] \tag{16}$$

$$= \frac{1}{\sqrt{2\pi}} \sum_{n=-\infty}^{\infty} e^{in(\phi_1 - \pi/2)} J_n(\Gamma) \delta(\omega - n\omega_1). \tag{17}$$

In the previous step we used the linearity of the Fourier transform $\mathcal{F}_\sqcup[c \cdot f(t)] = c \cdot \mathcal{F}_\sqcup[f]$. Note that 14 becomes

$$h_0 e^{i\phi_0} \delta(\omega - \omega_0) = \hat{f}(\omega) \star \left[\sum_{n=-\infty}^{\infty} e^{in(\phi_1 - \pi/2)} J_n(\Gamma) \delta(\omega - n\omega_1)\right]. \tag{18}$$

By performing the convolution,

$$\hat{f}(\omega) \star \delta(\omega - n\omega_1) = \int \hat{f}(\omega - \widetilde{\omega})\delta(\widetilde{\omega} - n\omega_1)d\widetilde{\omega} \tag{19}$$

$$= \hat{f}(\omega - n\omega_1) \tag{20}$$

brings us to

$$h_0 e^{i\phi_0}\delta(\omega - \omega_0) = \sum_{n=-\infty}^{\infty} e^{in(\phi_1 - \pi/2)} J_n(\Gamma)\hat{f}(\omega - n\omega_1). \tag{21}$$

For clarity and later convenience let us have $n \to -n$. We will use the other property of the Bessel Functions: []

$$J_{-n}(z) = (-1)^n J_n(z). \tag{22}$$

This results in the Gustafson Algorithm:

$$\boxed{h_0 e^{i\phi_0}\delta(\omega - \omega_0) = 2\sum_{n=-\infty}^{\infty} e^{-in(\phi_1 + \pi/2)} J_n(\Gamma)\hat{f}(\omega + n\omega_1)} \tag{23}$$

An erroneous factor of two has been added to the algorithm which, as we will see, makes it suitable for use with real data. It is not obvious whether or not this algorithm in its current form can be used on real data because it assumed a complex waveform. This assumption allowed for a clean demodulation, which is marred if one takes the real of this function. This is due to the nonlinearity of the real operator as demonstrated by

$$\Re\{a \cdot b\} \neq \Re\{a\} \cdot \Re\{b\}; \qquad a, b \in \mathbb{C}. \tag{24}$$

We can use equation (23) as a guess as to find the form that will work with a real valued waveform. Upon comparison of the Fourier transforms of the complex and real waveforms (see appendices A and B) we see they are very similar. The real looks like the complex with the addition of terms that are mirrored about $\omega = 0$.

What would we have to do to get the Fourier transform of the real to look exactly like the Fourier transform of the complex wave? Unsurprisingly, not much because they are such similar functions. All we would have to do is drop the extra mirrored terms.

We would only have to multiply each term by some phase factor to get the real Fourier transform to look like the sum of two complex Fourier transforms. After adding in the phase factors we plug the Fourier transform of the real wave into the Gustafson Algorithm which will yield two delta functions instead of one (because the Fourier transform of the real is the sum of two complex wave Fourier transforms). One delta function will be at $+\omega_0$, this is the one of interest, and the other will be at $-\omega_0$. If we carry the search out over only positive values of ω_0 then we can throw out all terms in red of the Fourier transform of the real wave because they are inconsequential to the algorithm. Finally, the phase terms that were added to the Fourier transform of the real to take on the morphology of the complex can be incorporated into the Gustafson Algorithm. The only remaining thing to do is multiply the Gustafson Algorithm by two (explain why).

A. Discussion of The Gustafson Algorithm

It is not explicitly obvious how The Gustafson Algorithm, Eq. 23, can be used as a search algorithm capable of determining the values $h_0, \omega_0, \omega_1, \phi_0, \phi_1, \Gamma$. In this section we seek to make obvious what Eq. 23 can do.

Assume the data takes on the real form of a frequency modulated wave.

$$f(t) = \Re\left\{h_0 e^{i(\omega_0 t + \phi_0 + \Gamma\cos(\omega_1 t + \phi_1))}\right\} \tag{25}$$

Initially we do not know the values of $\omega_0, \omega_1, \phi_0, \phi_1, \Gamma$. To determine these values, we will try the values $\widetilde{\omega}_0, \widetilde{\omega}_1, \widetilde{\phi}_1, \widetilde{\Gamma}$ in the Gustafson Algorithm.

$$G_{\widetilde{\omega}_0,\widetilde{\omega}_1,\widetilde{\phi}_1,\widetilde{\Gamma}}\left[\hat{f}(\omega)\right] = \sum_{n=-\infty}^{\infty} e^{-in(\widetilde{\phi}_1+\pi/2)} J_n(\widetilde{\Gamma}) \hat{f}(\omega + n\widetilde{\omega}_1) \tag{26}$$

To see what this will yield we substitute the Fourier transform of the real of the waveform, equation (B6), into the above equation

$$\begin{aligned} G_{\omega_0,\omega_1,\phi_1,\Gamma}\left[\hat{f}(\omega)\right] =& \frac{h_0}{2} \sum_{m,n=-\infty}^{\infty} e^{i(m\phi_1 - n\widetilde{\phi}_1)} J_n(\widetilde{\Gamma}) J_m(\Gamma) \\ & \times \left[e^{i(\phi_0 + (m-n)\pi/2)} \delta(\omega - \omega_0 + n\widetilde{\omega}_1 - m\omega_1) + e^{-i(\phi_0 + (m+n)\pi/2)} \delta(\omega + \omega_0 + n\widetilde{\omega}_1 - m\omega_1) \right] \end{aligned} \tag{27}$$

We shall treat this in cases.

Case 1: Let us consider the most likely case when $\omega \neq \omega_0$ and $\omega_1 \neq \widetilde{\omega}_1$. Let us examine the term $\delta(\omega - \omega_0 + n\widetilde{\omega}_1 - m\omega_1)$ in the above equation. This delta function appears in the double sum and is a function of both m and n, so the $n\widetilde{\omega}_1 - m\omega_1$ is changing over the sums whereas the value $\omega - \omega_0$ is constant (not a function of m or n). This delta is non-zero only if

$$0 = \omega - \omega_0 + n\widetilde{\omega}_1 - m\omega_1 \tag{28}$$

$$\omega_0 - \omega = n\widetilde{\omega}_1 - m\omega_1 \tag{29}$$

This can happen twice or never. By the Euclidean Algorithm, the delta will only be non-zero if

$$\gcd\left(\widetilde{\omega}_1, \omega_1\right) = \omega_0 - \omega \tag{30}$$

If this is false, then the sum is zero. If it is true then call the values for m and n that make it true $\pm m_o$ and $\mp n_o$. So then the sum is

Placeholder for equation

Case 2: Now consider the case when $\omega = \omega_0$ and $\omega_1 \neq \widetilde{\omega}_1$. The term $\delta(\omega - \omega_0 + n\widetilde{\omega}_1 - m\omega_1)$ will only be non-zero if

$$\widetilde{\omega}_1 = \frac{m}{n}\omega_1 \tag{31}$$

Again, this can happen twice or never. If this is false, then the sum is zero.

Placeholder for equation

Case 3: Now consider the case when $\omega \neq \omega_0$ and $\omega_1 = \widetilde{\omega}_1$. The term $\delta(\omega - \omega_0 + n\widetilde{\omega}_1 - m\omega_1)$ will only be non-zero if

$$\omega - \omega_0 = (n-m)\widetilde{\omega}_1 \tag{32}$$

This too can happen twice or never. If this is false then the sum is zero.

