Robert Irwin

ECE 3522: Stochastics

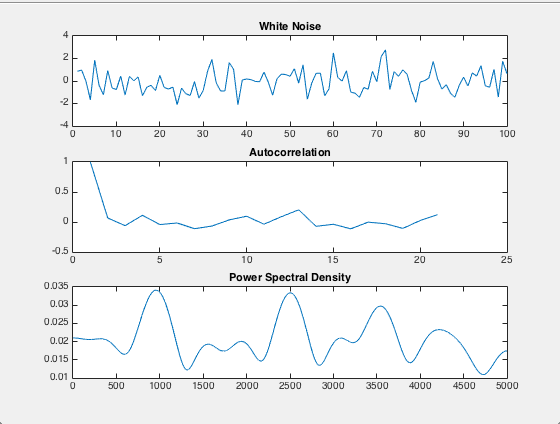
Department of Electrical and Computer Engineering, Temple University, Philadelphia, PA 1912

# Problem Statement

In this assignment we are attempting to find the autocorrelation of various functions. We will be analyzing a Gaussian white noise signal, an impulse function, a periodic impulse train, a sine wave, a portion of a sine wave, and a noisy sine wave. Once we have the autocorrelation of each of these functions, we will find the power spectral density of each function by computing the Fourier Transform of the autocorrelation function.

# Approach and Results

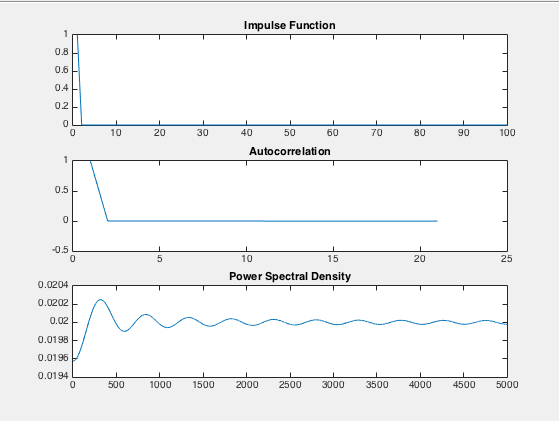
Our first step of the assignment was to define the functions in MatLab. This will be discussed further in the MatLab code discussion. The signal, autocorrelation function, and power spectral densities for the Gaussian white noise signal are shown below in Figure 1.



*Figure 1: Results for the Noise Signal*

As discussed in class, the autocorrelation function of Gaussian white noise should be 1 when tau equals 0, and 0 afterwards. This means that the autocorrelation is an impulse function, and it is what we see in the figure above. The slight error comes from the fact that we can not generate an infinite number of data points to construct the noise signal with. This means that the signal above is an approximation of white noise. We also know that the Fourier transform of an impulse function is a flat line in the frequency domain. If we look closely at the scale of the power spectral density plot, we notice it is extremely small. This tells us that the power spectral density is approximately constant, which is what we expect. Again, this error can be attributed to the way we generated the white noise signal.

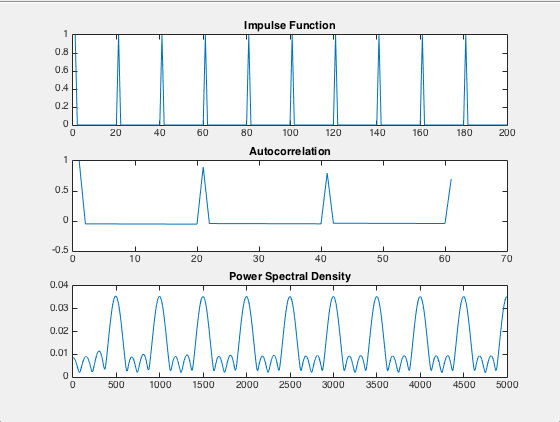
Now we will look at the results for the impulse function. They are seen below in Figure 2.



*Figure 2: Results for Impulse Function*

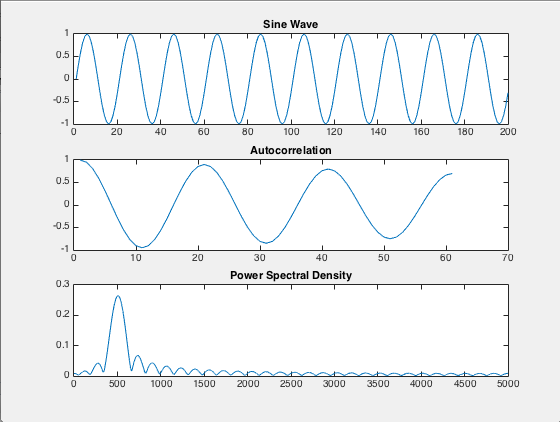
In the figure above it is evident that the autocorrelation function of an impulse quickly drops to 0. This is because an impulse function is zero everywhere except at x = 0. We also notice that the power spectral density is basically a flat line in the frequency domain. This is because the autocorrelation function is basically an impulse function, which we know has a Fourier Transform that is a flat line.

