Binghamton University

Thomas J. Watson School of Engineering and Applied Science

Passive Sonar with DFT

Matthew Simiele and Nevin Nedumthakady

EECE 301 – Signals and Systems

Professor Mark Fowler

December 7th, 2018

Introduction:

The Passive Sonar with DFT project was developed to analyze a sonar system using the Discrete Fourier Transform (DFT) and model the systems functionality using Matlab. Many applications of signal processing involve receiving a signal sinusoidal in nature. However, since components (receivers) and the environment (water in the case of sonar) modify the transmitted signal a significantly reduced signal encapsulated in noise is actually received by the system. The nature of the DFT of a sinusoid allows us to find the signal within the noise.

The analysis of a Passive Sonar system was completed using a singular sinusoid combined with a set of random noise values. The analysis of the system included changing the Signal to Noise Ratio (SNR) and analyzing how the signal responded. The sound command of MATLAB was used to hear the signal allowing us to distinguish between signals with different SNR values. Also, while computing and graphing the Fast Fourier Transform (FFT) different levels of sampling and zero padding were used to help solidify the effect changing these values have on the visualization of the DFT.

Part A: Generate Sinusoidal Signal

The goal of part a of the project is to create a sinusoidal signal, plot the first 100 samples of the sinusoid vs time, and compute the DFT of the Sinusoid. The function to be analyzed is defined as follows:

$$x(t) = Acos\left(\frac{2\pi n F_c}{F_s}\right) \ (1)$$

Where F_s the sampling frequency is 40 kHz, n is the number of samples, A is the amplitude of the sinusoid ($\sqrt{2}$), and F_c the frequency of the sinusoid is 5400 Hz. The sampling frequency is chosen to be 40kHz because in order to properly sample a signal the sampling frequency must be two times greater than the signals bandwidth, for the sinusoid being analyzed the bandwidth is 5400Hz.

A big concept used in part a of the project is sampling as we send the continuous time signal through an Analog to Digital converter, we get a discrete time equivalent of the continuous time signal.

$$x(t) \rightarrow ADC \rightarrow x[n] = x(nT)$$

Where n is the number of samples and T is the sampling interval.

In figure 1 the first 100 samples of x[n] are plotted versus time. Since the sampling frequency is not an integer multiple of the frequency of the sinusoid, we see a variation in the sample locations from cycle to cycle. The variation in sample location occurs because the spacing of the samples causes the sinusoid to be sampled at different locations. Zooming in on each cycle of the sinusoid we can find the average length for each cycle in seconds. Using the relationship between the time between cycles of the sinusoid and the frequency of the sinusoid the frequency of the sinusoid can be defined as in equation 2.

$$F_c = \frac{1}{T_{avg}} \ (2)$$

After using MATLAB the average length of a cycles (T_{avg}) is .185 milliseconds. Plugging the value of the average time between cycles information in equation 2 gives a sinusoidal frequency of 5416 Hz which is about 5400 Hz. The calculation for the represented sinusoid in figure 2 proves the equation 2, averaging the cycle length of a sinusoid in which the sampling rate is not a multiple of the frequency of the sinusoid will give a good estimate to the actual frequency if the sinusoid.

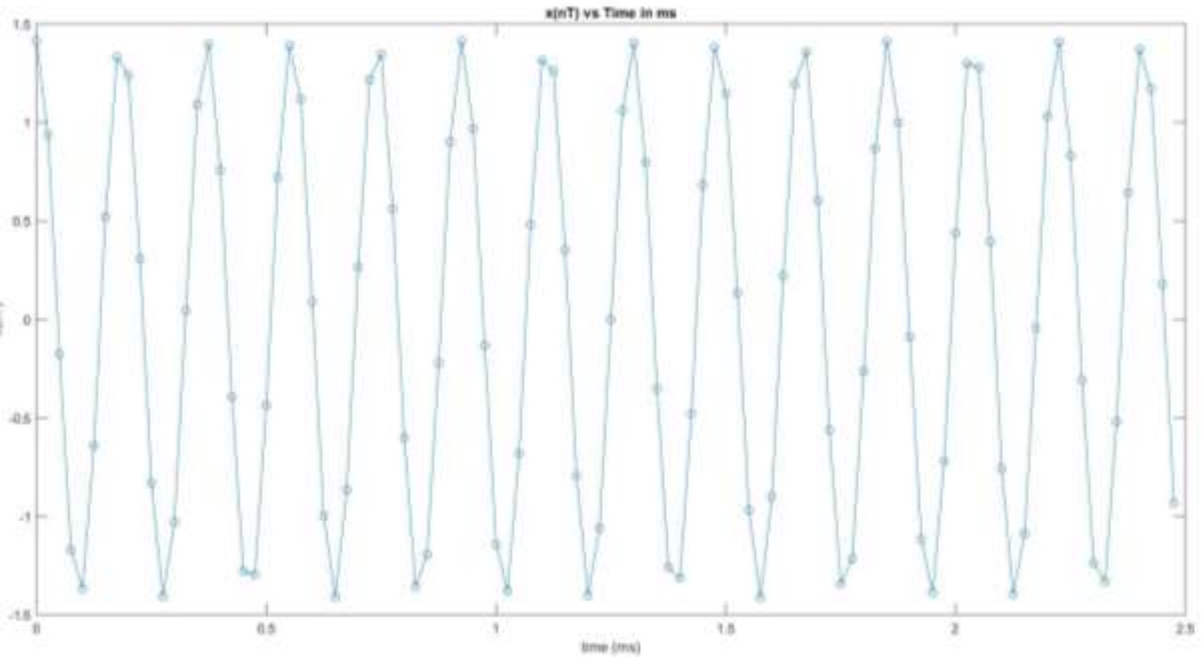

Figure 1: x(nT) vs time

When we compute the DFT of a n number of samples taken from a CT sinusoid we expect the plot of the DFT to look very similar to the DTFT of the signal since the points of the DFT lie on the DTFT curve. Since we are computing the DFT we are assuming the signal is of infinite duration or of duration longer than our computer can process. The DFT of the sinusoid is not the perfect representation of the DTFT of the signal but the DFT does in fact show the DTFT of the truncated signal of 4096 samples. Since we know our sampling frequency and our sinusoids frequency, we can properly guess the x limits will span -20kHz to 20kHz and the peaks will occur at $\pm 5400\ Hz$ due to our knowledge of the DTFT of a sinusoid, which is defined in equation 3.

$$DTFT\{\cos(\Omega_0 n)\} = \delta(\Omega + \Omega_0) + \delta(\Omega - \Omega_0)\ (3)$$

The sinusoid being analyzed has $\Omega_0 = 5400Hz$ which is the frequency of the sinusoid. In figure 2 we can see the DFT of the sinusoid truncated to 4096 samples.

As we can see in figure 2 the peak of the DFT occurs at 5400Hz confirming the DFT is computed correctly. Due to the symmetry of the DFT we know the other peak occurs at -5400Hz confirming the plotted DFT is equal to the DTFT defined in equation 2. By using MATLAB's fftshift command to compute the DFT we see the DFT centered at 0 ranging from $-F_s/2$ to $F_s/2$.

