[Tyler Olivieri]

ECE 3512: Stochastic Processing in Signals and Systems

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

# Problem Statement

Every signal in the real world will inherently have some noise. Understanding how to process signals with noise is an important step to understand how to signal process in the real world. The signal to noise ratio is the ratio of signal power to noise power. In this computer assignment, 500 hz sine waves are generated using various signal to noise ratios. The autocorrelation function, power spectral density, squared magnitude spectrum, and the squared magnitude spectrum will be computed and compared.

# Approach and Results

In Figure 1, the autocorrelation can be seen for the snr\_db ranging from -30:10:30. The autocorrelation was taken of . We expect the autocorrelation function of the sine to be a cosine. With a high signal to noise ratio, there is more signal power present and therefore the output of

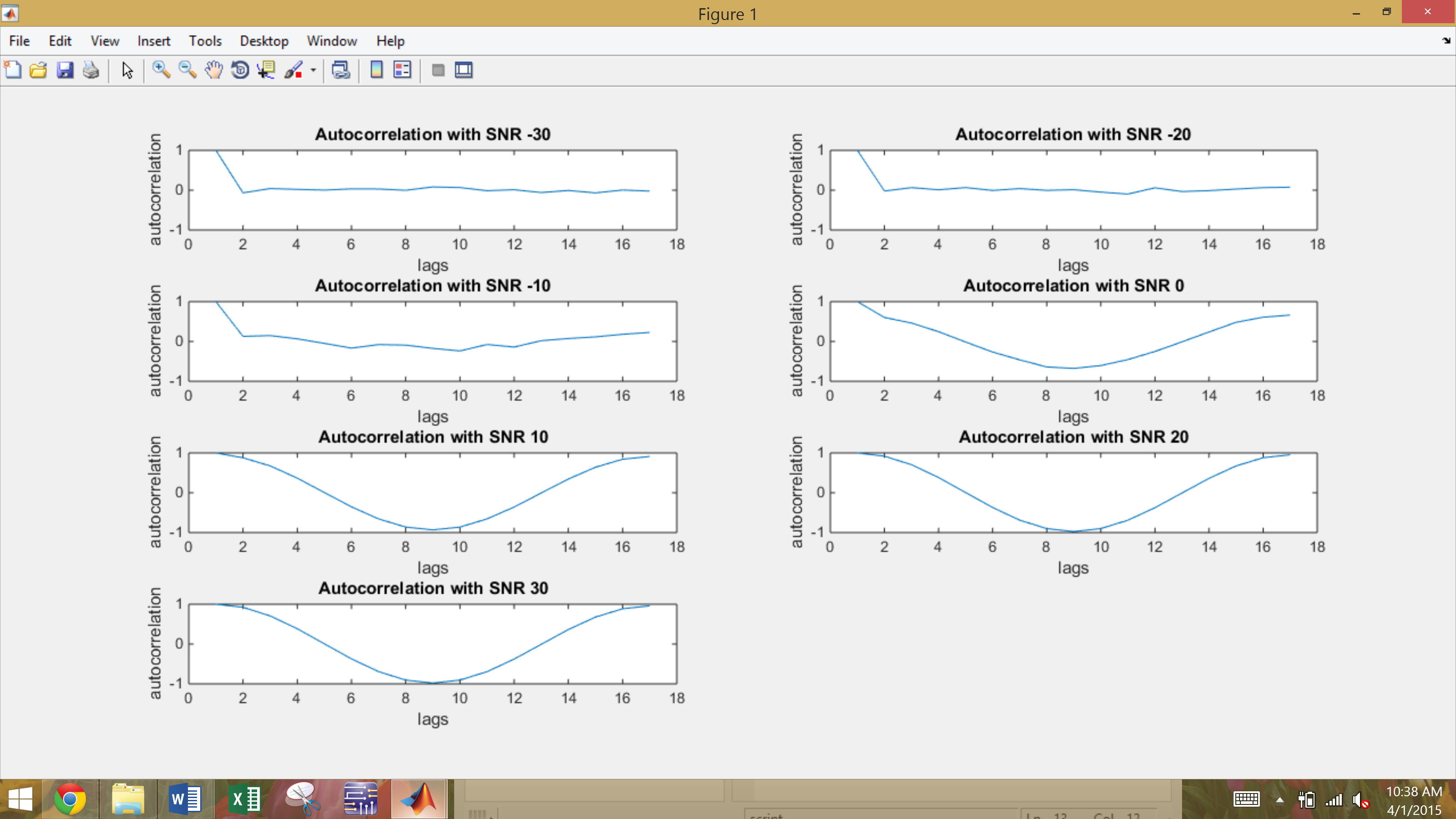


Figure 1: Autocorrleation with snr\_db ranging -30:10:30

In Figure 2, the FFT can be seen of the noisy sine wave signal (the same signal is used in the whole compute assignment). The results are as expected again. The Fourier transform shows the most energy at 500hz, the fundamental frequency of the sine wave. As the signal to noise ratio decreases and into the negatives the spectrum becomes more uniform and the sine wave is not as prominent. This can be deduced with the lack of energy at 500Hz.