Cases 1-3: In all of these cases

Case 4: Now consider the case when $\omega = \omega_0$ and $\omega_1 = \widetilde{\omega}_1$. The term $\delta(\omega - \omega_0 + n\widetilde{\omega}_1 - m\omega_1)$ will only be non-zero if $m = n$ which will happen an infinite number of times. Thus, the sum becomes

$$\frac{h_0}{2} e^{i\phi_0} \delta(0) \sum_{n=-\infty}^{\infty} e^{in(\phi_1 - \widetilde{\phi}_1)} J_n(\widetilde{\Gamma}) J_n(\Gamma) \tag{33}$$

Case 4a: Now if we let $\Gamma = \widetilde{\Gamma}$ and $\phi_1 = \widetilde{\phi}_1$ then the sum becomes

$$\frac{h_0}{2} e^{i\phi_0} \delta(0) \sum_{n=-\infty}^{\infty} J_n^2(\Gamma) = \frac{h_0}{2} e^{i\phi_0} \delta(0) \tag{34}$$

So if we guess every search parameter correctly, $\omega = \omega_0$, $\omega_1 = \widetilde{\omega}_1$, $\Gamma = \widetilde{\Gamma}$, and $\phi_1 = \widetilde{\phi}_1$, then we recover the

III. CARSON'S RULE AND THE NYQUIST FREQUENCY

In order too fully reconstruct the carrier frequency we would need to take an infinite sum over the sidebands. As this is impractical and unnecessary as many terms are insignificant, we need to find a way to limit the number of sidebands we are summing over. The first problem is locating the sidebands' characteristic frequency. There are infinite sideband frequencies and eventually they will be higher than the Nyquist frequency which will cause aliasing of the sideband frequency. The Nyquist frequency, ψ_N, is defined as the maximum frequency that can be accurately reconstructed and is equal to half of the sampling frequency, ψ_s.

$$\psi_N = \frac{1}{2}\psi_s \tag{35}$$

Any side band with a frequency of ψ_0 higher than the Nyquist frequency, but less than the sampling frequency, will be reconstructed with a frequency of $(\psi_s - \psi_0)$. That is the frequency will be under-sampled and an alias frequency will be produced. When this occurs the best case scenario is that the alias frequency is unique and does not correspond to the frequency of another lower frequency sideband. In this case we can use anti-aliasing techniques to correct for the aliasing and still obtain information about the amplitude and frequency of the higher side bands. The worst case scenario is that the aliased frequency is too close to another lower frequency sideband and this will cause two problems: a systematic error will be added to the measurement of the lower side band's amplitude, and we will be unable to determine the necessary information on what the amplitude of the higher frequency would be. To minimize the effect this might have, we would like any sidebands with a frequency over the Nyquist frequency to have as low an amplitude as possible. For this we can employ Carson's rule. Carson's rule can be understood to say that almost all (roughly 98 percent) of the power for a frequency-modulated sinusoidal signal is contained within a finite bandwidth B_T, defined by:

$$B_T = 2(\Delta\psi + \psi_m) \tag{36}$$

where $\Delta\psi$ is the peak frequency deviation of the instantaneous frequency $\psi(t)$ from the center carrier frequency ψ_c, and ψ_m is the highest frequency in the modulating signal. In our case the highest frequency of the modulating signal is the carrier frequency, ψ_c. From this calculation we can adjust our sampling frequency to ensure it is large enough in order to minimize aliasing. This also puts a limit on the number of sidebands we need to sum over to just those that fall within the bandwidth. The important thing to take away from this is that it takes an infinite bandwidth to perfectly transmit a phase or frequency modulated wave, regardless of how smooth it is, but in practice only a finite bandwidth is needed for an accurate approximation.

IV. ERROR ANALYSIS

While the search for monochromatic gravitational waves via the LIGO detector is relatively effortless, the adulteration of signals obtained from the detector by noise adds an appreciable number of complications to the problem. By the same token, techniques by which improved resistance of the LIGO detector to noise can be obtained has been investigated vigorously [**?**] [**?**].

A. Noise Sources

Since the signal being searched for is relatively stable but of little amplitude, the reliability of data obtained from the detector is related to the degree of noise therein. Among others, quantum, thermal, seismic and Newtonian noise have been implicated as prime drivers of LIGO data adulteration. Since the prime noise contributor is quantum, a summation of all the noise sources are approximately Gaussian. Thus, it suffices to use the properties of Gaussian noise to understand the volatility of the data.

B. Noise Elimination

As mentioned previously, quantum noise — in particular, thermal noise — has been implicated as a limiting factor in the sensitivity of gravitational wave inteferometers and hence a prime factor in the degradation of the signal quality; this noise source is generally modeled as Gaussian [5].

By low-pass filtering or spectral subtraction, it is possible to derive signals with high SNR; however, these widely-accepted approaches require information which can be difficult to obtain. Since the frequency of the monochromatic wave being searched for is not known, low-pass filtering may result in erroneous outputs. Spectral subtraction requires a sample of the noise; nevertheless, due to the low SNR of LIGO data, it is precarious to find segments with pure noise.

In this section, we introduce an alternate approach by which Gaussian noise in data may be reduced. This method utilizes the following principle: the nature of the noise is not known but the nature of the signal is known. Our methodology consists of subtracting estimated noise from the signal via singular value decomposition of the Fourier transform of intrinsic mode functions (see appendix) and assumes the noise is correlated across different channels. A significant part of this approach is motivated by [] and []. It is worth noting that because correlated Gaussian noise is prevalent, this approach may be used for a host of other signals.

1. Proposed Denoising Algorithm

In an effort to reduce the noise, we shall consider data from two channels. By the assumption that the noise observed is additive (and correlated), we yield the following:

$$\xi^{(1)}(t) = f^{(1)}(t) + \epsilon^{(1)}(t) \tag{37}$$

$$\xi^{(2)}(t) = f^{(2)}(t) + \epsilon^{(2)}(t) \tag{38}$$

where $\xi^{(n)}(t)$ is the data observed, $f^{(n)}(t)$ is the true signal and $\epsilon^{(n)}(t)$ is the noise obtained from the n-th channel (or gravitational wave interferometer) with $\epsilon^{(n)}(t) \sim \mathcal{N}(0, \sigma^2)$.

First, we disintegrate each observed signal into α intrinsic mode functions via empirical mode decomposition and arrange them into a $2\alpha \times T$ matrix $\mathbf{L}$:

$$\mathbf{L} = \begin{pmatrix} l^{(1)}_{1,t_1} & l^{(1)}_{1,t_2} & \cdots & l^{(1)}_{1,t_T} \\ l^{(1)}_{2,t_2} & l^{(1)}_{2,t_2} & \cdots & l^{(1)}_{2,t_T} \\ \vdots & \vdots & \vdots & \vdots \\ l^{(1)}_{\alpha,t_1} & l^{(1)}_{\alpha,t_2} & \cdots & l^{(1)}_{\alpha,t_T} \\ l^{(2)}_{\alpha+1,t_1} & l^{(2)}_{\alpha+1,t_2} & \cdots & l^{(2)}_{\alpha+1,t_T} \\ l^{(2)}_{\alpha+2,t_2} & l^{(2)}_{\alpha+2,t_2} & \cdots & l^{(2)}_{\alpha+2,t_T} \\ \vdots & \vdots & \vdots & \vdots \\ l^{(2)}_{2\alpha,t_1} & l^{(2)}_{2\alpha,t_2} & \cdots & l^{(2)}_{2\alpha,t_T} \end{pmatrix} \tag{39}$$

where $l^{(n)}_{p,t_q}$ is the $\left[\left((p-1)\text{mod }\alpha\right)+1\right]$-th intrinsic mode function at time t_q of $\xi^{(n)}(t)$, and with T being the total number of data points observed.

Thereafter, the fast Fourier transform of each intrinsic mode function in $\mathbf{L}$ is computed to obtain a matrix $\mathcal{F}(\mathbf{L})$. Singular value decomposition of the variance-covariance matrix obtained from $\mathcal{F}(\mathbf{L})$ is then computed to result in the following:

$$\Sigma = \mathbf{USV} \tag{40}$$

where $\Sigma = T^{-1}\left[\mathcal{F}(\mathbf{L})^{\mathbf{T}} - \mathbf{1}\mathbf{1}^{\mathbf{T}}\mathcal{F}(\mathbf{L})^{\mathbf{T}}T^{-1}\right]^{\dagger}\left[\mathcal{F}(\mathbf{L})^{\mathbf{T}} - \mathbf{1}\mathbf{1}^{\mathbf{T}}\mathcal{F}(\mathbf{L})^{\mathbf{T}}T^{-1}\right]$with $(\cdot)^{\mathbf{T}}$ being the transpose, the conjugate transpose $(\cdot)^{\dagger}$ and $\mathbf{1}$ a $T \times 1$ column matrix.