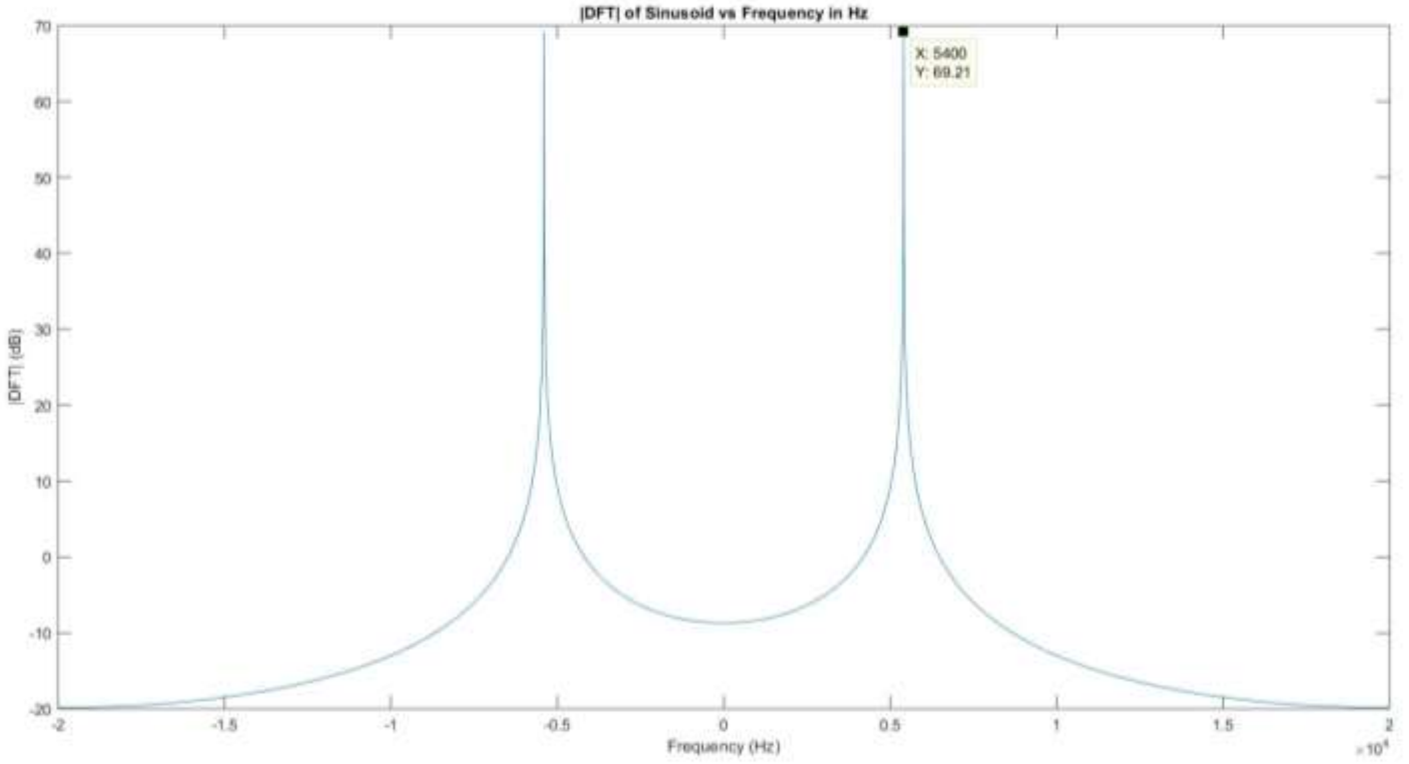

Figure 2: Magnitude of the DFT of the sinusoid in dB vs Frequency in Hz

Part B: Generate Random Noise

The next step in creating a realistic received signal is to generate the noise in which the signal will be encapsulated within. Since in Sonar the received signal is corrupted by noise in order to properly analyze a Sonar system you must be able to control the level of noise. In part b, a random noise signal is generated with the same length of vector x which contains the sinusoid of 32,768 samples. Using the MATLAB commands randn and size, a vector v, the noise vector, was able to be created to be the correct length. Since the noise vector is composed of random integers the noise is very erratic in nature. In figure 3 we can clearly see the erratic behavior. In figure 3 we see the first 1000 points of the noise vector plotted against time to show the erotic behavior of the noise. Since the randn command in MATLAB draws a scalar from the standard distribution the power of the signal is equal to 1 since the area under the standard distribution is equal to 1.

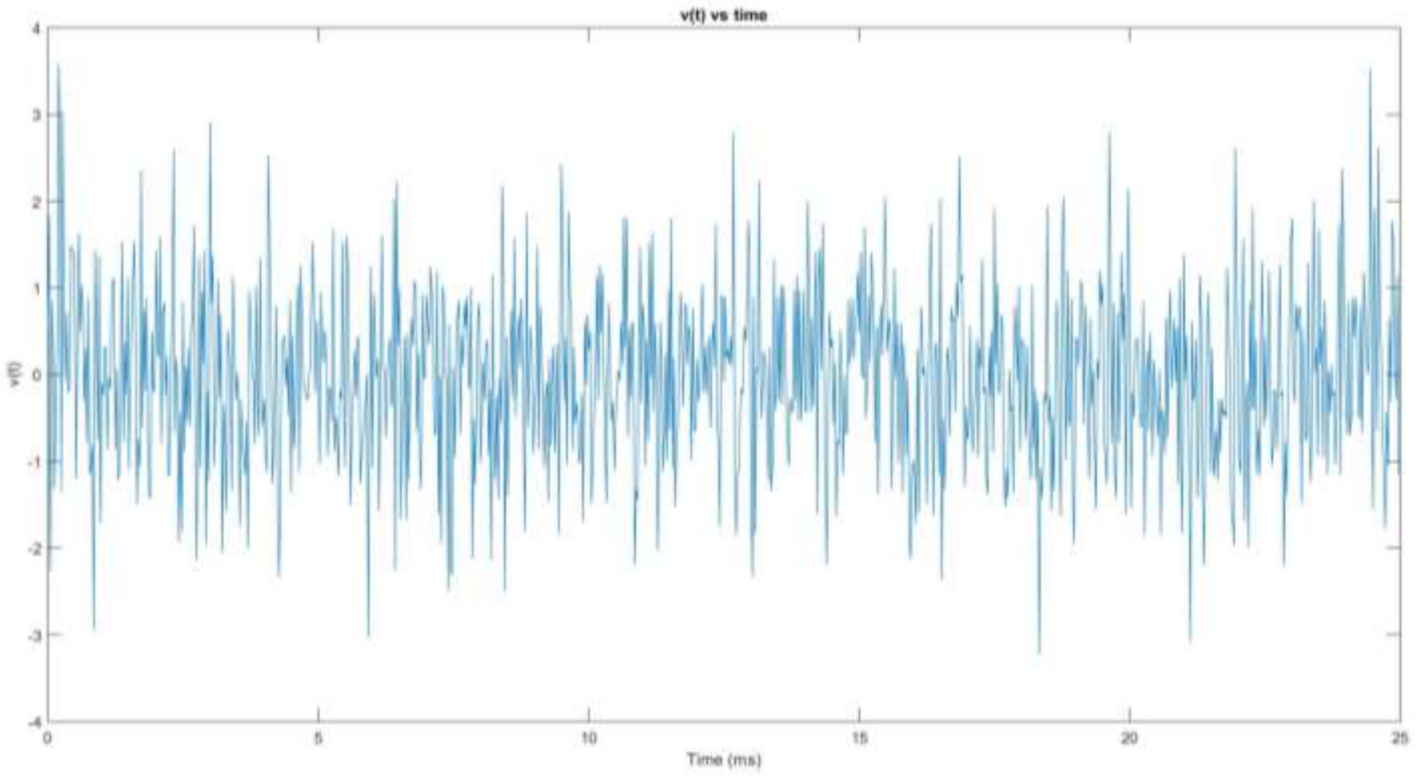

Figure 3: The first 1000 points of the noise vs time

Since in order to see the peaks of the sinusoid within the noise we will need to view the combined signal in the frequency domain. Due to the erratic nature of the noise in the time domain we don't expect to a very clean DFT of the signal. Since the time signal is so erratic when we compute the DFT the DTFT sees all the peaks of the time function and creates peaks at all frequencies. As you can see in figure 4 the magnitude of the DFT is very ragged but the top of the plot is very flat comparatively. By using MATLAB's fftshift command to compute the DFT we see the DFT centered at 0 ranging from $-F_S/2$ to $F_S/2$. The noise we are analyzing is called white noise which is noise which contains equal since the noise contains equal amounts of all frequencies. A very easy way in order to compare the signal to the noise is to use the noise floor power which is defined by equation 4.

$$P_{NF} = N * P_N = N * \left(\frac{1}{N}\right) * \sum v^2 \ (4)$$

Since, as we explained earlier, the power of the noise is equal to one we can further simplify equation 4. For the case where the noise power is one, equation 4 is simplified to define equation 5.

$$P_{NF} = N * 1 = N \ (5)$$

Since in our case we are taking 4,096 samples of the DFT we can see when the noise floor power level is converted to dB the noise floor power level is about 36 dB. The computed value of the noise floor power level is shown in equation 6.

$$P_{NF} = 10 * \log_{10} 4096 \approx 36 \ dB \ (6)$$

In figure 4 we can see the magnitude of the DFT of the noise with the noise floor power level sitting inside the noise. The noise floor level siting inside the noise is expected because the noise floor power level should lie inside the noise. The magnitude of DFT in dB of the noise can be calculated using equation 7.

