[Haiming Xu]

ECE 3512: Signals – Continuous and Discrete

Department of Electrical and Computer Engineering, Temple University, Philadelphia, PA 19122

# Problem Statement

In this assignment we must connect the concepts of signal power and variance. Create a function that generates a signal that consists of a sinewave plus Gaussian white noise assuming the sinewave has an amplitude of +/- 1.0. What is its power? Generate a Gaussian white noise signal such that the signal to noise ratio of the summed signal is “snr\_db” measured in dB. For different range of SNRs compute and plot the autocorrelation function for different sampling. Then process the signal corresponding to a 30 dB SNR through a digital filter of the form:



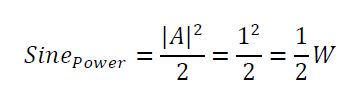
Compute the autocorrelation signal for y[n], and plot the magnitude spectrum of the Fourier Transform of the autocorrelation function. Similarly plot the square of the magnitude spectrum of the original signal, y[n]. Plot these on the same scale and compare/contrast these plots. Lastly, we explore the idea of the power spectral density function. The PSD is the Fourier transform of the autocorrelation function and tells us the power at certain frequencies of our signal.

# Approach and Results

First we create a table of the power in the entire signal given different SNR.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| SNR(dB) | -30 | -20 | -10 | 0 | 10 | 20 | 30 |
| Power(W) | 0.1 | 0.1 | 0.1 | 0.1 | 0.1 | 0.2 | 0.3 |

Our sine wave has an amplitude of ±1. For a pure sine wave the power is as follows:



To observe the autocorrelation of our signal given various SNRs for the first 16 shifts and the Fourier transform of the signal to note the frequency response.

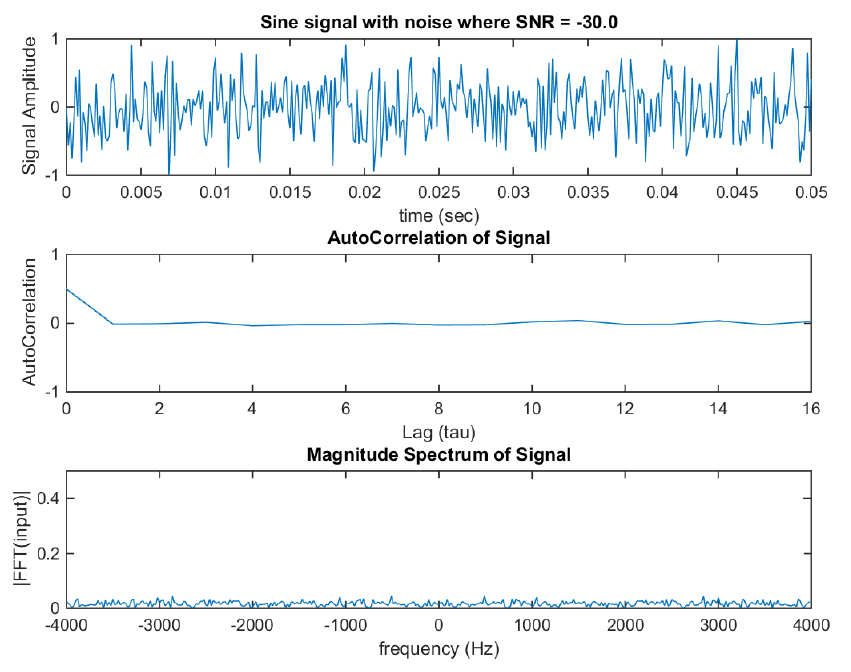


Figure 1: Sine with noise signal where SNR = -30db. ACF of Signal w/ Noise, and FFT of Signal w/ Noise