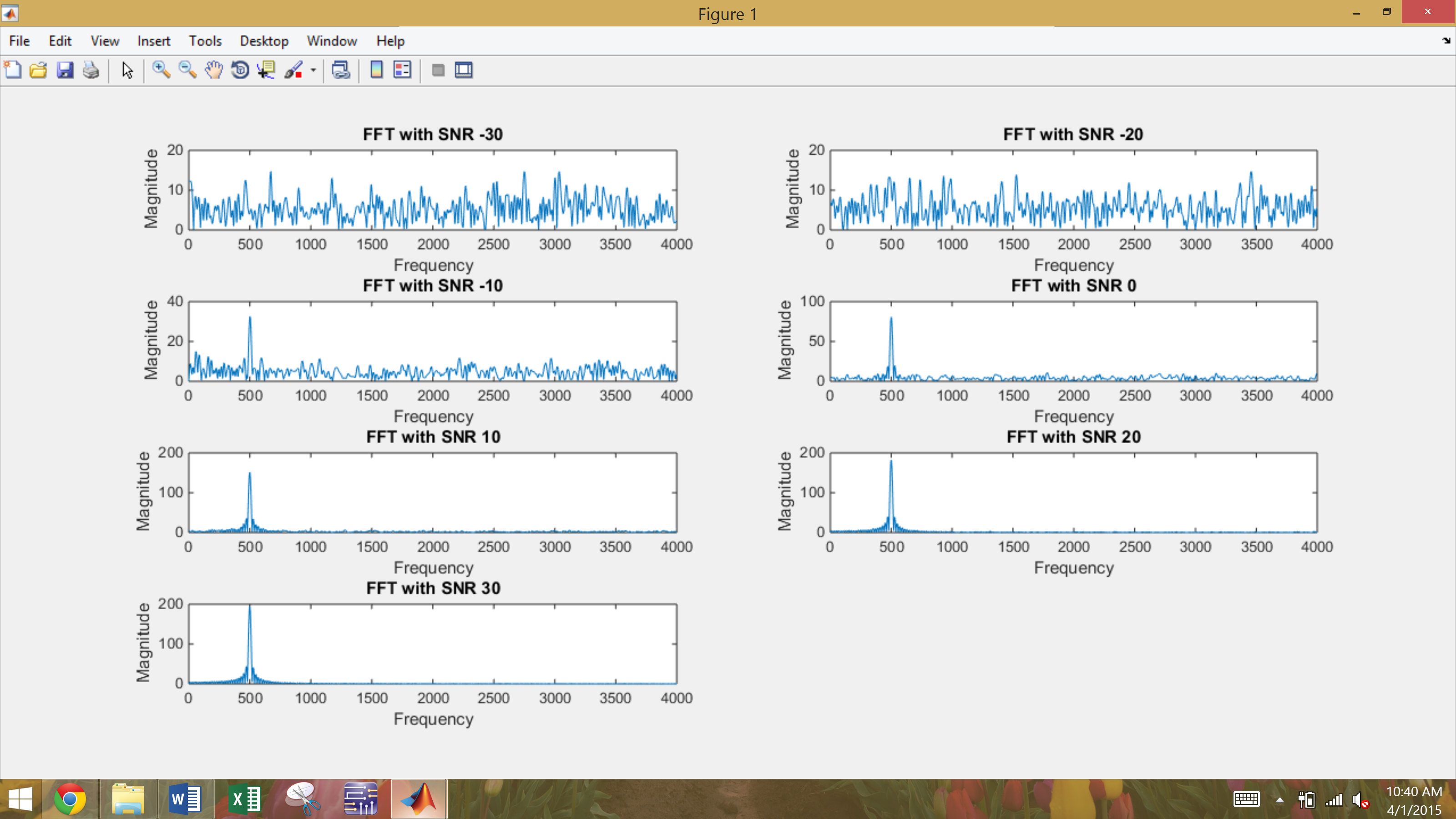


Figure 2: FFT with NFFT = 8192, fs = 8000, sin(1000\*pi\*t) snr\_db ranging -30:10:30

In Figure 3, the power spectral density can be seen compared to the squared magnitude spectrum. We can observe that they are not exactly the same but similar. With a snr\_db of 30, the sine wave is more present in the signal. Therefore, the power spectral density should be concentrated around the fundamental frequency of the sine wave.