$$|DFT(v)| = 20 * \log_{10}|v| \ (7)$$

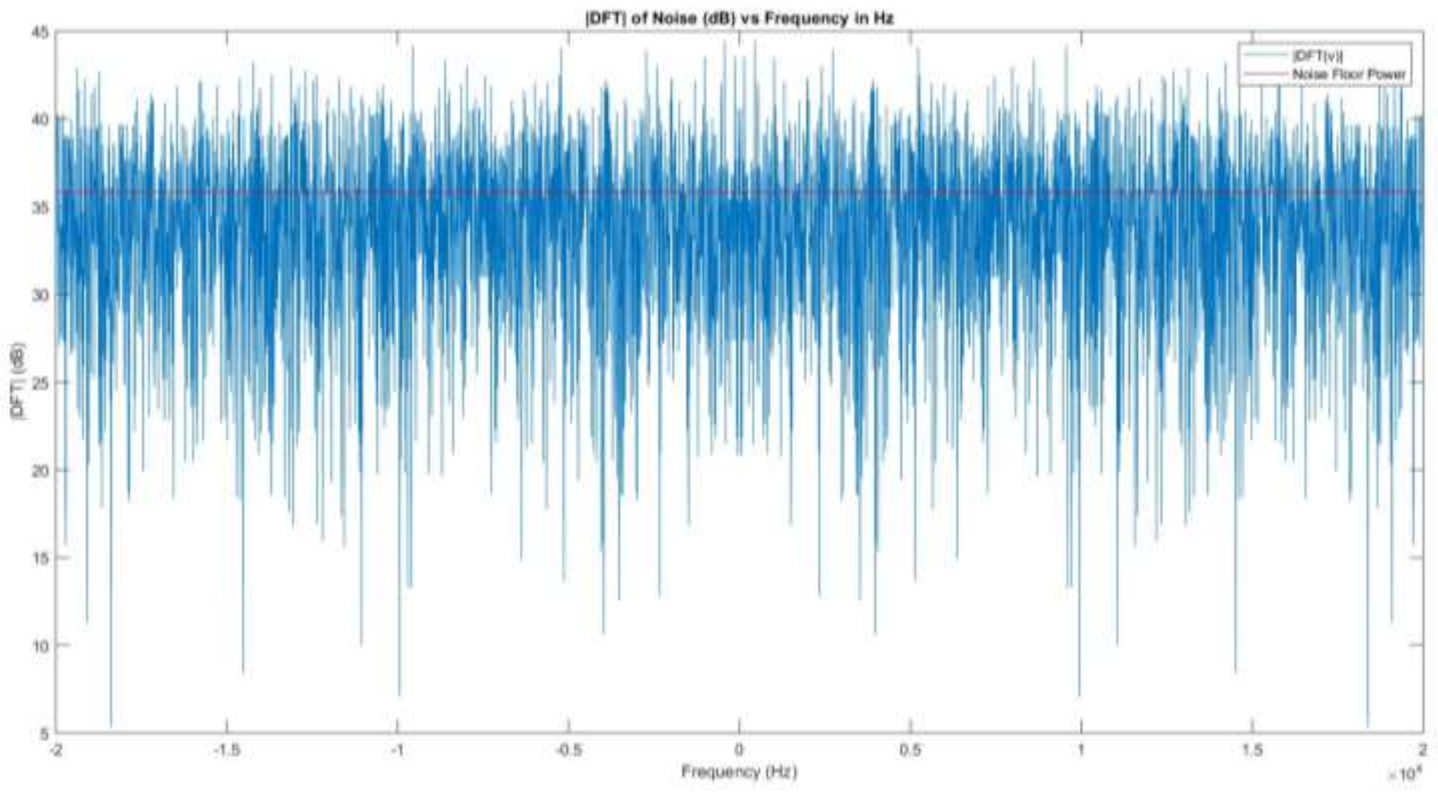

Figure 4: |DFT(v)| in dB and the noise floor power level in dB vs frequency

Part C: Combine Signal and Noise to Simulate Received Signal

Since we have found a way to express the noise of a signal and the signal without noise, we now need to simulate the system receiving the signal. The simulation for the system receiving a signal can be done in MATLAB using equation 8 where r[n] is the vector of the received signal, x[n] is the vector containing 32,768 time samples of the sinusoid and v[n] is the noise vector.

$$r[n] = x[n] + v[n] \quad (8)$$

A parameter of high importance in understanding the relationship between the signal and the noise is the input Signal to Noise Ratio (SNR), which is defined in equation 9.

$$SNR_{input} = \frac{Signal\ Power}{Noise\ Power} \quad (9)$$

In order to calculate we must determine the signal power and the noise power. Earlier in part B the noise power was defined while calculating the noise floor power. We will use the same idea to define the signal power in order to create a way of calculating the SNR value in MATLAB. The input SNR is defined further in equation 10. Since throughout the report the SNR value is expressed in decibels equation 11 will define the equation of the input SNR expressed in decibels.

$$SNR_{input} = \frac{\frac{1}{N}\sum_{n=0}^{N-1} x^2[n]}{\frac{1}{N}\sum_{n=0}^{N-1} v^2[n]} = \frac{\sum_{n=0}^{N-1} x^2[n]}{\sum_{n=0}^{N-1} v^2[n]} \quad (10)$$

$$SNR_{dB} = 10 * \log_{10}\frac{Signal\ Power}{Noise\ Power} = 10 * \log_{10}\left(\frac{\sum_{n=0}^{N-1} x^2[n]}{\sum_{n=0}^{N-1} v^2[n]}\right) (11)$$

Since we have not modified the signals in any way, yet we expect the SNR value to be 1 and 0 in dB because the power of the noise is equal to the power of the signal. In the figures in part c the received signal and the simulated signal will also be graphed in dB in order to see changed in the amplitude of the signals for the following parts easier. In figure 5 we see the received signal, sinusoidal signal, and the SNR value of the unmodified signals. Using the MATLAB command in equation 12 an SNR value of 0.0410 dB was computed and using equation 13 defined in the project description an SNR value the 0.0410 dB was confirmed.

$$SNR_{computed} = 10 * log10\left({sum(x.^2)}/{sum(v.^2)} \right) \quad (12)$$

$$SNR_{real} = 10 * log10\left({sum(x.^2)}/{sum((r-x).^2)} \right) \quad (13)$$

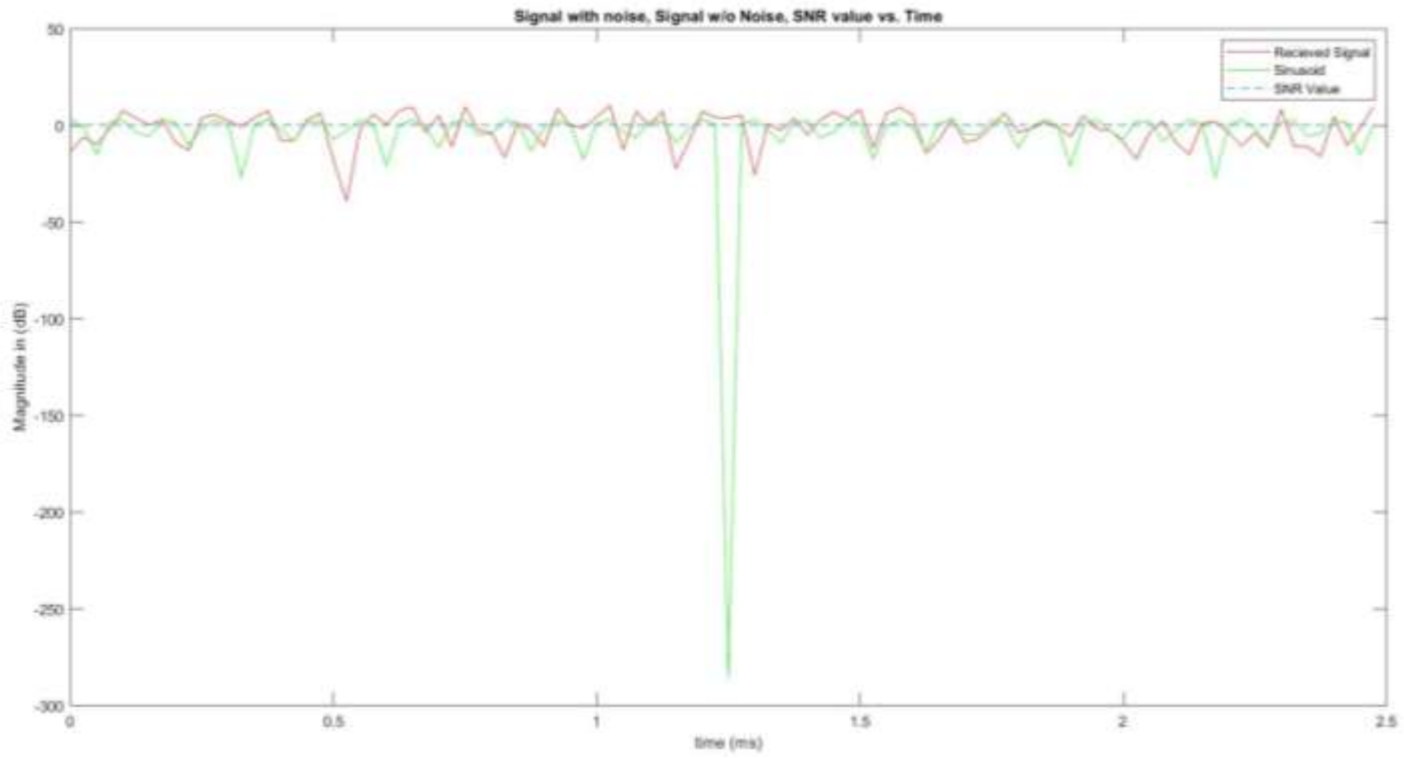

Figure 5: Received Signal, Signal w/o Noise, SNR Value in dB vs time

In figure 5 you can see the received signal is modified slightly by the noise but retains its basic shape and power. The signal is modified slightly because the noise has the same power as the original signal. When using the MATLAB sound command, you can clearly hear the sinusoid within the noise at about the same volume as the noise. MATLAB's sound command confirms the sound is corrupted by the noise with the SNR of 0 dB but still is audible.

