EE 215 Semester Project

SPECTRAL ANALYSIS USING FOURIER TRANSFORM

Department of Electrical and Computer Engineering
Missouri University of Science and Technology

Table of Contents

Notice: In the cosine function section of the results, the plot titles for Figures 5, 7, 9, and 11 are incorrect. The three plots in each figure should read "N = 80 (a = 10)," "N = 400 (a = 50)," and "N = 800 (a = 100)," respectively.

Introduction

During this day and age, technological improvements are made every day and the world must try and keep up with the rapid growth. In order to make these improvements, one must understand what is going on with the system that is in use. It is important to know how certain inputs will affect the system and its output. Several questions get raised. Is the correct input being carried through? Does the system create unwanted noise? Can the system handle it? These become answered when signal analysis is done on the system.

The signals coming into the system will be in continuous time, however, the system does not have the capabilities to use a signal with an infinitely small time step, so sampling is done to make the process easier. A sampling frequency is chosen based on the frequency of the signal and the signal is sampled at time intervals equal to the inverse of the sampling frequency. This creates a discrete time signal, which can now be used by the system.

It is vital to if the signal passing through the system is truly the desired signal. Also, the system is not perfect or ideal, so there will be noise or leakage created. Before allowing the system to run through its process, control signals engineers must first analyze what is going on with the signals. It is easier to do this when the signal is in its frequency domain rather than its time domain. This is done using a technique known as Fourier transforms. A simple way of describing a Fourier transform is taking a signal and turning it into a series of sinusoidal functions. Now that the signal is in the frequency domain, it is much easier to see what is desired: the spectral content of the signal. MATLAB will be the tool that will allow for this analysis.

Procedure

The analysis of spectral content was conducted on two separate continuous time signals: $x(t) = \cos(2\pi Ft)$, where $F = 5{,}000\ Hz$, and $x(t) = rect(t/a)$, where $a = 10$. These signals are first going to be sampled and converted in to discrete time signals. To get a better grasp of Fourier transforms and how to understand the spectral content, four different methods will be used to obtain the transforms.

The first method is to use the definition of discrete time Fourier transform directly. The definition states that if you have a discrete signal $x[n]$, then the Fourier transform $X(F) = \sum_{n=-\infty}^{\infty} x[n]e^{-j2\pi Fn}$. A function for this summation was created in MATLAB to simplify coding later. The following script is the code for this DTFT function:

This function calls three separate variables: the discrete time vector "n", the frequency vector "f", and the discrete signal "x". The output provides the DTFT "X". The division by the length of the time vector is a method of correcting the amplitude. This is necessary for time unlimited signals like the cosine function. However, it is not needed for a time limited signal like the rectangular pulse. This script is mainly for coding the first method, but it will also be used in the second method.

The next way a finding the Fourier transform is using a modified version of the previous method. This one begins with finding the theoretical DTFT of the signals using known transformation pairs. Then, using the code from above, the numerical DTFT is found of the

window function, in this case a rectangle function, which depicts the window size of the original signal. Both of these Fourier transforms are then convolved in MATLAB to get the modified DTFT that is desired.

The third method is applying an algorithm known as the discrete Fourier transform, or DFT. This algorithm, like the DTFT, requires the use of a summation to acquire the Fourier transform. The algorithm states $X[k] = \sum_{n=0}^{N-1} x[n]e^{-j2\pi nk/N}$, where x[n] is the discrete time signal, N is the window size or number of samples, and k is the harmonic number, which is equivalent to the sampling frequency divided by the window size. A function was created in MATLAB for this summation just as one was made for the DTFT. The code is as follows:

This function makes the actual coding for the simulations easier and cleaner. It requires four inputs: the discrete time vector, the harmonic number vector, the window size, and the discrete signal. The output required an amplitude correction similar to the DTFT.

The final way of finding the Fourier transform is using a pre-built function in MATLAB. This function is called the Fast Fourier Transform, or FFT. The FFT uses the same algorithm as the DFT, however it only requires the user to input the discrete time signal and then it calculates a rather accurate depiction of the Fourier transform. This is very commonly used, however it is the final method shown as a comparison to the other methods, which tend to have some more accuracy.

These methods can provide the wrong results if minimum requirements are not met for two parameters: the sampling frequency fs and the window size N. The goal of this project is to

find the best combination of the two parameters to give an accurate depiction of the transform and that will not be too hard on an embedded system. There is a methodical way of choosing these parameters as long as the signal is periodic, like the cosine signal that will be analyzed first. The sampling frequency will be chosen based on the Nyquist criteria, which states that the signal should be sampled at twice the signal's frequency or higher. The number or samples or window size is determined by the ratio between the sampling frequency and the signal's natural frequency. The size can then be changed by multiplying it by a constant of the user's choosing. Several frequencies below, at, and above the Nyquist rate shall be tested along with various window sizes to determine, which is the best to use. For the rectangle function, which does not have a period, the previous way of finding the parameters does not work. This requires more experimenting. Several different sampling frequencies must be tested start low and increasing and the same goes for the window size. Then, the best combination must be chose from these results keeping two questions in mind. Is this an accurate enough depiction of my signal? Is this feasible for an embedded system with limited memory?

Experimental Results

Part 1: The Cosine Signal

The first signal analyzed was $x(t) = \cos(2\pi Ft)$, where $F = 5{,}000\ Hz$. The first step in the procedure is to sample it can change it from continuous to discrete time. This was done by testing with several different sampling frequencies and constant window size and then a constant frequency with differing windows. Shown below is the continuous signal along with discrete plots with varying sampling frequencies and window sizes.

