Matthew LeDuc
ECEN 5244
Fall 2023
HW 2

1 Question 1

(a) We know that

$$\hat{\sigma}_N^2 = \frac{1}{N-1}\sum_{i=1}^{N} y_i^2 \tag{1.1}$$

and additionally that the χ^2 estimator given by

$$\chi^2 = \sum_{i=1}^{N} \frac{y_i^2}{\sigma^2}$$

has mean $N-1$ and variance $2N-2$. Now we wish to evaluate $\mathbb{E}\left[\left(\hat{\sigma}_N^2 - \sigma^2\right)^2\right]$. Using the expression in Equation 1.1, we can calculate that

$$\mathbb{E}\left[\left(\hat{\sigma}_N^2 - \sigma^2\right)^2\right] = \mathbb{E}\left[\left(\frac{1}{N-1}\sum_{i=1}^{N} y_i^2 - \sigma^2\right)^2\right] =$$

$$\frac{\sigma^4}{(N-1)^2}\mathbb{E}\left[\left(\chi^2 - (N-1)\right)^2\right] = \frac{\sigma^4}{(N-1)^2}\mathbb{E}\left[\left(\chi^2 - \chi^2\right)^2\right]$$

The expectation is exactly the variance of χ^2, which we know to be $2N-2$, and so therefore,

$$\mathbb{E}\left[\left(\hat{\sigma}_N^2 - \sigma^2\right)^2\right] = \frac{2\sigma^4}{(N-1)}.$$

Since the standard deviation is the square root of the variance, we find that

$$\text{STD}\left(\hat{\sigma}_N^2\right) = \sqrt{\frac{2}{N-1}}\sigma^2$$

as desired.

(b) First, we are assuming that the last 3 autocorrelation coefficients are exactly zero due to noise in the data. Now, since the ℓ^2 norm is induced by an inner product, we can write that

$$\mathbb{E}\left[\left|U_k W \vec{\delta\rho}\right|^2\right] = \mathbb{E}\left[\langle U_k W \vec{\delta\rho}, U_k W \vec{\delta\rho}\rangle\right]$$

Focusing just on the inner product, unitarity of the U_k implies that $\langle U_k x, U_k y\rangle = \langle x, y\rangle \ \forall x, y \in H$, with H the Hilbert space associated with the inner product [1]. So, we have that

$$\langle U_k W\vec{\delta\rho}, U_k W\vec{\delta\rho}\rangle = \langle W\vec{\delta\rho}, W\vec{\delta\rho}\rangle$$

W is a diagonal matrix with real values along the diagonal, thus it is self-adjoint, so we can equivalently write that $\langle W\vec{\delta\rho}, W\vec{\delta\rho}\rangle = \langle \vec{\delta\rho}, W^*W\vec{\delta\rho}\rangle = \langle \vec{\delta\rho}, W^2\vec{\delta\rho}\rangle$. Denoting the diagonal elements of W as w_n, we can expand the inner product to find that, choosing to index the sum so that the change in the 0-lag autocorrelation is denoted $\delta\rho_0$,

$$\langle \vec{\delta\rho}, W^2\vec{\delta\rho}\rangle = \sum_{n=4-N}^{N-4} w_n^2 \left(\delta\rho_n\right)^2$$

Now, by linearity of the sum and of expectation, we have that

$$\mathbb{E}\left[\left|U_k W\vec{\delta\rho}\right|^2\right] = \sum_{n=4-N}^{N-4} w_n^2 \mathbb{E}\left[\left(\delta\rho_n\right)^2\right]$$

By analogy with the case discussed in Lecture 8, we can calculate that

$$\mathbb{E}\left[\left(\delta\rho_n\right)^2\right] = \frac{(1-\rho_n^2)^2}{N-3-|n|}$$

and therefore that

$$\mathbb{E}\left[\left|U_k W\vec{\delta\rho}\right|^2\right] = \sum_{n=4-N}^{N-4} w_n^2 \frac{(1-\rho_n^2)^2}{N-3-|n|}$$

2 Question 2

(a) Using the transfer function identification techniques from Lecture 10, I found that the given transfer function has one zero at $s = -\frac{1}{2}$ and a pole with multiplicity 2 at $s = -20$. The real and imaginary parts of the given transfer function data and the fit are shown in Figure 2.1.

