Digital Signal Processing Lab
Experiment 5
Power Spectrum Estimation

Name: Anand Jhunjhunwala
Roll No: 17EC30041
Group No: 63

Objective:

The goal of the assignment is to estimate the PSD of a signal using two methods, first Welch nonparametric method and second parametric method i.e The Yule-Walker AR model method.

Theory:

In this experiment, we first generated a random gaussian sequence r(n) of zero mean and sigma square variance.
Then we passed this generated signal through some digital filter H(z) to generate the output sequence x(n).
Now using H(z) we can get the known PSD of x(n) using the formula: $\left|H\left(e^{j2\pi f}\right)\right|^2 \sigma_r^2$

In part A, we estimated the above-known PSD using Welch Non-parametric method, i.e averaging modified periodogram.
Detailed steps of implementation are given in the problem document.

In part B, we estimated the same PSD using parametric methods i.e The Yule-Walker AR model. In this method, we first estimate H(z) and then use it to calculate PSD.
The detailed steps are shown below.

Step.

1. Obtain the autocorrelation estimate of the sequence $x(n)$ given by,

$$r_{xx}(m) = \frac{1}{N}\sum_{n=0}^{N-m-1} x(n)x(n+m) \qquad m \geq 0$$

2. Find out the estimated AR model parameters $a_1, a_2, \ldots, a_p$ as,

$$\begin{bmatrix} r_{xx}(0) & r_{xx}(1) & \cdots & r_{xx}(p-1) \\ r_{xx}(1) & r_{xx}(0) & \cdots & \cdots \\ \cdots & \cdots & \cdots & \cdots \\ r_{xx}(p-1) & \cdots & \cdots & r_{xx}(0) \end{bmatrix} \begin{bmatrix} a_1 \\ a_2 \\ \cdot \\ a_p \end{bmatrix} = -\begin{bmatrix} r_{xx}(1) \\ r_{xx}(2) \\ \cdot \\ r_{xx}(p) \end{bmatrix}$$

3. Obtain the estimated variance $\hat{\sigma}_{rp}^2$ as,

$$\hat{\sigma}_{rp}^2 = r_{xx}(0) + \sum_{k=1}^{p} a_k r_{xx}(k)$$

4. Construct the estimated AR model $\hat{H}(z)$ by,

$$\hat{H}(z) = \frac{1}{1 + \sum_{k=1}^{p} a_k z^{-k}}$$

Matlab code for part A [File name: assign5.m]

- This code is responsible for performing all part A of the experiment stepwise.

Code snippet:

```
mean = 0;
sigma = 5;
N= 128;
noise= sigma.*randn(N,1) + mean;
b = [1, -0.5, 0.7];
a = [1, -0.9, 0.8, -0.729];
X = filter(b,a,noise);
L = 8;
M = 16;
D = 0;
X_div = [;];
for i=1:L
    X_div(i,:) = X((1+(i-1)*M):(M + (i-1)*M));
end
n = 0:1:(M-1);
hamm_win = 0.54 - 0.46*cos(2*pi*n/(M-1));
U = sumsqr(hamm_win)/M;
P_n = [;];
for i=1:L
    P_n(i,:) = X_div(i,:).*hamm_win;
end
f = -0.5:0.01:0.5;
COS = 0;
SIN = 0;
P_f = zeros(L, length(f));
for j = 1:L
    for F = 1:length(f)
        COS = 0;
        SIN = 0;
        for i = 1:M
            COS = COS + cos(2*pi*f(F)*i)*P_n(j,i);
            SIN = SIN + sin(2*pi*f(F)*i)*P_n(j,i);
        end
        P_f(j,F) = (COS^2 + SIN^2)/(M*U);
    end
end
Pw_f = zeros(L,length(P_f(1,:)));
for i = 1:L
    Pw_f = Pw_f + P_f(i,:);
end
Pw_f = Pw_f./L;
[h, w] = freqz(b,a,128);
```

Code explanation:

The code is fully modular which takes the input as a mean, sigma, filter coefficients a and b, L, M, D and produce known PSD in h, and estimated PSD in Pw_f.