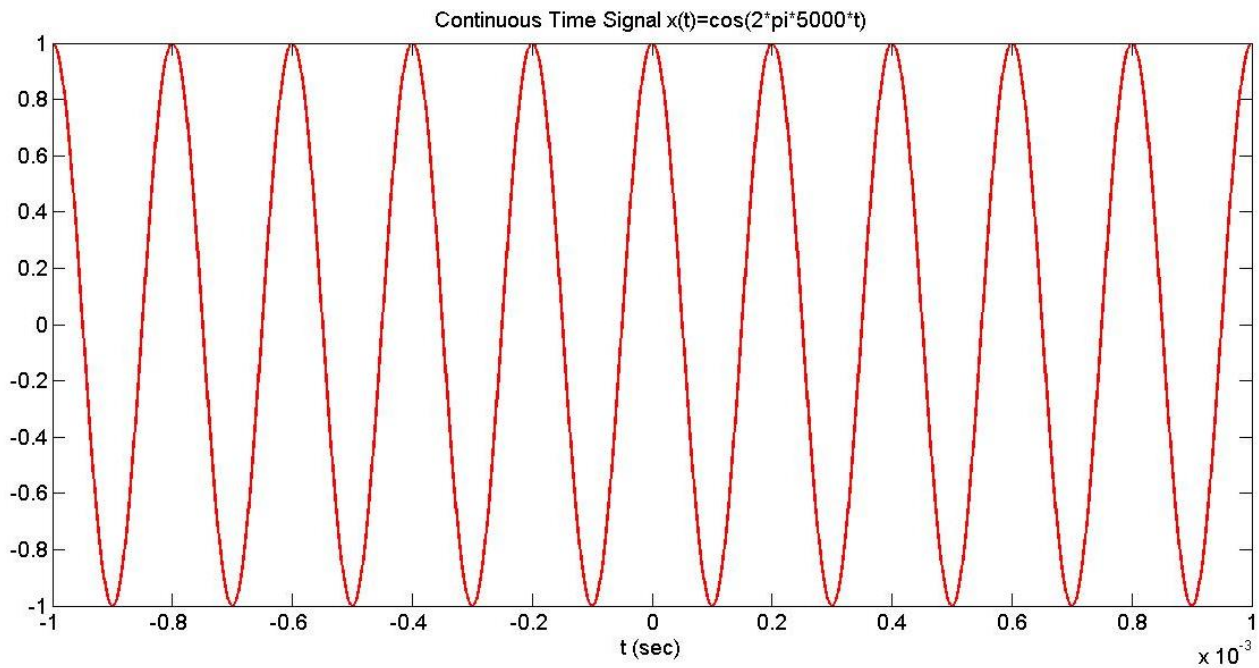

Figure 1: The plot of $x(t) = \cos(2\pi 5{,}000t)$

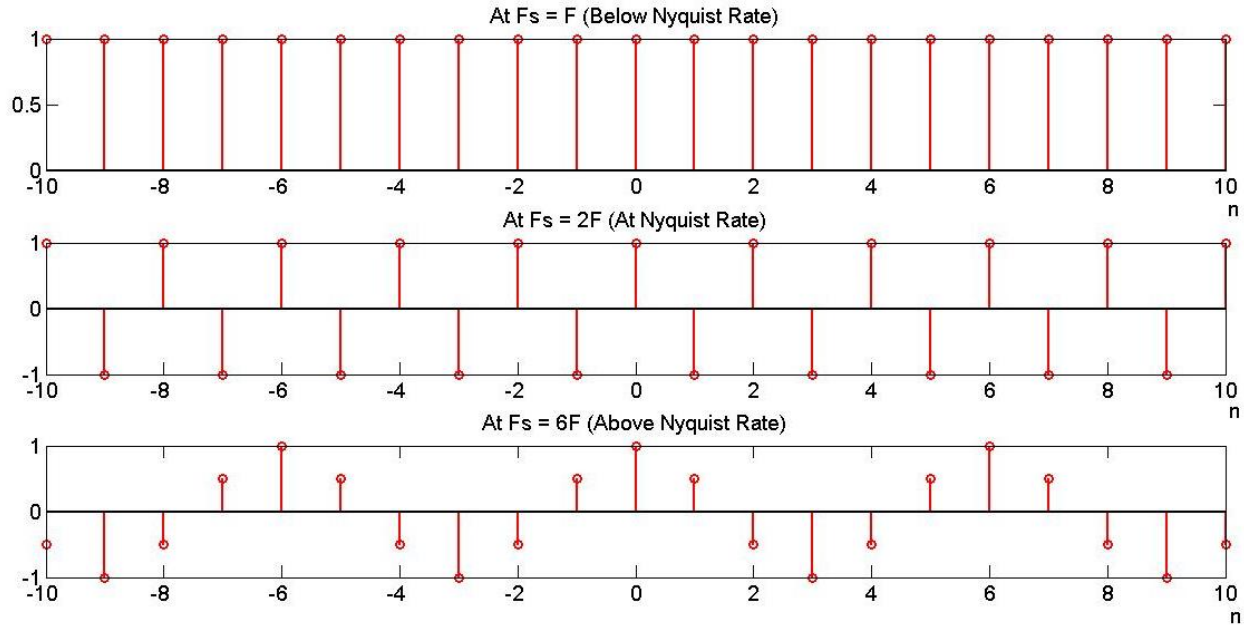

Figure 2: Plots of $x[n] = \cos(2\pi 5{,}000 nTs)$, where Ts, the inverse of Fs, varies

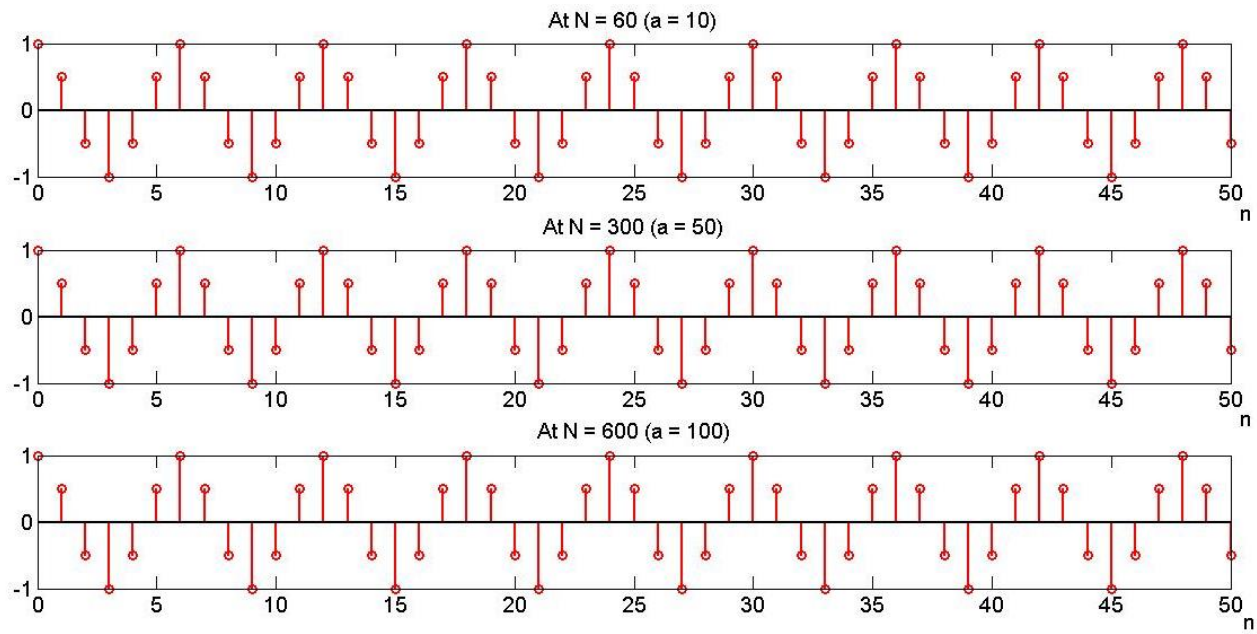

Figure 3: Plots of $x[n] = \cos(2\pi 5{,}000 nTs)$, where n varies

In Figure 2, it is obvious that the higher sampling frequencies present a better depiction of the original signal. More than three were tested, but those in Figure 2 were the most important to display. The first plot in the figure has a sampling frequency equal to F, which puts it below the Nyquist rate. As it can be observed, the outcome is one at every instance of n. At the Nyquist rate as in the second plot, the maxima and minima are all that can be seen. However, above the Nyquist rate, particularly at six times F, the minimum number of points to describe the plot as a cosine are now present. In Figure 3, 6F was chosen to be the constant sampling frequency as the window size was changed. This was done by varying the constant A, which is multiplied by the ratio of Fs by F. Three values of A were used in the plots in Figure 3: 10, 50, and 100. However, changing the window size had no effect on the appearance of the discrete signal itself.

These discrete signals were then run through the DTFT code to find the numerical Fourier transform, but from this point on, 8F will be used because 6F caused issue further in with the modified DTFT. The following two figures are plots from the DTFT code.