The fast Fourier transforms are then decorrelated to form a matrix $\mathbf{R}$ by employing the following:

$$\mathbf{R} = \mathbf{U}^{\mathbf{T}}\mathbf{L}. \tag{41}$$

After the decorrelation procedure above, we seek to apply a threshold Δ which dictates the approximate percentage of statistically irrelevant information about the Fourier transform we shall discard. Prior to the thresholding procedure, the eigenvectors that contribute to approximately Δ percent of the variance of the transform by virtue of their respective eigenvalues is estimated using the following formulation:

$$y = \underset{b}{\operatorname{argmin}} \left(\left| \frac{\sum_{j=1}^{(b)} \lambda_j}{trace\{\mathbf{S}\}} - \frac{\Delta}{100} \right| \right) \tag{42}$$

where y is the index denoting the y^{th} eigenvalue from λ_y with $\lambda = diag(\mathbf{S})$.

To apply the threshold, the projections with eigenvalues whose indices are less than y are retained and the rest are replaced with zero row vectors:

$$\mathbf{R}_r = \begin{cases} \mathbf{R}_r & , r < y \\ \mathbf{0} & , r \geq y \end{cases}$$

where $\mathbf{R}_r$ the r^{th} row of the matrix $\mathbf{R}$. This preserves the vectors that explain closest to, but less than, Δ percent of the total variance.

The fast Fourier transforms are then recorrelated to obtain denoised intrinsic mode functions in the following manner:

$$\tilde{\mathbf{L}} = \mathbf{VR}. \tag{43}$$

The real component of the inverse fast Fourier transform of each row of $\tilde{\mathbf{L}}$ is then computed to obtain a matrix $\mathbf{D}$ of similar architecture with denoised intrinsic mode functions. Thereafter, the reconstructed signals $\hat{\xi^{(n)}}(t)$ are obtained as follows:

$$\hat{\xi^{(n)}}(t_q) = \sum_{u=(n-1)\alpha+1}^{n\alpha} \mathbf{d}_{u,q} \tag{44}$$

where $\mathbf{d}_{u,q}$ is the value at the u^{th} row and q^{th} column of $\mathbf{D}$.

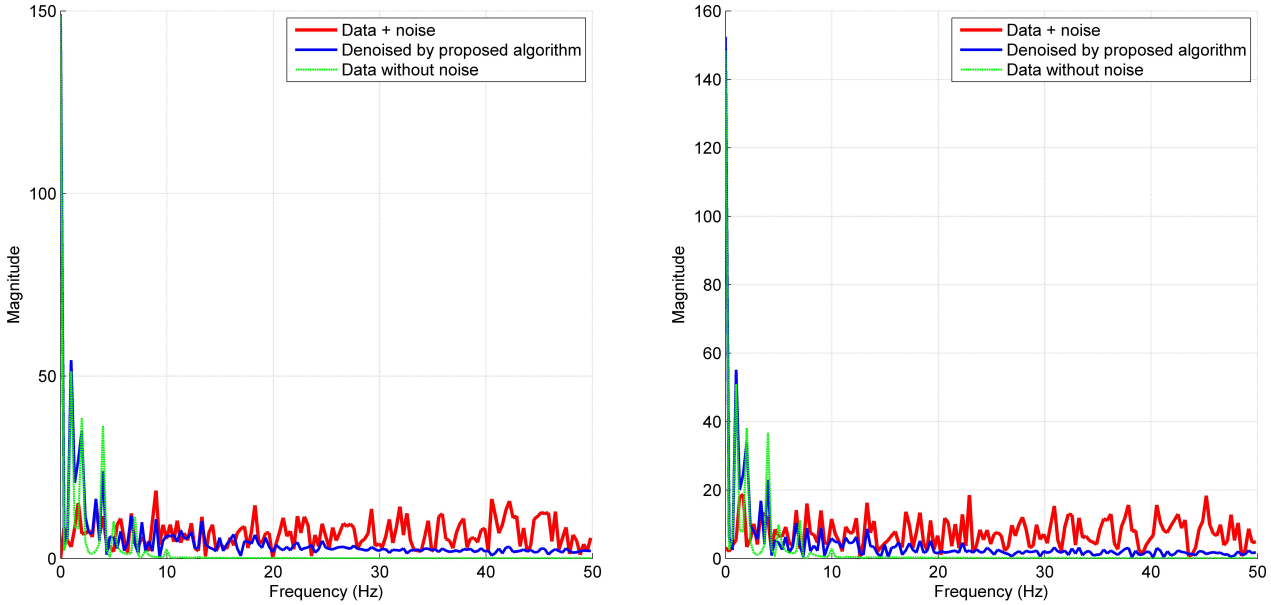
2. Discussion

In contrast with denoising techniques such as low-pass filtering, this approach does not require the knowledge of the highest frequency of the signal being recorded. However, the knowledge of how much variance of the data to keep is integral to successful denoising — this is usually unknown and techniques have been proposed to solve this non-trivial problem [].

To demonstrate the potential of this procedure in removing correlated Gaussian noise, two frequency modulated signals that are phase shifted from each other are adulterated with correlated Gaussian noise. The original signals are then reconstructed using the proposed denoising algorithm with a threshold of 99 percent — that is, eigenvectors that account for closest to (but, less than) 99 percent of the variation of the data obtained from the set of multichannel intrinsic mode functions are retained. In Figure 2a, it can be easily observed that the algorithm extinguished an appreciable amount of the high frequency component that was not previously present in the data and selectively reduced the magnitude of the Fourier transform of the noise components with low frequency. Figure 2b was no different in terms of its denoising performance. It is worth noting that both results do have some discrepancy — and their discrepancies are not the same. This feature is due to the fact that, although the noise sources are correlated, the noises are different.

A fundamental inquiry that needs a resolution is about how the proposed algorithm responds to different SNRs. To address this, the deviations of the Fourier transforms of reconstructed signals as a function of a myriad of SNRs were computed from the Fourier transform of the original signal. These deviation scores were measured as the mean squared errors between the reconstructed signals and the original signal. The threshold employed in this investigation was 99 percent. From Figure 3, it can be noted that the performance of the proposed algorithm increases with an increasing SNR; however, this increase in performance is limited. Since the proposed approach disregards parts of the data that are statistically irrelevant (with respect to the threshold chosen), it is plausible that parts of the original data may have been erased. This infers that an adaptive threshold that depends on the structure of the data and the structure of the noise to be extinguished may be most appropriate for this algorithm.

Since a fundamental assumption the proposed algorithm makes is that the noise sources are correlated, it is important to investigate how the correlation affects its performance. To this end, a procedure similar to that of Figure 3 was implemented; nevertheless, a second independent variable was added — the correlation between the noise sources. Further, the performance for denoising in each channel was not analyzed jointly. From Figure 4a, it



(a) Fourier transform of the proposed denoising approach for channel 1.

(b) Fourier transform of the proposed denoising approach for channel 2.

FIG. 2: Output of the proposed denoising approach on two channels with signals of similar frequency modulation, but shifted in time. This procedure assumes the signal being recorded is from the same source and the noise observed in each channel are correlated.

