Vaughn Wiernicki

ECE 3522: Stochastic Processes in Signals and Systems

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

# Problem Statement

The purpose of this computer assignment was to have a function that can create a sine wave with given frequency and sampling time and add Gaussian White noise to the sine wave at various Signal to Noise ratios. We then had to observe the effect of these various ratios and illustrate them via a Fourier Transform. We then used a simple digital filter equation to filter the signal with added noise and see how filtering effected the autocorrelation and Fourier Transform.

# Approach and Results

The function was easily written by creating a sine wave with the frequency converted to Radians/second. The white noise was then generated by MATLAB’s “wng” function and summed with the sine wave to get our desired function. A for loop then cycled through and gathered the autocorrelation plots for each value of signal-noise ratio. Below are the plots illustrating the various values for signal-noise ratio. Each is titled appropriately:









Figure : Autocorrelations for Various Signal-Noise Ratios

We can see that the Autocorrelation function starts going wrong around -10 dB, slightly, or 0 dB, more noticeably so, but we don’t have any measurable impact until +10 dB. We then see that only the first value is similar and then later values plummet to around zero.

To then take the Fourier Transform was completed via MATLAB’s “fft” function and then both the -30 dB and 30 dB plot were attached. Below is a plot of the sine wave with -30 dB Signal-Noise ratio. We expect to see that the most prominent feature will be the spike at the frequency of the sinewave as the sine wave will be far more prominent than the noise added to the signal.



Figure : Fourier Transform for Signal-Noise Ratio of -30 dB

Meeting our expectations, we get one spike near the right hand value of frequency. The plot is centered at 4000 roughly and the frequency isn’t 500 Hz, but rather 3141 Hz away from the center as that incorporates the factor of 2π. We then have to retry this to see the same signal with a Signal-Noise ratio of 30 dB. This time we expect to see a Fourier Transform that looks all over the place as it should have a lot more energy at various other frequencies. Below is the plot of the Fourier Transform:



Figure : Fourier Transform for Signal-Noise Ratio of 30 dB

Again, we met expectations. We see that there are plenty of other frequencies that have similar energy levels. The noise introduced has added a lot more variation to the Fourier Transform.

We then used the equation y[n] = 0.5y[n-1] + x[n] to filter the data, at 30 dB Signal-Noise ratio again, and observe the effects that would have on both the autocorrelation and Fourier Transform. After filtering the data “autocorr” was then used. The plot for the autocorrelation is below:



Figure : Autocorrelation of the Data Filtered Sine Wave

We can see that this plot for the autocorrelation approaches zero a lot more smoothly than the unfiltered attempt did. This could be attributed to the fact that the data is accounting for the past data and the effect of that keeps building as the signal goes on. We then had to take the Fourier Transform of this vector of filtered data and the plot for that is below:



Figure : Fourier Transform of Data Filtered Sine Wave

We see that this time a lot of the lower frequencies are more attenuated, whereas higher frequencies are taking over in terms of their presence. This could mean that the filtering equation acted as a weak highpass filter and pared down some of the lower frequencies that were more prevalent before.

# MATLAB Code

clear; clc; clf; close all;

% Part 2:

f = 500;

s = 1;

fsamp = 8000;

snr\_db = [-30, -20, -10, 0, 10, 20, 30];

nlags = 16;

figs = zeros(length(snr\_db), nlags+1);

for a = 1:1:length(snr\_db)

 signal = generate\_sine(f, s, fsamp, snr\_db(a));

 figs(a, :) = autocorr(signal, nlags);

end

%Part 4:

npts = 8192;

minus30 = fft(generate\_sine(f, s, fsamp, snr\_db(1)), npts);

plus30 = fft(generate\_sine(f, s, fsamp, snr\_db(7)), npts);

magneg = abs(minus30);

magpos = abs(plus30);

%Part 5:

x = generate\_sine(f, s, fsamp, snr\_db(7));

y = zeros(1, length(x));

for a = 1:1:length(x)

 if ((a-1) == 0)

 y(a) = x(a);

 end

 if ((a-1) > 0)

 y(a) = 0.5\*y(a-1) + x(a);

 end

end

ACy = autocorr(y, nlags);

FTy = abs(fft(y, npts));

%%

%Plotting Section:

figure(1);

plot(figs(1, :));

title('Autocorrelation of -30 dB Signal-Noise Ratio:');

figure(2);

plot(figs(2, :));

title('Autocorrelation of -20 dB Signal-Noise Ratio:');

figure(3);

plot(figs(3, :));

title('Autocorrelation of -10 dB Signal-Noise Ratio:');

figure(4);

plot(figs(4, :));

title('Autocorrelation of 0 dB Signal-Noise Ratio:');

figure(5);

plot(figs(5, :));

title('Autocorrelation of 10 dB Signal-Noise Ratio:');

figure(6);

plot(figs(6, :));

title('Autocorrelation of 20 dB Signal-Noise Ratio:');

figure(7);

plot(figs(7, :));

title('Autocorrelation of 30 dB Signal-Noise Ratio:');

%%

%Plotting Section 2:

figure(1);

plot(magneg);

xlim([(length(magneg)/2), length(magneg)]);

title('Fourier Transform for Signal-Noise Ratio of -30 dB');

figure(2);

plot(magpos);

xlim([(length(magneg)/2), length(magneg)]);

title('Fourier Transform for Signal-Noise Ratio of 30 dB');

%%

%Plotting Section 3:

figure(1);

plot(ACy);

title('Autocorrelation of Data Filtered Sine with Signal-Noise Ratio of 30 dB');

figure(2);

plot(FTy);

xlim([(length(FTy)/2), length(FTy)]);

title('Fourier Transform for Data Filtered Signal-Noise Ratio of 30 dB');

function [signal] = generate\_sine(f, s, fsamp, snr\_db)

%generate\_sine

% Takes the inputs of frequency(in Hz), Time duration (in secs),

% Sampling frequency(in Hz) and Signal-to-Noise Ratio(in dB) and

% creates a signal of a sine wave with noise added to it.

%Generate Sine:

t = 0:(1/fsamp):s;

omega = 2\*pi\*f;

sine = sin(omega.\*t);

%Generate Gaussian White Noise:

noise = wgn(1, length(t), snr\_db);

%Generate Signal:

signal = zeros(1, length(t));

for i = 1:1:length(t)

 signal(i) = sine(i) + noise(i);

end

end

# Conclusions

We can see that noise can really distort a signal once the ratio goes more in favor of the noise. Autocorrelation shows that the noise can distort the signal to the point that it’s not identical and aperiodic with respect to itself. This can all be attributed to the fact that the noise generated was random. The Fourier Transform also illustrates that the noise has a lot to do with what frequencies we see more or less of. Filtering the data can help to eliminate noise, if a proper filtering algorithm is employed.