In order to change the SNR, we need to define an equation in which an integer modifies the power of the noise signal. In equation 14 the value sqrt(b) is added as a multiplier to the noise in order to set the SNR value.

$$SNR_{dB} = 10 * \log_{10}\left(\frac{\sum_{n=0}^{N-1} x^2[n]}{\sum_{n=0}^{N-1} \left(\sqrt{b}v\right)^2[n]} \right) \quad (14)$$

Equation 14 must be modified to solve for the multiplier sqrt(b) to obtain a desired SNR. The derivation of the equation to solve for the multiplier b is outlined in equations 15 through 18, with equation 18 giving the final answer for the derivation of b.

Equation 15 shows a modified version of equation 14 where log properties were applied in order to solve for sqrt(b).

$$SNR_{dB} = 10 * \log_{10} x^2 - 10 * \log_{10}\left(\sqrt{b}v\right)^2 \; (15)$$

$$10 * \log_{10}\left(\sqrt{b}v\right)^2 = 10 * \log_{10} x^2 - SNR_{dB} \; (16)$$

$$\log_{10}\left(\sqrt{b}v\right)^2 = \frac{10 * \log_{10} x^2 - SNR_{dB}}{10} \; (17)$$

$$b = \frac{10^{\frac{10*\log_{10} x^2 - SNR_{dB}}{10}}}{v^2} \; (18)$$

After obtaining the mathematical expression for b in equation 18, the MATLAB equivalent expression for b is defined in equation 19.

$$b = (10\verb|^|((10 * log10(sum(x.\verb|^|2)) - desired\_SNR)/10))/(sum(v.\verb|^|2)) \; (19)$$

Figure 6 shows the plot of the received signal, sinusoidal signal, and the SNR value vs time, with the desired SNR value being -20dB. Before looking at the graph we know for the case of the SNR value being -20dB causes our Signal to Noise ratio will be very low so the signal will become very encapsulated within the noise. Using MATLAB's sound command to simulate the received signal all of what you can hear is the noise, no sinusoid is easily heard confirming the sound is engulfed by the noise. When looking at figure 6 you can see the noise plays a significant role here. The received signal is much larger than the sinusoid which is because the power of the noise is much greater than the power of the sinusoid.

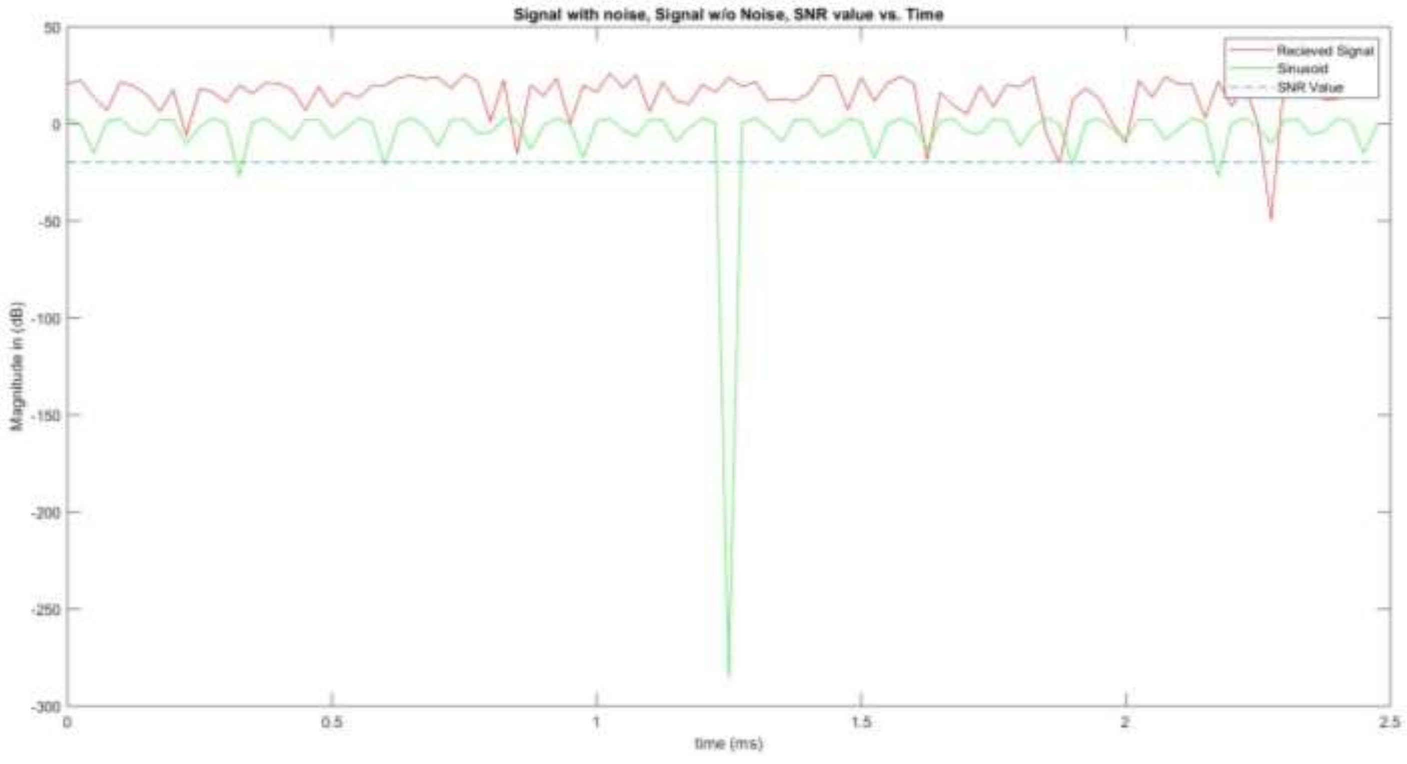

Figure 6: Received Signal, Signal w/o Noise, SNR Value of approximately -20 dB vs time

In the MATLAB code the b value is calculated to account for the full length of the signal not just the sampled section graphed above in figure 6 and below in figures 7 and 8. Since the b value is computed for the entire signal the SNR of the sampled signal is slightly off from what the intended SNR was. Using equation 12 the

SNR for the condensed signal was computed to be -19.7876 dB which is about -20 dB and using equation 13 to check to see if the computed SNR is close to the real SNR in MATLAB the SNR was found to be -19.7876 dB.

Figure 7 shows the plot of the received signal, sinusoidal signal, and the SNR value vs time, with the desired SNR value being 10dB. Before looking at the graph we know the signal to noise ratio is very high, so the signal power is greater than the noise. Thus, the received signal will look much more like the sinusoid than in figure 6. After using the sound command in MATLAB our assumption is confirmed because the signal is heard clearly through the noise. Using equation 12 the SNR for the condensed signal was computed to be 10.2124 dB which is about 10 dB and using equation 13 to check to see if the computed SNR is close to the real SNR in MATLAB the SNR was found to be 10.2124 dB.