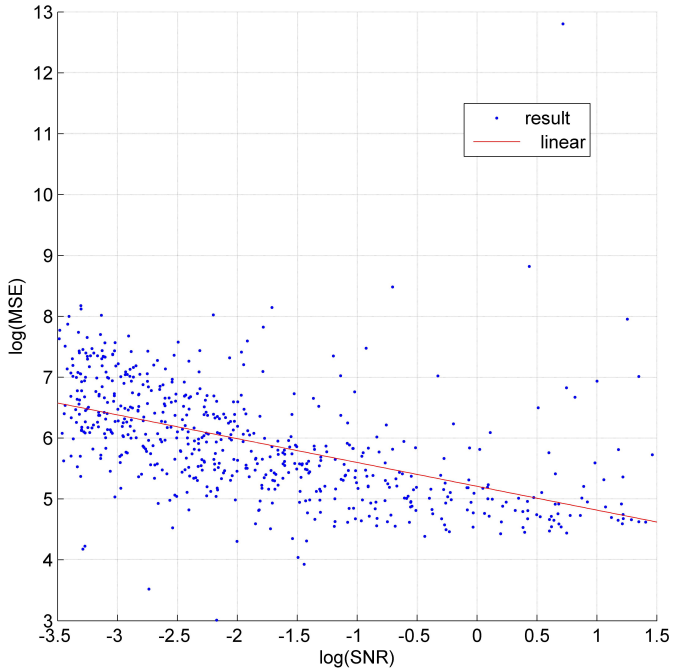
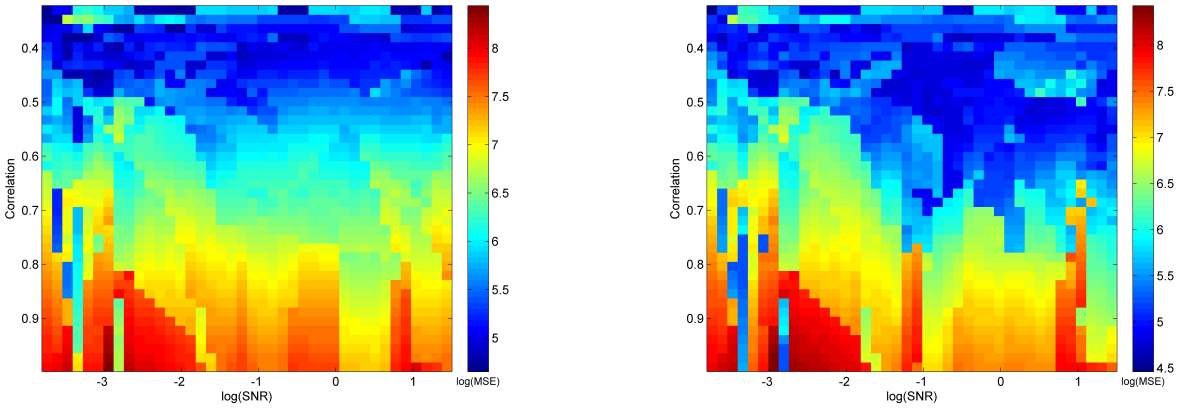


FIG. 3: The algorithm performs better with increasing SNR; however, this performance plateaus at high SNRs. The correlation coefficient $r^2 = -0.6543$ with $p < 0.05$.

can be noticed — as insinuated before — that the algorithm's performance plateaus at high SNRs and progressively worsens as the same threshold is more likely to remove parts of the original data. Interestingly, the proposed approach works well with noise sources of tolerably high correlation; however, it fails to produce quality results when the correlation is really high.

The potential of this approach has been outlined above. With a few modifications, this denoising algorithm may improve its capability and, to some extent, (depending on the structure of the data and the noise) could be modified



(a) Response of the algorithm with changing noise correlations and SNRs for channel 1.

(b) Response of the algorithm with changing noise correlations and SNRs for channel 2.

FIG. 4: Behavior of the proposed algorithm with respect to differing SNRs and noise correlations. This figure provides an insight to the performance of the proposed denoising algorithm (via the mean squared error) under the outlined conditions.

to deal with uncorrelated noise. In the spirit of demonstrating the robustness of the Gustafson algorithm against noise, the results from the search algorithm will be analyzed without no attempt to remove of noise.

V. IMPLEMENTING THE GUSTAFSON ALGORITHM IN C++

A. Calculating Bessel Functions

Known problems with the algorithm include calculating large order bessel functions. The bessel functions of natural number order are given by

$$J_n(z) = \sum_{m=0}^{\infty} \frac{(-1)^n}{m!(m+n)!} \left(\frac{z}{2}\right)^{2m+n} \tag{45}$$

so calculating the nth Bessel function involves calculating factorials greater than n!. The highest n for which we can store n! in an unsigned long integer type, the largest positive integer type, is 20:

$$2^{64} - 1 = 18,446,744,073,709,551,615 \tag{46}$$

$$< 21! = 51,090,942,171,709,440,000 \tag{47}$$

if we are willing to sacrafise some precision to truncation we can use a double precision floating point integer which allows values upto $1.797,693,134,862,315,7 \cdot 10^{308}$

$$170! \approx 7.25 \cdot 10^{306} < 1.797,693,134,862,315,7 \cdot 10^{308} < 171! \approx 1.24 \cdot 10^{309} \tag{48}$$

This is not acceptable hovever, because $|J_n(z)| \leq 1$, so when performing the sum we need precision all the way down to at the very least 0.1, which would require a mantissa with roughly 308 digits. This is not only impracticle, but it is by far one of the slowest ways one could calculate values of the Bessel functions. This whole buiseness of wresteling with factorials can be avoided by calculating the Bessel Functions using Bessel's Equation 49 [3].

$$x^2 J_n''(x) + x J_n'(x) + (x^2 - n^2) J_n(x) = 0 \tag{49}$$

The standard perscription for numerically solving differential equations is first to discretize the independent variable $x \to x_i = x_{min} + i \cdot \delta x$. Then use the limit definitions of the derivatives; we will use the three point

definitions because their error goes as $O(\delta x^2)$ rather than using the two point definition with an error that goes as $O(\delta x)$ [**?**]. Also, because there are only one order of Bessel Functions in Bessel's Equation, the order is implied by the n that appears, so we will drop the subscript of n, $J_n \to J$, and we will further adopt the notation $J(x_i) = J_i$.

$$J_i' = \frac{J_{i+1} - J_{i-1}}{2\delta x} \tag{50}$$

$$J_i'' = \frac{J_{i-1} - 2J_i + J_{i+1}}{\delta x^2} \tag{51}$$

So the discretized form of Bessel's Equation is

$$x_i^2 \frac{J_{i-1} - 2J_i + J_{i+1}}{\delta x^2} + x \frac{J_{i+1} - J_{i-1}}{2\delta x} + (x^2 - n^2)J_i = 0 \tag{52}$$

$$\left(\frac{x_i^2}{\delta x^2} - \frac{x_i}{2\delta x}\right) J_{i-1} + \left(\left(1 - \frac{2}{\delta x^2}\right) x_i^2 - n^2\right) J_i + \left(\frac{x_i^2}{\delta x^2} + \frac{x_i}{2\delta x}\right) J_{i+1} = 0 \tag{53}$$

Where we have collected like terms of J_i. For convenience we define coefficients to simplify the above equation.

$$a_i = \frac{x_i^2}{\delta x^2} - \frac{x_i}{2\delta x} \tag{54}$$

$$b_i = \left(1 - \frac{2}{\delta x^2}\right) x_i^2 - n^2 \tag{55}$$

$$c_i = \frac{x_i^2}{\delta x^2} + \frac{x_i}{2\delta x} \tag{56}$$

These coefficients can be simplified by using the definition for x_i, and taking $x_{min} = 0$.

$$a_i = i\left(i - \frac{1}{2}\right) \tag{57}$$

$$b_i = i^2\left(\delta x^2 - 2\right) - n^2 \tag{58}$$

$$c_i = i\left(i + \frac{1}{2}\right) \tag{59}$$

So the discretized Bessel equation with simplified coefficients is

$$a_i J_{i-1} + b_i J_i + c_i J_{i+1} = 0 \tag{60}$$

Wrighting out the system of equations explicitly:

$$b_1 J_1 + c_1 J_2 = -a_1 J_0 \tag{61}$$

$$a_2 J_1 + b_2 J_2 + c_2 J_3 = 0 \tag{62}$$

$$a_3 J_2 + b_3 J_3 + c_3 J_4 = 0 \tag{63}$$

$$\vdots \tag{64}$$

$$a_{n-1} J_{n-2} + b_{n-1} J_{n-1} + c_{n-1} J_n = 0 \tag{65}$$

$$a_n J_{n-1} + b_n J_n = -c_n J_{n+1} \tag{66}$$

Which suggests Bessel Equation be written as a matrix equation.