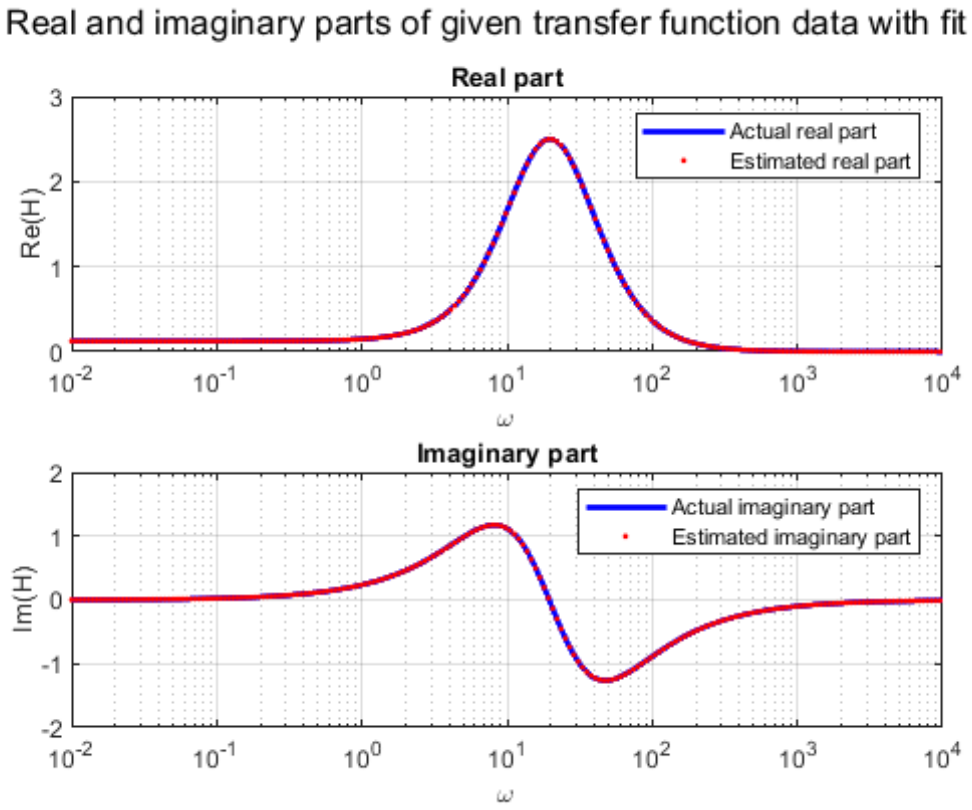

Figure 2.1: Estimated transfer function compared with data

(b) The estimate is very sensitive to noise. As shown in Figure 2.2, even noise standard deviations on the order of 10^{-5} cause significant differences in the estimates of the values of the imaginary parts of the poles. The estimates of the real part and the zero are both slightly more robust, however they are not perfect and eventually, as $\sigma \to 5 \times 10^{-4}$, the root estimate approaches -1500 and, even though the coefficients remain real, due to (presumably) numerical trouble the poles are no longer complex conjugates of each other for $\sigma \gtrapprox 3.5 \times 10^{-4}$, as shown in Figure 2.3.

(c) I would consider smoothing the data in some way to attempt to remove the effects of the noise. In addition, the predictor variables (s, s^2, etc) are strongly correlated, so we can expect issues solving the system of equations in the presence of noise. Rewriting the design matrix using an appropriate change of basis will remove this correlation and allow for a more stable solve in the presence of noise.

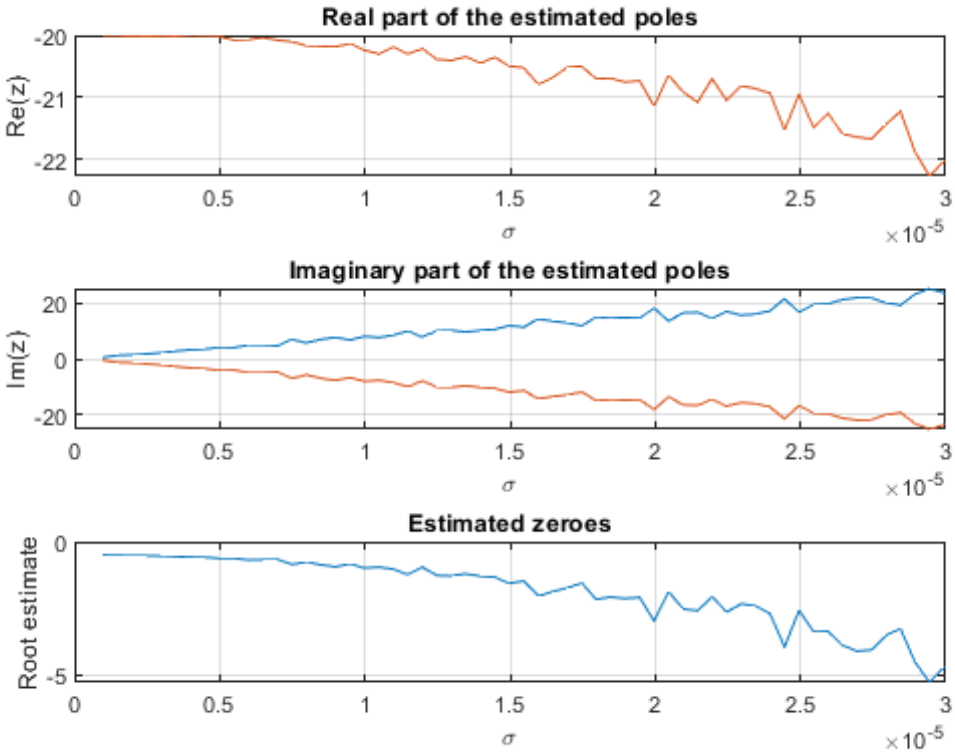

Figure 2.2: Estimated poles and zeros of the transfer function

(d) In this problem, the data spans several orders of magnitude in frequency. For this reason, I believe that it is more appropriate to determine the goodness of the fit using the correlation coefficient ρ than the χ^2 metric.

