# A Coherent Search for Frequency Modulated Waves

Curtis Rau

By building on the method established in [2], this paper primarily seeks to introduce the Gustafson algorithm for removing frequency/phase modulation from a signal. The Gustafson algorithm uses the time series data as its input and outputs the original single tone carrier frequency. This method is powerful because it works in the frequency rather than time domain — which easily allows for filtering of the signal. Here, we will focus on the Gustafson algorithm's applications to LIGO and the search for gravitational waves; nevertheless, there are many other possible applications for the algorithm such as radio wave astronomy and communications systems using frequency modulated signals. Since one of the pertinent issues in any detection algorithm is signal adulteration by noise, we shall also introduce a novel approach to deal with correlated noise from a multichannel perspective.

## I. INTRODUCTION

General Relativity postulates that information about the spatial distribution of matter in the universe is transmitted via a gravitational field [1]. 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 [7]. 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 three solar masses [3]. 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 [6]. 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.

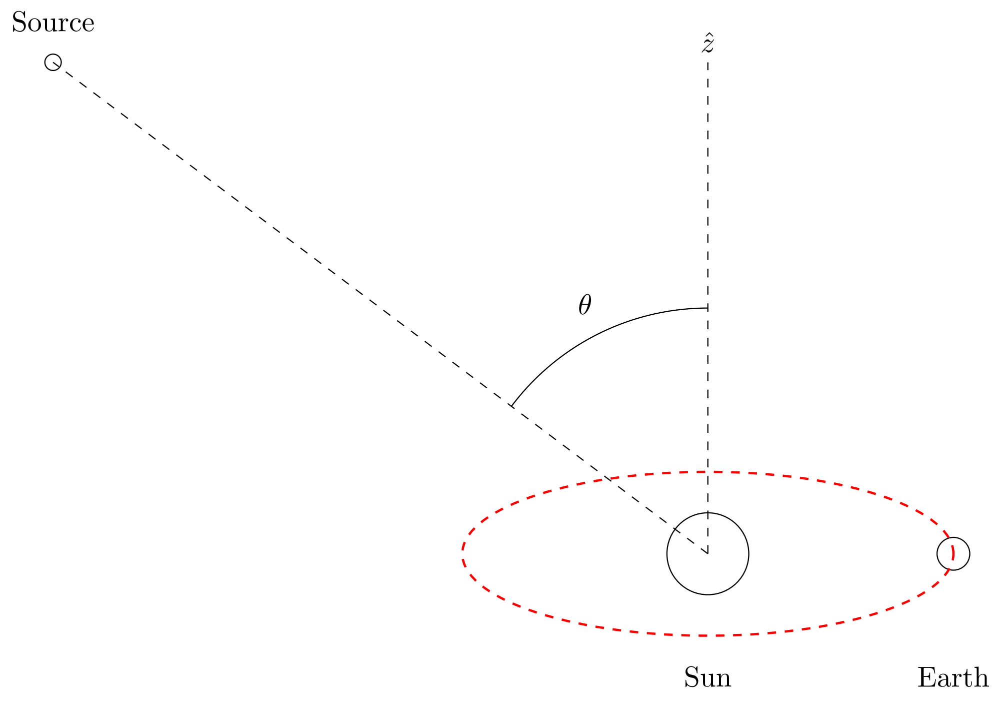


**FIG. 1**

By assuming that the wave travels at the speed of light, we can capture it as a Lorentz invariant 4-vector, **K** with the carrier frequency being the first entry and the second through fourth entries represent 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:

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

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

where $\theta$ is the polar angle from the axis perpendicular to the orbit of Earth, $\gamma$ is the Lorentz factor from special relativity, $\omega_e$ is the angular frequency of the Earth's rotation about the Sun, $k$ is the magnitude of the wave number, $c$ is the speed of light and $\beta$ 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, $\Gamma$ is the modulation index, $\omega_0$ is the frequency of the wave to be detected, $\phi_e$ is the relative phase difference between the Earth's rotation about the Sun and the wave to be detected.

An important note is that $\Gamma$ 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 scenario is that when the compact stars are in a binary system, one or both objects produce waves but the stars rotate around each other causing an analogous phase modulation. Withal, 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 $\omega_0$, $\omega_1$, $\phi_0$, $\phi_1$, and $\Gamma$ where $\phi_0$ is the phase of the wave to be detected at $t = 0$, with $\omega_1$ and $\phi_1$ being generalizations of $\omega_e$ and $\phi_e$ respectively. This is effectively a Fourier transform as a function of 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, because of the results in the previous section. 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: $N_t = N_{\omega_0} = N_{\omega_1}$ [8] [4, P. 251]. 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 $\mathcal{F}_t[f(t)](\omega) = \hat{f}(\omega)$. By invoking the convolution theorem yields,

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

By invoking the Jacobi-Anger Expansion, equation (C1), equation 12 becomes

$$\frac{h_0 e^{i\phi_0}}{\sqrt{2\pi}} \delta(\omega - \omega_0) = \hat{f}(\omega) \star \mathcal{F}_t\left[\sum_{n=-\infty}^{\infty} e^{in\pi/2} J_n(-\Gamma) e^{in(\omega_1 t+\phi_1)}\right]. \tag{13}$$

By employing equation (C2) we observe that

$$\mathcal{F}_t\left[\sum_{n=-\infty}^{\infty} e^{in\pi/2} J_n(-\Gamma) e^{in(\omega_1 t+\phi_1)}\right] = \sum_{n=-\infty}^{\infty} (-1)^n e^{in\pi/2} e^{in\phi_1} J_n(\Gamma) \mathcal{F}_t\left[e^{in\omega_1 t}\right] \tag{14}$$

$$= \frac{1}{\sqrt{2\pi}} \sum_{n=-\infty}^{\infty} e^{in(\phi_1-\pi/2)} J_n(\Gamma) \delta(\omega - n\omega_1). \tag{15}$$

In the previous step we used the linearity of the Fourier transform $\mathcal{F}_t[c \cdot f(t)] = c \cdot \mathcal{F}_t[f]$. Note that 13 becomes

$$h_0 e^{i\phi_0} \delta(\omega - \omega_0) = \hat{f}(\omega) \star \left[\sum_{n=-\infty}^{\infty} e^{in(\phi_1-\pi/2)} J_n(\Gamma) \delta(\omega - n\omega_1)\right]. \tag{16}$$

By performing the convolution,

$$\hat{f}(\omega) \star \delta(\omega - n\omega_1) = \int \hat{f}(\omega - \widetilde{\omega}) \delta(\widetilde{\omega} - n\omega_1) d\widetilde{\omega} \tag{17}$$

$$= \hat{f}(\omega - n\omega_1) \tag{18}$$

brings us to the almost final answer

$$h_0 e^{i\phi_0} \delta(\omega - \omega_0) = \sum_{n=-\infty}^{\infty} e^{in(\phi_1-\pi/2)} J_n(\Gamma) \hat{f}(\omega - n\omega_1). \tag{19}$$

For clarity and later convenience let's have $n \to -n$. Let us employ a property of the Bessel functions (C3) This results in the Gustafson Algorithm:

$$\boxed{h_0 e^{i\phi_0}\delta(\omega-\omega_0) = 2\sum_{n=-\infty}^{\infty} e^{-in(\phi_1+\pi/2)} J_n(\Gamma)\hat{f}(\omega+n\omega_1)} \tag{20}$$

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

$$\Re\{a\cdot b\} \neq \Re\{a\}\cdot\Re\{b\}; \qquad a, b \in \mathbb{C}. \tag{21}$$

Naively we try plugging the Fourier Transform of the real waveform (eq. B6) into the Gustafson Algorithm (eq. 20) and see what happens. Perhaps unsurprisingly using ether the real or complex waveforms yield the same result. Notice that the Fourier Transform of the real wave looks very similar to the Fourier Transform of the complex wave (see appendices A and B). The difference being that the Fourier Transform of the real is symmetric about $\omega = 0$ up to a phase factor. The vectors corresponding to the positive carrier frequency rotate counterclockwise, whereas the vectors corresponding to the negative carrier frequency rotate clockwise (neglecting the rotation due to $\phi_1$ which is the same for both). **This phase factor is the reason why the Gustafson Algorithm only picks out the positive value of $\omega_0$.** If it is unclear what this means refer to figure (7e).

Now we will try to build an intuitive conceptual model of the Gustafson Algorithm. The complex numbers form a two dimentional vector field over the real numbers. This means the Fourier Transform is a collection of three dimentional vectors. If we plot the vectors in cylindrical coordinates, where the z-axis is frequency, the distance from the z-axis represents the amplitude of that frequency, and the polar angle represents the phase of that frequency component, then we get something like figure (2).