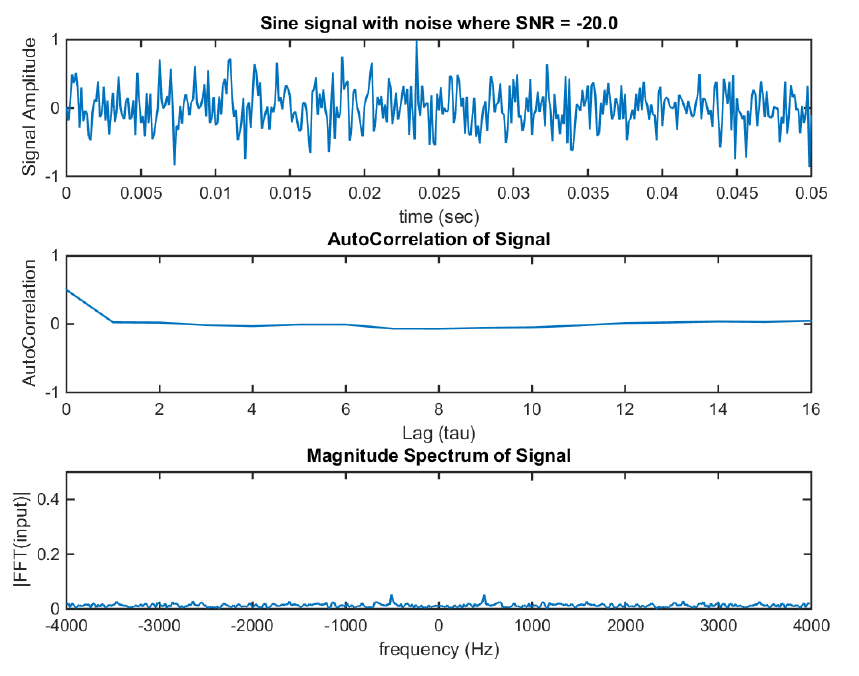


Figure 2: Sine with noise signal where SNR = -20db. ACF of Signal w/ Noise, and FFT of Signal w/ Noise

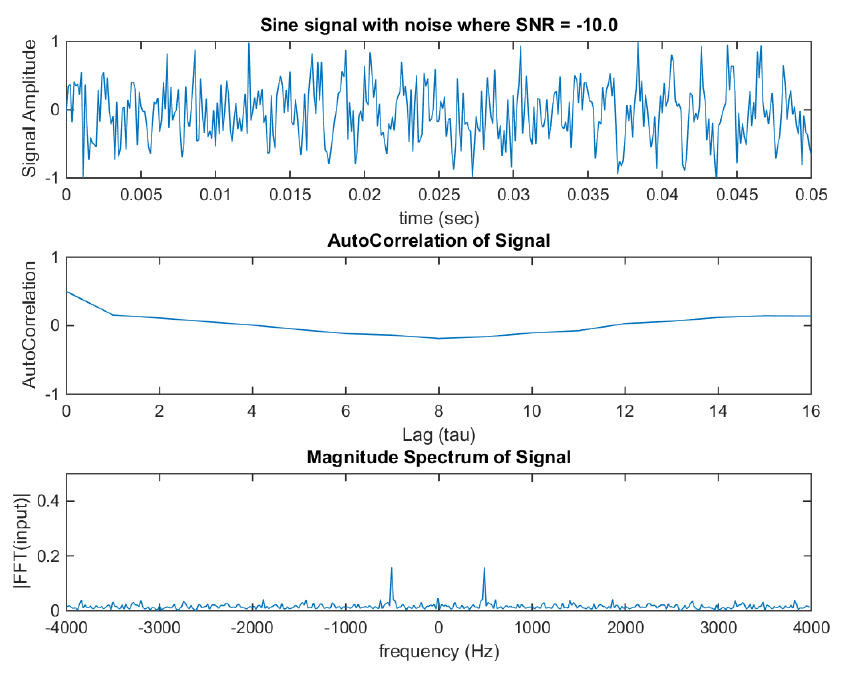


Figure 3: Sine with noise signal where SNR = -10db. ACF of Signal w/ Noise, and FFT of Signal w/ Noise