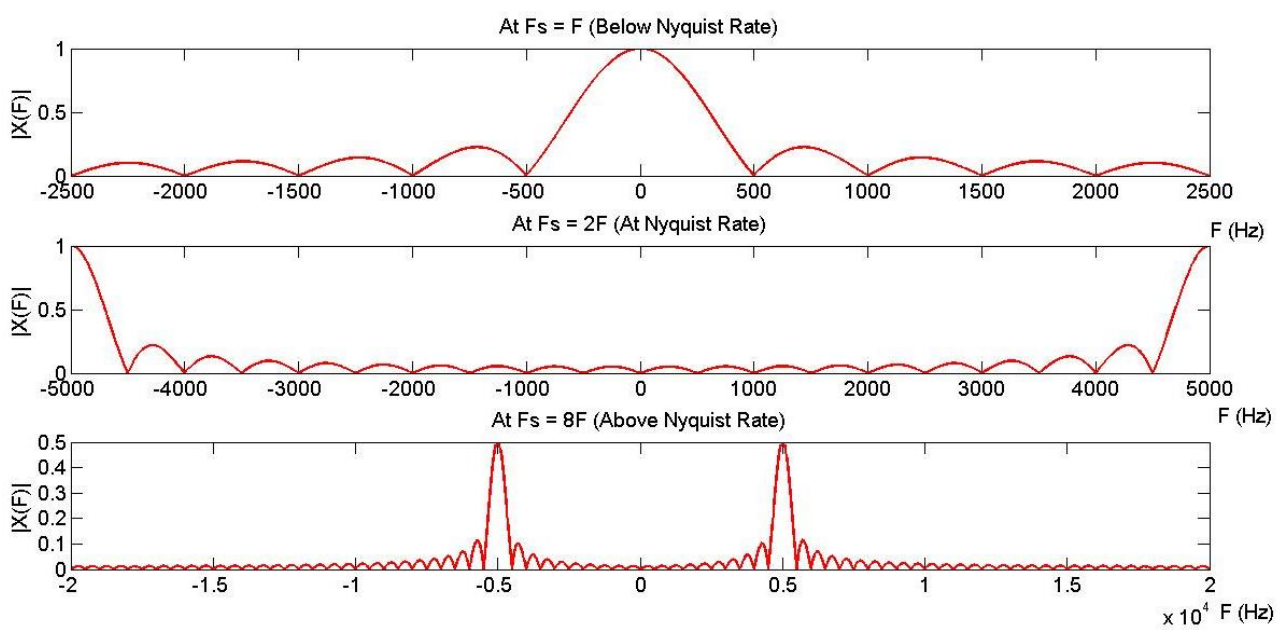

Figure 4: Amplitude plots of X(F) with varying Fs

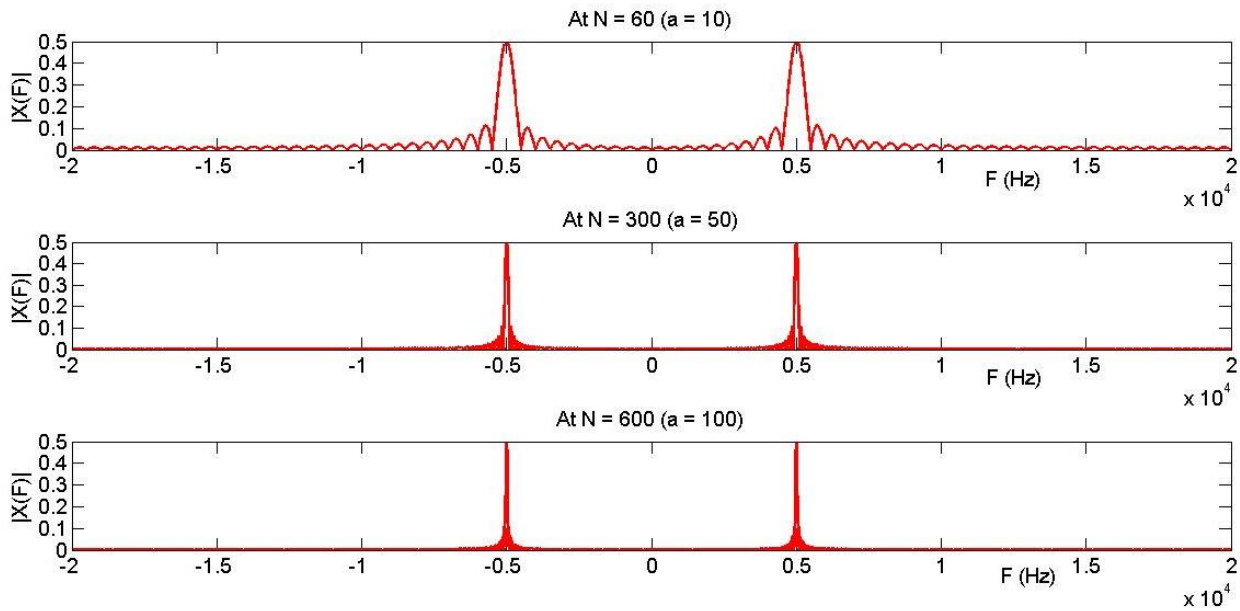

Figure 5: Amplitude plots of X(F) with varying A

Theoretically, within one period of Fs there should be two impulses at ±5,000 Hertz with amplitudes of 0.5. This numerical method will produce leakage, so instead of impulses, Sinc functions will appear at ±5,000 Hertz. However, it can be observed that is not the case for a sampling frequency under the Nyquist rate as in the first plot of Figure 4. At the Nyquist rate, the Sincs are in the correct location, but the amplitude is 1.0, which is incorrect. The third plot at Fs equaling 40,000 Hertz depicts an output close to what is desired. It has the Sinc profile at ±5,000 Hertz and the amplitude is at 0.5. Figure 5 plots use the 40,000 Hertz sampling frequency, but use the values of A from earlier. It is observed now that as the window size increases, the output begins to get closer to impulses instead of Sinc functions. At A=100, the plot looks best, but the plot produced from A=50 also depicts an accurate Fourier transform with minimal leakage.

The theoretical Fourier transform of $x[n] = \cos(2\pi 5{,}000 nTs)$ is found using a table of standard Fourier transform pairs provided by the instructor of the course. The pair related to this particular signal is $\cos(2\pi F_0 n) \xleftrightarrow{F} (1/2)\,[comb(F - F_0) + comb(F + F_0)]$. For the signal in this case, $F_0 = 5{,}000Ts$. This is coded in to MATLAB along with rectangle functions with widths equal to the window size of each case. The DTFT code was then used to find the numerical DTFT of each window function. These are then convolved with their correlating analytical DTFT. This convolution produces the modified DTFT and should be nearly equivalent to the numerical DTFT in Figures 4 and 5. Below are the resulting plots for the modified DTFT.