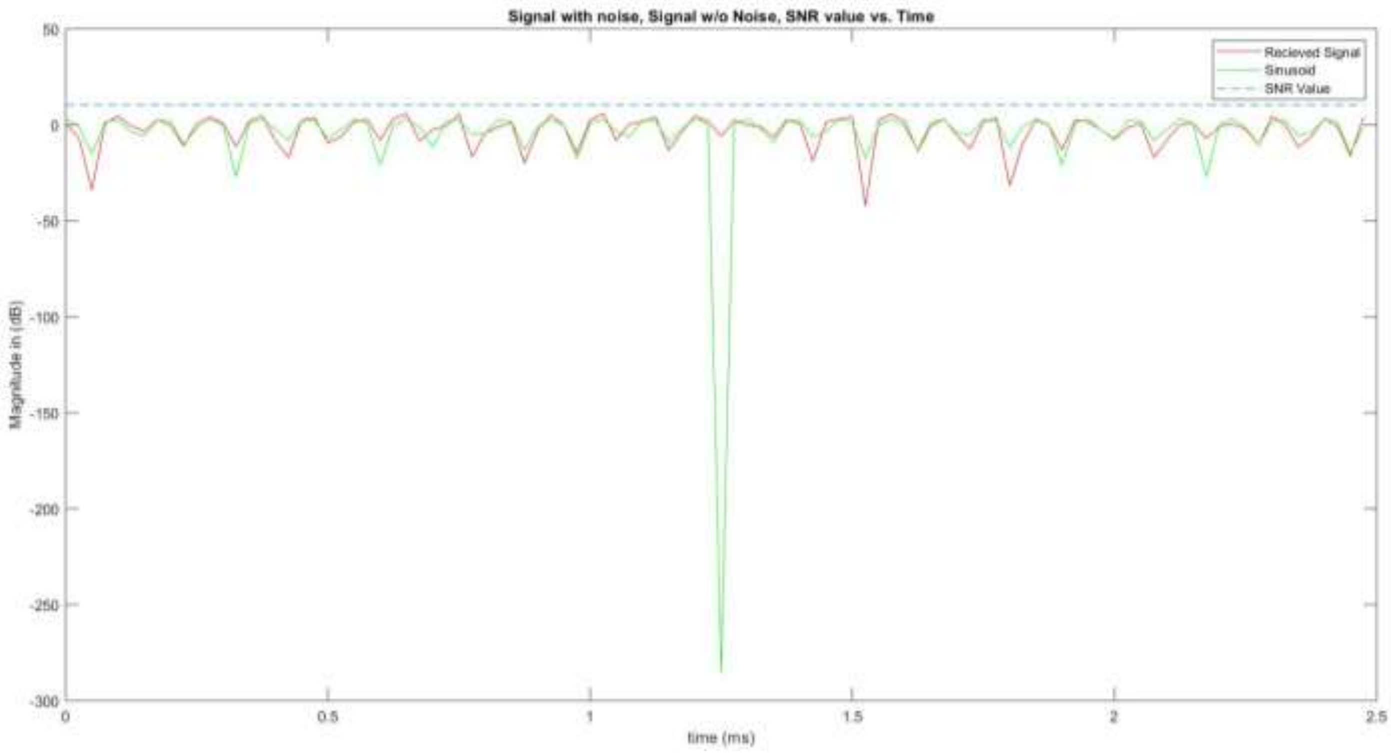

Figure 7: Received Signal, Signal w/o Noise, SNR Value of approximately 10 dB vs time

Figure 8 shows the plot of the received signal, sinusoidal signal, and the SNR value vs time, with the desired SNR value being 30dB. Before looking at the graph we know the signal to noise ratio is very high, so the signal power is much greater than the noise. In the graph the received signal is very similar to the original sinusoid because the power of the original sinusoid is basically maintained due to the noise power being very low. After using the sound command in MATLAB our assumption is confirmed because the signal is heard clearly through a nearly inaudible noise. Using equation 12 the SNR for the condensed signal was computed to be 30.2124 dB which is about 30 dB and using equation 13 to check to see if the computed SNR is close to the real SNR in MATLAB the SNR was found to be 30.2124 dB.

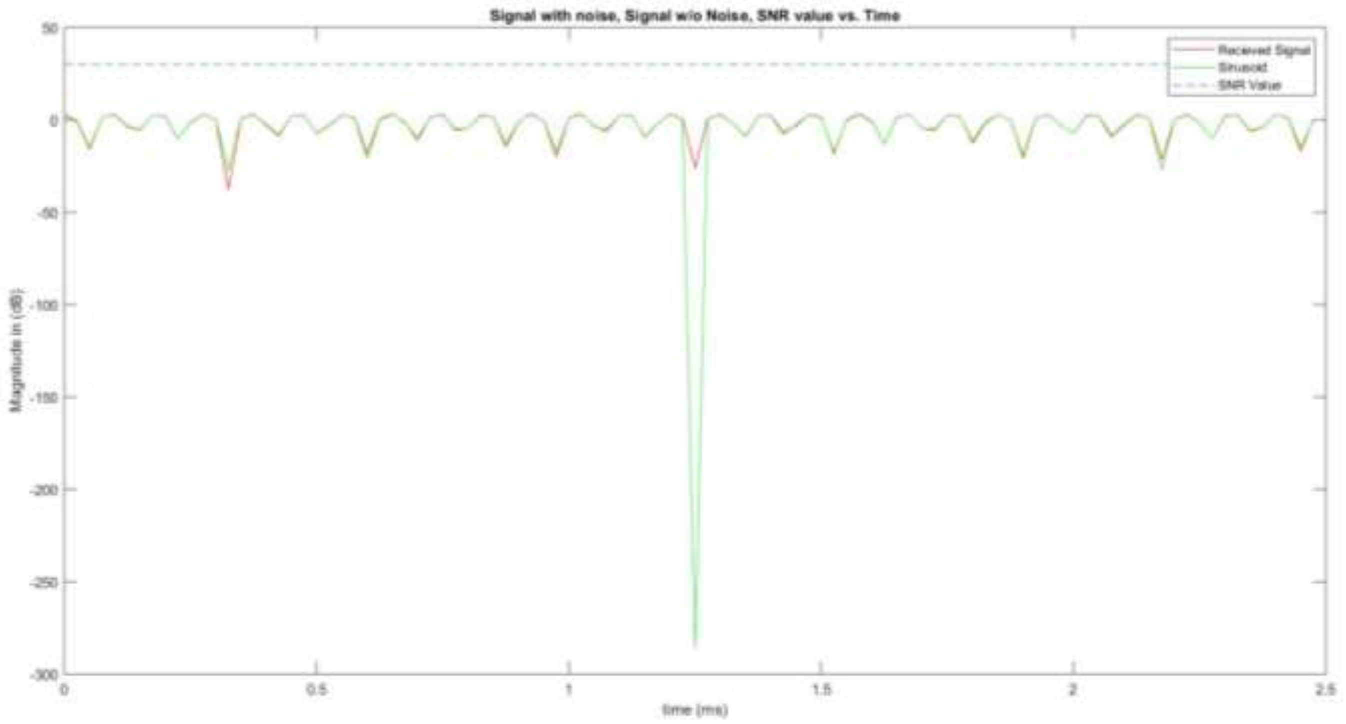

Figure 8: Received Signal, Signal w/o Noise, SNR Value of approximately 30 dB vs time

Since MATLAB uses the randn command to generate the noise every time the program is run the noise will be different but will exhibit all the same properties.

Part D: DFT Processing of Noisy Signals

Up to now all the necessary information to complete part d of the project has been found. In part a, the sinusoidal signal was generated to find within the noise. In part b, the noise the sinusoidal signal will be corrupted by was created. In part c, we were able to control the Signal to Noise ratio to simulate a favorable system. The final part of the project will analyze the affect of changing the number of samples of the DFT, both in the zero padded and non-zero padded case. An SNR value of 20 dB will be used for the following part of the project. As seen above with the SNR at 20 dB we can still see the signal in the time domain. We will see for the case where the SNR value is 20dB pulling the signal out of the noise is very easy when applying the DFT. We can use equation 11 in order to find the power of the signal if the power of the noise is equal to 1μW. We can see the commutations to solve for the signal power in equations 20 to 22, where x^2 is the power of the signal.

$$20dB = 10_{log10}\frac{x^2}{1\mu W}\ (20)$$

$$10^2 = \frac{x^2}{1\mu W}\ (21)$$

$$x^2 = 1\mu W * 100 = 100\mu W\ or\ 0.1mW\ (22)$$

As we explored earlier the DFT of the sinusoidal signal is defined by two delta functions centered at the origin placed at $\pm\Omega_0$, which is equal to 5400Hz for the sinusoid in which the DFT will be computed, see figure 2 for reference. Also, we looked at the frequency domain view of the noise in figure 4, since the noise is so erratic in the time domain the noise causes a very erratic signal in the frequency domain. Since we know how both signals look in the time and frequency domain and how the combined signal looks in the time domain, we can make a very good guess as to how the combined signal will act in the frequency domain. Since the combined signal looks very similar to the sinusoidal signal in the time domain, the frequency domain should inherit the characteristics of the sinusoidal signal in the frequency domain. The signal still inherits at least some characteristics of the noise but since the power of the noise is much less than the power of the signal and the noise will sit significantly below the peak at $\pm\Omega_0$. We can clearly see the noise sits significantly below the peak in figure 9 for all N number of samples. Since the desired SNR value is stated we can come up with a decent guess at the noise floor power level using equations 4 and 10.

For the derivation for the Noise Floor Power level SNR, in terms of N, will not be expressed in decibels.

In equation 10 SNR is defined as:

$$SNR = \frac{Signal\ Power}{Noise\ Power} = \frac{x^2}{v^2}$$

At SNR = 20dB (SNR = 100):

$$100 = \frac{x^2}{v^2}$$

Solving for Noise Power (v^2) gives:

$$v^2 = \frac{x^2}{100}$$

Using the definition for noise floor power:

$$P_{NF} = v^2 * N = \frac{x^2}{100} * N$$

Converting to decibels gives:

$$P_{NF}(dB) = 10 * \log_{10}\left(\frac{x^2}{100} * N\right) \ (23)$$

Using equation 23 as a basis when the number of samples of a signal with a defined SNR increases the noise floor power level also increases. The relationship between the number of sample and the noise floor power level occurs because as you obtain more samples of white noise the intensity of the noise will increase because with more samples some frequencies can be repeated causing larger peaks in the frequency domain.