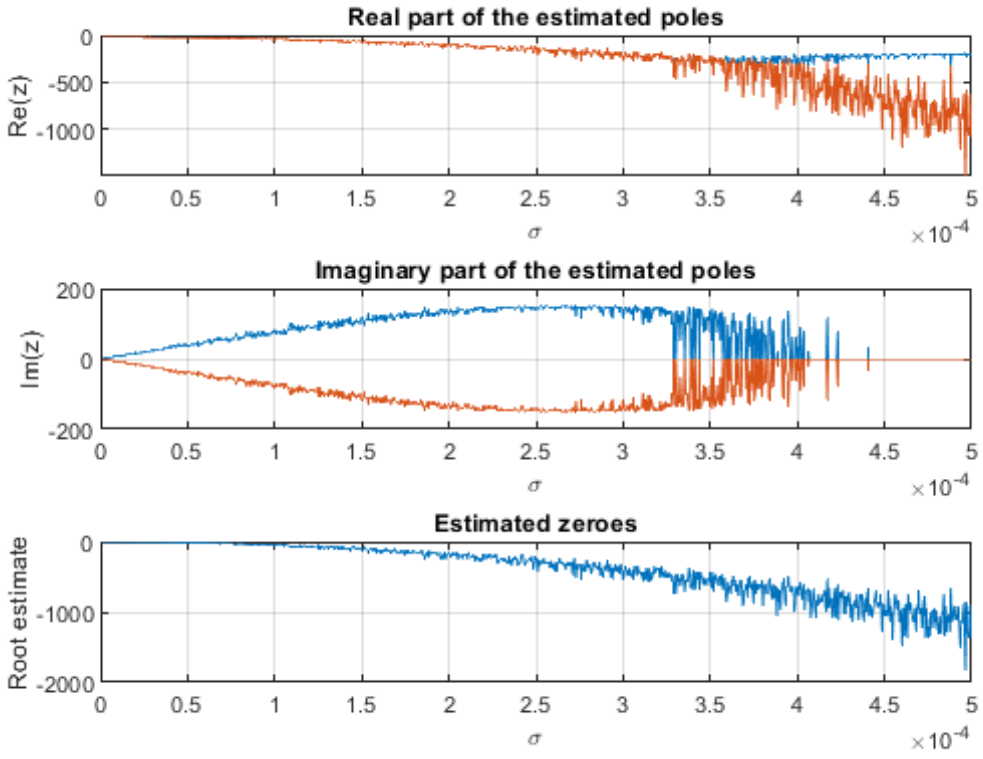

Figure 2.3: Estimated poles and zeros of the transfer function

3 Question 3

(a) The Welch Periodogram spectral estimate using M realizations of an N point FFT with window function $w[n]$ is given by

$$\hat{S}(k\Delta f) = \frac{1}{MN^2 w_{ss}} \sum_{j=1}^{M} |\hat{x}_{j,k}|^2 \tag{3.1}$$

where $w_{ss} = \sum_{n=1}^{N} w[n]^2$ and $\hat{x}_{j,k}$ is the estimate of the k^{th} Fourier Coefficient from the j^{th} FFT. The results of this spectral estimate are shown in Figure 3.1. The estimate was performed using a Tukey window with constant 0.1. As a note, MATLAB's fft normalization meant that the result needed to be divided by $\sqrt{N}$ in order to match the normalization from the notes.

The spectrum shows evidence of signals at approximately 2.7 and 17 Hz, and additionally some evidence of brown noise below 5 Hz, where the spectrum decreases by approximately 15 dB per decade.

(b) The Blackman-Tukey spectral estimate relies on the Wiener-Khinchin Theorem, which says that the autocorrelation function and the power spectral density are a Fourier

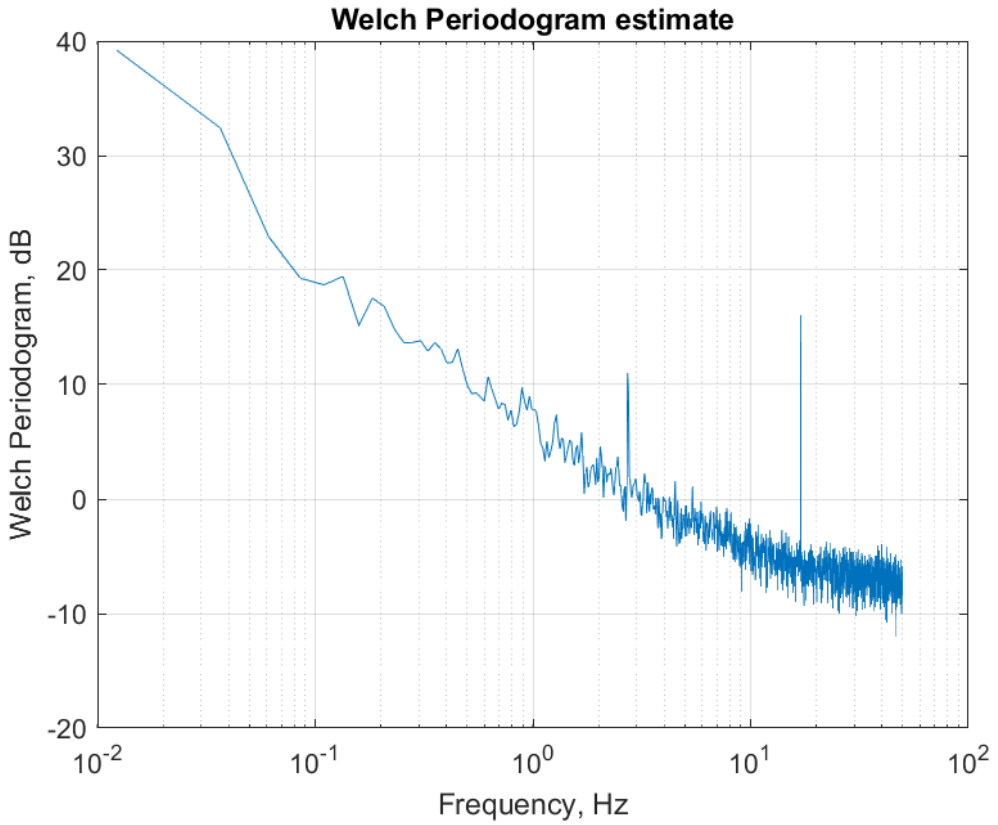

Figure 3.1: Welch Periodogram Estimate

Transform pair. The autocorrelation function can be calculated as

$$\hat{R}[n] = \mathbb{E}\left[x[k]x[k-n]\right]$$

which is implemented as the convolution of the signal and its time reversal. This implementation automatically calculates the two-sided autocorrelation, giving us twice the spectral resolution. The average autocorrelation estimate for the windowed signal is shown in Figure 3.2. The spectrum is estimated as $\hat{S}(k\Delta f) = \mathcal{F}\left[\hat{R}[n]\right]$, and shown in Figure 3.3. As a note, a spectral estimate was calculated using each estimate of the autocorrelation function and these estimates were averaged, I did not use the averaged autocorrelation function shown in Figure 3.2 to calculate this spectrum.

This estimate is similar to the Welch estimate, with signals present at approximately 2.7 and 17 Hz and similar evidence of brownish noise.