Now we look at the results for the periodic impulse train.



*Figure 3: Results for Impulse Train*

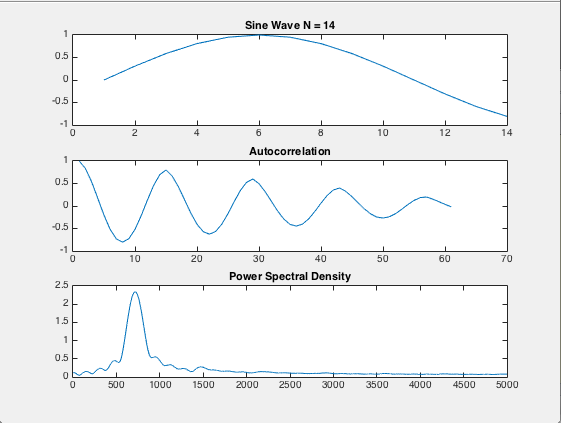
Here we see that there is a spike in the autocorrelation function everywhere there is an impulse in the original function. We also see that the power spectral density is repeated every 500 Hz.



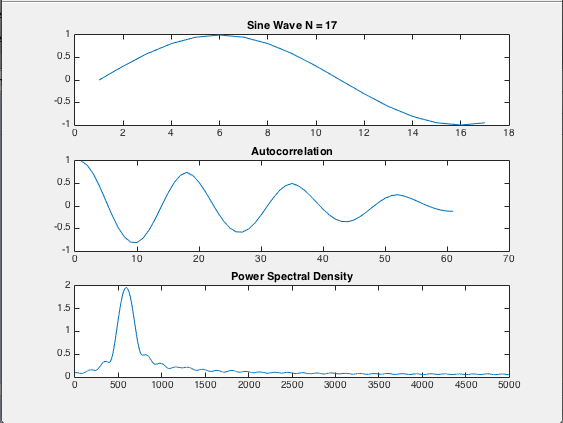
*Figure 4: Results for Sine Wave*

As we expect, the autocorrelation function of a sine ave is a cosine wave of identical frequency. Because the sine and cosine are of frequency 500 Hz, we expect an impulse in the frequency domain at 500 Hz. This is in fact what we observe.

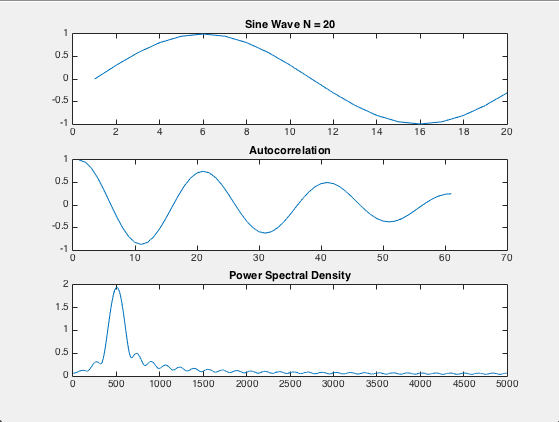
Now we look at the autocorrelation and power spectral density of sine waves that are aperiodic.



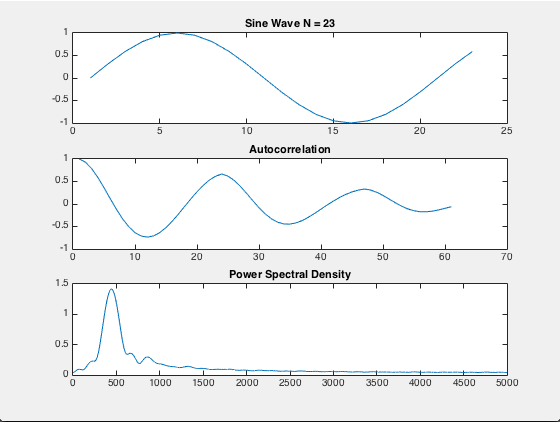
*Figure 5: Period 20 Samples, Only 14 Samples*