In figure 9 all the assumptions about how the combined signal will be viewed in the frequency domain stated up until now will be confirmed. The value of the noise floor power level represented by the solid red line slowly shifts up as we increase the number of samples. By using MATLAB’s fftshift command to compute the DFT we see the DFT centered at 0 ranging from $-F_s/2$ to $F_s/2$.

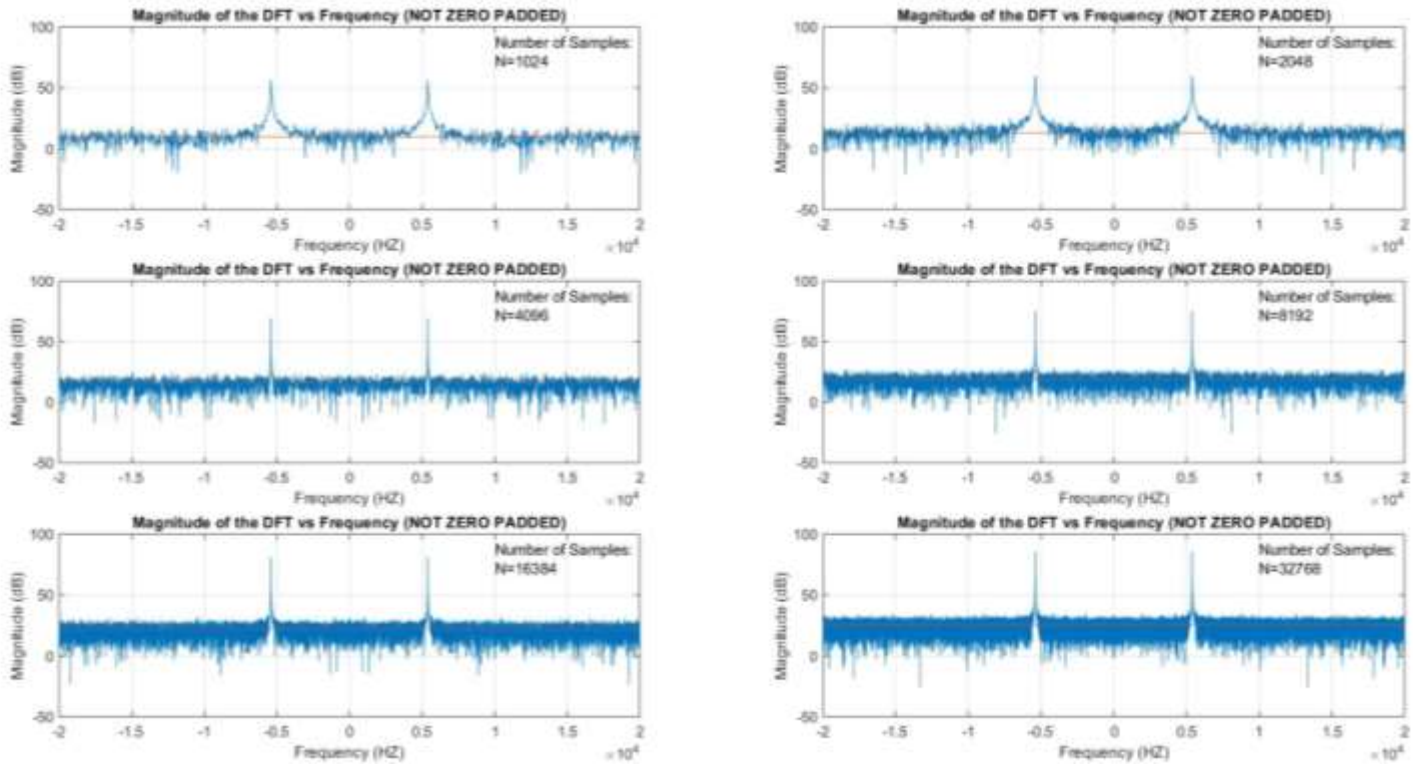

Figure 9: |DFT| vs Frequency (Hz) Non-Zero padded for N samples from 1024 to 32768

As shown in figure 9 the DFT progressively starts to look better with the increasing number of samples used to compute the DFT. The DFT begins to look better due to an increasing number of samples because as the number of samples increases the spacing between the samples of the DTFT decreases. This is because smearing error occurs when the not enough samples are used to compute the DFT. Decreasing the sampling spacing allows the DFT to look more and more like the signal we expect. The sampling spacing is defined in equation 24.

$$Spacing\ Between\ Samples = \frac{F_s}{N}\ (24)$$

In order to analyze how the passive sonar system responds to the received signal we need to calculate the difference between the peak of the received signal in the frequency domain and the noise floor power. The SNR out value is defined in equation 25.

$$SNR_{OUT} = (DFT\ Peak\ in\ dB) - P_{NF}(in\ dB)\ (25)$$

Figure 10 shows the relationship between the Signal to Noise ratio of the DFT and the number of samples taken to compute the DFT. The relationship between the SNR out and the number of samples cannot be easily defined because the number of samples changes the noise floor power as well as the peak of the DFT. For the case of no zero padding applied the signal the difference, defined in equation 25, is not large enough to cause the relationship between the SNR out and the number of samples to be linear.

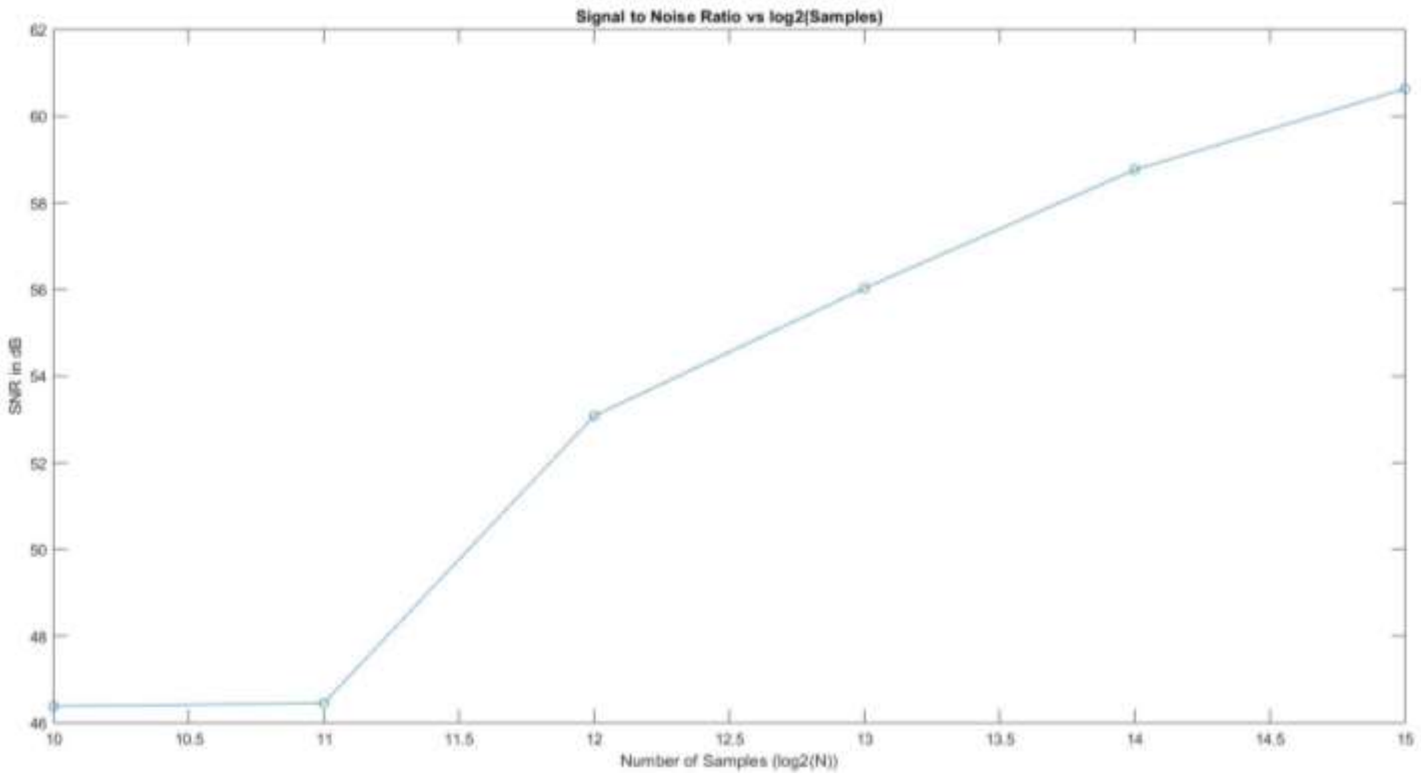

Figure 10: SNR Out in dB vs log2(N) (Non-Zero Padded Case)

So far, we have analyzed how the system responds to no zero padding and in figures 11 and 12 we will apply zero passing to the DFT to see the effect zero padding has on how the signal is visualized using the DFT and the effect zero padding has on the SNR out values.