$$\begin{pmatrix} b_1 & c_1 & & & \\ a_2 & b_2 & c_2 & & \\ & a_3 & b_3 & c_3 & \\ & & \vdots & \vdots & \\ & & a_{n-1} & b_{n-1} & c_{n-1} \\ & & & a_n & b_n \end{pmatrix} \begin{pmatrix} J_1 \\ J_2 \\ J_3 \\ \vdots \\ J_{n-1} \\ J_n \end{pmatrix} = \begin{pmatrix} -a_1 J_0 \\ 0 \\ 0 \\ \vdots \\ 0 \\ -c_n J_{n+1} \end{pmatrix} \tag{67}$$

This is a tridiagonal matrix system that charactorized a one dimentional differential equation. It can be easilly solved using the standard Thomas Algorithm. To program this in C++ we need all the arrays to be indexed from 0, so

$$\begin{pmatrix} B_0 & C_0 & & & \\ A_1 & B_1 & C_1 & & \\ & A_2 & B_2 & C_2 & \\ & & \vdots & \vdots & \\ & & A_{N-4} & B_{N-4} & C_{N-4} \\ & & & A_{N-3} & B_{N-3} \end{pmatrix} \begin{pmatrix} J_1 \\ J_2 \\ J_3 \\ \vdots \\ J_{N-3} \\ J_{N-2} \end{pmatrix} = \begin{pmatrix} -A_0 J_0 \\ 0 \\ 0 \\ \vdots \\ 0 \\ -C_{N-3} J_{N-1} \end{pmatrix} \tag{68}$$

$$A_i = a_{i+1} = (i+1)\left(i + \frac{1}{2}\right) \tag{69}$$

$$B_i = b_{i+1} = (i+1)^2\left(\delta x^2 - 2\right) - n^2 \tag{70}$$

$$C_i = c_{i+1} = (i+1)\left(i + \frac{3}{2}\right) \tag{71}$$

1. The Thomas Algorithm

The Thomas Algorithm is a lightning fast way to solve tridiagonal systems like equation 68. The algorithm consists of two steps. First there is a forward substitution which eliminates the A_i's, and modifies the B_i's and the vector on the right which we will call the source vector S_i; $\quad \forall i \in [0, N-3]$.

$$R_i = R_i - \frac{A_i}{B_{i-1}} R_{i-1}; \qquad \forall i \in [1, N-3] \tag{72}$$

$$B_i \to B_i - \frac{A_i}{B_{i-1}} C_{i-1}; \qquad \forall i \in [1, N-3] \tag{73}$$

$$S_i \to S_i - \frac{A_i}{B_{i-1}} S_{i-1}; \qquad \forall i \in [1, N-3] \tag{74}$$

Then there is a backwards substitution eliminating the C_i's, again modifying the S_i's. Finally we solve for J_i.

$$R_i = R_i - \frac{C_i}{B_{i+1}} R_{i+1}; \qquad \forall i \in [N-4, 0] \tag{75}$$

$$S_i \to S_i - \frac{C_i}{B_{i+1}} S_{i+1}; \qquad \forall i \in [N-4, 0] \tag{76}$$

$$J_i = \frac{S_{i-1}}{B_{i-1}}; \qquad \forall i \in [N-2, 1] \tag{77}$$

The time it takes to perform the Thomas Algorithm goes as N [].

2. Boundary Conditions

To solve this second-order differential equation we will need two boundary conditions. The boundary condition at $x = 0$ is simple.

$$J_n(0) = \begin{cases} 1 & : n = 0 \\ 0 & : n \neq 0 \end{cases}$$

For the other boundary condition we *could* use the asymtotic form of the Bessel functions. Theorem 5.1 in [3] states:

Parameters	Point by Point Using Integral	Thomas Algo. Solving Bessel's Equation	Speed Increse
n = 3 N = 500 $z_{max} = 20.0$	0.051786 s	0.00013800 s	375
n = 14 N = 50,000 $z_{max} = 100.0$	113.33 s	0.003937 s	28,785

TABLE I: N is the number of points returned, n is the order of the bessel function, and the points calculated were for $z \in [0, z_{max}]$

For each $n \in N$ there is a constant $C_n \in R$ such that, if $x \geq 1$, then

$$\left| J_n(x) - \sqrt{\frac{2}{\pi x}} \cos\left(x - \frac{\pi}{4}(2n+1)\right) \right| \leq \frac{C_n}{x^{3/2}} \tag{78}$$

This is an asymptotic expansion, so it is only valid for large values of x. It turns out the accuracy of the solution to bessels equation relies heavilly on the accuracy of the endpoint at $x \neq 0$. Figures 5a and 5b illistrate this point quite well. The conclusion is that the asymtotic form of the bessel functions is not accurate enough for even fairly large valuse of z. We can also do

$$J_n(z) = \frac{1}{\pi} \int_0^{\pi} \cos\left[z \sin\theta - n\theta\right] d\theta \tag{79}$$

Which is computed using a Riemann Sum

$$J_n(z) \approx \frac{1}{N} \sum_{i=0}^{N+1} \cos\left[z \sin(i \cdot dx) - n \cdot i \cdot dx\right]; \qquad dx = \frac{\pi}{N} \tag{80}$$

The function being integrated becomes more oscillitory as n increases. Numerical analysis shows that n gives the number of zeros the function has on the interval $(0, \pi)$ for $z = 0$. For $n = 0$ the number of zeros is given by $Floor(\frac{2z}{\pi})$. We want a way to ensure the calculation is accurate regardless of how oscillatory it is, but we also want it to be as fast as possible, so we use these facts regarding the zeros to ensure there are roughly the same number of integration points between zeros. By taking the number of (evenly spaced) integration points to be

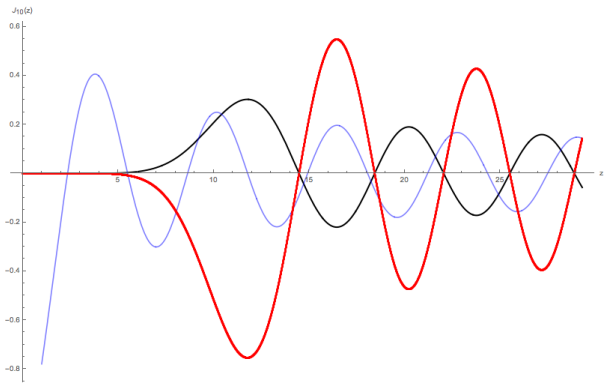
$$N = 100 \cdot (n+1) \cdot Floor\left(\frac{2z}{\pi} + 1\right) \tag{81}$$

we acheved excellent results (accurate consistantly to 16 decimal places) for a wide range of z and n. The $+1$'s are to ensure accurate results when z or n are zero. The algorithm can have problems when J is extremely small. Then the function can be off by many orders of magnitude, but because it is essentially zero, it does not seem to effect our results when used as a boundary condition for solving Bessel's Differential Equation.

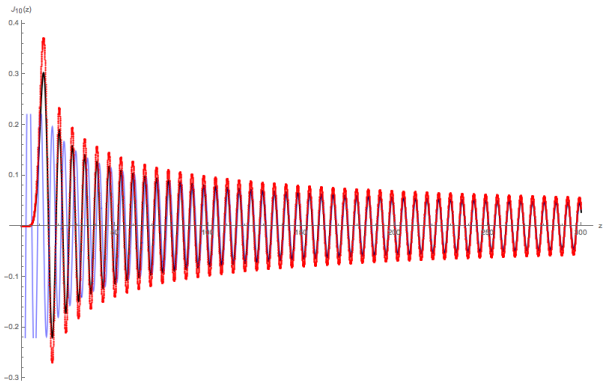
3. Calculating Bessel Functions: Results

The ultimate benchmark for sucess in calculating Bessel Functions in this manner (using the Thomas Algorithm to solve Bessel's Differential Equation) is to compare it to the stardard way of calculating the value at each point using the Riemann Sum (as defined in equation 80 section V A 2). So how much faster is it? This depends on the values of n and z because we use a dynamic number of integration points as given by eq. 81. Table I gives the speed increse for various parameters.

