SIOC 221A Homework 5 **Julia Dohner**

Prof. Sarah Gille Due Date: November 2, 2017

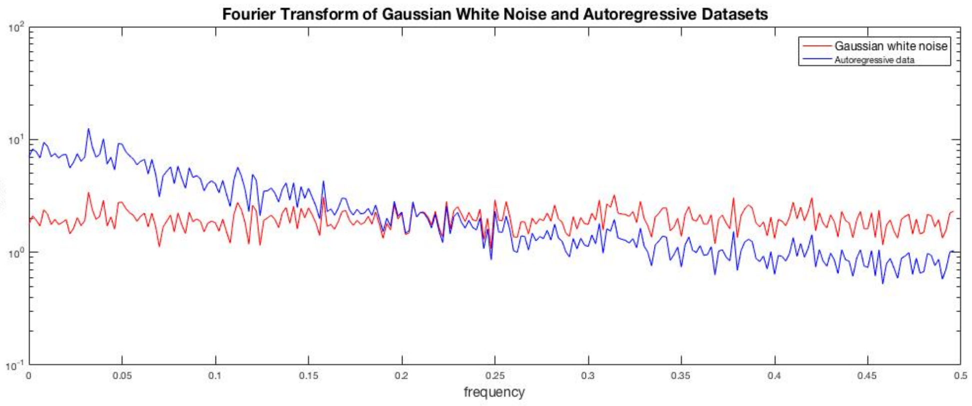
3. Use Monte Carlo simulation to verify the χ^2 error bar.

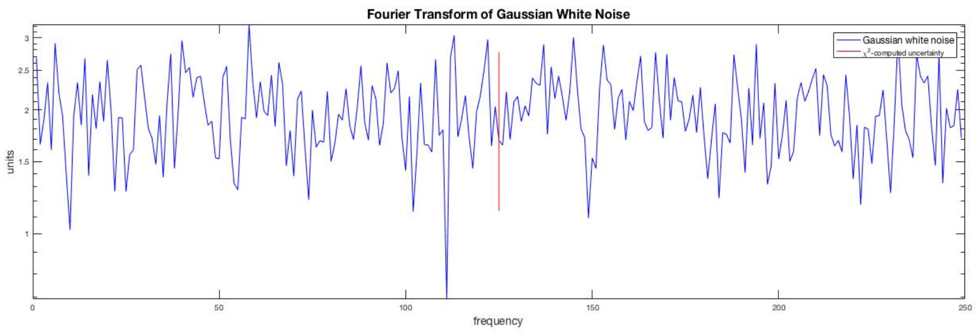
The PDF of the spectra of the data generated for the Monte Carlo simulation seems to have a Gaussian distribution. I believe that the PDF is consistent with my expectations for a variable with a χ^2 distribution with high degrees of freedom.

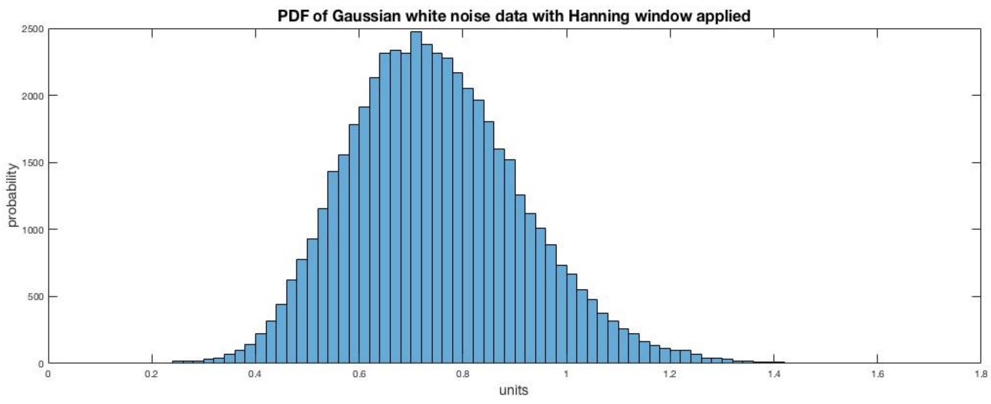
After sorting my PDF values, the ratio of the upper limit to the lower limit of the Gaussian Monte Carlo-simulated data is 1.8628. I don't know how "consistent" is defined here, but the ratio of the upper to lower limits from question 1 is 2.4287, so they're roughly in the same range but are not identical. Using more data and more segments for my initial analysis might make these ratios converge. Note: I recognize the problematic y-axis units on my PDF plot; this is the result of not having properly normalized my data when computing the PDF. I used the plot simply for visualization purposes to see whether the distribution appeared to be Gaussian.

4. Evaluate whether windowing alters degrees of freedom.

Applying a Hanning window to the Gaussian Monte Carlo-generated data results in a ratio of the upper to lower limits of 1.8792, which is close to the result of the data without having applied a Hanning window. I'd take this outcome to mean that applying a Hanning window does not alter the estimated uncertainties because [in this particular case, where there is no overlap of windows,] it does not change the number of degrees of freedom of the data, which the the sole determinant of the uncertainty.