To understand the difference between figures 9 and 11 the concept of zero padding must be explained further. Zero Padding is the process in which you “tack” on zeros to the end of your signal in order to trick the DFT into thinking there are more samples then there actually are. Zero padding will not make the a bad DFT look more like the DTFT. Zero padding will only just make the bad DFT look better. The only way to make a DFT look more like the DTFT is to take more samples. The difference between a zero padded signal and a non-zero padded signal is evident in figures 11 and 9. Even with zero padding the DFT of the 1024 samples just looks like a better version of the non-zero padded DFT in figure 9, zero padding does not make the DFT of 4096 zero padded samples look more like DFT of 4096 samples with no zero padding. Figure 9 shows the magnitude of the DFT of the received signal with zero padding out to 4 times the number of samples of the received signal vs frequency. By using MATLAB’s fftshift command to compute the DFT we see the DFT centered at 0 ranging from $-F_s/2$ to $F_s/2$.

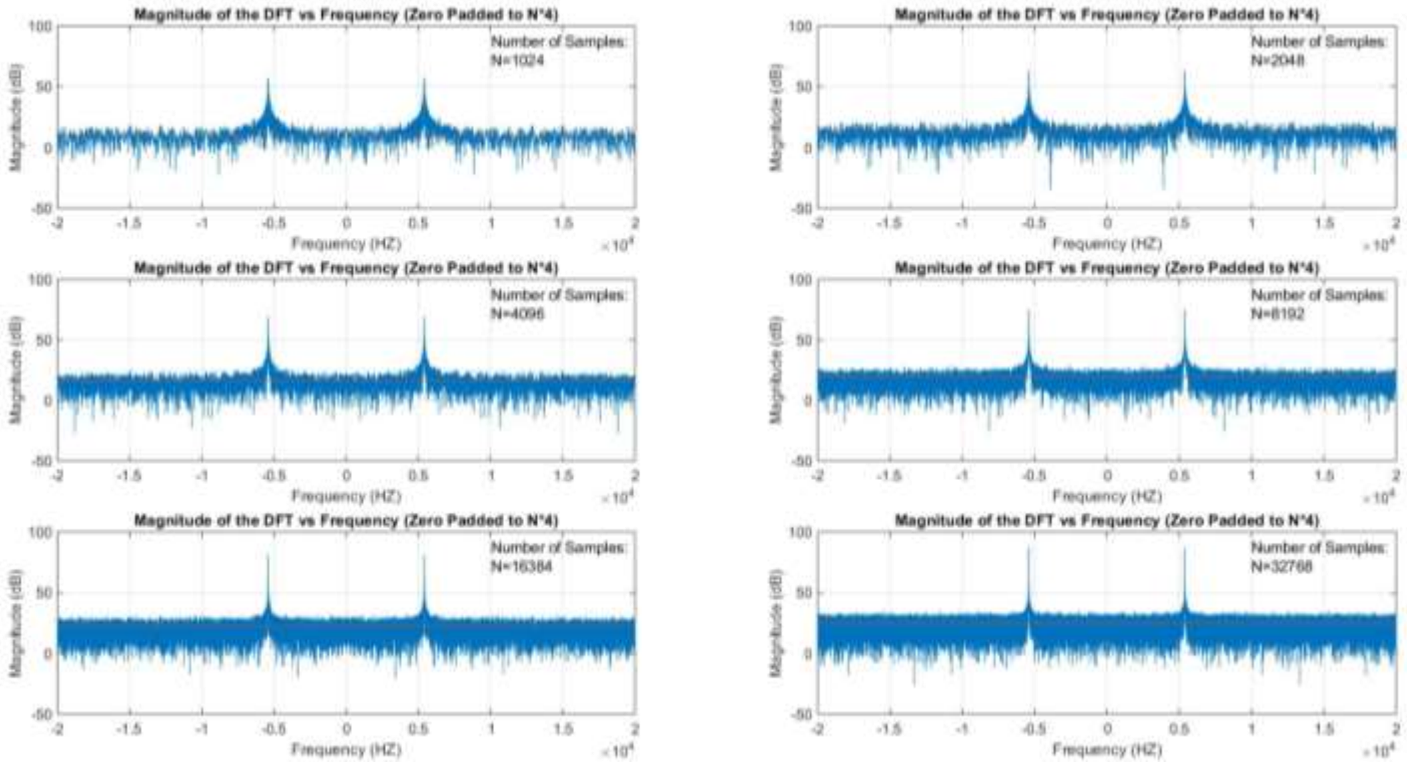

Figure 11: |DFT| vs Frequency (Hz) Zero padded to 4*N for N samples from 1024 to 32768

Although zero padding may not turn a bad DFT into a good one zero padding does have some upside. Since the number of samples is increased when you zero pad the spacing between samples is decreased. When zero padding is applied a more accurate reading of your DFT will be obtained. The zero-padding processing allows more of the signal to be seen than when no zero padding is used thus making applying zero padding to the DFT better than not zero padding.

Using the same process defined in equation 25 the output SNR value is computed again for when zero padding is applied during processing. For the non-zero padding processing there was no apparent trend but when zero padding is applied there is a linear trend, shown in figure 12. The extra samples create a smaller sampling spacing allowing for the DFT peak to be large enough to counteract the change in the noise floor power level in order to create a linear trend.

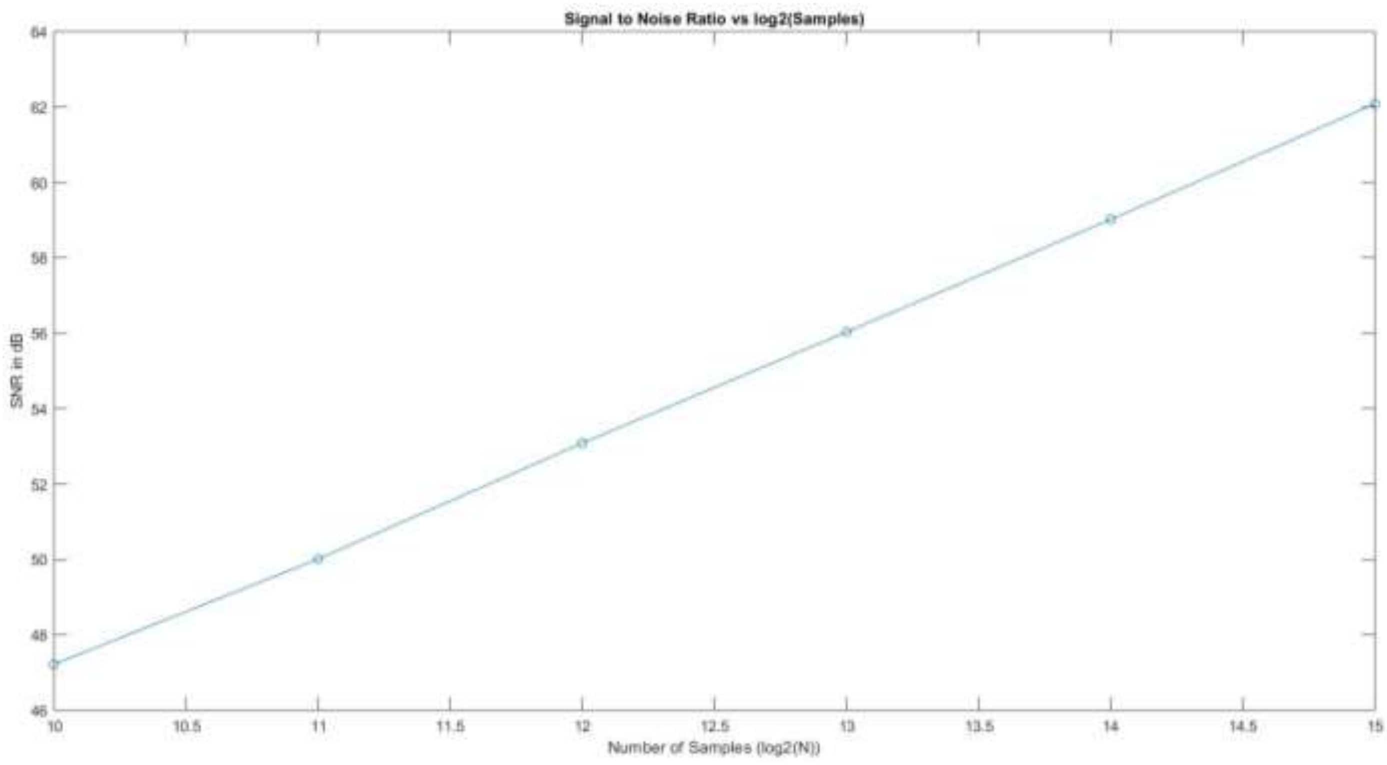

Figure 10: SNR Out in dB vs log2(N) (Zero Padded Case)

In equation 23 we defined the noise floor power level for the given SNR of 100 (20dB) converting the power of the noise floor into a general formula we get equation 26.

$$P_{NF} = \left(\frac{x^2 * N}{SNR_{in}}\right) \quad (26)$$

Since for a sinusoidal signal the power of x is equal to one, we can simplify equation 26 further to find the noise floor power of a given signal with power 1, shown in equation 27.

$$P_{NF} = \frac{N}{SNR_{in}} \quad (27)$$

To determine the peak of the DFT we need to derive the equation for the DTFT of a discrete time-domain cosine sequence, the derivation for the peak of the DFT of a sinusoidal signal is shown in equations 28 through 32.