Table 6 gives three examples of Bessel Functions that were calculated using the Thomas Algorithm to solve Bessel's Equation.



(a) Notice how the asymtotic form and the one calculated by our differential equation solver match up at the maximum value for z, but do not match up with the true solution. This has far reaching consequences. Whatever multiplicative factor the endpoint is off by seems to carry through for every other point. In this case the multiplicative factor (asymtotic / true) is almost exactly -2.5. Notice how this inverts the solution and makes it 250% larger.



(b) For much larger values of z the asymtotic form is a better approximation to the true value. In this case the multiplicative factor (asymtotic / true evaluated at $z = 300$) is roughtly 1.2 which results in the soultion being roughly 20% larger than it should be.

FIG. 5: The blue curve is the asymtotic form of the bessel functions as given in equation 78, the black curve is $J_{10}(x)$ as given by Mathematica, and the red curve is the one calculated by our differential equation solver. Note, both of these solutions were calculated using 10,000 points. Even though the solution in figure 5a has 10 times the point density, it is limited in accuracy by the endpoint!

VI. C++ IMPLEMENTATION OF THE GUSTAFSON ALGORITHM: RESULTS

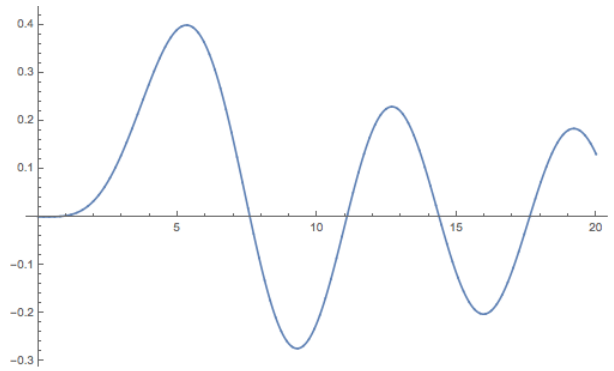
-Step size in omega0 and omega1 must be small because they are deltas. -Step size in phi1 can be larger because it is less sharply peaked. -Step size in Gamma can be even larger because it is relatively smooth.

VII. CONCLUSION

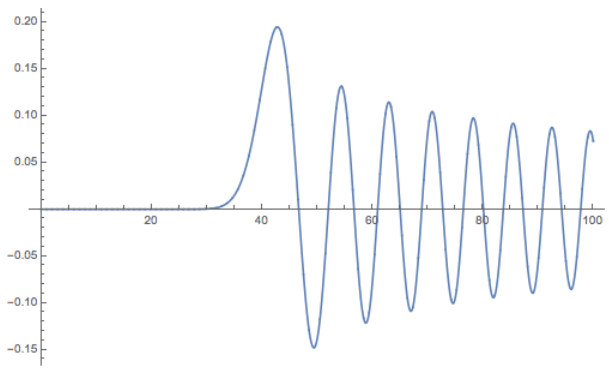
Recap the benifits of working in the frequency domain vs the time domain. It is the belief of this author that the methods outlined in this paper are unlikely to be useful for searching for gravitational waves.

Appendix A: The Fourier Transform of The Complex Function

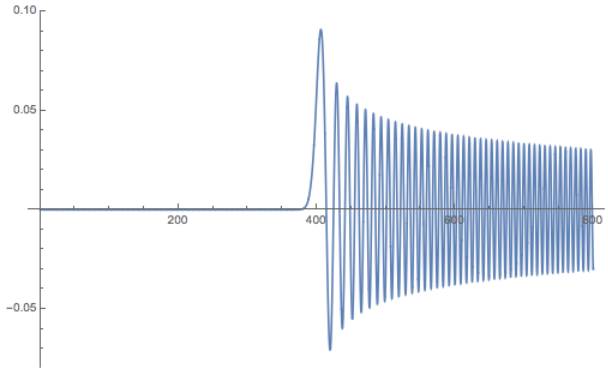
The results this and the following section are quite surprising. Looking at the instentanious frequency of a frequency modulated wave, it takes on every value between $\omega_0 \pm \omega_1$. There are an uncountably infinite number of frequencies in this band, but essentially every frequency in this band does not appear in the Fourier Transform! In fact, there are only two or three of these frequencies present in the Fourier Transform. Moreover the bandwidth for a frequency modulated wave is infinite. A countably infinite number of frequencies called *sidebands* are evenly sapced at integer multiples of ω_1 from the *carrier frequency* ω_0. Interestingly for particular values of the *modulation index* Γ it is possible for the carrier frequency to not be present in the Fourier Transform! This occures at the zeros of the zeroth Bessel Function.



(a) n=4, numPts=10,000, xMax=20, comptime=0.00082



(b) n=40, numPts=10,000, xMax=100, comptime=0.00092



(c) n=400, numPts=10,000, xMax=800, comptime=0.00200; About half the computation time went to calculating the endpoint!

FIG. 6

$$\mathcal{F}_t\left[h_0 e^{i(\omega_0 t+\phi_0+\Gamma\cos(\omega_1 t+\phi_1))}\right] = h_0 e^{i\phi_0}\mathcal{F}_t\left[e^{i\omega_0 t}e^{i\Gamma\cos(\omega_1 t+\phi_1)}\right] \tag{A1}$$

$$= h_0 e^{i\phi_0}\mathcal{F}_t\left[e^{i\omega_0 t}\right] \star \mathcal{F}_t\left[e^{i\Gamma\cos(\omega_1 t+\phi_1)}\right] \tag{A2}$$

$$= h_0 e^{i\phi_0}\delta(\omega-\omega_0) \star \mathcal{F}_t\left[\sum_{n=-\infty}^{\infty} e^{in\pi/2}J_n(\Gamma)e^{in(\omega_1 t+\phi_1)}\right] \tag{A3}$$

$$= h_0 e^{i\phi_0}\delta(\omega-\omega_0) \star \sum_{n=-\infty}^{\infty} e^{in\pi/2}e^{in\phi_1}J_n(\Gamma)F_t\left[e^{in\omega_1 t}\right] \tag{A4}$$

$$= h_0 e^{i\phi_0}\delta(\omega-\omega_0) \star \sum_{n=-\infty}^{\infty} e^{in(\phi_1+\pi/2)}J_n(\Gamma)\delta(\omega-n\omega_1) \tag{A5}$$

$$= h_0 e^{i\phi_0}\sum_{n=-\infty}^{\infty} e^{in(\phi_1+\pi/2)}J_n(\Gamma)\delta(\omega-\omega_0) \star \delta(\omega-n\omega_1) \tag{A6}$$

$$= h_0 e^{i\phi_0}\sum_{n=-\infty}^{\infty} e^{in(\phi_1+\pi/2)}J_n(\Gamma)\delta(\omega-\omega_0-n\omega_1) \tag{A7}$$

In this derivation we start by using the linearity of the Fourier Transform, then we invoke the Convolution Theorem, use the Jacobi-Anger Expansion, again use the linearity of the Fourier Transform, and the last few steps are simple Dirac Delta Function manipulations.