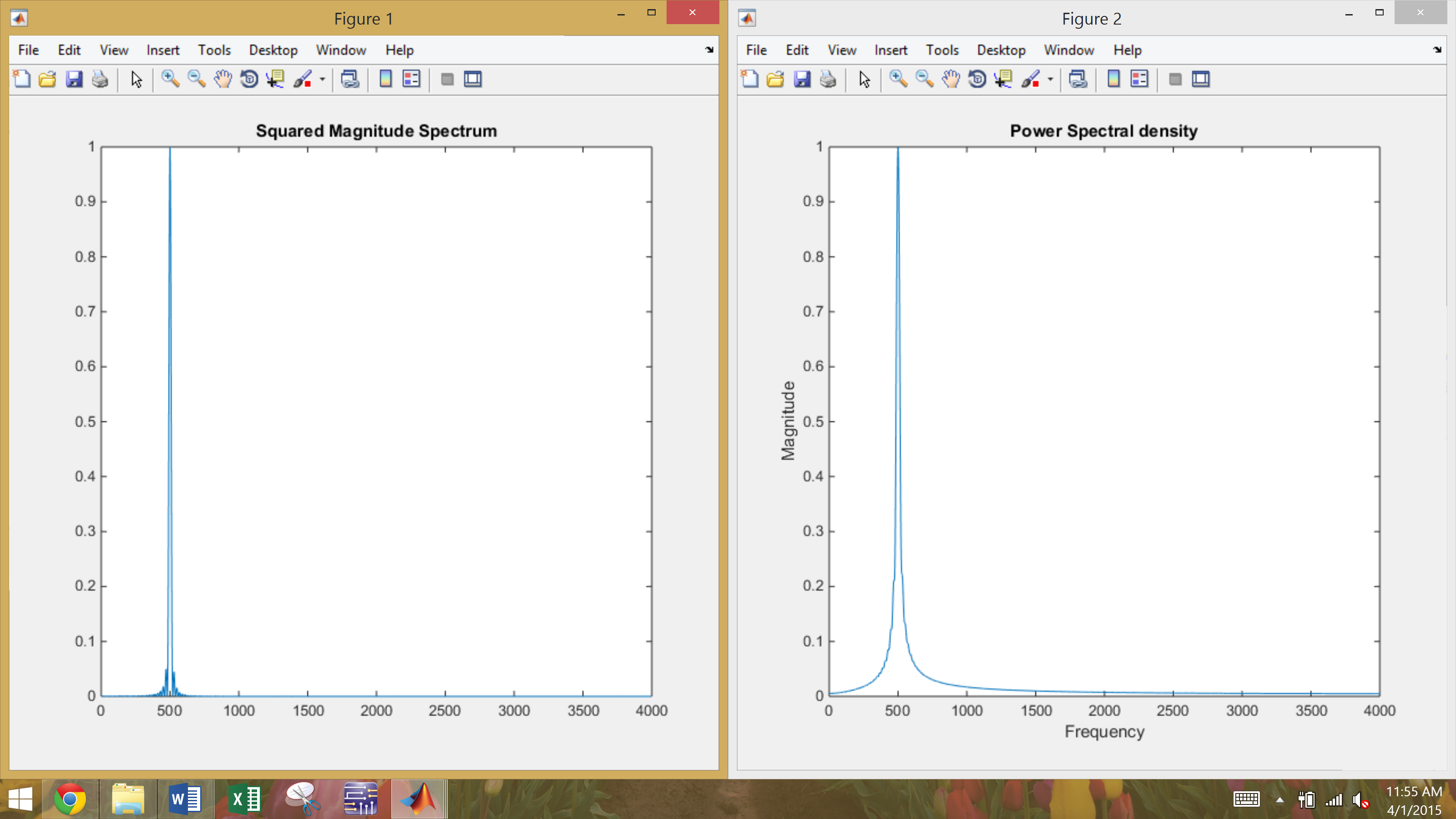


Figure 3: Power spectral density vs squared magnitude spectrum

# MATLAB Code

The function generate\_sine generates a noisy sine wave with a given frequency, sampling frequency, length of time, and signal to noise ratio in decibels. First, the power of the sine wave is computed, which is . The power of the noise is computed by dividing the power of the sine wave by the snr magnitude. The power of the noise is added to the power of the sine to obtain the output wave. It is then normalized by the max value. The function Tyautocorr computes and plots the autocorrelation of an input signal given the amount of lags. TyFFT computes and plots the FFT of an input signal given NFFT and fs.