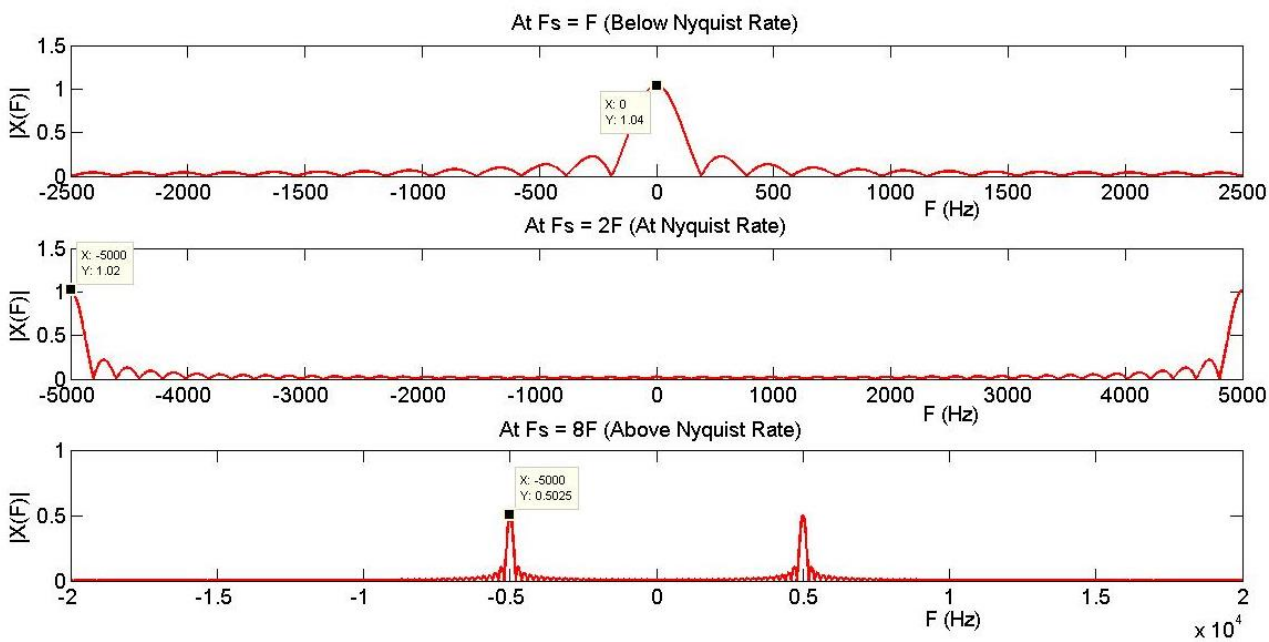

Figure 6: Amplitude plots of modified X(F) with varying Fs

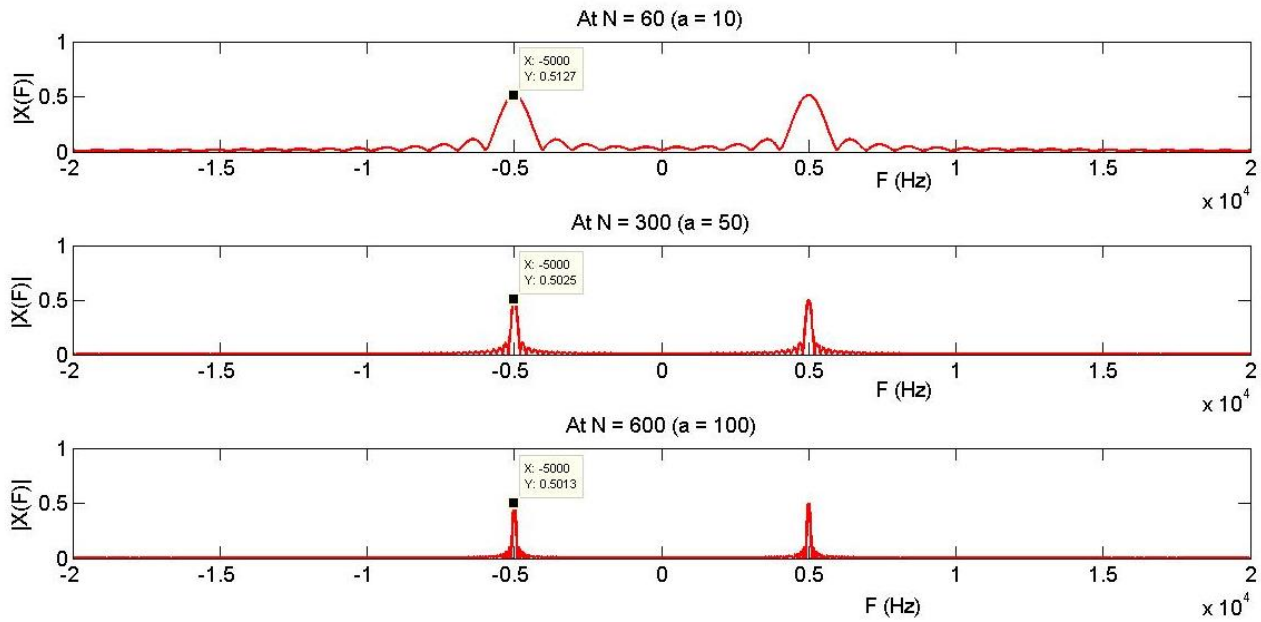

Figure 7: Amplitude plots of modified X(F) with varying A

Just like the DTFT, sampling at or under the Nyquist rate yields incorrect results. Eight times the frequency is again the best sampling frequency. Something that observed using the data tips on the plots, the amplitude is slightly off from 0.5. From Figure 7, it can be seen that A=50 still provides an adequate plot, but it still has a slightly different amplitude than it should. As the window size continues to increase, the amplitude should eventually correct itself.

The idea of the Nyquist criteria truly comes into play when dealing with the discrete Fourier transform. It is still used to define the sampling frequency and window size, but it is value of N also creates the bounds for the summation in the DFT algorithm. The same parameters used in the first two methods were once again established in MATLAB along with the new parameter k, which is a vector from -2N to +2N. These parameters were run through the DFT code created earlier and the following plots were produced.