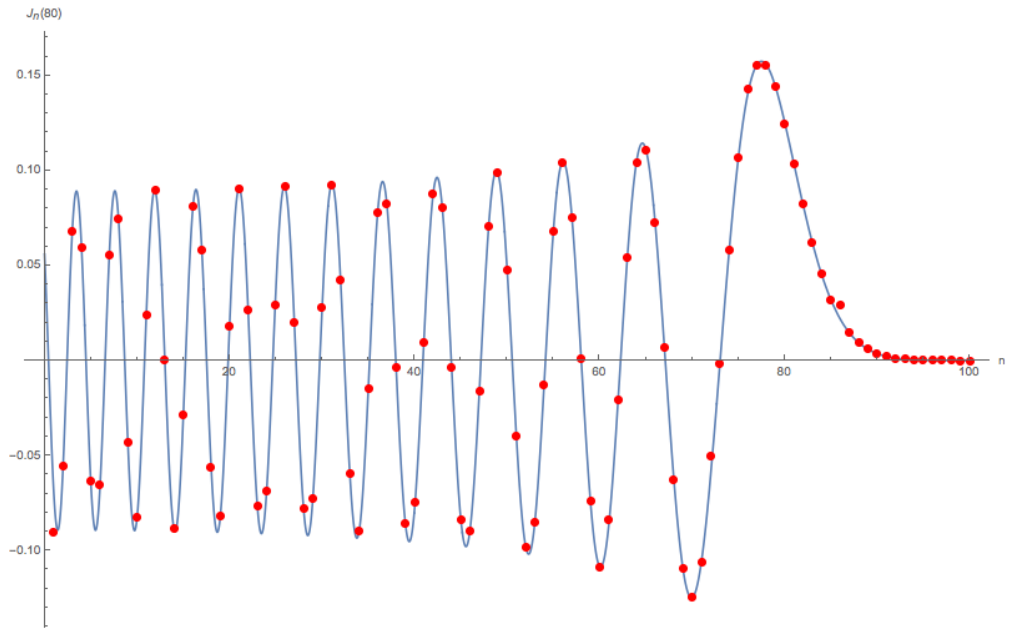


FIG. 7: This plot is a plot of $J_n(80)$ for values of $n \in (0, 100)$. The blue curve was generated by the BesselJ function of Mathematica. The red dots were calculated by my code. The two sets of data are in excelent agreement at except perhaps $n = 0, 85$ in which case there is a slight disagreement. This is a small selection of the $400,000$ data points my code generated in 0.33 seconds (on 5 year old a loptop). It solved the Bessel differential equation for 4000 steps in $\Gamma \in (0, 100)$, then saves the output, and moved on to solve Bessel's Equation for the next value of n until all values of Γ and n were solved for.

Appendix B: The Fourier Transform of The Real-Valued Function

$$\mathcal{F}_t\left[\Re\left\{h_0 e^{i(\omega_0 t+\phi_0+\Gamma\cos(\omega_1 t+\phi_1))}\right\}\right] = \frac{1}{2}\mathcal{F}_t\left[h_0 e^{i(\omega_0 t+\phi_0+\Gamma\cos(\omega_1 t+\phi_1))} + h_0 e^{-i(\omega_0 t+\phi_0+\Gamma\cos(\omega_1 t+\phi_1))}\right] \tag{B1}$$

$$= \frac{h_0}{2}\left[e^{i\phi_0}\mathcal{F}_t\left[e^{i(\omega_0 t+\Gamma\cos(\omega_1 t+\phi_1))}\right] + e^{-i\phi_0}\mathcal{F}_t\left[e^{i(-\omega_0 t-\Gamma\cos(\omega_1 t+\phi_1))}\right]\right] \tag{B2}$$

From the previous section we see that

$$\mathcal{F}_t\left[e^{i(\omega_0 t+\Gamma\cos(\omega_1 t+\phi_1))}\right] = \sum_{n=-\infty}^{\infty} e^{in(\phi_1+\pi/2)} J_n(\Gamma)\delta(\omega-\omega_0-n\omega_1) \tag{B3}$$

and by replacing ω_0 with $-\omega_0$ and Γ with $-\Gamma$, and using the fact that $J_n(-\Gamma) = (-1)^n J_n(\Gamma)$ we also have

$$\mathcal{F}_t\left[e^{i(-\omega_0 t-\Gamma\cos(\omega_1 t+\phi_1))}\right] = \sum_{n=-\infty}^{\infty} e^{in(\phi_1-\pi/2)} J_n(\Gamma)\delta(\omega+\omega_0-n\omega_1) \tag{B4}$$

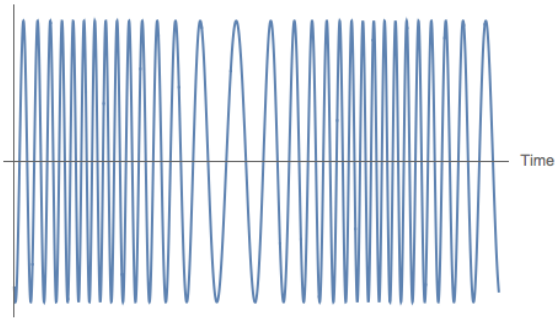
so the Fourier Transform of the real part of our frequency/phase modulated signal is

$$\begin{aligned}\mathcal{F}_t\left[\Re\left\{h_0 e^{i(\omega_0 t+\phi_0+\Gamma\cos(\omega_1 t+\phi_1))}\right\}\right] &= \frac{1}{2}h_0\left[e^{i\phi_0}\sum_{n=-\infty}^{\infty} e^{in(\phi_1+\pi/2)}J_n(\Gamma)\delta(\omega-\omega_0-n\omega_1)\right.\\ &\left.+\, e^{-i\phi_0}\sum_{n=-\infty}^{\infty} e^{in(\phi_1-\pi/2)}J_n(\Gamma)\delta(\omega+\omega_0-n\omega_1)\right]\\ &= \frac{1}{2}h_0\sum_{n=-\infty}^{\infty} e^{in\phi_1}J_n(\Gamma)\left[e^{i(\phi_0+n\pi/2)}\delta(\omega-\omega_0-n\omega_1)\right.\\ &\left.+\, e^{-i(\phi_0+n\pi/2)}\delta(\omega+\omega_0-n\omega_1)\right]\end{aligned} \tag{B5}$$

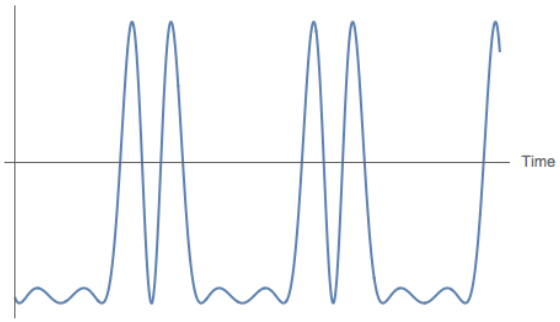
$$\hat{f}(\omega) = \frac{1}{2}h_0\sum_{n=-\infty}^{\infty} e^{in\phi_1}J_n(\Gamma)\left[e^{i(\phi_0+n\pi/2)}\delta(\omega-\omega_0-n\omega_1)+e^{-i(\phi_0+n\pi/2)}\delta(\omega+\omega_0-n\omega_1)\right] \tag{B6}$$

Notice that, by taking the fourier transform of just the real part of our frequency modulated function it has added the terms in red. (Go into more detail why that is) The other important change is that the X_n and Y_n terms were added (Again, go into more detail why that is).