%function generate\_sine  
% generates sine wave with gaussian white noise  
% with inputs:  
%f- frequency in hertz  
%t - duration in seconds  
%fs- sample frequency  
%snr\_db - signal to noise ratio in db  
%test case  
% clc;clear;  
% f = 1;  
% t = 1;  
% fs = 8000;  
% snr\_db = -30;  
%  
function u = generate\_sine(f,t,fs,snr\_db)  
  
  
%generate sine wave  
sample\_period = 1/fs;  
t\_vector = [0:sample\_period:t];  
sine = sin(2\*pi\*f\*t\_vector);  
  
power\_sine = 1/2; %---power of sine.o.. A^2/2  
% if(snr\_db < 0)  
% power\_noise = (power\_sine)\*mag2db(snr\_db);  
% else  
power\_noise = (power\_sine)/db2mag(snr\_db);  
  
%generate gaussian white noise  
u1 = rand(length(sine),1);  
u2 = rand(length(sine),1);  
y = sqrt(-2\*log(u1)).\*cos(2\*pi\*u2);  
  
%---- generate noise with snr  
%mean\_noise = 0;  
noise = power\_noise\*y; % + mean\_noise  
  
%generate sine plus gaussian noise  
u\_unormalized = sine' + noise;  
%normalize  
u = u\_unormalized/max(u\_unormalized);

%x- input  
% - lags  
function acf = Tyautocorr(x,lags)  
  
acf = autocorr(x,lags);  
plot(acf)  
title('Autocorrelation of Signal')  
xlabel('lags')  
ylabel('autocorrelation')

%x----input function  
%NFFT ----- nfft  
% Fs----fs  
function F = TyFFT(x,NFFT,Fs)  
  
 Y = fft(x, NFFT);  
 F = Fs/2\*linspace(0,1,NFFT/2+1);  
 NF = abs(Y(1:NFFT/2+1))/max(abs(Y(1:NFFT/2+1)));  
 plot(F,NF);  
 title('FFT')  
 xlabel('Frequency')  
 ylabel('Magnitude')

Below, the main script of the computer assignment takes all given variables and uses the above functions to plot and compute various things that is needed.

%Tyler Olivieri  
clc;clear;  
  
f = 500;  
fs = 8000;  
t = .05;  
lags = 16;  
NFFT = 8192;  
index = 0;  
figure(1);  
for snr = -30:10:30  
 index = index + 1;  
 y = generate\_sine(f,t,fs,snr);  
 subplot(4,2,index)  
 acf = Tyautocorr(y,lags);  
 string = sprintf('FFT with SNR %d',snr);  
 fft =TyFFT(y,NFFT,fs);  
 title(string);  
end

clc;clear;  
f = 500;  
fs = 8000;  
t = .05;  
NFFT = 8192;  
snr = 30;  
x = generate\_sine(f,t,fs,snr);  
y(1) = x(1);  
for n = 2:length(x)  
 y(n) = .5\*y(n-1) + x(n);  
end  
yFFT = fft(x, NFFT);  
%squared mag spectrum  
yFFT = yFFT.^2;  
F = fs/2\*linspace(0,1,NFFT/2+1);  
%Normalize  
NF =abs(yFFT(1:NFFT/2+1))/(max(abs(yFFT(1:NFFT/2+1))));  
figure(1);  
plot(F,NF);  
title('Squared Magnitude Spectrum');  
  
lags = length(y)-1;  
Ryy = autocorr(y,lags);  
%Normalize  
Ryy = Ryy/max(Ryy);  
figure(2);  
Sw = TyFFT(Ryy,NFFT,fs);  
title('Power Spectral density')

# Conclusions

In conclusion, the signal to noise ratio defines the power of the signal to the power of the noise in the signal. A lower signal to noise ratio shows a very noisy signal. In Figure 1, the autocorrelation of the sine wave at various snr can be seen. The autocorrelation of a sine is a cosine and an almost perfect cosine can be seen with a high signal to noise ratio. The noise started to effect the sine wave at 0db snr. The FFT of the noisy sine wave was taken at different snr which can be seen in Figure 3. The higher the snr, the more sine wave is present, therefore the fft will be concentrated around the fundamental frequency with low energy at all other frequencies. Flatter but moderate energy will be observed with a low signal to noise ratio. The power spectral density could be seen to be very similar to the squared magnitude of the FFT. This makes sense because the power spectral density is a view of how the power is spread throughout all frequencies. Meanwhile, when squaring the FFT you are essentially finding the power in the frequency domain. The filter has compounding energy, as you are adding the energy of the sample previous sample to the current sample. I would expect the power spectral density through the filter to have more energy.