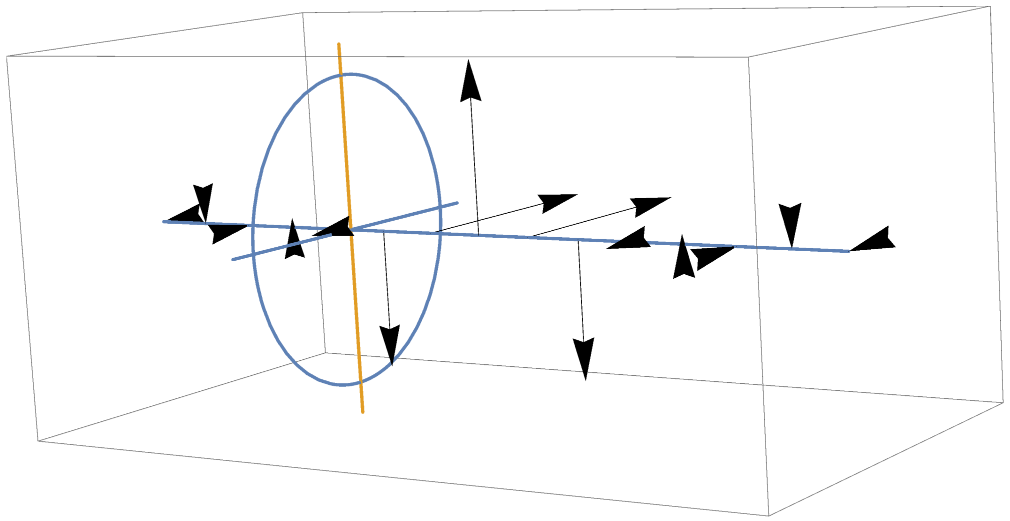


**FIG. 2:** The 3D representation of the Fourier Transform of the complex waveform.

The Gustafson Algorithm rotates these vectors, multiplies them by a constant, and takes the vector sum of them. Apparently this vector sum is zero unless the frequency $\omega$ is exactly equal to $\omega_0$ in which case they sum to a vector of infinite length. The unincluded vectors of the real waveform in figure (2) are symmetric about $\omega = 0$ except that they rotate the opposite way around the frequency axis. This asymmetry in phase is why the Gustafson Algorithm is asymmetric. It rotates all vectors the same direction. Apparently the algorithm rotates vectors corresponding to $+\omega_0$ so that they constructively add, whereas it rotates vectors corresponding to $-\omega$ in the wrong direction so they distructively add.

## A. The Gustafson Algorithm as a Search Algorithm

It is not explicitly obvious how The Gustafson Algorithm, Eq. 20, 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. 20 can do.

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

$$f(t) = \Re\left\{h_0 e^{i(\omega_0 t+\phi_0+\Gamma\cos(\omega_1 t+\phi_1))}\right\} \tag{22}$$

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 Search Algorithm.

$$G_{\widetilde{\omega}_0,\widetilde{\omega}_1,\widetilde{\phi}_1,\widetilde{\Gamma}}\left[\hat{f}(\omega)\right] = \sum_{n=-\infty}^{\infty} e^{-in(\widetilde{\phi}_1+\pi/2)} J_n(\widetilde{\Gamma})\hat{f}(\omega+n\widetilde{\omega}_1) \tag{23}$$

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

$$\begin{aligned} G_{\omega_0,\omega_1,\phi_1,\Gamma}\left[\hat{f}(\omega)\right] =& h_0 \sum_{m,n=-\infty}^{\infty} e^{i(m\phi_1-n\widetilde{\phi}_1)} J_n(\widetilde{\Gamma})J_m(\Gamma) \\ &\times\left[e^{i(\phi_0+(m-n)\pi/2)}\delta(\omega-\omega_0+n\widetilde{\omega}_1-m\omega_1)+e^{-i(\phi_0+(m+n)\pi/2)}\delta(\omega+\omega_0+n\widetilde{\omega}_1-m\omega_1)\right] \end{aligned} \tag{24}$$

Notice that the erroneous factor of two in front of the Gustafson Algorithm (Eq. 20) has canceled with the factor of one half in front of the Fourier Transform of the real waveform (Eq. B6). This equation is far to complicated to see what is going on, so we will treat it in cases.

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 in detail. 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

$$0 = \omega-\omega_0+n\widetilde{\omega}_1-m\omega_1 \tag{25}$$

$$\omega_0-\omega = n\widetilde{\omega}_1-m\omega_1 \tag{26}$$

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

$$\gcd(\widetilde{\omega}_1,\omega_1) = \omega_0-\omega \tag{27}$$

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

$$G_{\widetilde{\omega}_0,\widetilde{\omega}_1,\widetilde{\phi}_1,\widetilde{\Gamma}}\left[\hat{f}(\omega)\right] = h_0 e^{i\phi_0}\delta(0)J_n(\widetilde{\Gamma})J_m(\Gamma)\cos(m\phi_1-n\widetilde{\phi}_1)\times\begin{cases}(-1)^{(m-n)/2} & ,m+n \text{ is even}\\ i(-1)^{m-n} & ,m+n\text{ is odd}\end{cases} \tag{28}$$

Note that the only way this could give us the true amplitude $h_0e^{i\phi_0}$ is if $\widetilde{\Gamma}=\Gamma=0$, $\widetilde{\phi}_1=\phi_1$, and $m=n=0$. This is because the Bessel Functions are always less than or equal to one. The only time equality holds is for $J_0(0)=1$. Moreover if $m=n=0$ then $\widetilde{\omega}_0=\omega_0$ which violates our assumption. Therefore in this case, when $\omega\neq\omega_0$ and $\omega_1\neq\widetilde{\omega}_1$, the amplitude of the greatest peak will always be less than the true amplitude of the wave.

**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

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

This could happen never, or a countably infinite number of times. If this is false, then the sum is zero.

**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

$$\omega - \omega_0 = (n-m)\widetilde{\omega}_1 \tag{30}$$

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

**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

$$\frac{h_0}{2} e^{i\phi_0} \delta(0) \sum_{n=-\infty}^{\infty} e^{in(\phi_1 - \widetilde{\phi}_1)} J_n(\widetilde{\Gamma}) J_n(\Gamma) \tag{31}$$

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

$$\frac{h_0}{2} e^{i\phi_0} \delta(0) \sum_{n=-\infty}^{\infty} J_n^2(\Gamma) = \frac{h_0}{2} e^{i\phi_0} \delta(0) \tag{32}$$

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 true amplitude of the wave, as indicated by the Gustafson Algorithm.

## B. Stepping over the Modulation Index

How big should the step size over the modulation index be?

Should it be constant, or a function of the modulation index itsself?

$$\begin{aligned} G_{\omega_0,\omega_1,\phi_1,\Gamma}\left[\hat{f}(\omega)\right] =& h_0 \sum_{m,n=-N_{sb}(\Gamma)}^{N_{sb}(\Gamma)} e^{i(m\phi_1 - n\widetilde{\phi}_1)} J_n(\widetilde{\Gamma}) J_m(\Gamma) \\ & \times \left[ e^{i(\phi_0 + (m-n)\pi/2)} \delta(\omega - \omega_0 + n\widetilde{\omega}_1 - m\omega_1) + e^{-i(\phi_0 + (m+n)\pi/2)} \delta(\omega + \omega_0 + n\widetilde{\omega}_1 - m\omega_1) \right] \end{aligned} \tag{33}$$

assume all parameters are correct except the modulation index. ($\omega = \omega_0$, $\phi_1 = \widetilde{\phi}_1$, and $\omega_1 = \widetilde{\omega}_1$)

$$\begin{aligned} G_{\omega_0,\omega_1,\phi_1,\Gamma}\left[\hat{f}(\omega)\right] =& h_0 \sum_{m,n=-N_{sb}(\Gamma)}^{N_{sb}(\Gamma)} e^{i(m-n)\phi_1} J_n(\widetilde{\Gamma}) J_m(\Gamma) \\ & \times \left[ e^{i(\phi_0 + (m-n)\pi/2)} \delta((n-m)\omega_1) + e^{-i(\phi_0 + (m+n)\pi/2)} \delta(2\omega_0 + (n-m)\omega_1) \right] \end{aligned} \tag{34}$$

this expression is only non-zero when m=n or $\frac{2\omega_0}{(m-n)} = \omega_1$ we ignore the second case for the time being. assuming m=n we have

$$G_{\omega_0,\omega_1,\phi_1,\Gamma}\left[\hat{f}(\omega)\right] = h_0 \delta(0) e^{i\phi_0} \sum_{n=-N_{sb}(\Gamma)}^{N_{sb}(\Gamma)} J_n(\widetilde{\Gamma}) J_n(\Gamma) \tag{35}$$

using the property of the Bessel Functions given in equation (C6) and the fact that terms outside of the finite sum are vanishingly small we have

$$G_{\omega_0,\omega_1,\phi_1,\Gamma}\left[\hat{f}(\omega)\right] = h_0 \delta(0) e^{i\phi_0} J_0(\widetilde{\Gamma} - \Gamma) \tag{36}$$

The question is, how many steps should we take in Γ when performing the search? This presents an interesting oppertunity to break out the asymptotic form of the bessel functions:

so

$$J_0(x) \to \sqrt{\frac{2}{\pi x}} \cos\left(x - \frac{\pi}{4}\right) \tag{37}$$

so the periodicity of $J_0$ is roughly $2\pi$ – an approximation which is valid for all values of Γ.

$$\boxed{\partial\Gamma = \frac{2\pi}{\nu_\Gamma}} \tag{38}$$

the number of steps in modulation index should be

$$\boxed{N_\Gamma = \left[\!\left[ \frac{(\Gamma_{max} - \Gamma_{min})}{2\pi} \right]\!\right] \nu_\Gamma + 1} \tag{39}$$

where $\nu_\Gamma$ is the number of steps in Γ over an interval of length $2\pi$. Valuse as low as 10 for $\nu_\Gamma$ seem appropriate. We can take so few steps in gamma because (36) is such a smooth function.