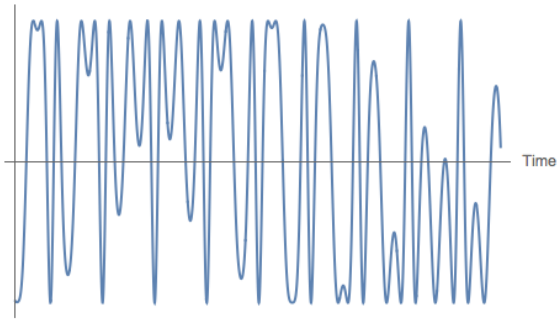
Appendix C: Selected Waveforms



(a) When the carrier frequency is much higher than the modulation frequency the frequency modulated wave is "nice". $\Gamma = 9.88, \omega_0 = 3.62, \omega_1 = 0.21, \phi_1 = 3.14$



(b) When the carrier frequency is a multiple of the modulation frequency one can get exotic waveforms. One example of the exotic signals cyclic frequency modulation can produce. $\Gamma = 2.88, \omega_0 = 1.5, \omega_1 = 0.75, \phi_1 = 0$



(c) When the carrier and modulation frequencies are almost equal the resulting FM wave is not so nice. This is a particularly nasty waveform that could be mistaken as noise by looking at it. $\Gamma = 3, \omega_0 = 0.42, \omega_1 = 0.46, \phi_1 = 0.1$

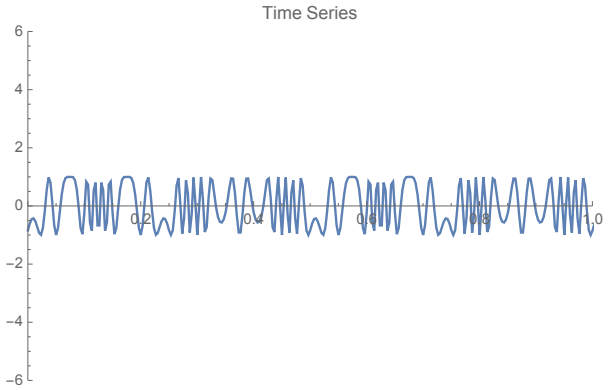
[1] Deanna Emery. A coherent three dimensional fft based search scheme for gravitational waves from binary neurton star systems. *LIGO DCC*, (T1500307-v2), September 2015.

[2] B.P. Abbott et al. (LIGO Scientific Collaboration and Virgo Collaboration). Observation of gravitational waves from a binary black hole merger. *Physical Review Letters*, 116(061102):1–16, 12 February 2016.

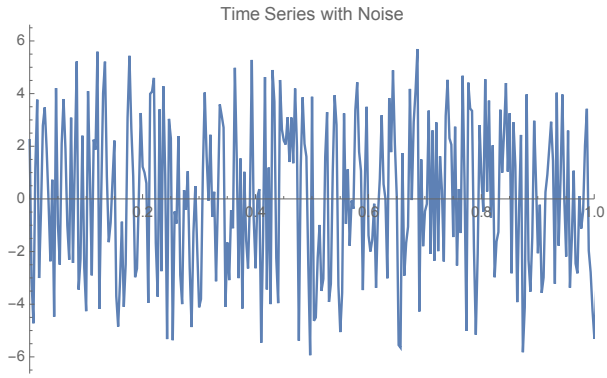
[3] Gerald B. Folland. *Fourier Analysis and its Applications*. Number ISBN 0-534-17097-3. Brooks and Cole Publishing Company, 1992.

[4] David J. Griffiths. *Introduction to Quantum Mechanics*. Number ISBN 0-13-111892-7. Pearson, second edition edition, 2005.

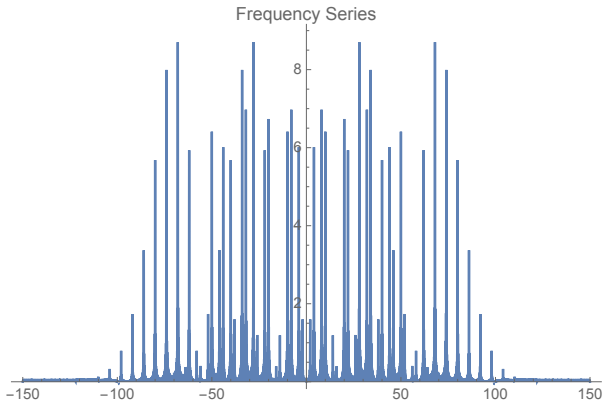
[5] Peter R. Saulson. *Fundamentals of Interferometric Gravitational Wave Detectors*. World Scientific Publishing Company, March 31, 1994.



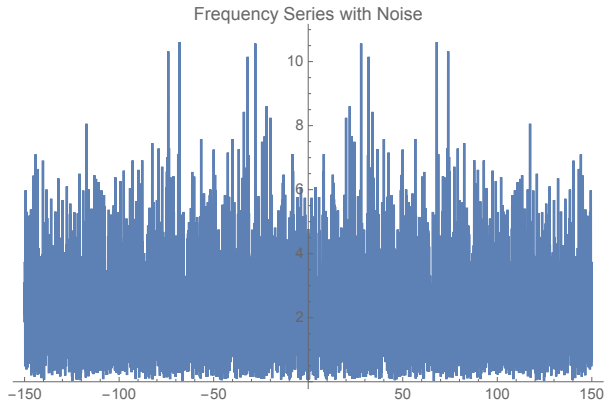
(a)



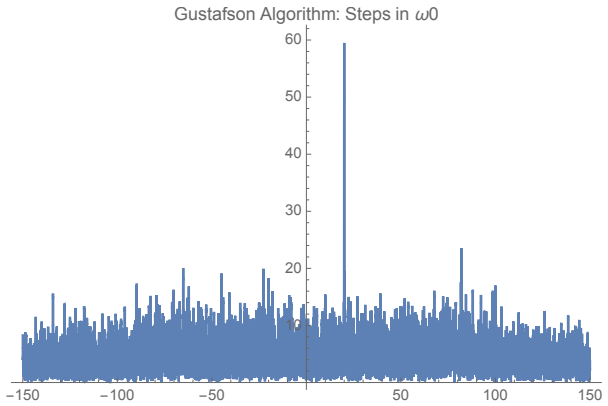
(b)



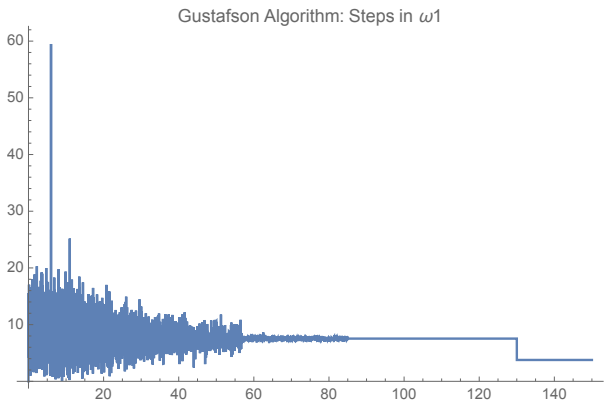
(c)



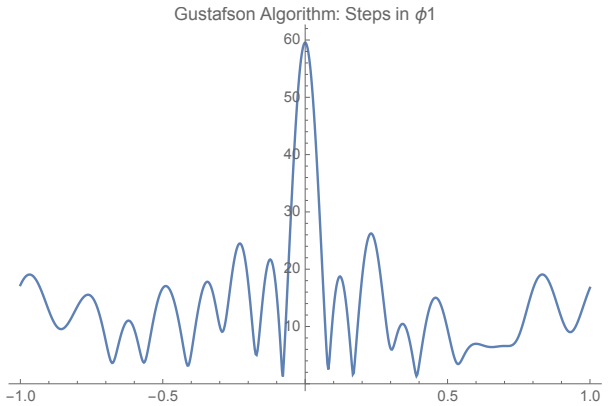
(d)



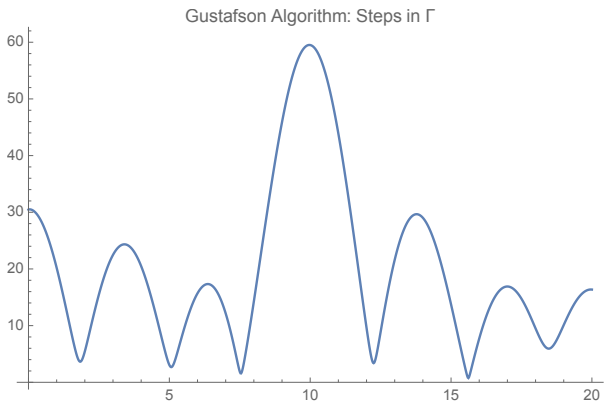
(e)



(f)



(g)



(h)

FIG. 8: Sampeling Rate = 300Hz, Carrier = 20Hz, Mod Frequency = 6Hz, Mod Index = 10, Time Series Duration = 10s