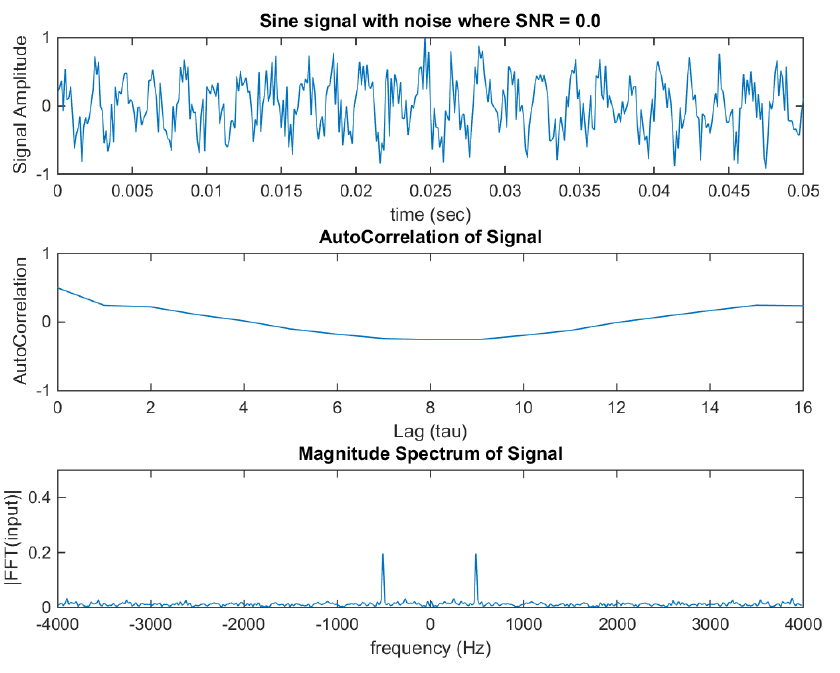


Figure 4: Sine with noise signal where SNR = 0.0db. ACF of Signal w/ Noise, and FFT of Signal w/ Noise

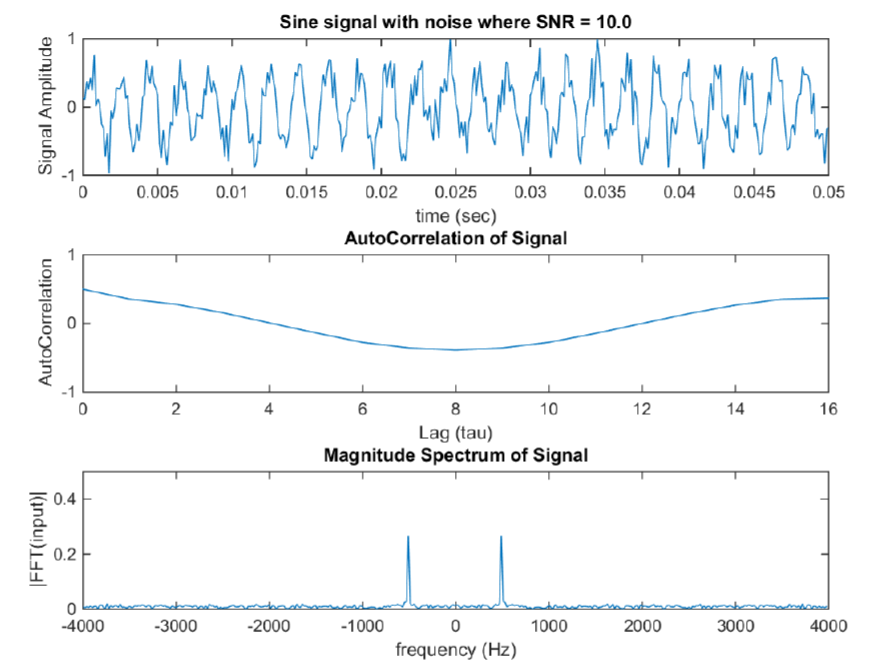


Figure 5: Sine with noise signal where SNR = 10db. ACF of Signal w/ Noise, and FFT of Signal w/ Noise