This is ture in the high SNR domain, but what about when there is a lot of noise? Then we have

$$G_{\omega_0,\omega_1,\phi_1,\Gamma}\left[\hat{f}(\omega)\right] = h_0\delta(0)e^{i\phi_0} \sum_{n=-N_{sb}(\Gamma)}^{N_{sb}(\Gamma)} J_n(\Gamma)(J_n(\widetilde{\Gamma}) + Rand(n)) \tag{40}$$

where rand(n) is a random number that has been added onto each sideband. We can distribute and write

$$G_{\omega_0,\omega_1,\phi_1,\Gamma}\left[\hat{f}(\omega)\right] = h_0\delta(0)e^{i\phi_0}\left[\sum_{n=-N_{sb}(\Gamma)}^{N_{sb}(\Gamma)} J_n(\Gamma)J_n(\widetilde{\Gamma}) + \sum_{n=-N_{sb}(\Gamma)}^{N_{sb}(\Gamma)} J_n(\Gamma)Rand(n)\right] \tag{41}$$

$$= h_0\delta(0)e^{i\phi_0}\left[J_0(\widetilde{\Gamma} - \Gamma) + \sum_{n=-N_{sb}(\Gamma)}^{N_{sb}(\Gamma)} J_n(\Gamma)Rand(n)\right] \tag{42}$$

in the high noise (define high noise, may be actually close to SNR = unity!!) limit this is

$$G_{\omega_0,\omega_1,\phi_1,\Gamma}\left[\hat{f}(\omega)\right] = h_0\delta(0)e^{i\phi_0} \sum_{n=-N_{sb}(\Gamma)}^{N_{sb}(\Gamma)} J_n(\Gamma)Rand(n) = random \tag{43}$$

which tells us nothing about the true modulation index. This is just some random smooth function.

If the SNR isn't too large we can hope to detect GW's with this algorithm.

### C. Stepping over the Relative Phase

How big should the step size over the phase be?

Should it be constant, or a function of the phase itsself?

$$\begin{aligned} G_{\omega_0,\omega_1,\phi_1,\Gamma}\left[\hat{f}(\omega)\right] =& h_0 \sum_{m,n=-N_{sb}(\Gamma)}^{N_{sb}(\Gamma)} e^{i(m\phi_1 - n\widetilde{\phi}_1)} J_n(\widetilde{\Gamma})J_m(\Gamma) \\ &\times \left[e^{i(\phi_0+(m-n)\pi/2)}\delta(\omega - \omega_0 + n\widetilde{\omega}_1 - m\omega_1) + e^{-i(\phi_0+(m+n)\pi/2)}\delta(\omega + \omega_0 + n\widetilde{\omega}_1 - m\omega_1)\right] \end{aligned} \tag{44}$$

assume all parameters are correct except the phase index. ($\omega = \omega_0$, $\Gamma = \widetilde{\Gamma}$, and $\omega_1 = \widetilde{\omega}_1$)

$$G_{\omega_0,\omega_1,\phi_1,\Gamma}\left[\hat{f}(\omega)\right] = h_0 \sum_{m,n=-N_{sb}(\Gamma)}^{N_{sb}(\Gamma)} e^{i(m\phi_1 - n\widetilde{\phi}_1)} J_n(\Gamma) J_m(\Gamma) \times \left[e^{i(\phi_0+(m-n)\pi/2)}\delta((n-m)\omega_1) + e^{-i(\phi_0+(m+n)\pi/2)}\delta(2\omega_0 + (n-m)\omega_1)\right] \tag{45}$$

this expression is only non-zero when m=n or $\frac{2\omega_0}{(m-n)} = \omega_1$ we ignore the second case for the time being. assuming m=n we have

$$G_{\omega_0,\omega_1,\phi_1,\Gamma}\left[\hat{f}(\omega)\right] = h_0\delta(0)e^{i\phi_0} \sum_{n=-N_{sb}(\Gamma)}^{N_{sb}(\Gamma)} e^{in(\phi_1-\widetilde{\phi}_1)} J_n^2(\Gamma) \tag{46}$$

$$= h_0\delta(0)e^{i\phi_0} J_0\left(2\Gamma \sin\left(\frac{\phi_1 - \widetilde{\phi}_1}{2}\right)\right) \tag{47}$$

Invoking the asymptotic expansion from the previous section

$$J_0\left(2\Gamma \sin\left(\frac{\phi_1 - \widetilde{\phi}_1}{2}\right)\right) \approx \frac{\sin\left(2\Gamma \sin\left|\frac{\phi_1-\widetilde{\phi}_1}{2}\right|\right)}{\sqrt{\pi\Gamma\sin\left|\frac{\phi_1-\widetilde{\phi}_1}{2}\right|}} \tag{48}$$

The step size in relative phase should be

$$\boxed{\partial\phi_1(\Gamma) = \frac{\pi}{[\![\Gamma]\!]\nu_{\phi_1}}} \tag{49}$$

and the number of steps would be

$$\boxed{N_{\phi_1}(\Gamma) = 2[\![\Gamma]\!]\nu_{\phi_1} + 1} \tag{50}$$

where $\nu_{\phi_1}$ is the number of steps in $\phi_1$ over roughly one oscillation of (???). STEP SIZE IN RELATIVE PHASE IS DEPENDENT ON MODULATION INDEX!!!!!

This suggests there is a natural order to perform the search in. The outer most for loop should be over modulation index, then the second for loop should be over relative phase, the inner loops would be over the frequencies, hoverver there is no perfered oreder to these.

## D. Stepping over the frequencies

The infinite limits on the sums must be made finite in any computational implementation. By the ***Riemann-Lebesgue Lemma:*** *If $f \in L^1$, then $\hat{f}(\omega) \to 0$ as $\omega \to \pm\infty$ [4, P.217].* this is well justified. The question is, what is an appropriate number of sidebands to sum over? Power is distributed to higher order sidebands as the modulation index increases, thus the number of sidebands to include in the sum should be determined by $\Gamma$. Upon inspection of the Fourier Transform plotted for various values of $\Gamma$ a trend appears. Going out from the carrier frequency in ether direction, the power in the sidebands oscillates around, slowly increasing until a maximum is reached, after which the power rapidly goes to zero. We therefore want to find the value of $n$ that maximizes $J_n(\Gamma)$ for any value of $\Gamma$. This $n$, multiplied by some constant, gives the number of sidebands to include in the sum. The following relation, which could not be found in existing literature, was discovered while researching this project. It will be stated without proof.

***Rau's Theorem:*** *For any $z \in (\pi, \infty)$ the value of $\nu$ that maximizes $J_\nu(z)$ asymtotically approaches $\nu \to z - \log_\pi(z)$.*

We can use this to calculate how many sidebands we need to sum over.

$$N_{sb}(\Gamma) = \left[\!\left[ 3 + 1.1 \left( \Gamma - \left\{ \begin{array}{ll} 1 & : \Gamma \leq \pi \\ \log_\pi \Gamma & : \Gamma > \pi \end{array} \right. \right) \right]\!\right] \tag{51}$$

The multiplicative factor of 1.5 and the additive constant of 1 are to make the relation work for smaller modulation indicies. This ensures at least 98% of the power is included in the sum for all values of $\Gamma$.