Hamm_win : define a window that has to be used, here hamming window is used.

Matlab code for part B [File name: assign5B.m]

- This code is responsible for performing all part A of the experiment stepwise.

Code snippet:

```
mean = 0;
sigma = 5;
N= 128;
noise= sigma.*randn(N,1) + mean;
b = 1;
a = [1, -0.9, 0.8, -0.729];
X = filter(b,a,noise);
[h, w] = freqz(b,a,128);
p = 6;
r = zeros(p+1);
for i = 0:p
    for j = 1:(N-i)
        r(i+1) = r(i+1) + X(j)*X(j+i);
    end
    r(i+1)= r(i+1)/N;
end
mat = zeros(p,p);
mat2 = zeros(1,p);
for i = 1:p
    mat2(1,i) = r(i+1);
end
for i = 1:p
    for j = 1:p
        mat(i,j) = r(abs(i-j)+1);
    end
end
mat_inv = inv(mat);
coff_a = mat2*mat_inv;
coff_a = coff_a';
sigma_new = 0;
for i=1:p
    sigma_new = sigma_new + coff_a(i,1)*r(i+1);
end
sigma_new = sigma_new + r(1);
b_new = 1;
a_new = ones(p+1);
for i=1:p
    a_new(i+1) = coff_a(i);
end
[h_new, w_new] = freqz(b_new,a_new(:,1),128);
subplot(1,2,1);
l1 =  (abs(h).^2).*sigma^2;
l2 = flip(l1);
l = [l2' l1'];
plot(-length(abs(h)):length(abs(h))-1,l);
title('Known PSD using H(f)');
subplot(1,2,2);
l2_new =  (abs(h_new).^2).*sigma_new;
l1_new = flip(l2_new);
l_new = [l2_new' l1_new'];
plot(-length(abs(h_new)):length(abs(h_new))-1,l);
title('Estimated PSD');
```

Code explanation:

The code is similar to the previous one, p define dimension of parameters mat and mat2 are two matrics with r as their elements, as in step 2, coff_a contains estimated coefficients and a_new, b_new are new filter coefficients. h_new contains estimated filter value.

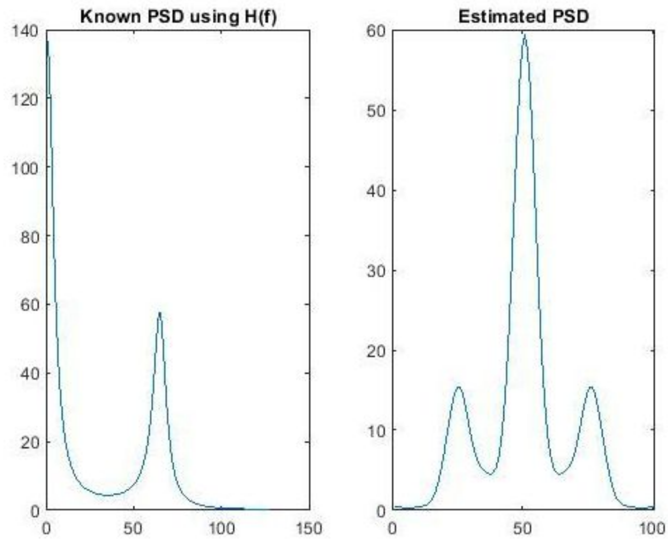
NOTE: Later I merged both the code in one file named assign5.m.