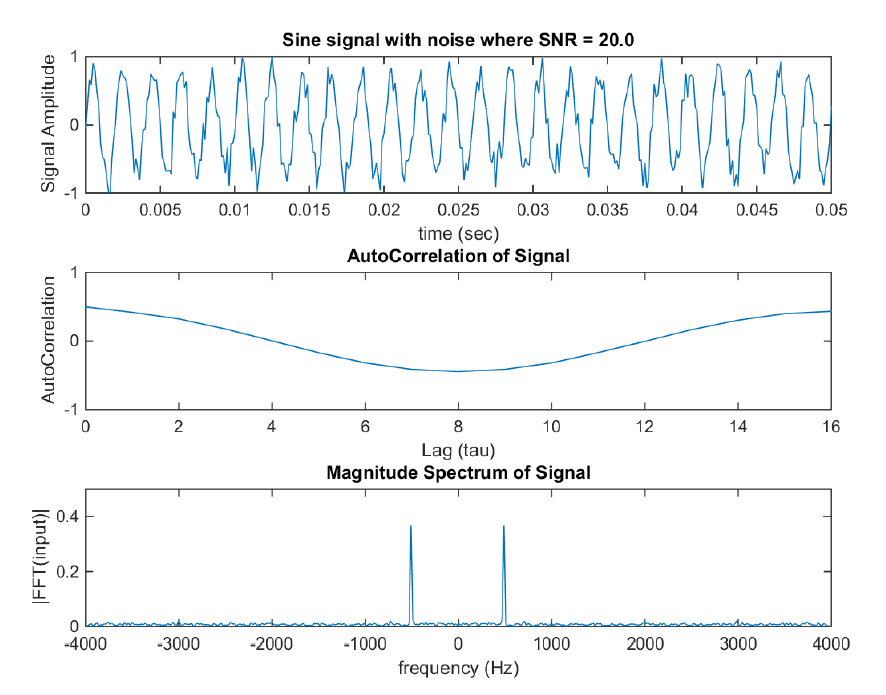


Figure 6: Sine with noise signal where SNR = 20db. ACF of Signal w/ Noise, and FFT of Signal w/ Noise

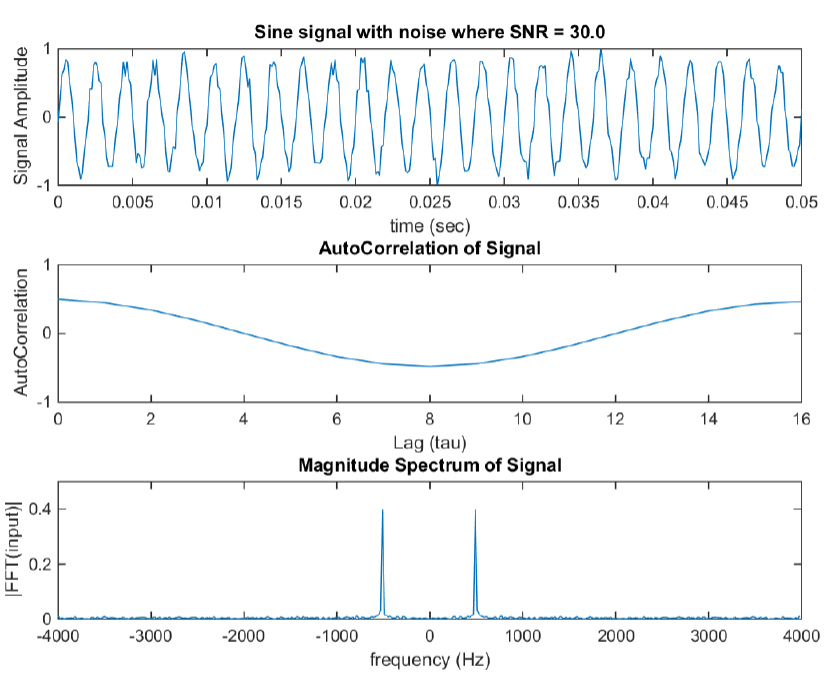
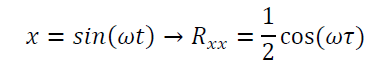