-the size of the bessel matrix is based on the max value of gamma in the search. -performing the algorithm suggests that steps in gamma may not need to be very small because we're not stepping over a delta function

## E. How Long should it take?

$$\boxed{N_\Gamma = \left[\!\left[ \frac{(\Gamma_{max} - \Gamma_{min})}{2\pi} \right]\!\right] \nu_\Gamma + 1} \tag{52}$$

$$\boxed{N_{\phi_1}(\Gamma) = 2[\![\Gamma]\!]\nu_{\phi_1} + 1} \tag{53}$$

$$\boxed{N_{f_0} = \left[\!\left[ \frac{(f_{0max} - f_{0min})}{\partial f} \right]\!\right] + 1} \tag{54}$$

$$\boxed{N_{f_1} = \left[\!\left[ \frac{(f_{1max} - f_{1min})}{\partial f} \right]\!\right] + 1} \tag{55}$$

$$N_{sb}(\Gamma) = \left[\!\left[ 3 + 1.1 \left( \Gamma - \left\{ \begin{array}{ll} 1 & : \Gamma \leq \pi \\ \log_\pi \Gamma & : \Gamma > \pi \end{array} \right. \right) \right]\!\right] \tag{56}$$

Comp time

$$\sum_{N_\Gamma} N_{\phi_1} N_{f_0} N_{f_1} N_{sb} = N_{f_0} N_{f_1} \sum_{N_\Gamma} (2[\![\Gamma]\!]\nu_{\phi_1} + 1) \left[\!\left[ 3 + 1.1 \left( \Gamma - \left\{ \begin{array}{ll} 1 & : \Gamma \leq \pi \\ \log_\pi \Gamma & : \Gamma > \pi \end{array} \right. \right) \right]\!\right] \tag{57}$$

## III. IMPLEMENTING THE GUSTAFSON ALGORITHM IN C++

## IV. CALCULATING A SINGLE BESSEL FUNCTION AT A SINGLE POINT: $J_n(z)$ FOR A SINGLE $n$ AND $z$

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

$$J_n(z) = \sum_{m=0}^{\infty} \frac{(-1)^n}{m!(m+n)!} \left( \frac{z}{2} \right)^{2m+n} . \tag{58}$$

Hence, calculating the $n$-th Bessel function in this manner involves calculating factorials greater than $n!$. The highest $n$ for which we can store $n!$ in an unsigned long integer type is 20:

$$2^{64} - 1 = 18,446,744,073,709,551,615 \tag{59}$$

$$< 21! = 51,090,942,171,709,440,000. \tag{60}$$

If we are willing to sacrifice some precision to truncation (which is a good idea if we wish to persue this avenue), we can use a double precision floating point integer which allows values upto $1.797,693,134,862,315,7 \cdot 10^{308}$

$$170! \approx 7.25 \cdot 10^{306} < 1.797,693,134,862,315,7 \cdot 10^{308} < 171! \approx 1.24 \cdot 10^{309} \tag{61}$$

This is not acceptable however, because $|J_n(z)| \leq 1;\ \forall n \wedge z$. Therefore, 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 impractical, but it is by far one of the slowest ways one could calculate values of the Bessel functions.

This whole business of wrestling with factorials can be avoided by calculating the Bessel Functions using the integrad definition of a Bessel Function

$$J_n(z) = \frac{1}{\pi}\int_0^\pi \cos\left[z\sin\theta - n\theta\right]d\theta. \tag{62}$$

Which is computed using a trapazoidal Riemann sum:

$$J_n(z) \approx \frac{1}{N}\left(\frac{1+\cos(n\pi)}{2} + \sum_{i=1}^{N-1}\cos\left[z\sin(i\cdot dx) - n\cdot i\cdot dx\right]\right); \qquad dx = \frac{\pi}{N} \tag{63}$$

The function being integrated becomes more oscillatory as $n$ and $z$ increase. 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. We will use the number of zeros of the integrand as a measure of how oscillatory it is, and dynamically determin the number of integration points based on that fact. The integrand being a transcendental function makes the explicit formula for the number of zeros unruley, though it does exist.

$$N_0 = 2\left\lfloor \frac{\sqrt{z^2-n^2} - n\arccos(n/z)}{\pi} + \frac{1}{2}\right\rfloor + \left\lfloor n + \frac{1}{2}\right\rfloor \tag{64}$$

The above formula will be off by +1 whenever the expression being floored is exactly an integer. We can put an upper bound on the number of zeros however in an easy way.

$$N_0 \leq 2\left\lfloor \frac{z}{\pi} + \frac{1}{2}\right\rfloor + \left\lfloor n + \frac{1}{2}\right\rfloor \tag{65}$$

The upper bound on the number of zeros is the one we use to determine the number of integration points. The exact expression takes longer to compute, possibly longer than it takes to perform the sum over the additional integration points. 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

$$N = 100\left(2\left\lfloor \frac{z}{\pi} + \frac{1}{2}\right\rfloor + \left\lfloor n + \frac{1}{2}\right\rfloor + 1\right) \tag{66}$$

The eronious +1 at the end is to ensure that if $n$ and $z$ are small, there will still be enough integration points to ensure accurate results. To finish off the calculation an if-statement is added to the beginning of the function to check if $z = 0$. In this case the condition

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

The algorithm can have problems when $J$ is extremely small. We achieved excellent results (accurate consistently to 16 decimal places) for a wide range of $z$ and $n$.

## V. CALCULATING MANY BESSEL FUNCTIONS AT ONE POINT: $J_n(z)$ FOR MANY $n$ AND A SINGLE $z$

The Gustafson Algorithm requires many values of the Bessel Functions. Specifically to search over a particular modulation index one needs many values of $J_n(\Gamma)$ for a range of $n$. Rather than using the numeric integration method from the previous section to calculate these values one at a time it is possible to compute them all at once. This often reduces the computation time by many orders of magnitude.

We start with the recursion relation

$$J_{n-1}(z) - \frac{2n}{z}J_n(z) + J_{n+1}(z) = 0 \tag{67}$$

The system of equations implicit in the above equation, written explicitly, is

$$-\frac{2}{z}J_1 + J_2 = -J_0 \tag{68}$$

$$J_1 - \frac{4}{z}J_2 + J_3 = 0 \tag{69}$$

$$J_2 - \frac{6}{z}J_3 + J_4 = 0 \tag{70}$$

$$\vdots \tag{71}$$

$$J_{n-2} - \frac{2(n-1)}{z}J_{n-1} + J_n = 0 \tag{72}$$

$$J_{n-1} - \frac{2n}{z}J_n = -J_{n+1} \tag{73}$$

This can be expressed as a tridiagonal matrix equation which is also easily solved via the Thomas Algorithm. In C++ indexing:

$$\begin{pmatrix} B_0 & 1 & & & & \\ 1 & B_1 & 1 & & & \\ & 1 & B_2 & 1 & & \\ & & \vdots & \vdots & & \\ & & 1 & B_{N-4} & 1 \\ & & & 1 & B_{N-3} \end{pmatrix} \begin{pmatrix} J_1 \\ J_2 \\ J_3 \\ \vdots \\ J_{N-3} \\ J_{N-2} \end{pmatrix} = \begin{pmatrix} -J_0 \\ 0 \\ 0 \\ \vdots \\ 0 \\ -J_{N-1} \end{pmatrix} \tag{74}$$

where $B_i$ is

$$B_i = -\frac{2(i+1)}{z} \tag{75}$$

Notice the $J_i$ vector is indexed from 1. This is because the first and last elements of $J$ are not being solved for. They are the boundary conditions, and they must be calculated via the numeric integration method. In practice this is the bottle neck for this method, as the results section will show. The Thomas Algorithm will be covered in detail in a proceeding section.

-error in values increases as N increases.

## VI. CALCULATING A SINGLE BESSEL FUNCTION AT MANY POINTS: $J_n(z)$ FOR A SINGLE $n$ AND MANY $z$

An alternative method to calculating an array of Bessel Functions which uses the same trick from the previous section is presented here. This time we calculate Bessel Function values for a fixed $n$ and a range of $z$.

This will be acomplished by solving Bessel's Equation [4].

$$x^2 J_n''(x) + xJ_n'(x) + (x^2 - n^2)J_n(x) = 0 \tag{76}$$

The standard prescription for numerically solving differential equations is first to discretize the independent variable $x \to x_i = x_{min} + i \cdot \delta x$, and then employ the limit definitions of the derivatives. We will use the three point definitions because their error goes as $\mathcal{O}(\delta x^2)$ rather than using the two point definition with an error that goes as $\mathcal{O}(\delta x)$ [**?** ]. Both the two point and three point methods take the same ammount of time to compute. Further, because there are only one order of Bessel functions in Bessel's equation, the order is implied by the $n$ that appears in equation (76), so we will drop the subscript of $n$, $J_n \to J$, and adopt the notation $J(x_i) = J_i$.

$$J_i' = \frac{J_{i+1} - J_{i-1}}{2\delta x} \tag{77}$$

$$J_i'' = \frac{J_{i-1} - 2J_i + J_{i+1}}{\delta x^2}. \tag{78}$$

Therefore the discretized form of Bessel's equation is

$$x_i^2 \frac{J_{i-1} - 2J_i + J_{i+1}}{\delta x^2} + x\frac{J_{i+1} - J_{i-1}}{2\delta x} + (x^2 - n^2)J_i = 0 \tag{79}$$

and by collecting like terms of $J_i$ we have

$$\left(\frac{x_i^2}{\delta x^2} - \frac{x_i}{2\delta x}\right) J_{i-1} + \left(\left(1 - \frac{2}{\delta x^2}\right) x_i^2 - n^2\right) J_i + \left(\frac{x_i^2}{\delta x^2} + \frac{x_i}{2\delta x}\right) J_{i+1} = 0 \tag{80}$$

For convenience we define coefficients to simplify the above equation.