Results:

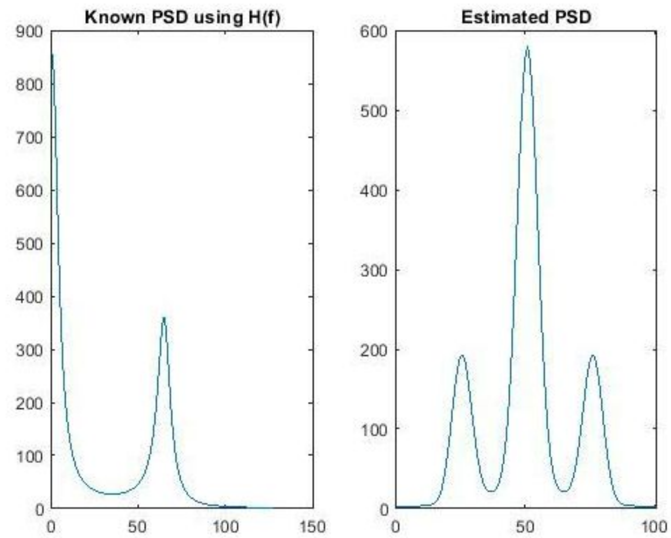
Part A

H(z): Given in question

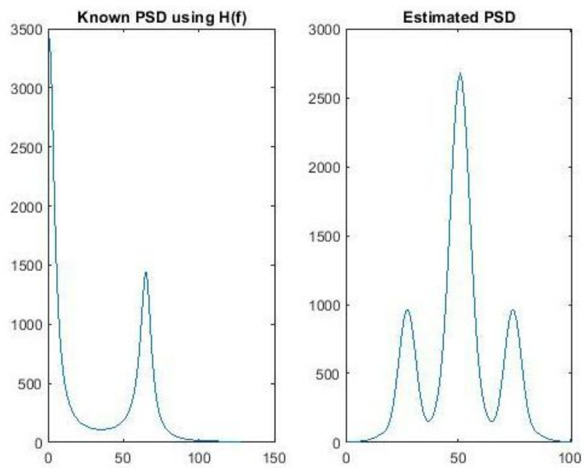
Sigma: 2



Sigma 5

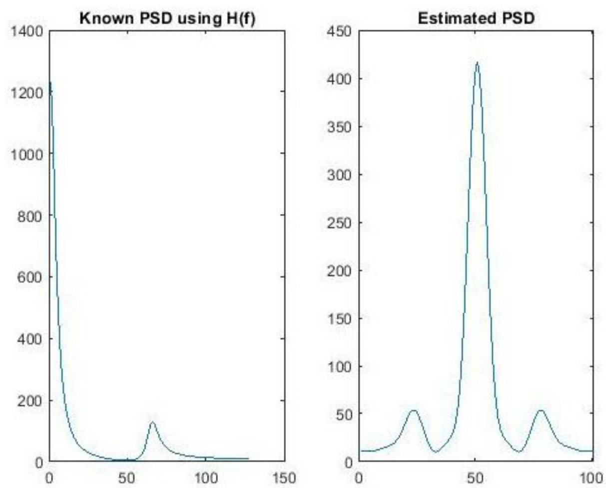


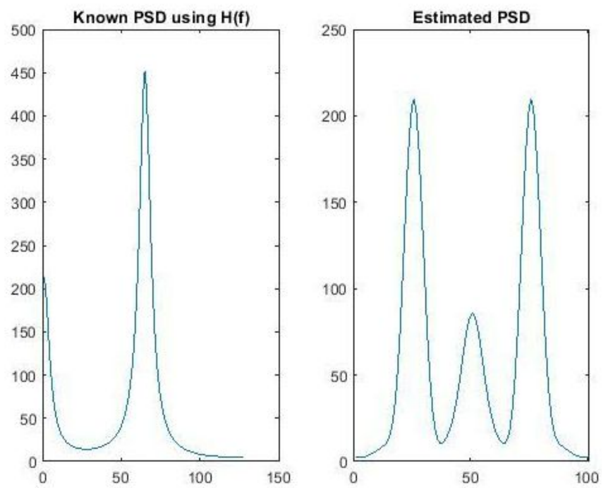
Sigma 10:



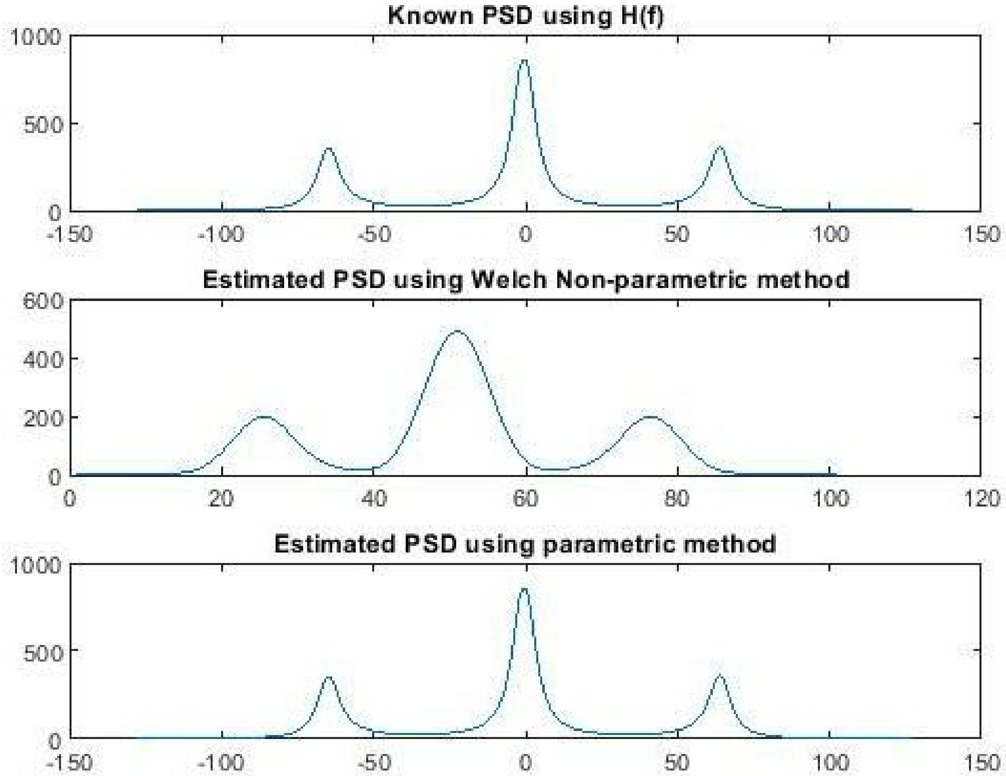
H(z): Modified

Sigma: 2

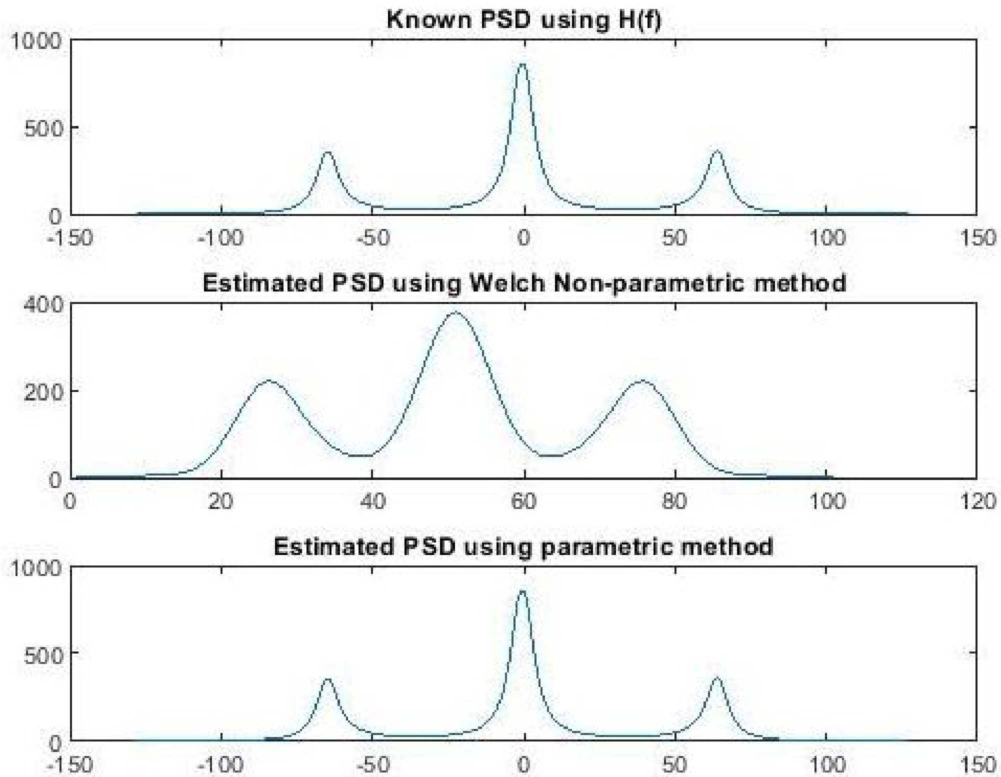