Figure 7: Sine with noise signal where SNR = 30db. ACF of Signal w/ Noise, and FFT of Signal w/ Noise

The plots above showcase the SNRs effect on the signal itself. For a pure sine wave we expect an auto-correlation of a cosine that is time invariant:



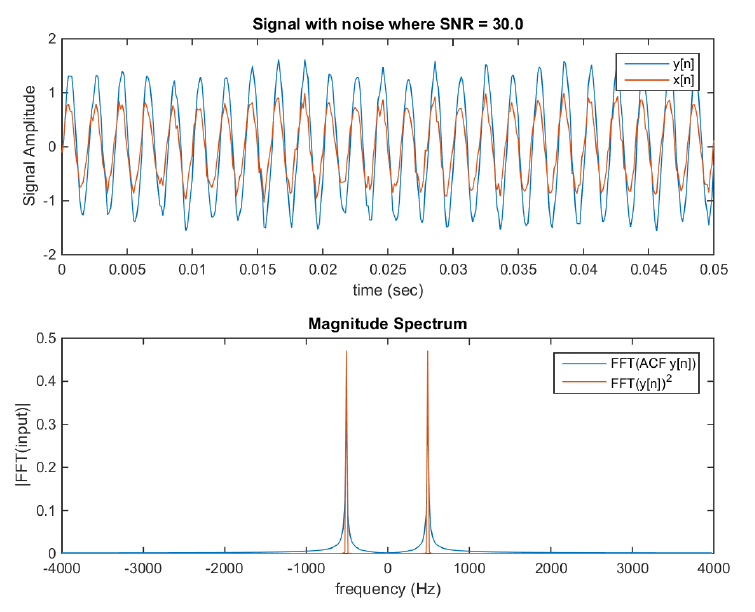
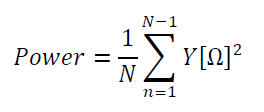


Figure 8: Signal with noise x[n] and filtered signal y[n]. PSD and Squared Fourier transform of filtered signal

Then we apply a filter that takes a signal and adds half the amplitude of the previous part in the filter. The real observation in this part of the assignment is to compare the power spectral density to the power calculated using conventional means.



# MATLAB Code

function sig = devGenerate\_SineN(Apeak, freqHz, tMax, samFreq, snrDB)

% Define time between samples

dt = 1/samFreq;

% Define time Vector

t = 0:dt:tMax;

numSam = length(t);

% Create Signal

x = Apeak\*sin(freqHz\*2\*pi\*t);

% Find power in the sine wave

powX = Apeak^2/2;

% Find amplitude of noise

%SNRDB = (Asignal/Anoise)in DB

powNoise = powX/db2mag(snrDB);

noiseMean = 0;

noiseVar = powNoise;

noiseStd = sqrt(noiseVar);

xNoise = devNormDist(noiseMean, noiseStd, numSam);

% Create Noisy Signal and Normalize

sigNoisy = x + xNoise;

sig = sigNoisy/max(sigNoisy)\*Apeak;

end

function ccf1 = devAutoCorr(Signal, lags)

[ccf,lags,bounds] = autocorr(Signal,lags);

ccf = ccf\*max(Signal)^2/2;

plot(lags, ccf);

maxA = round(max(ccf));

ylim([-maxA maxA]);

xlabel('Lag (tau)');

ylabel('AutoCorrelation');

title('AutoCorrelation of Signal');

ccf1 = ccf;

end

function FTM = devFFTMag2(Signal, fs)

samL = length(Signal);

df = fs/samL;

NFFT = 2^nextpow2(samL);

f = (-fs/2) : df : (fs/2)-df;