$$a_i = \frac{x_i^2}{\delta x^2} - \frac{x_i}{2\delta x} \tag{81}$$

$$b_i = \left(1 - \frac{2}{\delta x^2}\right) x_i^2 - n^2 \tag{82}$$

$$c_i = \frac{x_i^2}{\delta x^2} + \frac{x_i}{2\delta x} \tag{83}$$

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

$$a_i = i\left(i - \frac{1}{2}\right) \tag{84}$$

$$b_i = i^2\left(\delta x^2 - 2\right) - n^2 \tag{85}$$

$$c_i = i\left(i + \frac{1}{2}\right) \tag{86}$$

In effect, the discretized Bessel's equation with simplified coefficients is

$$a_i J_{i-1} + b_i J_i + c_i J_{i+1} = 0 \tag{87}$$

By writing out the system of equations explicitly, we have:

$$b_1 J_1 + c_1 J_2 = -a_1 J_0 \tag{88}$$

$$a_2 J_1 + b_2 J_2 + c_2 J_3 = 0 \tag{89}$$

$$a_3 J_2 + b_3 J_3 + c_3 J_4 = 0 \tag{90}$$

$$\vdots \tag{91}$$

$$a_{n-1} J_{n-2} + b_{n-1} J_{n-1} + c_{n-1} J_n = 0 \tag{92}$$

$$a_n J_{n-1} + b_n J_n = -c_n J_{n+1} \tag{93}$$

This suggests Bessel's equation can be written as a matrix equation:

$$\begin{pmatrix} b_1 & c_1 & & & & \\ a_2 & b_2 & c_2 & & & \\ & a_3 & b_3 & c_3 & & \\ & & \vdots & \vdots & & \\ & & a_{n-1} & b_{n-1} & c_{n-1} \\ & & & a_n & b_n \end{pmatrix} \begin{pmatrix} J_1 \\ J_2 \\ J_3 \\ \vdots \\ J_{n-1} \\ J_n \end{pmatrix} = \begin{pmatrix} -a_1 J_0 \\ 0 \\ 0 \\ \vdots \\ 0 \\ -c_n J_{n+1} \end{pmatrix}. \tag{94}$$

where we have omitted the zeros for readability. This is a tridiagonal matrix system which can be easily solved using the standard Thomas Algorithm. To program this in C++ we need all the arrays to be indexed from 0. By virtue of this indexing system, we yield:

$$\begin{pmatrix} B_0 & C_0 & & & & \\ A_1 & B_1 & C_1 & & & \\ & A_2 & B_2 & C_2 & & \\ & & \vdots & \vdots & & \\ & & A_{N-4} & B_{N-4} & C_{N-4} \\ & & & A_{N-3} & B_{N-3} \end{pmatrix} \begin{pmatrix} J_1 \\ J_2 \\ J_3 \\ \vdots \\ J_{N-3} \\ J_{N-2} \end{pmatrix} = \begin{pmatrix} -A_0 J_0 \\ 0 \\ 0 \\ \vdots \\ 0 \\ -C_{N-3} J_{N-1} \end{pmatrix} \tag{95}$$

$$A_i = a_{i+1} = (i+1)\left(i + \frac{1}{2}\right) \tag{96}$$

$$B_i = b_{i+1} = (i+1)^2 \left(\delta x^2 - 2\right) - n^2 \tag{97}$$

$$C_i = c_{i+1} = (i+1)\left(i + \frac{3}{2}\right) \tag{98}$$

Again, notice the $J$ vector is indexed starting from 1 for the same reason as before. The next section explains how to solve these tridiagonal matrix systems.
-error in values decreases as N increases

## VII. THE THOMAS ALGORITHM

The Thomas Algorithm is a lightning fast way to solve tridiagonal systems like equation 95. 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 $S_i$; $\quad \forall i \in [0, N-3]$.

$$R_i = R_i - \frac{A_i}{B_{i-1}} R_{i-1}; \qquad \forall i \in [1, N-3] \tag{99}$$

$$B_i \to B_i - \frac{A_i}{B_{i-1}} C_{i-1}; \qquad \forall i \in [1, N-3] \tag{100}$$

$$S_i \to S_i - \frac{A_i}{B_{i-1}} S_{i-1}; \qquad \forall i \in [1, N-3] \tag{101}$$

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

$$R_i = R_i - \frac{C_i}{B_{i+1}} R_{i+1}; \qquad \forall i \in [N-4, 0] \tag{102}$$

$$S_i \to S_i - \frac{C_i}{B_{i+1}} S_{i+1}; \qquad \forall i \in [N-4, 0] \tag{103}$$

$$J_i = \frac{S_{i-1}}{B_{i-1}}; \qquad \forall i \in [N-2, 1] \tag{104}$$

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

## VIII. BEST APPROACH FOR CALCULATING BESSEL FUNCTIONS

If one is only using the Gustafson Algorithm to reconstruct the power contained in a signal then values of Bessel Functions for a constant modulation index $\Gamma$ and a range of $n$ are needed. Clearly the method one should use to calculate the necessary Bessel Functions for this case is the one outlined in section (V).

However if one intends to use the Gustafson Algorithm as a search algorithm over a range of modulation indices, then one needs a matrix of Bessel Function values running over a range of $n$ and $z$. The question becomes, which method is best? Since speed is the goal (assuming all ways of calculating the Bessel matrix are equally accurate) the question hinges on finding the bottle necks. The bottle necks are definitely in calculating the boundary conditions.

This is were the two methods are not equal in speed.

The number of boundary points (which are calculated using the integration method (VERY SLOW)) for method (V) is $N_\Gamma$

The number of boundary points (which are calculated using the integration method (VERY SLOW)) for method (VI) is $NSB_{max}$

In general $N_\Gamma \gg NSB_{max}$ so method (VI) will be faster.

However we can do something clever and use method (VI) to calculate the boundary conditions for method

we want the matrix (double array) to be indexed as J[G][n]. This is the way it will be accessed and will be much faster to read than J[n][g]. For speed we also want to write the matrix in the order J[G][n] so that it does not have to be transposed after it is calculated.

What we will do is calculate all values of $J_0(\Gamma); \quad \Gamma \in [0, \Gamma_{max}]$ using the constant n method. Then we calculate $J_{NSB_{max}}(\Gamma); \quad \Gamma \in [0, \Gamma_{max}]$ using the constant n method.

Then we calculate $J_n(\Gamma); \quad \Gamma in()$... using the constant gamma method.

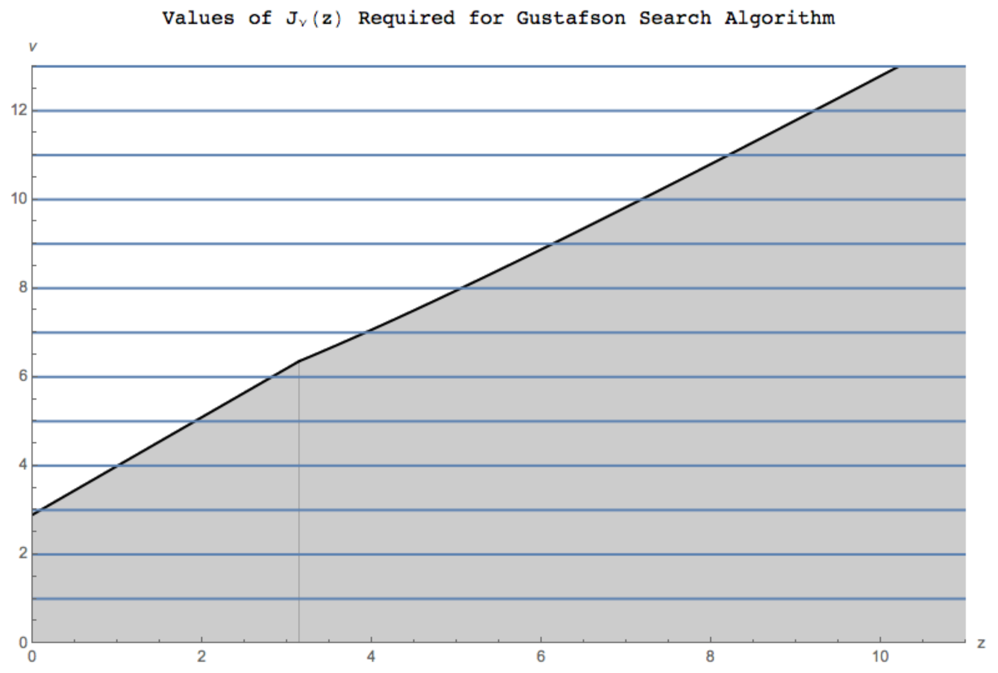


**FIG. 3:** The blue horizontal lines are the values in the $z-\nu$ plain that are required to perform the Gustafson Search Algorithm. (this plain should be semi-infinite; only a small portion is shown) The values that lie within the shaded region are the ones that should be used if formula ... is used.

It is faster (by several orders of magnitude) to simply calculate a square chunk of values and only use some of them rather than only calculate the values needed. Were trading wasted memory for speed!