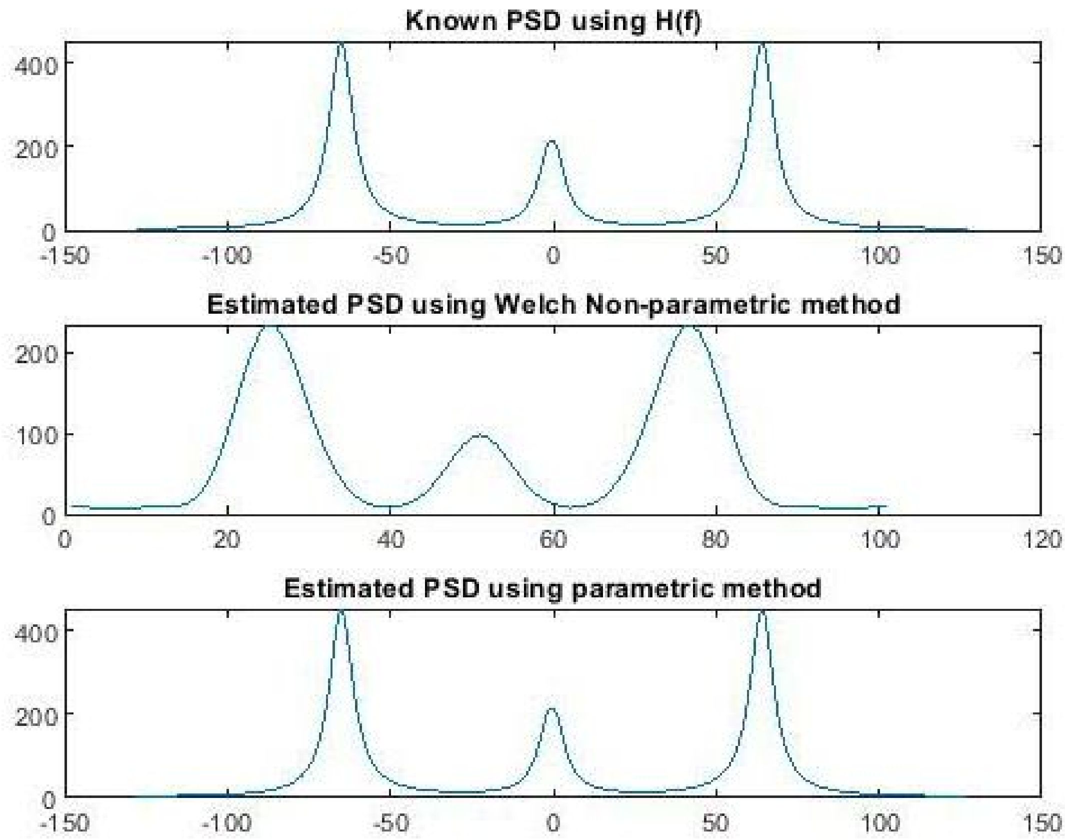
Part B and A combined:
H(z): Given in question
Sigma: 5
P: 6