**

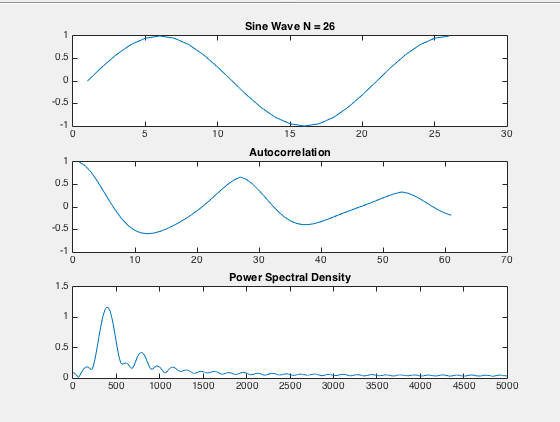
*Figure 6: 17 Samples*

**

*Figure 7: 20 Samples*



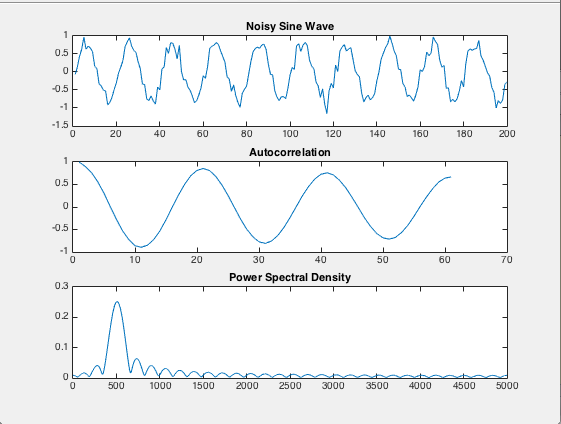
*Figure 8: 23 Samples*

**

*Figure 9: 26 Samples*

We see that when the sine wave does not complete a full period, there are sharp edges in the autocorrelation that correspond to the number of samples in the signal. We also see that the power spectral density is still an impulse around 500 Hz.

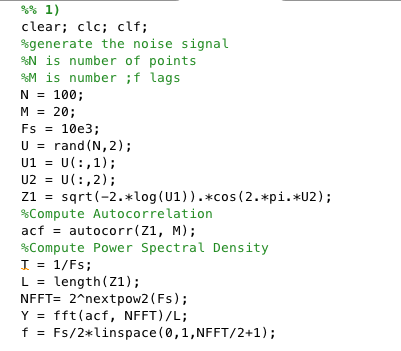
Now we will look at the autocorrelation and power spectral density of a noisy sine wave with a SNR of 10 dB.



*Figure 10:Noisy Sine Wave*

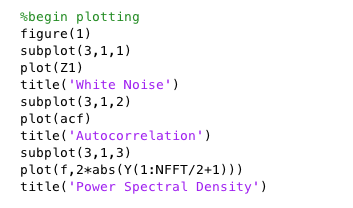
Here we see that the autocorrelation of the noisy sine wave is still a clean cosine wave, and the power spectral density is an impulse at the frequency of the sine wave. This leads us to believe that the autocorrelation function can act as a filter.

# MATLAB Code



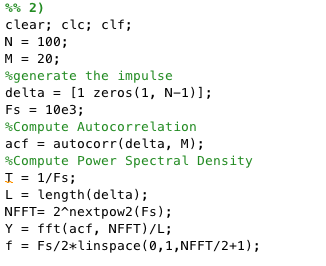
*Figure 11: Matlab Code for the White Noise Signal*

Notice that we are creating the white noise signal using the Box Muller approach. This takes two sets of normally distributed random numbers, and creates a Gaussian distributed data set. We then pass the data to the MatLab function autocorr(), and then pass the autocrrelation function to fft() to obtain the power spectral density.



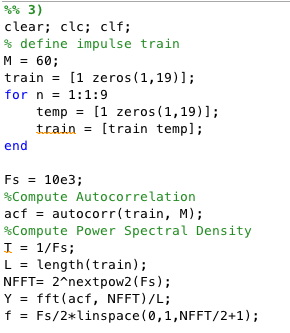
*Figure 12: Code for Plotting*

The only thing worth noting about this code is that we plot 2 times the absolute value of the output of the fft() function. This is because we only plot half of the Fourier transform, and because we are interested in the magnitude of the Fourier transform, not the complex value.



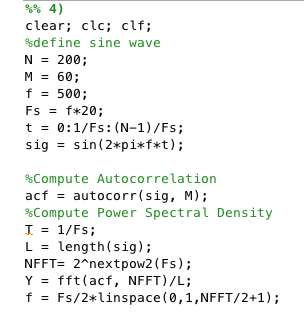
*Figure 13: Code for Impulse*

Notice that we create the impulse by creating a vector with initial value of 1, followed by a string of zeros. The rest of this code has been described above, and is plotted in the same way.