Starting with a discrete time-domain cosine sequence x[n] as follows:

$$x[n] = Acos\left(\frac{2\pi nk}{N}\right) \quad (28)$$

Where A is the amplitude of the cosine wave, k is the integer number of complete sinusoidal cycles occurring in the N number of samples, and variable n is the time index. The desired N-point DFT of x[n] is shown in equation 29.

$$X[k] = \sum_{n=0}^{N-1} Acos\left(\frac{2\pi nk}{N}\right) e^{-\frac{j2\pi k\mathrm{n}}{N}} \quad (29)$$

Using Euler's formula, equation 29 can be simplified further to the form in equation 30.

$$X[k] = \frac{A}{2}\sum_{n=0}^{N-1} e^{-\frac{j2\pi(k-m)\mathrm{n}}{N}} + \frac{A}{2}\sum_{n=0}^{N-1} e^{-\frac{j2\pi(k+m)\mathrm{n}}{N}} \quad (30)$$

Because the left summation in equation 30 represents a positive-frequency spectral component and the positive-frequency summation is the summation needed to continue deriving the magnitude of the peak of the DFT. Equation 31 shows equation 30 when only dealing with the positive frequency components.

$$X[k] = \frac{A}{2}\sum_{n=0}^{N-1} e^{-\frac{j2\pi(k-m)\mathrm{n}}{N}} \quad (31)$$

The summation in equation 31 is a geometric series, in order to solve equation 31 for to obtain the output magnitude of the DFT the series must be evaluated when $k = m$, which is shown in equation 32.

$$X[k = m] = \frac{A}{2}\left[\sum_{n=0}^{N-1} e^{n*0} = 1 + e^{0} + e^{2*0} + \cdots + e^{(N-1)*0}\right] = \frac{AN}{2} \quad (32)$$

The peak of the DFT of a sinusoid in terms of N and the noise floor power level in terms of N and SNR have been defined so the SNR out can be defined as follows in equation 33.

$$SNR_{OUT} = 20 * \log_{10}\left(\frac{AN}{2}\right) - 10 * \log_{10}\left(\frac{N}{SNR_{in}}\right) \quad (33)$$

Where A is the amplitude of the sinusoid which for the sinusoid the DFT is computed ($\sqrt{2}$), N is the number of samples without zero padding and SNR_{in} is the desired SNR value set by the user.

When the desired SNR of the input is -40dB (10^{-4}) what will the number of samples needed be?

Assuming the SNR out ((DFT Peak in dB)-P_NF (in dB)) needs be about 10dB equation 33 can be rewritten as follows.

$$10dB = 20 * \log_{10}\left(\frac{\sqrt{2}N}{2}\right) - 10 * \log_{10}(10000 * N)$$

Solving for N gives an answer of 262,144 samples before zero padding after rounding up. The magnitude of the DFT vs frequency, for the number of samples equal to 262,144 zero padded out to 1,048,576 is shown in figure 13. The peak of the sinusoid is shown using a MATLAB tool, the x coordinate is 5400 Hz this is the frequency of the sinusoid being analyzed so the peak of the signal is visible through the noise using the DFT when computing 262,144 samples zero padded out to 1,048,576 samples.

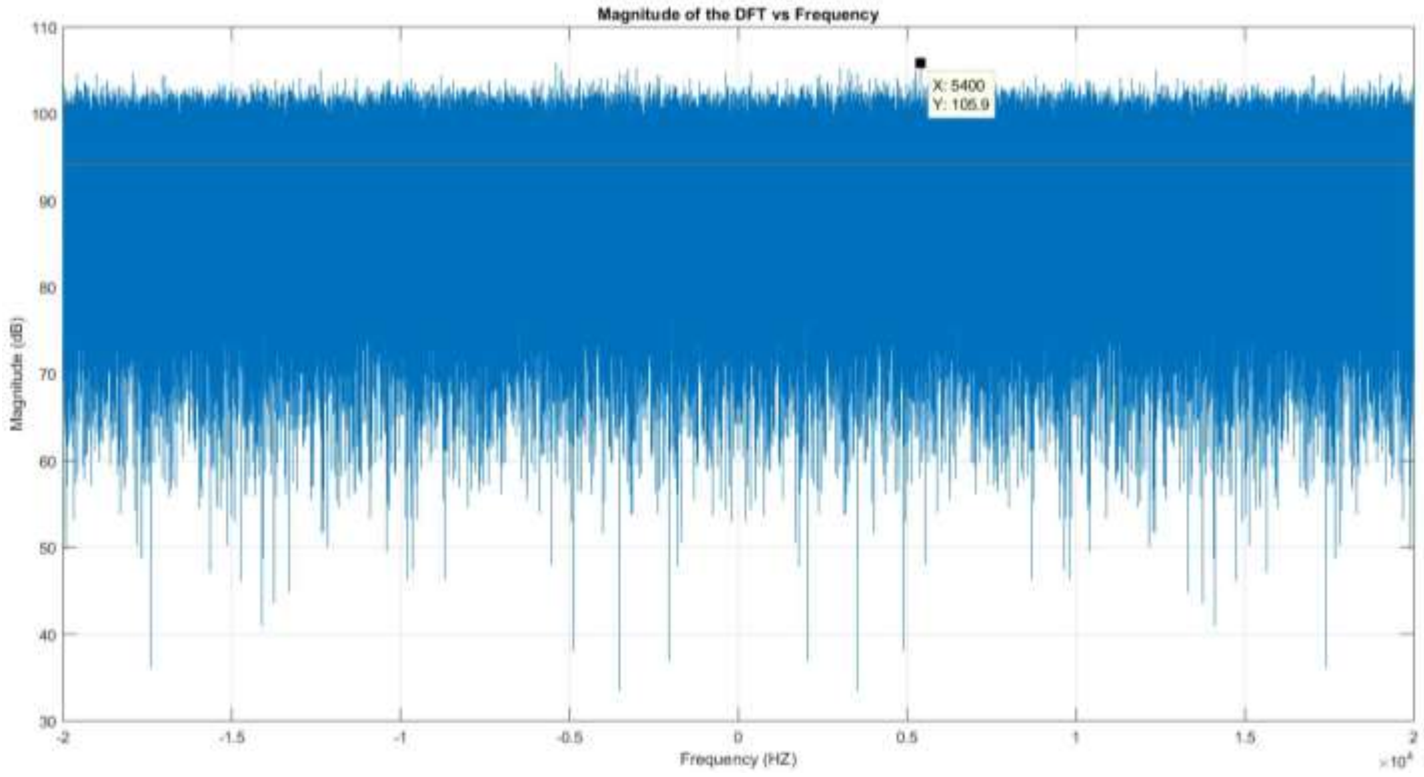

Figure 13: |DFT| in dB vs frequency in Hz for a signal with SNR value of -40dB (Peak Labeled)

Conclusion:

The main purpose of the Passive Sonar with DFT project was to analyze how a Passive Sonar system works using MATLAB and the knowledge learned from EECE 301 (Signals and Systems). The Passive Sonar with DFT project focused mainly on being able to pull signals out from within noise. Being able to pull signals out of the noise is very important because all components (receivers, etc.) and environments (air, water, etc.) in the real work will supply noise to a system. The ability to pull the signal out from the noise ensures the signal cannot get lost within noise. The relationship of the DTFT of the noise and the sinusoidal signal defines how the DFT of the received signal is visualized. Since the DTFT of the sinusoidal signal has two delta functions shifted by the frequency of the sinusoid. The magnitude of the DFT of the sinusoidal signal of the same power of the noise is higher than the magnitude of the DFT of the noise so the signal will be visible in the noise. Due to the theory behind zero-padding, explained in part d, a better peak of the DFT was able to be found when zero-padding was applied than without. Zero-padding of the DFT allowed for a linear relationship between the log base two of the number of samples of the DFT before zero padding and the signal to noise ratio of the system after the DFT was computed to be formed. The linear relationship between the SNR after the DFT after the DFT is computed and the number of samples gives a basis for how to define finding the correct number of samples to take when looking for a desired SNR out based on the specifications of the signal you want to receive and the signal to noise ratio of the received signal. The relationship between the SNR value and the number of samples easily defines a way to drag the signal out of the noise for all input SNR values.