(c) The all-pole model for spectral estimation relies on the assumption that the process is an autoregression of the form

$$x_n = a_0 n_n + \sum_{k=1}^{m} a_k x_{n-k}$$

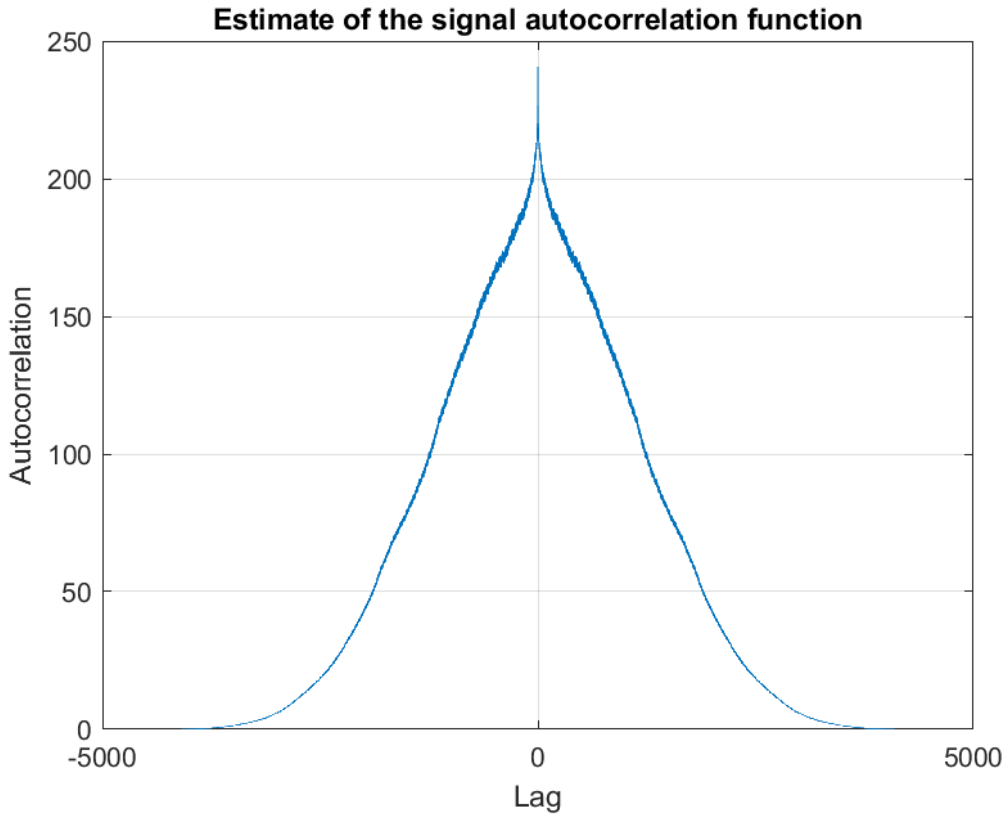

Figure 3.2: Autocorrelation Estimate

where $n_n \sim N(0,1)$ are uncorrelated. We assume a spectrum of the form

$$\hat{S}(\omega) = \frac{a_0}{\left|1 + \sum_{k=1}^{m} a_k \exp\left(2\pi i \omega k\right)\right|^2} \tag{3.2}$$

and use the values of the correlation coefficients at lags $1, 2, ..., m$ to attempt to find the a_k. Doing this with $m = 200$, the approximation of the spectrum is given in Figure 3.4. Note that the spectrum is steeper than the other models, as the denominator enforces a nearly constant 20 dB/decade rolloff, but there is still evidence of signals with frequencies 2.7 and 17 Hz. This estimate also uncovers what may be a smaller signal at 10 Hz.

(d) The desired information is shown in Table 3.1. There appear to be 3 distinct noise

Estimator	Brown noise	Pink noise	White noise	Signals	Signal levels
Welch	<1 Hz	1-10 Hz	>10 Hz	2.7 and 17 Hz	11 and 18 dB
Blackman-Tukey	<1 Hz	1-10 Hz	>10 Hz	2.7 and 17 Hz	11 and 18 dB
Autoregressive	<1 -10 Hz	None observed	>10 Hz	2.7,10, and 17 Hz	28,-8, and 20 dB