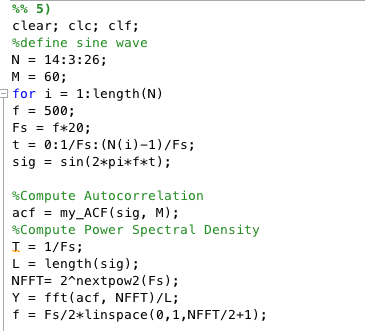
*Figure 14: Code for Impulse Train*

This time we create the signal by making a 20 sample impulse function, and concatenate an identical signal on it 9 times to give us 200 total samples. The rest of the code and plotting is identical to what was described above.



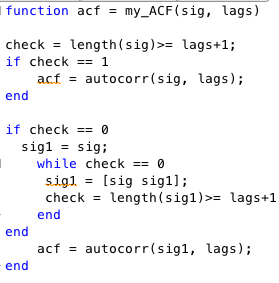
*Figure 15: Matlab Code for Sine Wave*

The sine wave is generated by selecting a desired frequency. For visual purposes 500 Hz was selected. We then define a time vector for the sine wave by denoting the total number of samples, N, and going from 0-N-1 with increments of one divided by the sampling frequency.. The rest of the code is identical to what is described above.



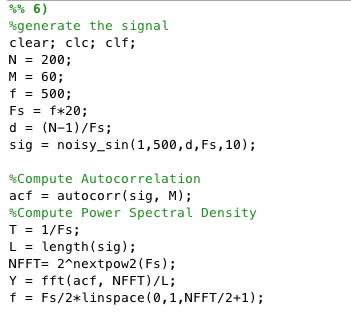
*Figure 16: Matlab Code for Part 5*

In this code, we have various N’s to generate our sine waves with. For each N however, the period is still 20 samples. The sine wave is defined in the same way as before. For this part of the assignment, a different autocorrelation function had to be implemented. The user-defined autocorrelation function is described below. It is important to note that this code is repeated for each specified value of N.



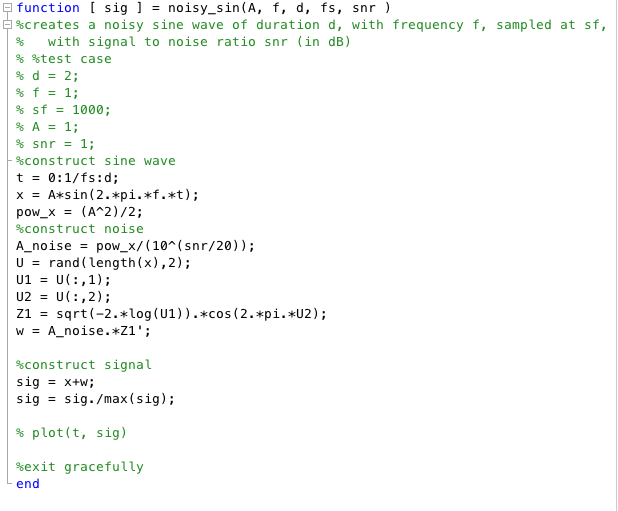
*Figure 17: my\_ACF*

The first step of this function checks to see whether the number of samples is greater than the number of lags desired plus one. If this is true, we compute the autocorrelation function as normal. If this condition fails however, we concatenate the signal to itself and re-check the condition. This process continues until the necessary condition is satisfied. Once it is, we compute the autocorrelation using Matlab’s autocorr() function.



*Figure 18: Matlab Code for Noisy Sine Wave (SNR = 10dB)*

This code is identical to what was seen previously, with the exception of generating the signal. The noisy sine wave with specified signal to noise ratio was generated using the user-defined function noisy\_sin(). This function is explained below.



*Figure 19: Function: noisy\_sin*

In this function we first define a time vector, t, based off the sampling frequency, and duration. The sample frequency and duration are both inputs to the function. Based off this we generate a clean sine wave with amplitude A. We then use the box-muller-transform to generate Gaussian noise. This signal is then multiplied by A\_noise which essentially gives it the desired energy for the proper signal to noise ratio. We then normalize the signal for plotting purposes.

# Conclusions

In this assignment we have learned that the power spectral density is the magnitude of the Fourier transform of the autocorrelation function. We also showed that the autocorrelation function acts as a filter for signals. This is an important concept because in the real world, all signals are noisy. If we try to simply analyze the noisy signal in the frequency domain, we will get misleading results because of the extra frequency content from the noise of the signal. If we take the autocorrelation of the signal first, and then find the power spectral density, we can accurately describe the true signal’s frequency content, if the signal to noise ratio is not too small.