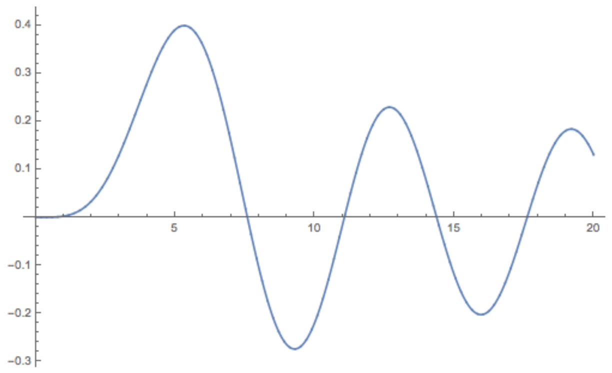
## IX. CALCULATING BESSEL FUNCTIONS: RESULTS

The ultimate benchmark for success in calculating Bessel Functions in this manner (using the Thomas Algorithm to solve Bessel's Differential Equation) is to compare it to the standard way of calculating the value at each point using the Riemann Sum (as defined in equation 63 section **??**). 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. 66. Table I gives the speed increase for various parameters.

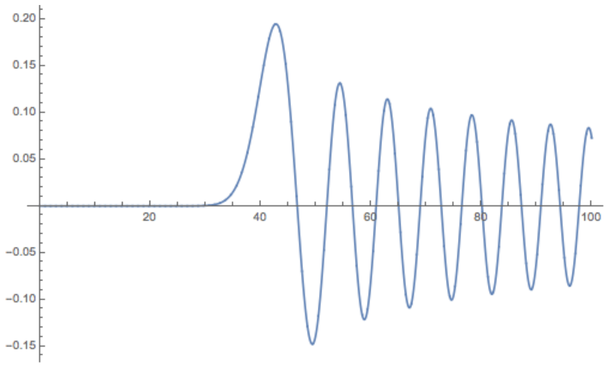
| Parameters | Point by Point Using Integral | Thomas Algo. Solving Bessel's Equation | Speed Increase |
|---|---|---|---|
| n = 3<br>N = 500<br>$z_{max} = 20.0$ | 0.051786 s | 0.00013800 s | 375 |
| n = 14<br>N = 50,000<br>$z_{max} = 100.0$ | 113.33 s | 0.003937 s | 28,785 |

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

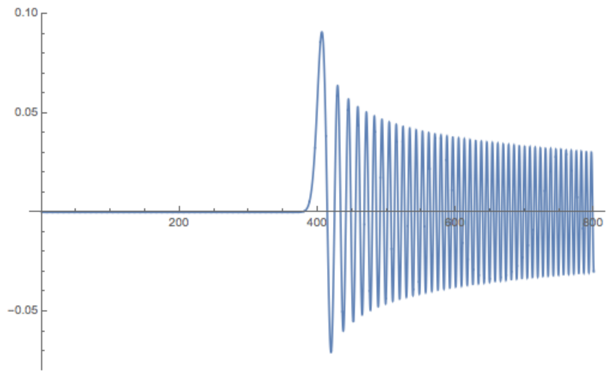
Figure 4 provides three examples of Bessel Functions that were calculated using the Thomas Algorithm to solve Bessel's Equation.



**(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 end-point!

**FIG. 4**

In Figure 5, it can be observed that the standard approach and the suggested approach are in agreement except at $n = 0, 85$. This is a small subset of the 400,000 data points generated in 0.33 seconds (on 5 year old a laptop). It solved the Bessel differential equation for 4000 steps in $\Gamma \in (0, 100)$, then saved the output, and moved on to solve Bessel's equation for the next value of n until all values of $\Gamma$ and n were solved for.

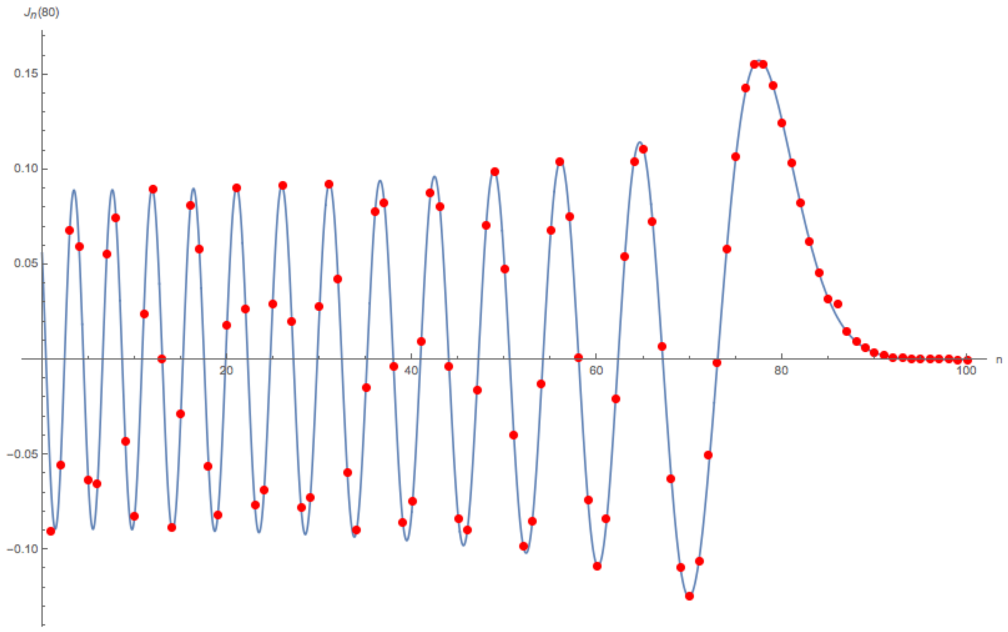


**FIG. 5:** This plot is a plot of $J_n(80)$ for values of $n \in (0, 100)$. The blue curve was generated by the BesselJ function of Mathematica. The red dots were calculated by our code. The two sets of data are in excellent agreement at except perhaps $n = 0, 85$ in which case there is a slight disagreement. This is a small selection of the $400,000$ data points my code generated in 0.33 seconds (on 5 year old a laptop). 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 $\Gamma$ and n were solved for.

### A. Implementing the Gustafson Algorithm

A practical implementation requires some modifications to the Gustafson Algorithm. Instead of calculating the negative order Bessel Functions we can use the relation given in equation (**??**). Restricting the algorithm in this way reduces computation time and memory requirements.

$$h_0 e^{i\phi_0}\delta(\omega-\omega_0) = 2\left(\sum_{n=-\infty}^{-1} e^{-in(\phi_1-\pi/2)} J_{-n}(\Gamma)\hat{f}(\omega+n\omega_1) + \sum_{n=0}^{\infty} e^{-in(\phi_1+\pi/2)} J_n(\Gamma)\hat{f}(\omega+n\omega_1)\right) \tag{105}$$

$$= 2\left(J_0(\Gamma)\hat{f}(\omega) + \sum_{n=1}^{\infty} e^{-in\pi/2} J_n(\Gamma)\left[e^{in\phi_1}\hat{f}(\omega-n\omega_1) + e^{-in\phi_1}\hat{f}(\omega+n\omega_1)\right]\right) \tag{106}$$

-because the FT is symmetric about omega0 it is tempting to simply multiply each sideband by 2 and sum over one side. -for robustness against noise we want to sum over sidebands on both sides of omega0, not just one.

## X. ALIASING AND FREQUENCY FOLDING

Typically the Discrete Fourier Transform runs over both positive and negative frequencies. Since the data considered here is real, the negative frequencies contain no additional information and can be ignored. This restriction cuts the time required to perform the DFT in half. The frequencies the search has access to are between zero and the Nyquist Frequency. It is possible for frequencies outside of this range to contain a significant portion of the power. These frequencies will be folded.

-only search over positive values of omega0. -only search over positive values of omega1.

-using aliased sidebands adds some computational overhead in calculating the new frequency. How much? probably not a lot.

now to calculate the new frequency bin for aliased sidebands.

for positive frequencies Calculate the folding zone:

$$N_{FZ} = \left\lceil \frac{|f|}{f_{nq}} \right\rceil \tag{107}$$

if $N_{FZ}$ is 0 or 1, then the frequency is unaliased. If this is not the case and $N_{FZ}$ is even, then the frequency is aliased and is said to be in an even folding zone. Frequencies in these zones are mirrored and encure a phase shift of $\pi$.

$$\hat{f}[f] \rightarrow -\hat{f}[f_{nq} - f \pmod{f_{nq}}] \tag{108}$$

If $N_{FZ}$ is greater than 1 and odd, then the frequency is aliased and said to be in an odd folding zone. Frequencies in these zones are translated and do not encure a phase shift.