Table 3.1: Noise and signal processes for the three estimates

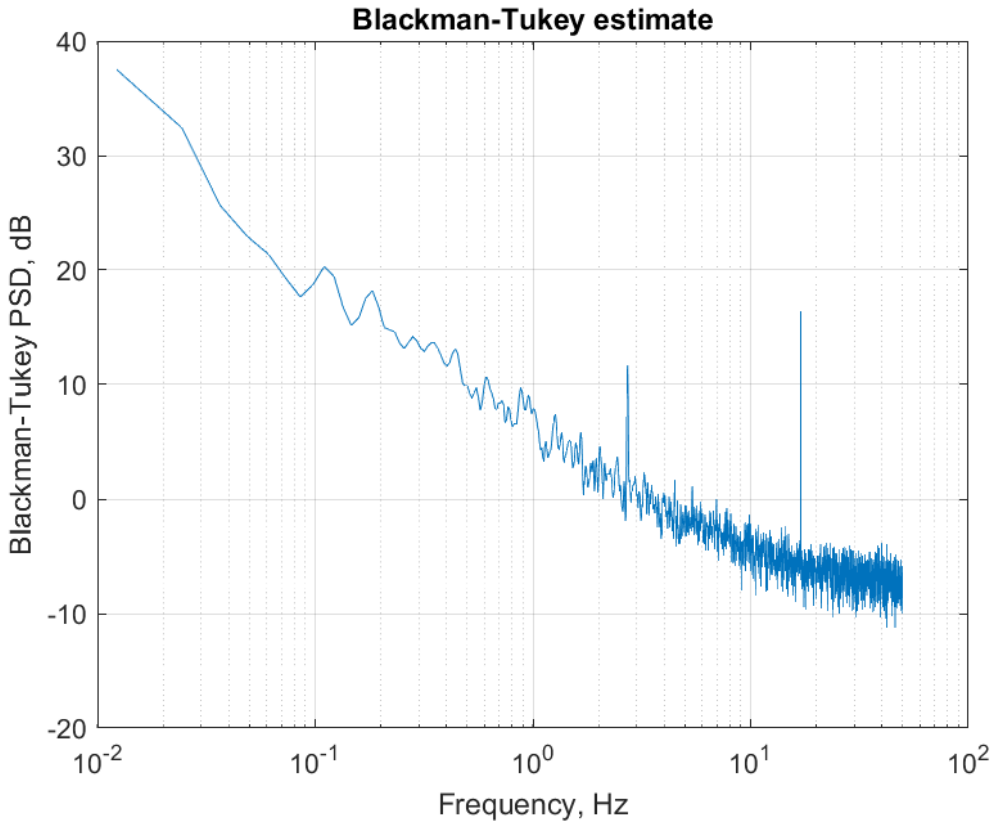

Figure 3.3: Blackman-Tukey Estimate

processes and 2, maybe 3 signals present in the data. According to the Blackman-Tukey and Welch estimators, the signals have approximate levels of 11 dB at 2.7 Hz and 18 dB at 17 Hz. However, the autoregressive estimator has the 2.7 Hz signal at about 28 dB and a bonus 10 Hz signal with a level around -8 dB. The autoregressive process also doesn't see any pink noise like the other two, instead seeing only brown noise until reaching high frequencies where the noise appears white.

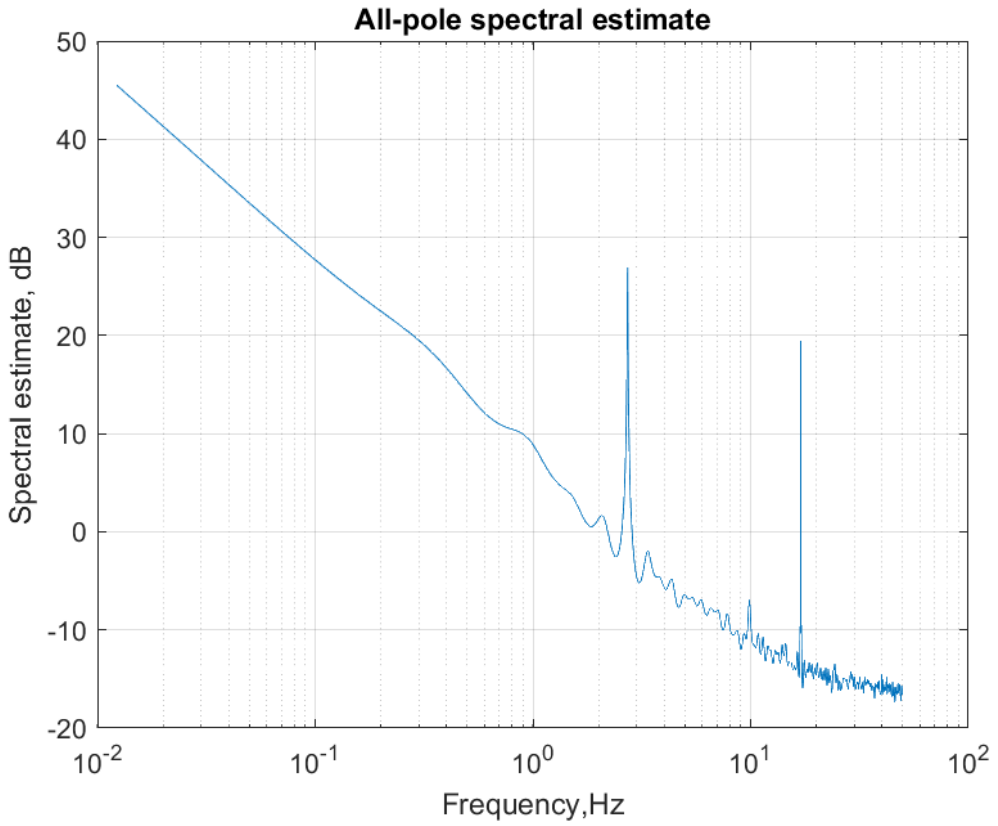

Figure 3.4: Spectral estimate using an all-pole model

4 Question 4

The optical delay is caused by changes in the path length. For a given path length difference d, the optical delay will be given by $t = d/c$ where c is the speed of light. So, we can determine what the acceptable variation in optical delay is, and then we can use that to determine the acceptable variation in the path length.

Since the sampling grid error is random, we are assuming that the signal is now random. In the presence of time dependent sampling point error $\delta(t)$ with $\mathbb{E}[\delta(t)] = 0$, the spectrum of a sinusoidal signal $x(t) = A\cos(2\pi f_0(t+\delta(t))+\phi)$ is given by $S_x(f) = \mathcal{F}[\mathbb{E}[R_x(t)]]$ where

$$\mathbb{E}[R_x(t)] = R_{x_0}(t)\mathbb{E}[\cos(2\pi f(\delta(0) - \delta(-t))]$$

and $R_{x_0}(t)$ is the autocorrelation function under zero error. If we assume that $\delta(0)$ and $\delta(-t)$ are correlated zero-mean Gaussian variables with variance σ_τ^2 and correlation coefficient $\rho = \exp\left(-\frac{t^2}{\tau^2}\right)$, then we find that, as shown in the notes,

$$S_x(f) = \exp(-(2\pi f)^2\sigma_\tau^2)S_{x_0}(f) + (1 - \exp(-(2\pi f)^2\sigma_\tau^2))\pi\tau\left[S_{x_0}(f) * \exp(-(\pi f\tau)^2)\right].$$

Since $S_{x_0}(f) = \frac{A^2}{2}(\delta(f-f_0)+\delta(f+f_0))$, the convolutions are $\exp\left(-[\pi(f \pm f_0)\tau]^2\right)$.