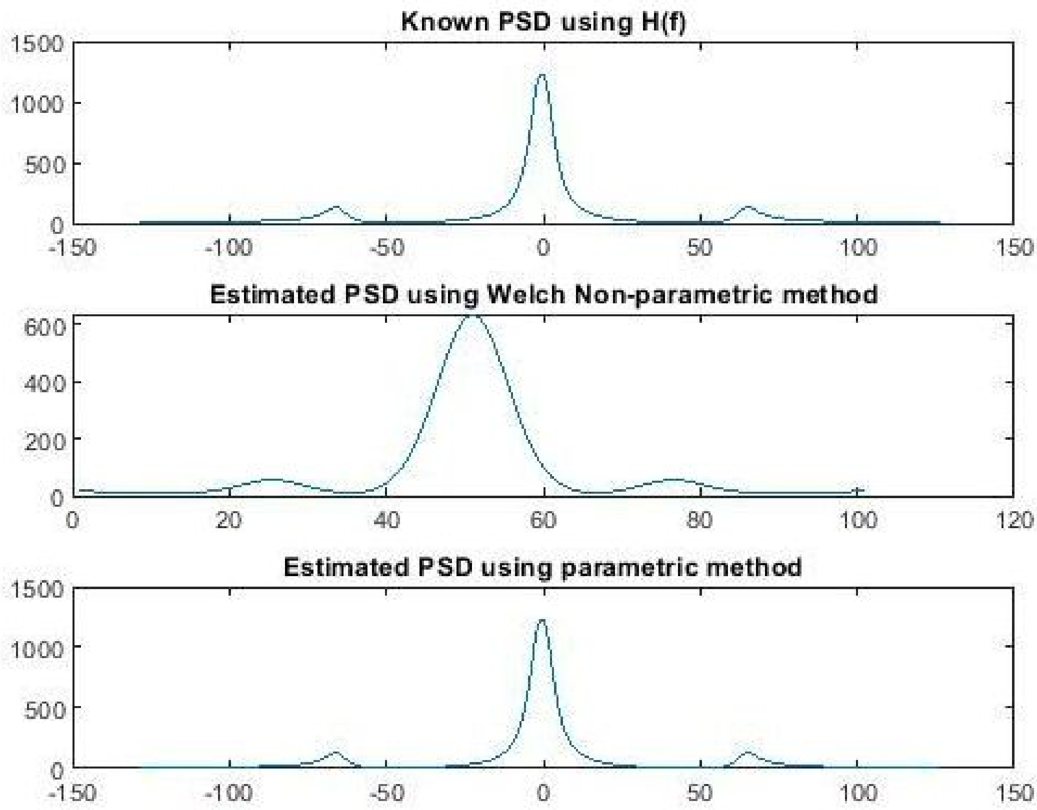


P: 10 | Sigma: 5



H(z): Modified
P:6 | Sigma: 5





This link contains all the original Matlab code and audio signals:
https://drive.google.com/open?id=16JQJtUDYqGH-0U_KxsGff-jSVUaKMXAX

Discussions:

In part A

- The estimated PSD is quite similar to the actual one for all H(z).

- With the increase in sigma, the estimation becomes peakier and close to actual one however significant effect can't be seen with the increase in sigma.

- PSD estimation is found to be better in the case of parametric estimation than non-parametric one.

In part B

- In part B, I observed that predicted and actual PSD are quite similar indicating the parametric method works better.

- The better performance of the parametric method can be due to the fact that in this method we try to estimate H(z) first then use it to calculate PSD, rather than directly estimating PSD which can be prone to errors as in the non-parametric case.

- We do not use any windowing in this method because AR methods normally use linear prediction that extrapolates the signal and they have reduced side lobes.

- A lower model order results in a smooth spectrum and less resolution. But a higher-order model may have false peaks due to extra poles that will appear in the filter function.