$$\hat{f}[f] \rightarrow \hat{f}[f \pmod{f_{nq}}] \tag{109}$$

-the number of bins in the fourier transform is (only including positive frequencies)

$$N_{FS} = \frac{f_{nq}}{df} + 1 \tag{110}$$

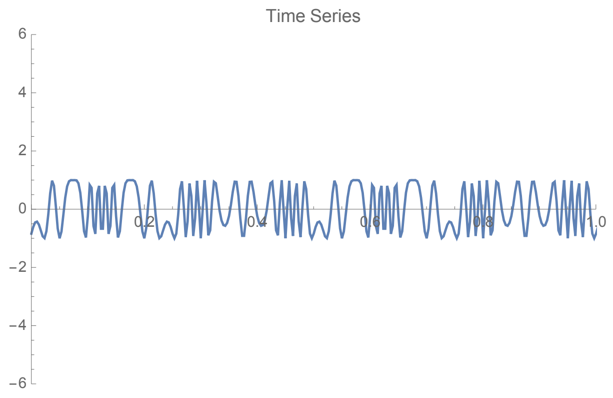
now we need to translate this into c++. In our fourier transform array $f_{nq} \rightarrow N_{FS} - 1$ and $\hat{f}(f_i = df * i) \rightarrow \hat{f}(i)$
now we can just change the index: if n is 0 or 1 do nothing
if n is even: (where i is the ith sideband) $N_{FS} - 1 - N_{df} * i \pmod{N_{FS} - 1}$
$N_{df} = \lfloor f_1/df \rfloor$
if n is odd: $N_{df} * i \pmod{N_{FS} - 1}$

## XI. RESULTS AND DISCUSSION

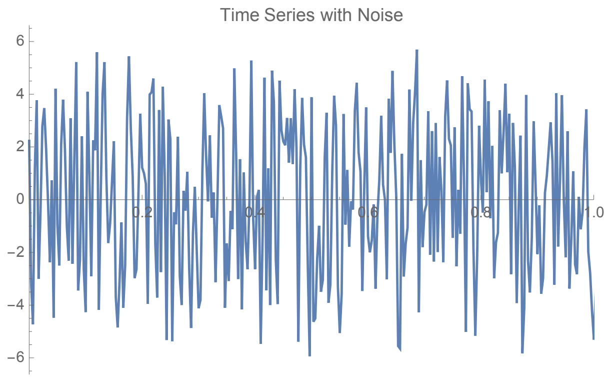
Since frequency modulated signals may sometimes look like noise in the time domain (see appendix), it is imperative to ascertain that the Gustafson algorithm provides accurate information about the carrier frequency. For this purpose, the Gustafson algorithm was designed to be implemented in Fourier space.

By observing Figures 6a and 6b, we can see that noise added to the frequency modulated signal makes it almost impossible to visually characterize. In practice, most frequency modulated signals encountered — especially those of astronomical sources and signal susceptible to environmental noise — are of this form. A simple approach to identifying relevant features of the frequency modulated signal adulterated with noise will be to search for patterns therein. However, since the noise is sampled randomly it is nearly impossible to extract patterns without some form of transformation. The most obvious transform to employ that allows us access the frequencies that compose the noisy signal is its Fourier transform.

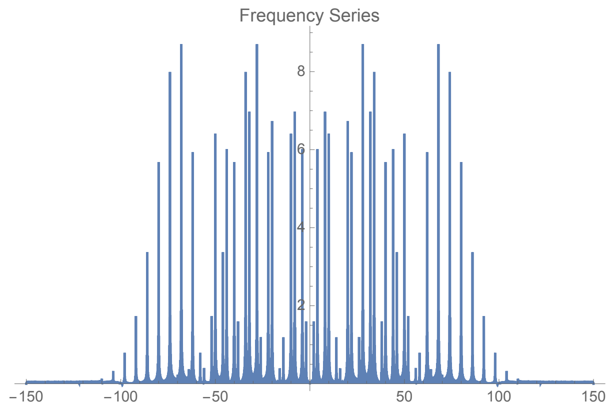
In Figures 6c and 6d the Fourier transform of the frequency modulated signal and that of the signal adulterated with noise are shown respectively. In the space of the Fourier transform, the frequencies that compose the original signal can be clearly observed. However, this task gets challenging when observing the Fourier transform of the noisy signal. Note that although a high degree of noise has been added, major peaks can still be seen. This is a function of the SNR; with a high enough noise amplitude, the Fourier transform will fail in providing useful information. This suggests that although the Fourier transform is a powerful technique when it comes to ignoring noise within a continuous signal, it is limited by the power of the noise.



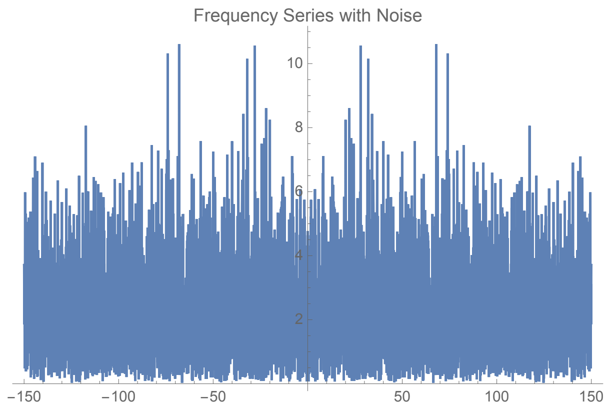
**(a)** Frequency modulated signal.



**(b)** Frequency modulated signal with noise.



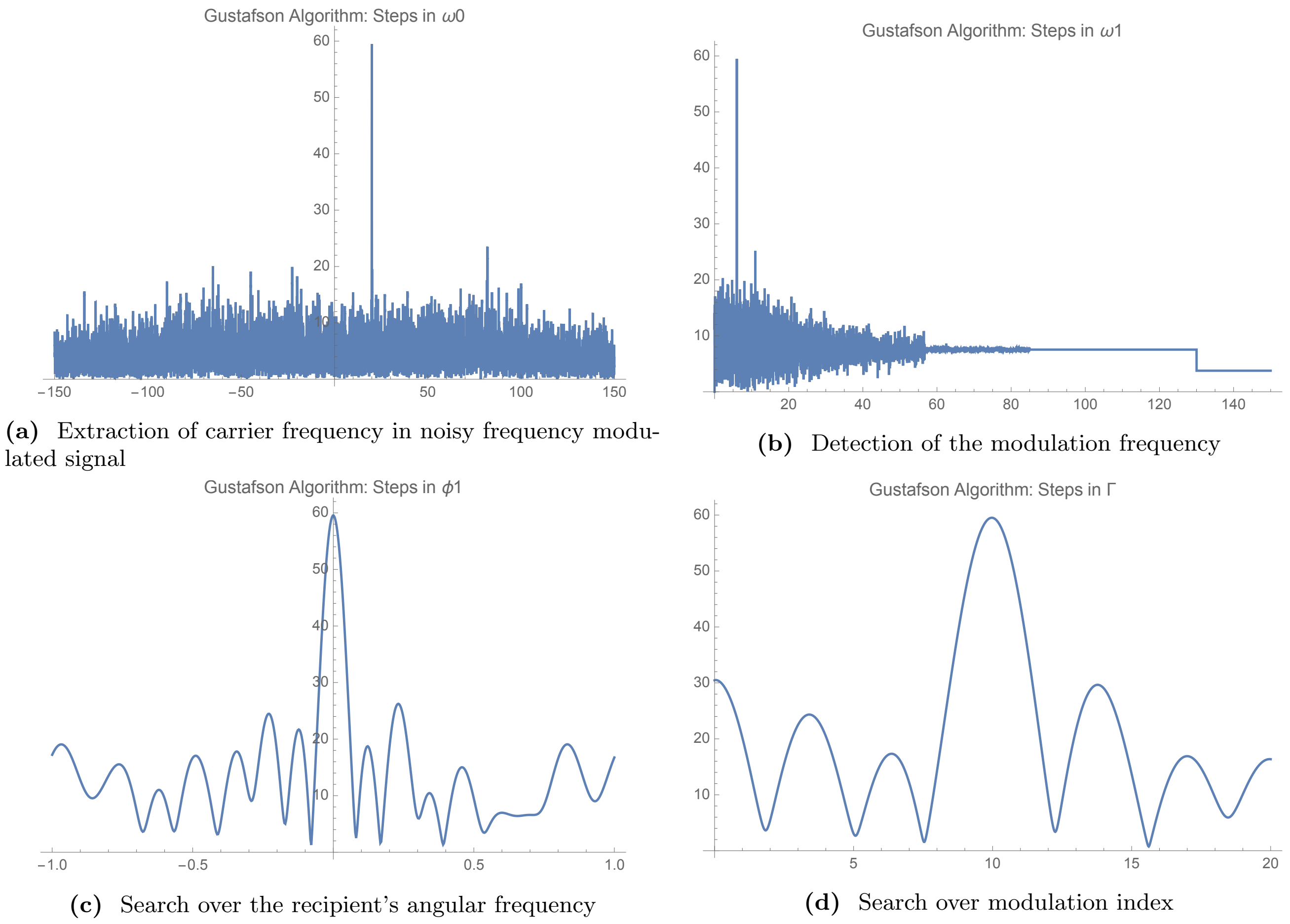
**(c)** Fourier transform of the frequency modulated signal



**(d)** Fourier transform of (b) above

**FIG. 6:** Frequency modulated signal and its noisy version in the time series and Fourier space. The sampling rate for these traces was 300Hz. The carrier frequency was 20Hz, while the modulaton frequency was 6Hz. It is worth noting that these represent 10 seconds of recording with a modulation index of 10.