From all of this, we have landed at

$$S_x(f) = \frac{A^2}{4}\exp\left(-\left(2\pi f\right)^2\sigma_\tau^2\right)\left(\delta(f-f_0)+\delta(f+f_0)\right)$$
$$+\frac{A^2}{4}\left(1-\exp\left(-\left(2\pi f\right)^2\sigma_\tau^2\right)\right)\pi\tau\left[\exp\left(-\left[\pi(f-f_0)\tau\right]^2\right)+\exp\left(-\left[\pi(f+f_0)\tau\right]^2\right)\right]$$

We wish to have less than 1% error at f_0, so we wish to have $\left|\frac{S_x(f_0)}{s_{x_0}(f_0)}-1\right| < 0.01$. We can assume that $\tau < \frac{1}{f_s} < \frac{1}{2f_0}$, where f_s is the sample rate, to satisfy the Nyquist Criterion, thus $\tau < \frac{\lambda}{2c} \approx 5\times10^{-15}$. We will simplify things by saying that $\tau \approx 0$. Combining all of this, we have that

$$\left|\frac{S_x(f_0)}{s_{x_0}(f_0)}-1\right| = \left|1-2\pi\tau+(2\pi\tau-1)\exp(-(2\pi f_0)^2\sigma_\tau^2)\right| \approx 1-\exp(-(2\pi f_0)^2\sigma_\tau^2)$$

We can bound σ_τ^2 as follows:

$$1-\exp(-(2\pi f_0)^2\sigma_\tau^2) < 0.01$$
$$\exp(-(2\pi f_0)^2\sigma_\tau^2) > 0.99$$
$$\sigma_\tau^2 < \frac{-\log(0.99)}{(2\pi f_0)^2} \approx 2.5\times10^{-32}\text{s}^2 \implies \sigma_\tau < 1.6\times10^{-16}\text{s}.$$

Since the distance travelled by the light wave is equal to the speed of light multiplied by the travel time, we have that $\sigma_d < 4.7\times10^{-8}$ meters.

Now since we wish to have a frequency resolution of 1 GHz, we need to determine the necessary time delay to achieve the resolution, and then from this determine the length over which the tolerance must be maintained. When taking an N-point FFT, the frequency resolution is given by $\Delta f = \frac{1}{N\Delta t}$, so if we want $\frac{1}{N\Delta t} = 10^9$, we need to have $\Delta t = \frac{1}{N}\times10^{-9}$, so the delay channel must have a length of $\frac{0.3}{N}$ meters, which is the distance over which the tolerance must be maintained.

5 Question 5

(a) Let $x(t)$ be a white noise process and $y(t) = h(t) * x(t)$ be the filter output, then we have that

$$|\hat{y}(\omega)|^2 = |H(\omega)|^2 |\hat{x}(\omega)|^2$$

and since $|\hat{x}(\omega)|^2 = \frac{1}{2}$, we have that

$$|\hat{y}(\omega)|^2 = \frac{1}{2}\left(\frac{1}{1+(\omega\tau)^2}\right).$$

The Wiener-Khinchin Theorem tells us that

$$R_{yy}(t) = \mathcal{F}^{-1}\left[|\hat{y}(\omega)|^2\right]$$

, and so we set about evaluating the inverse Fourier Transform. This tells us that

$$R_{yy}(t) = \frac{1}{4\pi}\int_{-\infty}^{\infty} \frac{\exp(i\omega t)}{1+(\omega\tau)^2}\mathrm{d}\omega$$

. Performing the change of variables $k = \omega\tau$, the integral becomes

$$R_{yy}(t) = \frac{1}{4\pi\tau}\int_{-\infty}^{\infty} \frac{\exp(ik\frac{t}{\tau})}{1+k^2}\mathrm{d}k$$

which can be readily evaluated using Cauchy's Residue Theorem by taking an integral around the upper half-plane, where the integrand has a pole at $k = i$. Doing this, we see that when $t \geq 0$,

$$R_{yy}(t) = \frac{2\pi i}{4\pi\tau}\mathrm{Res}_{k=i}\left[\frac{\exp(ik\frac{t}{\tau})}{1+k^2}\right] = \frac{1}{4\tau}\exp\left(-\frac{t}{\tau}\right).$$

Following a similar process for the lower half-plane, where now the pole is at $k = -i$, we find that when $t < 0$,

$$R_{yy}(t) = \frac{1}{4\tau}\exp\left(\frac{t}{\tau}\right).$$

So, in total, the autocorrelation function is given by

$$R_{yy}(t) = \frac{1}{4\tau}\exp\left(-\left|\frac{t}{\tau}\right|\right).$$

(b) The transfer function of an ideal bandpass filter is given by

$$H(f) = \chi\left(\left[f_0 - \frac{1}{2}B, f_0 + \frac{1}{2}B\right]\right) + \chi\left(\left[-f_0 - \frac{1}{2}B, -f_0 + \frac{1}{2}B\right]\right) \tag{5.1}$$

where $\chi(I)$ is the indicator function on I. Since $|H(f)|^2 = H(f)$, and letting $\omega = 2\pi f$ and $\beta = 2\pi B$, we have again that

$$|\hat{y}(\omega)|^2 = |H(\omega)|^2 |\hat{x}(\omega)|^2$$

and by the Wiener-Khinchin Theorem, that

$$R_{yy}(t) = \frac{1}{4\pi}\int_{-\infty}^{\infty} \exp(i\omega t)\left(\chi\left(\left[\omega_0 - \frac{1}{2}\beta, \omega_0 + \frac{1}{2}\beta\right]\right) + \chi\left(\left[-\omega_0 - \frac{1}{2}\beta, -\omega_0 + \frac{1}{2}\beta\right]\right)\right)\mathrm{d}\omega$$

$$= \frac{1}{4\pi}\left(\int_{-\omega_0-\frac{1}{2}\beta}^{-\omega_0+\frac{1}{2}\beta} \exp(i\omega t)\mathrm{d}\omega + \int_{\omega_0-\frac{1}{2}\beta}^{\omega_0+\frac{1}{2}\beta} \exp(i\omega t)\mathrm{d}\omega\right)$$