% Take the fourier transform. The fftshift will duplicate our FT for

% negative frequencies. (Plot is centered at zero)

FT = fft(Signal)/samL;

FTM = abs(FT);

plot(f,fftshift(FTM, 2));

xlabel('frequency (Hz)')

ylabel('|FFT(input)|');

title('Magnitude Spectrum');

end

clear; close all; clc

fs = 8E3;

f = 500;

sigAmp = 1;

timeL = .05;

tauMax = 16;

% Define time between samples

dt = 1/fs;

periodSig = 1/f;

sampleOneT = 2\*floor(periodSig\*fs);

% Define time Vector

timeVec = 0:dt:timeL;

samL = length(timeVec);

% Index Variable

i = 1;

for SNR = -30:10:30;

% Create Noisy Signal

sig = devGenerate\_SineN(sigAmp, f, timeL, fs, SNR);

figure('name','[ECE 3522] Class Assignment [9]');

subplot(3,1,1);

plot(timeVec, sig);

ylim([-sigAmp sigAmp]);

title(sprintf('Sine signal with noise where SNR = %0.1f',SNR));

xlabel('time (sec)');

ylabel('Signal Amplitude');

% Find Power of signal squared

pSig = sum(sig.^2)/samL;

fprintf(sprintf('Power computed taking SUM(sig[n]^2) = %0.001f where SNR = %0.1f\n', pSig, SNR));

% Part 2

subplot(3,1,2);

% Find AutoCorrelation for the first 16 lags

devAutoCorr(sig, tauMax);

% Part 4

subplot(3,1,3);

% Fourier Tranform

devFFTMag2(sig, fs);

title('Magnitude Spectrum of Signal');

ylim([0 0.5]);

end

% Part 5

for SNR = 30

% Generate your Signal

sig = devGenerate\_SineN(sigAmp, f, timeL, fs, SNR);

% Insatiate the filtered signal y = zeros(1,samL);

y(1) = sig(1);

% Digital Filter

for n = 2:samL

y(n) = 0.5\*y(n-1)+sig(n);

end

% Plotting Stuff

figure('name','[ECE 3522] Class Assignment [9]');

% Plot the Signal

subplot(2,1,1);

plot(timeVec, y);

hold on plot(timeVec, sig);

hold off

title(sprintf('Signal with noise where SNR = %0.1f',SNR));

xlabel('time (sec)');

ylabel('Signal Amplitude');

legend('y[n]', 'x[n]');

% Plot the FFT of AutoCorrelation Function

subplot(2,1,2);

acf = devAutoCorr(y, samL-1);

ftACF = devFFTMag2(acf\*2,fs);

% Plot the FFT of AutoCorrelation Function

subplot(2,1,2);

acf = devAutoCorr(y, samL-1);

ftACF = devFFTMag2(acf\*2,fs);

% Plot FFT of Signal

hold on df = fs/samL;

NFFT = 2^nextpow2(samL);

f = (-fs/2) : df : (fs/2)-df;

% Take the fourier transform. The fftshift will duplicate our FT for

% negative frequencies. (Plot is centered at zero)

FTY = fft(y)/samL;

FTMY = abs(FTY);

plot(f,fftshift(FTMY, 2).^2);

FTX = fft(sig)/samL;

FTMX = abs(FTX);

plot(f,fftshift(FTMX, 2).^2);

legend('FFT(ACF y[n])', 'FFT(y[n])^2', 'FFT(x[n])^2');

end

# Conclusions

In this assignment we have shown the construction of the autocorrelation of real world signals through the MatLab simulation.

Compare to the ideal cases we learn last year where there is no noise present in the signal, that we do not have to deal with filtering out your signal after it has been transmitted, in the stochastic process we produced how we can remove the noise to some extent.

Although noise is hard to eliminate since the random nature of its effects can never be predicted thus there is no one way to remove it. We can approximate what the characteristics of the original signal were, then made using techniques such as finding the power spectral density.