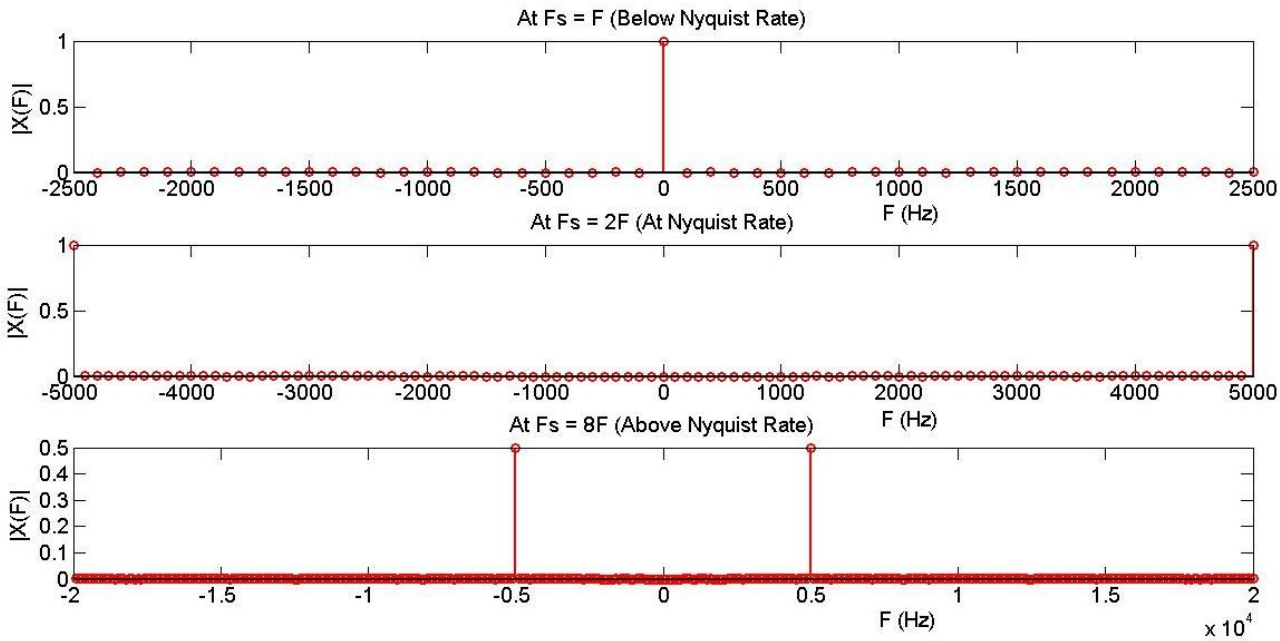

Figure 8: Amplitude plots of X(F) from DFT with varying Fs

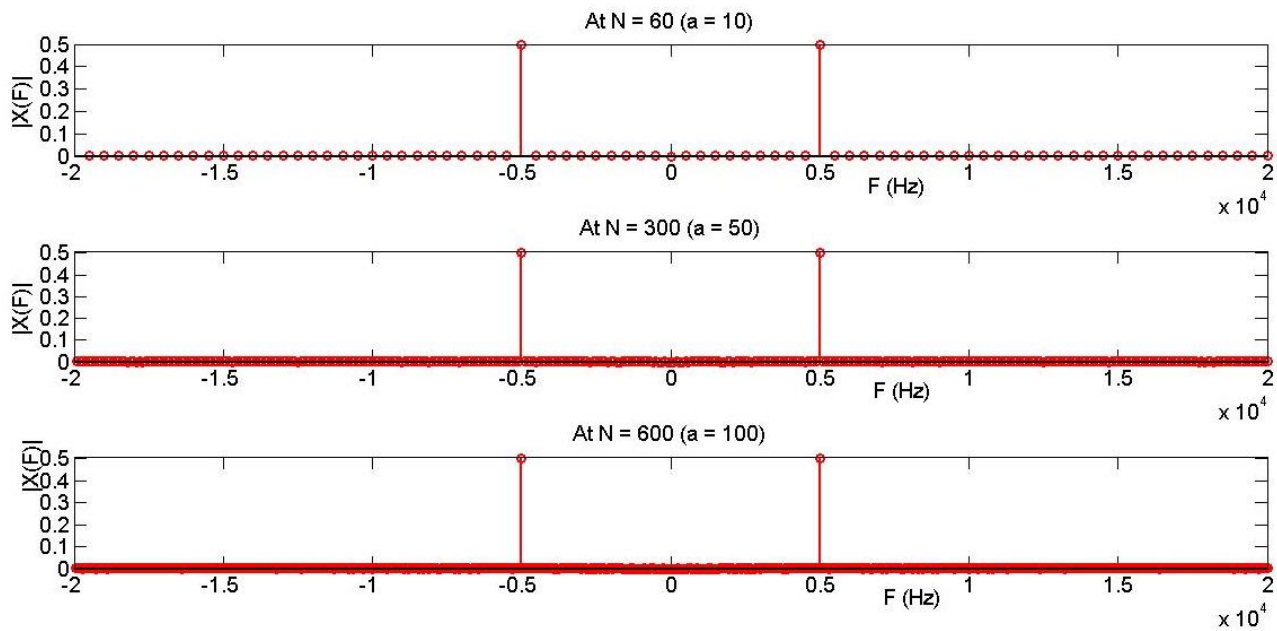

Figure 9: Amplitude plots of X(F) from DFT with varying A

These were plotted using the “stem” command in MATLAB, which produced an output that looks exactly like the theoretical Fourier transform. This method is a great way to avoid the leakage that comes with the numerical analysis. Like before, sampling at or under the Nyquist rate should be avoided and eight times the frequency still works great. Window size does not play to large of a factor in these results. As observed in Figure 9, the plots all have the same impulse at ±5,000 Hertz and amplitude of 0.5. The only difference is the number of samples visible, but even at A=10, this is still a viable output.

For the FFT method, the same parameters are again used and run through code as seen in Appendix I that includes the pre-built FFT function and creates a frequency range to plot against. Though this uses the DFT algorithm, it is still plotted normally instead of with stems. Here are the produced plots.

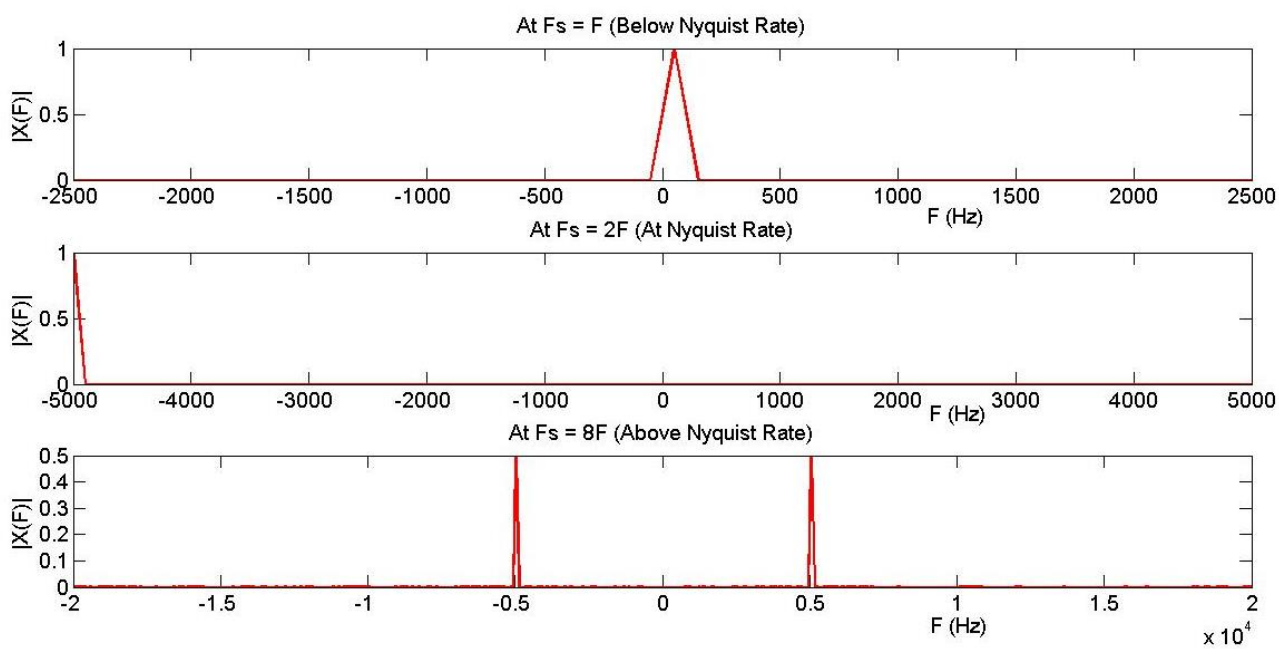

Figure 10: Amplitude plots of X(F) from FFT with varying Fs

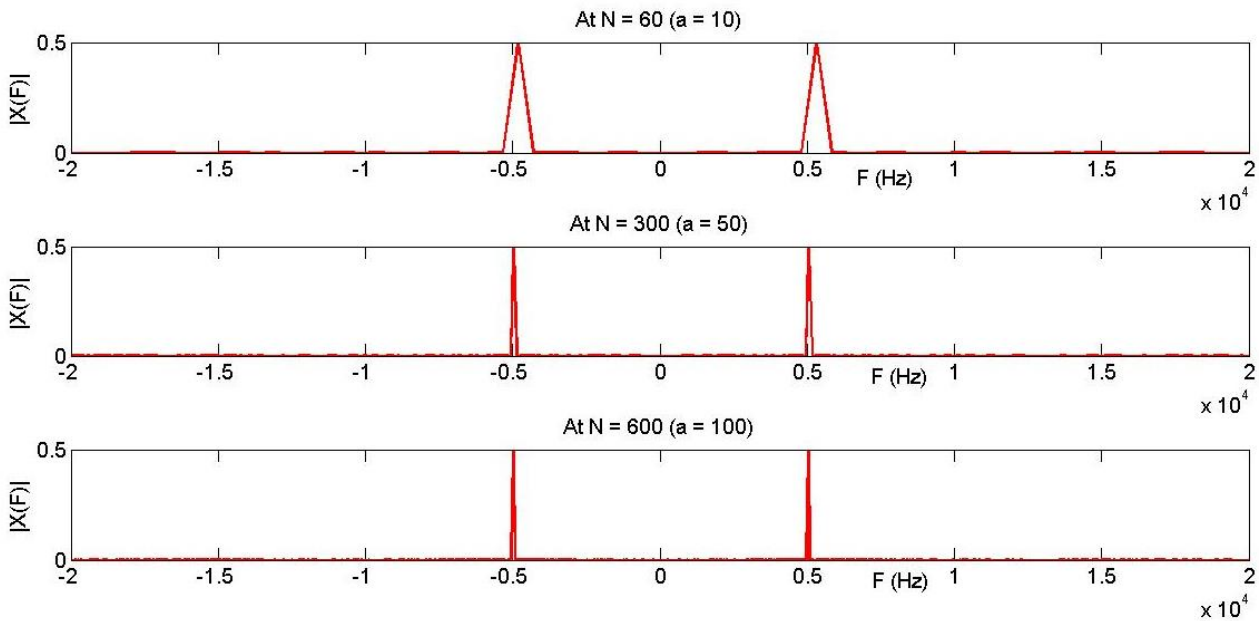

Figure 11: Amplitude plots of X(F) from FFT with varying A

Figure 10 again reiterates that 8F is the best sampling frequency to use. This time at the Nyquist rate, not only is the amplitude incorrect, the FFT failed in plotting the positive side of the spectrum. It is clear that the window size plays a major role in the outcome of the FFT function. The plot with a window size of 800 looks much more like an impulse than that of a window size of 400, however for all intents and purposes, 400 or A=50 will work perfectly fine.