Performing the change of variables $k = \omega + \omega_0$ in the first integral and $k = \omega - \omega_0$, the integral becomes

$$\frac{1}{4\pi}\left(\exp(-i\omega_0 t)\int_{-\frac{1}{2}\beta}^{\frac{1}{2}\beta} \exp(i\omega t)\mathrm{d}\omega + \exp(i\omega_0 t)\int_{-\frac{1}{2}\beta}^{\frac{1}{2}\beta} \exp(i\omega t)\mathrm{d}\omega\right)$$

$$= \frac{\cos(\omega_0 t)}{2\pi}\int_{-\frac{1}{2}\beta}^{\frac{1}{2}\beta} \exp(i\omega t)\mathrm{d}\omega = \frac{1}{\pi}\frac{\cos(\omega_0 t)\sin(\frac{\beta}{2}t)}{t}$$

so we have that the autocorrelation function of the noise is given by

$$R_{yy}(t) = \frac{1}{\pi}\frac{\cos(\omega_0 t)\sin(\frac{\beta}{2}t)}{t}.$$

6 Question 6

(a) If the power in the signal is $\sigma_s^2 = 1$ mW then the variance in the voltage is given by $\sigma_v^2 = 50$ mW. The discretization noise has variance $\sigma_n^2 = 0.01\sigma_v^2$, then since the discretization noise is uniformly distributed we have that

$$\sigma_n^2 = \frac{(\Delta V)^2}{12}$$

where ΔV is the voltage resolution of the converter, and so therefore

$$(\Delta V)^2 = 6\text{mW} \implies \Delta V \approx 0.0775\text{V}$$

Additionally, we have that

$$\Delta V = \frac{V_{max} - V_{min}}{2^{N_b}}$$

with $N_b = 12$, so the voltage range $V_{max} - V_{min}$ is given by $2^{12}\Delta V$. Using the value of ΔV from above, we see that the voltage range of the A/D converter is given by

$$V_{max} - V_{min} = 2^{12}\Delta V \approx 318\text{Volts.}$$

(b) In the case where a voltage signal regularly exceeds V_{max}, the observed discretization noise will no longer be uniformly distributed. Since the signal voltage is higher than V_{max}, this means that the observed disretization noise will be more commonly negative than positive and we cannot use the above expression for the variance of the discretization noise.

7 Code

7.1 Question 2

7.1.1 Run script

```
clear variables; close all;
rng(5684587)
data = load( 'hw2_2.mat' );
%% Wish to minimize error formula as given in the lecture 10 notes

H = data.d(:,2)+1j*data.d(:,3);

nZeros = 1;nPoles = 2; %Inputs: Number of desired poles and zeros

[a,b] = getTransferFn(data.d(:,1) ,H,nPoles, nZeros);
s = ( 1j*data.d(:,1 ) );%%s = i\omega!!!!!!!!!!!!!!!!!!!!!!!!!!!!
Sz = zeros(length(s), nZeros+1);
for ii = 0:nZeros
    Sz(:,ii+1) = s.^(ii);
end
Hnum = Sz*a;

Sp = zeros(length(s), 1+nPoles);
for ii = 0:nPoles
    Sp(:,ii+1) = s.^(ii);
end
Hdenom = Sp*b;

Hest = Hnum./Hdenom;

% figure;
% sgtitle('Real and imaginary parts of given transfer function data with fit')
% subplot(2,1,1)
%
% semilogx(data.d(:,1),real(H),'b','linewidth',2);
% hold on;semilogx( data.d(1:5:end,1),real(Hest(1:5:end)),'r.' )
% grid on
% legend('Actual real part','Estimated real part')
% title('Real part')
% xlabel('\omega')
% ylabel('Re(H)')
% subplot(2,1,2);
% semilogx(data.d(:,1),imag(H),'b','linewidth',2);
% hold on;semilogx( data.d(1:5:end,1),imag(Hest(1:5:end)),'r.' )
% grid on;
% legend('Actual imaginary part','Estimated imaginary part')
% title('Imaginary part')
% xlabel('\omega')
% ylabel('Im(H)')
% ylim([-2 2])
fprintf('No added noise\n');
```

```
fprintf('The transfer function has zeros at %.3f\n',roots(flip(a)));
poles = roots(flip(b));
fprintf('The transfer function has poles at %.3f and %.3f\n',poles(1),poles(2));

sigmas = (linspace(0.000000001,0.0005,1001));
poleEsts = zeros( length(sigmas),nPoles );
zeroEsts = zeros( length(sigmas),nZeros );
rhos = zeros( length(sigmas)+1,1 );
rhos(1) = abs(sum( conj(Hest).*H)) /sqrt( sum( abs(H).^2 )*sum(abs(Hest).^2));
for ii = 1:length(sigmas)
    noise = sigmas(ii)/sqrt(2)*(randn(1,length(H))+1j*randn(1,length(H))).';
    [a,b] = getTransferFn(data.d(:,1) ,H+noise,nPoles, nZeros);
    zeroEsts(ii) = roots(flip(a));
    if any(~isreal(b))
        keyboard
    end
    poleEsts(ii,:) = 1/2*[-b(2)+sqrt(b(2)^2-4*b(3)),-b(2)-sqrt(b(2)^2-4*b(3))]/b(3);
    Hfit = (Sz*a)./Sp*b;
    rhos(ii+1) = abs(sum( conj(Hfit).*H)) /sqrt( sum( abs(H).^2 )*sum(abs(Hfit).^2));
end

figure;
subplot(3,1,1)
plot(sigmas, (real((poleEsts))))
ylabel('Re(z)')
xlabel('\sigma')
title('Real part of the estimated poles')
grid on
subplot(3,1,2)
plot(sigmas, (imag((poleEsts))))
ylabel('Im(z)')
xlabel('\sigma')
title('Imaginary part of the estimated poles')
grid on
subplot(3,1,3)
plot(sigmas, zeroEsts);
grid on;
title('Estimated zeroes')
xlabel('\sigma')
ylabel('Root estimate')


figure;
subplot(3,1,1)
plot(sigmas, (real((poleEsts))))
ylabel('Re(z)')
xlabel('\sigma')
title('Real part of the estimated poles')
grid on
xlim([0,3*10^-5])
subplot(3,1,2)
plot(sigmas, (imag((poleEsts))))
ylabel('Im(z)')
xlabel('\sigma')
```

```
title('Imaginary part of the estimated poles')
grid on
xlim([0,3*10^-5])
subplot(3,1,3)
plot(sigmas, zeroEsts);
grid on;
title('Estimated zeroes')
xlabel('\sigma')
ylabel('Root estimate')
xlim([0,3*10^-5])
```