To test the robustness of the Gustafson algorithm against noise, we searched for the carrier frequency, modulation frequency, modulation index and the recipient's angular frequency for they noisy signal provided in Figure 6b. From Figure 7a we can see that the Gustafson algorithm appropriately separated out the signal (carrier frequency) from the noise even though it was noise obvious in the corresponding Fourier transform. It is worth noting that the search results in a transform that is asymmetric in frequency; this is a function of the fact that the Gustafson algorithm includes a rotation in Fourier space that is opposite to that of the Fourier transform. With Figure 7b, it can be observed that the Gustafson algorithm accurately obtained modulation frequency of the noisy signal. Since the search for the carrier frequency and modulation frequency is sensitive, it is important to use small steps; this will aid in preventing one from accidentally skipping over the correct solution. Like the previous results, the Gustafson algorithm was able to extract the correct recipient's angular frequency (Fig. 7c) and the modulation index (Fig. 7d) of the frequency modulated signal. Since the search for the modulation index and recipient's angular frequency is relatively smooth, one may take tolerably larger step sizes and still converge to the right solution.



**(a)** Extraction of carrier frequency in noisy frequency modulated signal

**(b)** Detection of the modulation frequency

**(c)** Search over the recipient's angular frequency

**(d)** Search over modulation index

**FIG. 7:** Application of the Gustafson algorithm to the noisy frequency modulated signal illustrated in Fig. 6b. Sampling Rate = 300Hz, Carrier = 20Hz, Mod Frequency = 6Hz, Mod Index = 10, Time Series Duration = 10s, SNR = 0.1

## XII. CONCLUSION

In this paper, the Gustafson algorithm has been introduced as well as a novel approach to noise reduction. Although both algorithms presented have been shown to be effective, they may require substantial modifications to be useful in practice. One major draw back of the Gustafson algorithm is the selection of a range of parameters to search over. An investigation of how one can introduce learning procedures such as stochastic gradient descent might be able to provide a steady solution to this issue.

## XIII. THE FUTURE

write a function to determine the standard deviation of the power in the fourier series.
use the standard deviation of the power as a criterion for determining possible candidates

### Appendix A: The Fourier Transform of The Complex Function

The results this and the following section are quite surprising. Looking at the instantaneous frequency of a frequency modulated wave, it takes on every value between $\omega_0 \pm \omega_1$. There are an uncountable 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 spaced at integer multiples of $\omega_1$ from the *carrier frequency* $\omega_0$. Interestingly for particular values of the *modulation index* $\Gamma$ it is possible for the carrier frequency to not be present in the Fourier Transform! This occurs at the zeros of the zeroth Bessel Function.

$$\begin{aligned}
\mathcal{F}_t\left[h_0 e^{i(\omega_0 t+\phi_0+\Gamma\cos(\omega_1 t+\phi_1))}\right] &= h_0 e^{i\phi_0}\mathcal{F}_t\left[e^{i\omega_0 t}e^{i\Gamma\cos(\omega_1 t+\phi_1)}\right] && \text{(A1)}\\
&= h_0 e^{i\phi_0}\mathcal{F}_t\left[e^{i\omega_0 t}\right]\star\mathcal{F}_t\left[e^{i\Gamma\cos(\omega_1 t+\phi_1)}\right] && \text{(A2)}\\
&= 2\pi h_0 e^{i\phi_0}\delta(\omega-\omega_0)\star\mathcal{F}_t\left[\sum_{n=-\infty}^{\infty} e^{in\pi/2}J_n(\Gamma)e^{in(\omega_1 t+\phi_1)}\right] && \text{(A3)}\\
&= 2\pi h_0 e^{i\phi_0}\delta(\omega-\omega_0)\star\sum_{n=-\infty}^{\infty} e^{in\pi/2}e^{in\phi_1}J_n(\Gamma)F_t\left[e^{in\omega_1 t}\right] && \text{(A4)}\\
&= 2\pi h_0 e^{i\phi_0}\delta(\omega-\omega_0)\star\sum_{n=-\infty}^{\infty} e^{in(\phi_1+\pi/2)}J_n(\Gamma)\delta(\omega-n\omega_1) && \text{(A5)}\\
&= 2\pi h_0 e^{i\phi_0}\sum_{n=-\infty}^{\infty} e^{in(\phi_1+\pi/2)}J_n(\Gamma)\delta(\omega-\omega_0)\star\delta(\omega-n\omega_1) && \text{(A6)}\\
&= 2\pi h_0 e^{i\phi_0}\sum_{n=-\infty}^{\infty} e^{in(\phi_1+\pi/2)}J_n(\Gamma)\delta(\omega-\omega_0-n\omega_1) && \text{(A7)}
\end{aligned}$$

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.

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

$$\begin{aligned}
\mathcal{F}_t\left[\Re\left\{h_0 e^{i(\omega_0 t+\phi_0+\Gamma\cos(\omega_1 t+\phi_1))}\right\}\right] &= \frac{1}{2}\mathcal{F}_t\left[h_0 e^{i(\omega_0 t+\phi_0+\Gamma\cos(\omega_1 t+\phi_1))}+h_0 e^{-i(\omega_0 t+\phi_0+\Gamma\cos(\omega_1 t+\phi_1))}\right] && \text{(B1)}\\
&= \frac{h_0}{2}\left[e^{i\phi_0}\mathcal{F}_t\left[e^{i(\omega_0 t+\Gamma\cos(\omega_1 t+\phi_1))}\right]+e^{-i\phi_0}\mathcal{F}_t\left[e^{i(-\omega_0 t-\Gamma\cos(\omega_1 t+\phi_1))}\right]\right] && \text{(B2)}
\end{aligned}$$

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 $\omega_0$ with $-\omega_0$ and $\Gamma$ 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

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

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

Notice that the Fourier Transform of the real wave looks very similar to the Fourier Transform of the complex wave. The difference being that the Fourier Transform of the real is symmetric about $\omega = 0$ up to a phase factor. The vectors corresponding to the positive carrier frequency rotate counterclockwise, whereas the vectors corresponding to the negative carrier frequency rotate clockwise (neglecting the rotation due to $\phi_1$ which is the same for both). **This phase factor is the reason why the Gustafson Algorithm only picks out the positive value of $\omega_0$.** If it is unclear what this means refer to figure (7e).

## Appendix C: Properties of the Bessel Functions

– Jacobi-Anger Expansion:

$$e^{iz\cos(\theta)} = \sum_{n=-\infty}^{\infty} e^{in\pi/2} J_n(z) e^{in\theta}. \tag{C1}$$

– Even order Bessel Functions are even, and odd order Bessel Functions are odd [5]. Explicitly

$$J_n(-z) = (-1)^n J_n(z) \tag{C2}$$

– [5]

$$J_{-n}(z) = (-1)^n J_n(z). \tag{C3}$$

– the asymptotic form of the bessel functions: Theorem 5.1 in [4] states:
*For each $n \in N$ there is a constant $C_n \in R$ such that, if $x \geq 1$, then*

$$\left| J_n(x) - \sqrt{\frac{2}{\pi x}} \cos\left(x - \frac{\pi}{4}(2n+1)\right) \right| \leq \frac{C_n}{x^{3/2}}. \tag{C4}$$

– ***Rau's Theorem:*** *For any $z \in (\pi, \infty)$ the value of $\nu$ that maximizes $J_\nu(z)$ asymtotically approaches $\nu \to z - \log_\pi(z)$.*
– Another of Curtis's Theorems:

$$\sum_{n=-\infty}^{\infty} J_n^2(z) e^{in\phi} = J_0\left(2z \sin\left(\frac{\phi}{2}\right)\right) \tag{C5}$$

–

$$J_0(x-y) = \sum_{n=-\infty}^{\infty} J_n(x) J_n(y) \tag{C6}$$

–

$$\sum_{n=-\infty}^{\infty} J_n^2(\Gamma) = 1 \tag{C7}$$

[1] Albert Einstein. *On gravitational waves.* Springer, 1990.
[2] Deanna Emery. A coherent three dimensional fft based search scheme for gravitational waves from binary neurton star systems. *LIGO DCC*, (T1500307-v2), September 2015.

[3] 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.

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

[5] Martin Kreh. Bessel functions. Project for the Penn State - Göttingen Summer School on Number Theory.

[6] Nadja S Magalhães, Warren W Johnson, Carlos Frajuca, and Odylio D Aguiar. Determination of astrophysical parameters from the spherical gravitational wave detector data. *Monthly Notices of the Royal Astronomical Society*, 274(3):670–678, 1995.

[7] Eric Poisson. Gravitational waves from inspiraling compact binaries: The quadrupole-moment term. *Physical Review D*, 57(8):5287, 1998.

[8] Charles Rader. Discrete fourier transforms when the number of data samples is prime. *Proceedings of the IEEE*, 56(6):1107–1108, 1968.