Part 2: The Rectangular Function

The second signal to be analyzed is $x(t) = rect(t/a)$, where $a = 10$. Just like with the cosine, this was converted into discrete time and then the Fourier transform was found using the same four methods. The only difference this time was that since the rectangular pulse has no frequency, the samples had to be taken at as many different amounts as possible and compared. Below are the Fourier transform plots produced, two for each of the four methods. One plot has three different sampling frequencies, while the other has different window sizes.

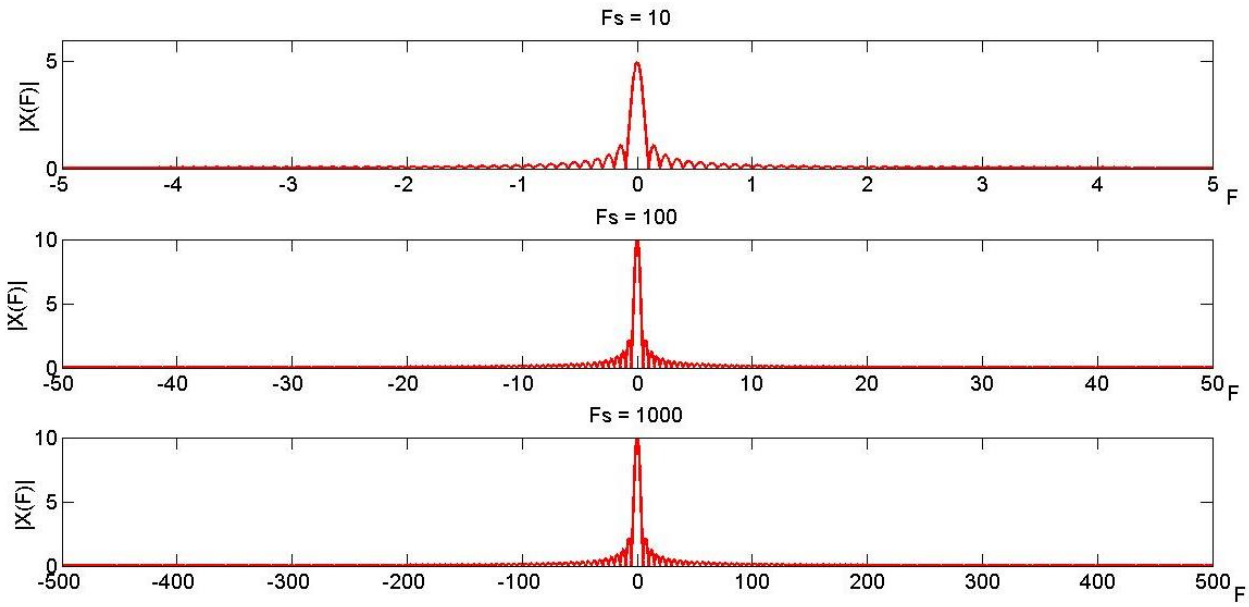

Figure 12: Amplitude plots of X(F) with varying Fs

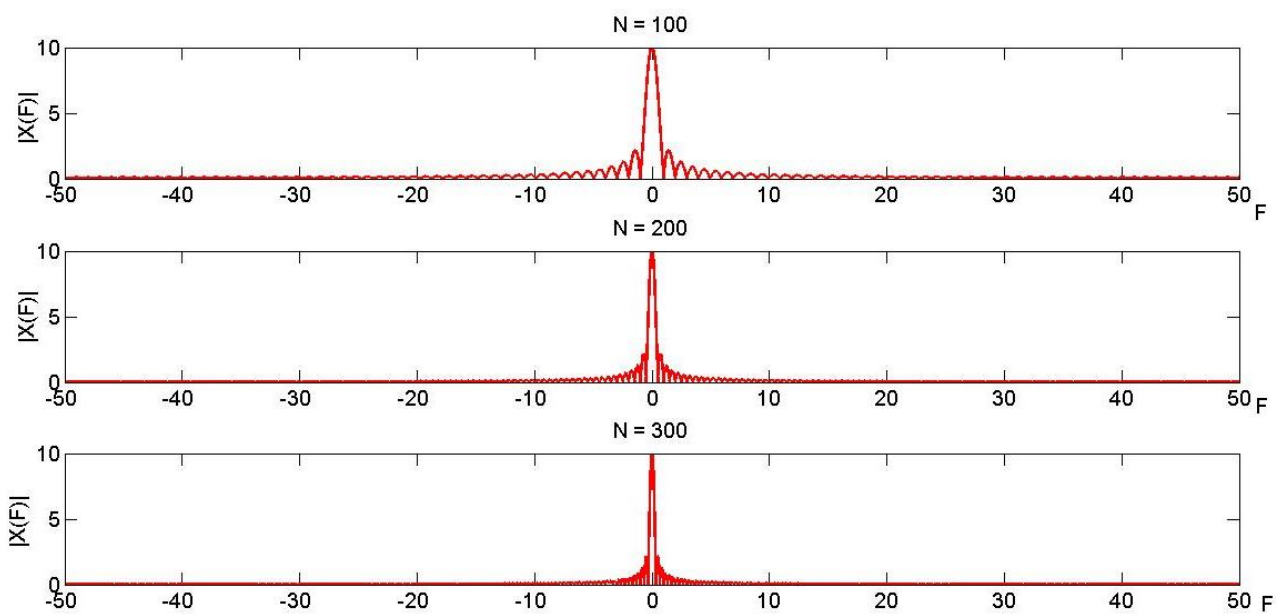

Figure 13: Amplitude plots of X(F) with varying N

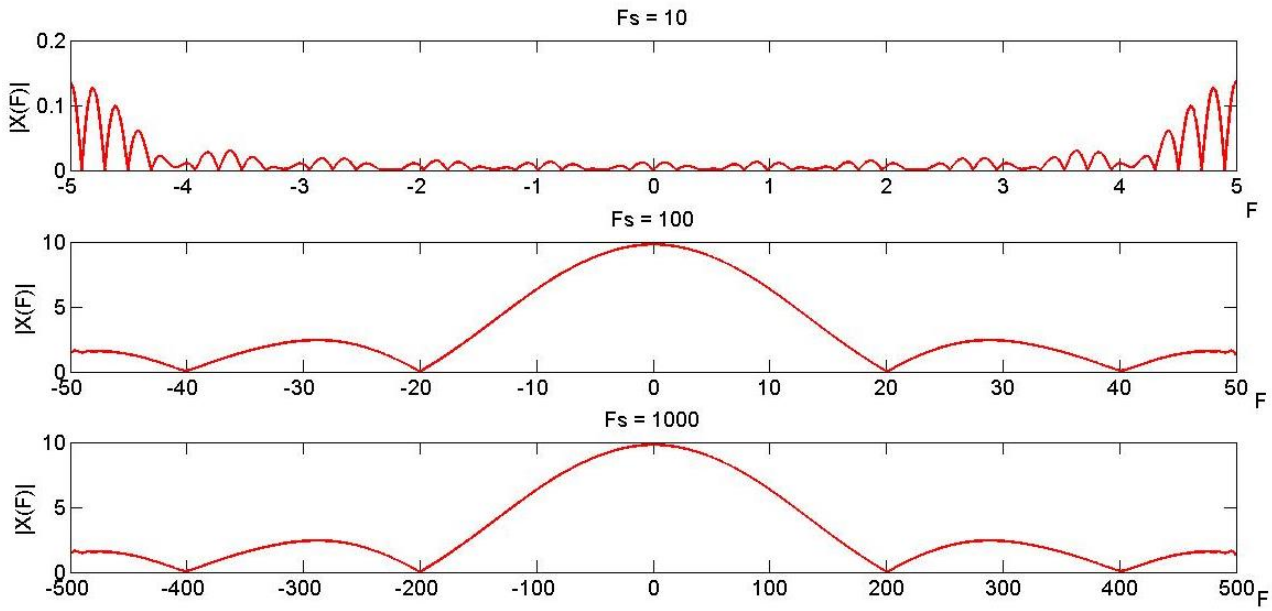

Figure 14: Amplitude plots of modified X(F) with varying Fs

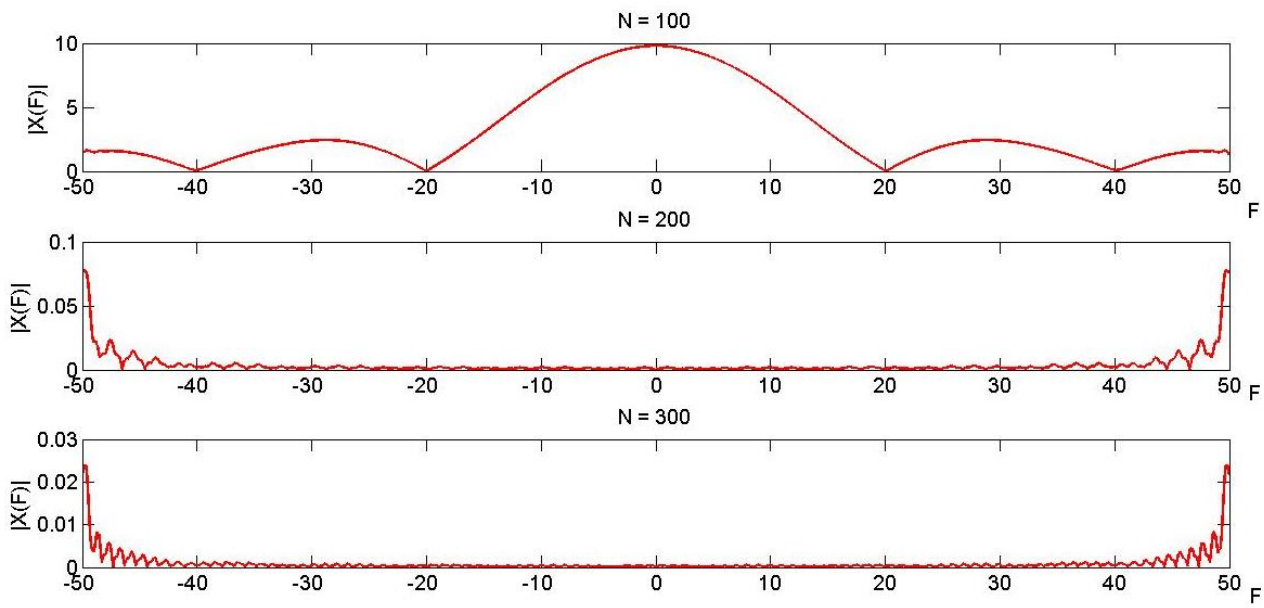

Figure 14: Amplitude plots of modified X(F) with varying N

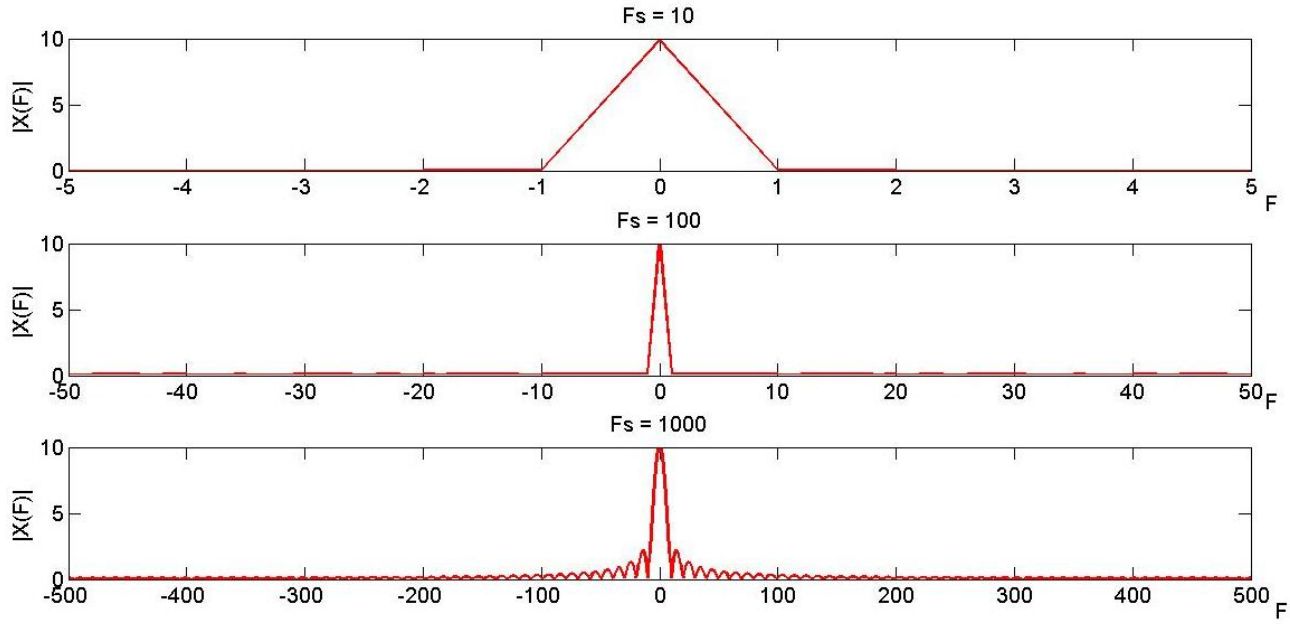

Figure 15: Amplitude plots of X(F) from DFT with varying Fs

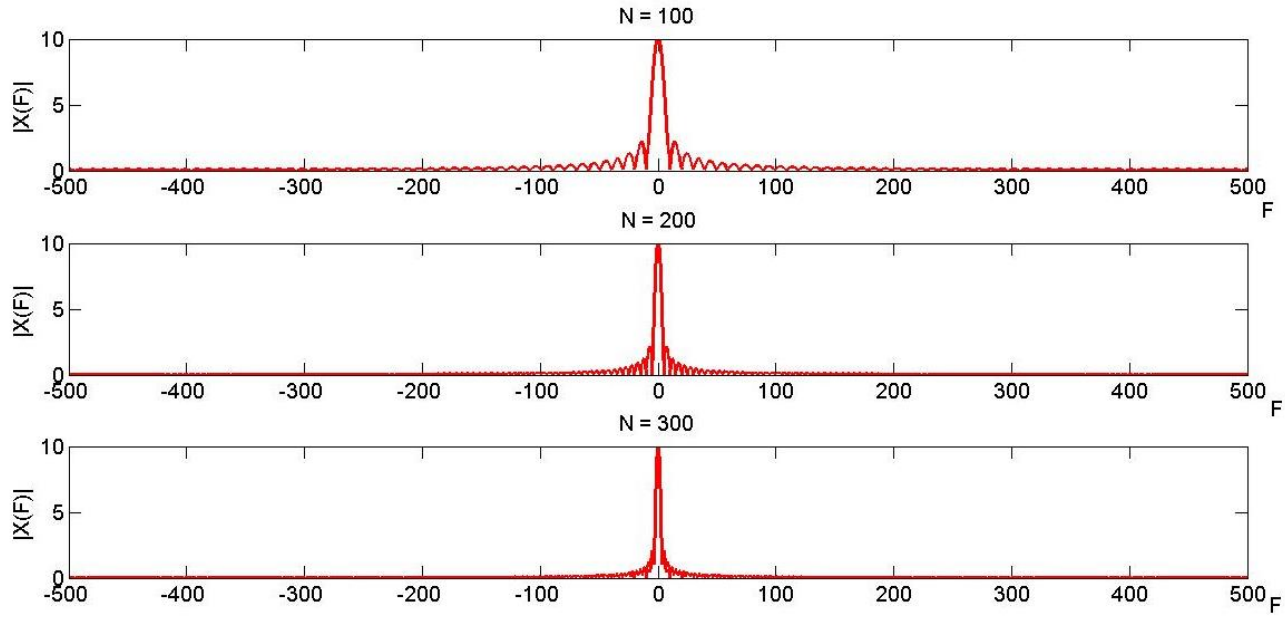

Figure 16: Amplitude plots of X(F) from DFT with varying N

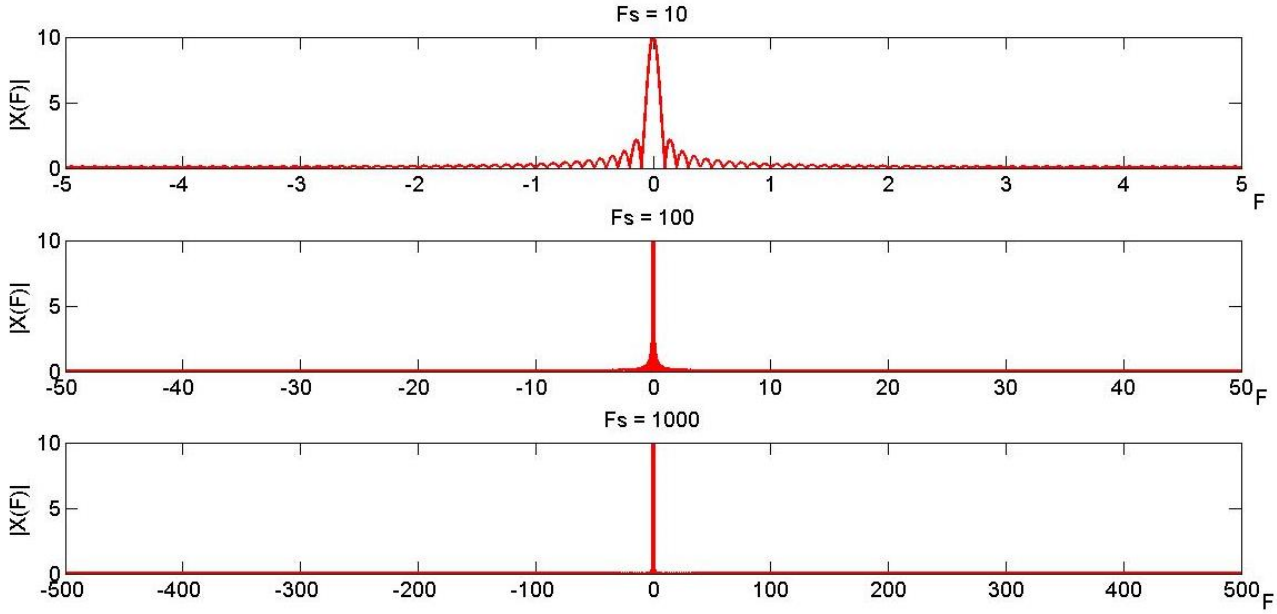

Figure 17: Amplitude plots of X(F) from FFT with varying Fs

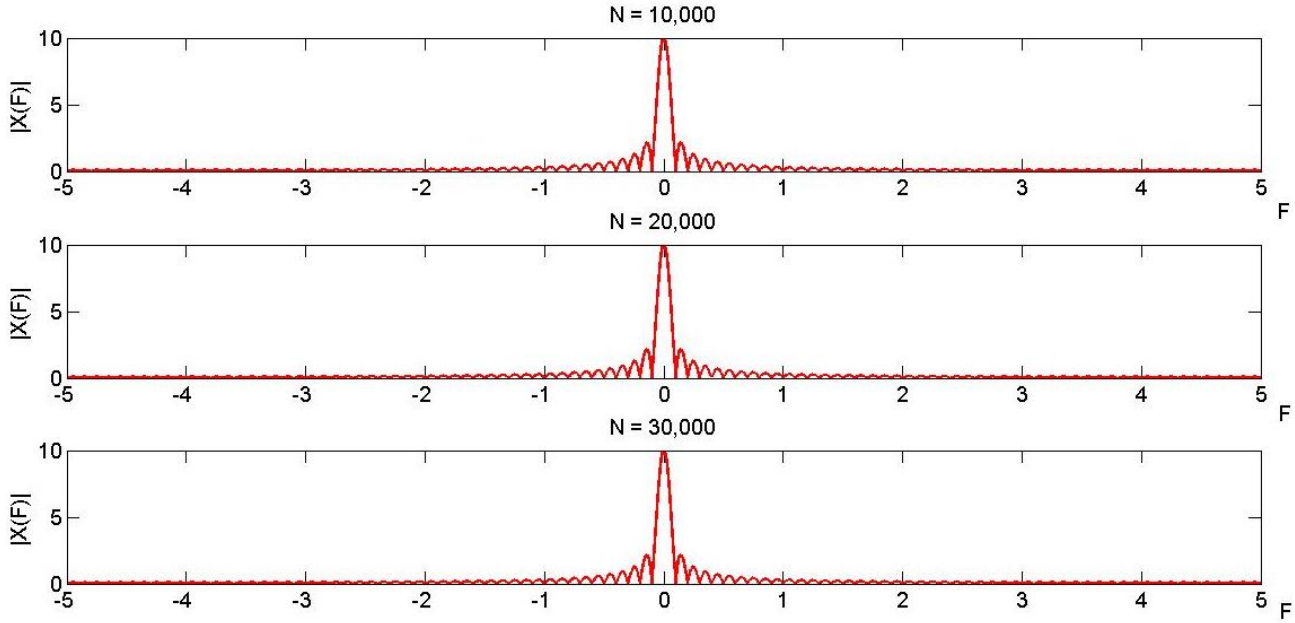

Figure 17: Amplitude plots of X(F) from FFT with varying N