7.1.2 Transfer function estimation code

```
function [a,b] = getTransferFn( w,H,nPoles, nZeros )
%Uses input w (radial frequency), H (values of the transfer function) data to approximate t
%rational function with nPoles poles and nZeros zeros.\
%varargin: 1 if w is given in terms of radial frequency, 0 if w is given as
%values on the unit circle
%Returns a (vector of coefficients on the numerator) and b (vector of
%coefficients on the denominator)
N = length(w);
z = ( 1j*w );%Values on the unit circle
z = reshape(z,N,1);%Ensuring they are column vectors
H = reshape(H,N,1);

Az = zeros(N,nZeros+1);Ap = zeros(N,nPoles);

for ii = 0:nZeros
    Az(:,ii+1) = z.^(ii);
end
for ii = 1:nPoles
    Ap(:,ii) = H.*z.^(ii);
end
A = [Az,Ap];%The whole design matrix

A2 = [real(A);imag(A)];
H2 = [real(H);imag(H)];
x = (A2'*A2)\(A2'*H2);%MATLAB backslash to solve. Want real coefficients
a = x(1:nZeros+1);%Numerator coefficients
b = [1;-x(nZeros+2:end)];%Denominator coefficients
end
```

7.2 Question 3

7.2.1 Run script

```
%% HW 2 problem 3
close all; clear variables;
data = load( 'hw2_3.mat' );
N = 2^12;
win = hamming(N);%Hamming window
wss = sqrt(sum(win.^2));
```

```
win = repmat(win/wss, 1,2^16/N);
d = reshape(data.d(:,2), N,[]);
% d = d-mean(d);
times = reshape((data.d(:,1)),N,[]);
windowedData = win.*d;%Columns of this matrix are the windowed data
%% Welch Periodogram
dhat = fft( windowedData );
dt = mean(diff( data.d(:,1) ));df = 1/(N*dt);
sHat = abs(dhat).^2;
s = mean(sHat.');
freqs = df*(0:N-1);freqs = freqs-mean(freqs);
figure;semilogx( freqs,10*log10(fftshift(s/sqrt(N))) );grid on;
xlabel('Frequency, Hz')
ylabel('Welch Periodogram, dB')
title('Welch Periodogram estimate')
saveas( gcf,'welch estimate.png' )
savefig(gcf, 'welch estimate.fig')
% ylim([0,50])
% BLackman Tukey Estimate
ac = [];
sBlackmanTukey = [];
for ii = 1:size(d,2)
    ac(ii,:) = conv(windowedData(:,ii),flip(windowedData(:,ii))) ;%Autocorrelation
    sBlackmanTukey(ii,:) = fft(ac(ii,:));
end
dfBT = df/2;
freqsBT = dfBT*(0:2*N-2);freqsBT = freqsBT-mean(freqsBT);
figure;semilogx(freqsBT, 10*log10(fftshift(abs(mean(sBlackmanTukey/sqrt(N))))));grid on
xlabel('Frequency, Hz')
ylabel('Blackman-Tukey PSD, dB')
title('Blackman-Tukey estimate')
saveas( gcf,'blackman tukey.png' )
savefig(gcf, 'blackman tukey.fig')
% ylim([0,50])
lags = (0:length(mean(ac))-1);
figure;plot(lags-mean(lags),mean(ac));
xlabel('Lag');ylabel('Autocorrelation')
title('Estimate of the signal autocorrelation function')
grid on
saveas( gcf,'autocorrelation function estimate.png' )
savefig(gcf, 'autocorrelation function esstimate.fig')
%% Autoregressive estimate
%Run on the entire data stream at once.
%all-pole model
acYuleWalker = conv( data.d(:,2) ,flip(data.d(:,2) ) );
acYuleWalker = acYuleWalker/max(acYuleWalker);
m = 200;%Number of poles

lag0Ndx = 2^16;

nDataPoints = 2^16-m-1;
X = zeros(nDataPoints, m);
for ii = 1:m
    X(:,ii) = acYuleWalker( lag0Ndx-ii+1:lag0Ndx+nDataPoints-ii );
```

```
end
b = -flip( acYuleWalker( lag0Ndx-nDataPoints:lag0Ndx-1 ) );

a = (X'*X)\X'*b;
a0 = acYuleWalker( lag0Ndx)+sum(acYuleWalker( lag0Ndx+1:lag0Ndx+m).*a);

z = exp(pi*1j*freqsBT/(freqsBT(end)));
Z = repmat(z.', 1, m);
for ii = 1:m
    Z(:,ii) = Z(:,ii).^ii;
end

S = a0./abs(1+Z*a).^2;
figure;semilogx( freqsBT, 10*log10(S) )
xlabel('Frequency,Hz')
ylabel('Spectral estimate, dB')
title('All-pole spectral estimate')
grid on;
% xlim([-50,50])
saveas( gcf,'all pole estimate.png' )
savefig(gcf, 'all pole esstimate.fig')
```

References

[1] Applied Analysis, John K. Hunter and Bruno Nachtergaele