PDF of Gaussian white noise data with Hanning window applied
probability
units
0
500
1000
1500
2000
2500
0
0.2
0.4
0.6
0.8
1
1.2
1.4
1.6
1.8

```
% file SIOC 221A HW 5
%
% author Julia Dohner
%
% due date November 2, 2017

clear all; close all;

%% Compute two spectra - attempt via Lecture 7 Notes

% compute two spectra for white noise and autoregressive datasets by
% breaking the data up into segments

N = 10000; % length of each chunk of data
M = 20; % number of segments splitting data into
p = N/M; % datapoints per segment

% generating 10,000 element dataset with Gaussian white noise
a=randn(N,1);
b(1) = a(1);

% generating a second dataset using autoregressive process
for i=2:length(a)
    b(i)=.5*b(i-1)+a(i);
end

a_reshape = reshape(a,N/M,M);
b_reshape = reshape(b,N/M,M);
a_f = fft(a_reshape);
b_f = fft(b_reshape);

a_amp = abs(a_f(1:p/2+1,:)).^2; % aplitude of a = abs(f_t(from 1 to 500/2+1, all columns)^2
a_amp(2:p/2,:) = 2*a_amp(2:p/2,:)/p; % normalizing values in amp_a (except for 0)
a_amp = a_amp(2:251,:); % dumping the mean
a_amp = a_amp(1:249,:); % dumping last value
b_amp=abs(b_f(1:p/2+1,:)).^2;
b_amp(2:p/2,:)=2*b_amp(2:p/2,:)/p;
b_amp = b_amp(2:251,:); % dumping the mean
b_amp = b_amp(1:249,:); % dumping last value

frequency=(0:p/2-2)/p; % for N even
a_amp_mean = mean(a_amp,2);
b_amp_mean = mean(b_amp,2);

figure
semilogy(frequency,a_amp_mean, '-r', frequency, b_amp_mean, '-b')
xlabel('\fontsize{14}frequency')
ylabel('\fontsize{14}units')
title('\fontsize{16}Fourier Transform of Gaussian White Noise and Autoregressive Datasets')
legend('\fontsize{12}Gaussian white noise','Autoregressive data');


%% add error bars to your spectra

nu = 2*floor(N/p);
err_high = nu/chi2inv(0.05/2,nu);
err_low = nu/chi2inv(1-0.05/2,nu);
ratio_chi2 = err_high/err_low;
```

```
figure(1)
semilogy(1:p/2-1, a_amp_mean, '-b', [p/4 p/4],[err_low err_high]*a_amp_mean(p/4), '-r')
xlabel('\fontsize{14}frequency')
ylabel('\fontsize{14}units')
title('\fontsize{16}Fourier Transform of Gaussian White Noise')
legend('\fontsize{12}Gaussian white noise','\chi^{2}-computed uncertainty');

figure(2)
semilogy(1:p/2-1, b_amp_mean, '-r', [p/4 p/4],[err_low err_high]*b_amp_mean(p/4), '-b')
xlabel('\fontsize{14}frequency')
ylabel('\fontsize{14}units')
title('\fontsize{16}Fourier Transform of Autoregressive Data')
legend('\fontsize{12}autoregressive data','\chi^{2}-computed uncertainty');

%% monte carlo simulation - Gaussian

% bigMatrix is a matrix (500x4,000) containing 200 500x20 matrices
bigMatrix_0 = randn(2000000,1);
bigMatrix = reshape(bigMatrix_0,500,4000);
%bigMatrix = randn(500,4000);
bigMatrix = fft(bigMatrix); % compute fft of each matrix

bigMatrix_amp=(abs(bigMatrix(1:p/2+1,:)).^2)/p; %needed to move the /p to this line from line↙
below
bigMatrix_amp(2:p/2,:)=2*bigMatrix_amp(2:p/2,:);
bigMatrix_amp = bigMatrix_amp(2:251,:); % dumping the mean


for i=1:20:4000
    bigMatrix_mean_amp(:,floor(i/20)+1) = mean(bigMatrix_amp(:,i:i+19),2);
end

%turn mean amplitude result into a column vector
bigMatrix_pdf = bigMatrix_mean_amp(:);
bigMatrix_pdf = sort(bigMatrix_pdf,'descend');
err_low_data = prctile(bigMatrix_pdf,0.025);
err_high_data = prctile(bigMatrix_pdf,0.975);
ratio_monteCarlo_Gaussian = err_high_data/err_low_data;

bigMatrix_pdf_hist = histogram(bigMatrix_pdf); %histogram just a way of visualizing, just need↙
info from raw values (have it once sorted)


%% repeat above steps for Gaussian white noise, but using a Hanning window

% Hanning window treatment of bigMatrix in Question #4
bigMatrix_4 = randn(500,4000);
dataHan = fft(detrend(bigMatrix_4).*(hann(500)*ones(1,4000)));

dataHan_amp=(abs(dataHan(1:p/2+1,:)).^2)/p;
dataHan_amp(2:p/2,:)=2*dataHan_amp(2:p/2,:);
dataHan_amp = dataHan_amp(2:251,:); % dumping the mean

for i=1:20:4000 %4000-19
    dataHan_mean_amp(:,floor(i/20)+1) = mean(dataHan_amp(:,i:i+19),2);
end

%turn mean amplitude result into a column vector
dataHan_pdf = dataHan_mean_amp(:);
dataHan_pdf = sort(dataHan_pdf,'descend');
err_low_data = prctile(dataHan_pdf,0.025);
err_high_data = prctile(dataHan_pdf,0.975);
ratio_Hann = err_high_data/err_low_data;

figure(3)
dataHan_pdf_hist = histogram(dataHan_pdf);
xlabel('\fontsize{14}units')
ylabel('\fontsize{14}probability')
title('\fontsize{16}PDF of Gaussian white noise data with Hanning window applied')
```