Just from a quick run through the plots, several discrepancies are easily observed. The theoretical Fourier transform of the rectangular pulse is a Sinc function. All of the methods were able to create transforms that were Sinc functions, however the modified DTFT method could not create the correct signal. The DTFT method was able to produce the correct transform no matter what sampling frequency was used and increasing window size just made it appear closer to an impulse at zero. The DFT and FFT methods required specific parameters in order to successfully create the proper transform. For the DFT, the sampling frequency of 100 was not enough to give it the Sinc profile, but the window size had about the same effect as the DTFT. The FFT required a very low sampling frequency and a very large sample size in order to get the correct transform, but once the window size is large enough it seems to no longer have any effect on the plot. So unlike the cosine function, where each method produced the same plot for each sampling frequency tested, the rectangular function required different parameters for each method with one method not even working correctly.

Conclusion

In this world of technology, signal processing is a very important area that requires much testing, analysis, and some common sense. The analysis of signals need to be done in the frequency domain, so the time signals coming into the system are converted via Fourier transform. Four methods had been applied in this experiment to determine the transformed signal. For a cosine signal, it is easy. Its period helps determine the best sampling frequency and number of samples to be taken. Through the tests run in this experiment, it is determined that most efficient sampling frequency is eight times the frequency of the signal. This allows the system to read the correct signal and yet is not too hard on an embedded system. An adequate window size or number of samples for a periodic signal like this cosine would be fifty times the ratio between the sampling frequency and actual frequency, so along with the recommended Fs, N should be equal to 400 samples.

With an aperiodic signal like the rectangular function, using the same four methods are more difficult. The modified DTFT, for instance, is not a valid way to find the correct Fourier transform. Also, unlike with the cosine, one set of sampling frequency and sample size does not allow the other three methods to create the same transform plots. The DTFT code was able to handle any parameter that went through it and it would output the correct plot, so the recommended Fs was 100 Hz and sample size of 100 samples. For the DFT, a higher sampling frequency is needed such as 1,000 Hz and the number of samples could remain the same as with the DTFT because it had no effect. The FFT was much different. It required a very low frequency of 10 Hz and high number of samples like 100,000 samples. All of the options could still be easily handled by an embedded processor.