Digital Signal Processing Lab
Experiment -5

Name: Harshavardhan Alimi
Roll No.:18EC10021

Power Spectrum Estimation

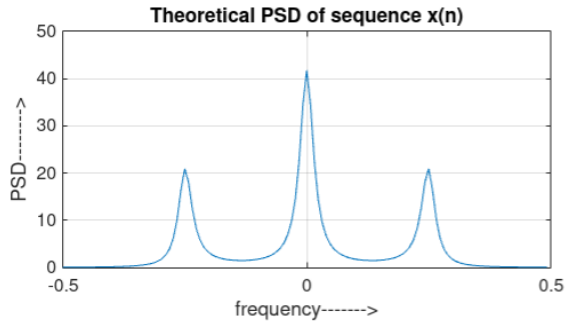
Aim: Estimation of power spectral density using

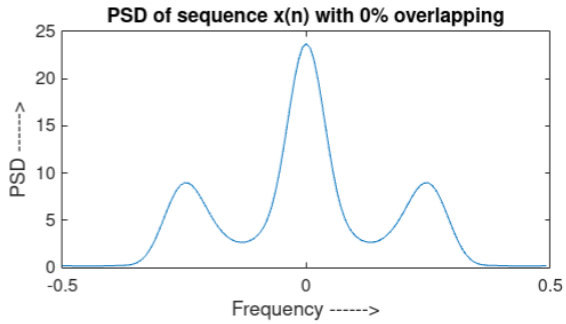
i) welch non-parametric method

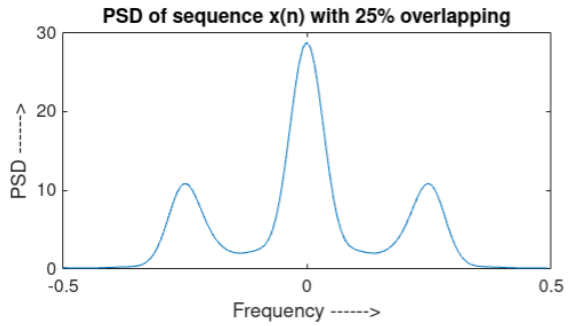
ii) yule walker AR model - parametric method

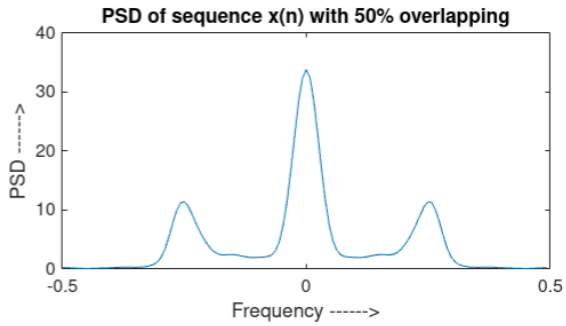
Results:

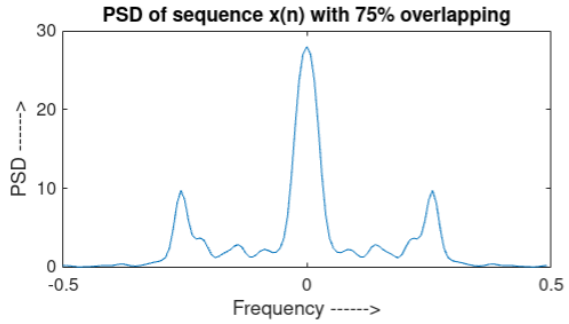
a) Welch's method:

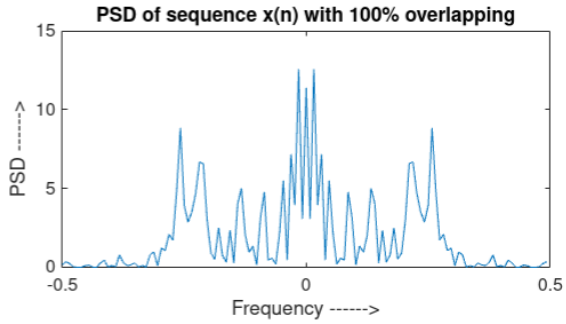




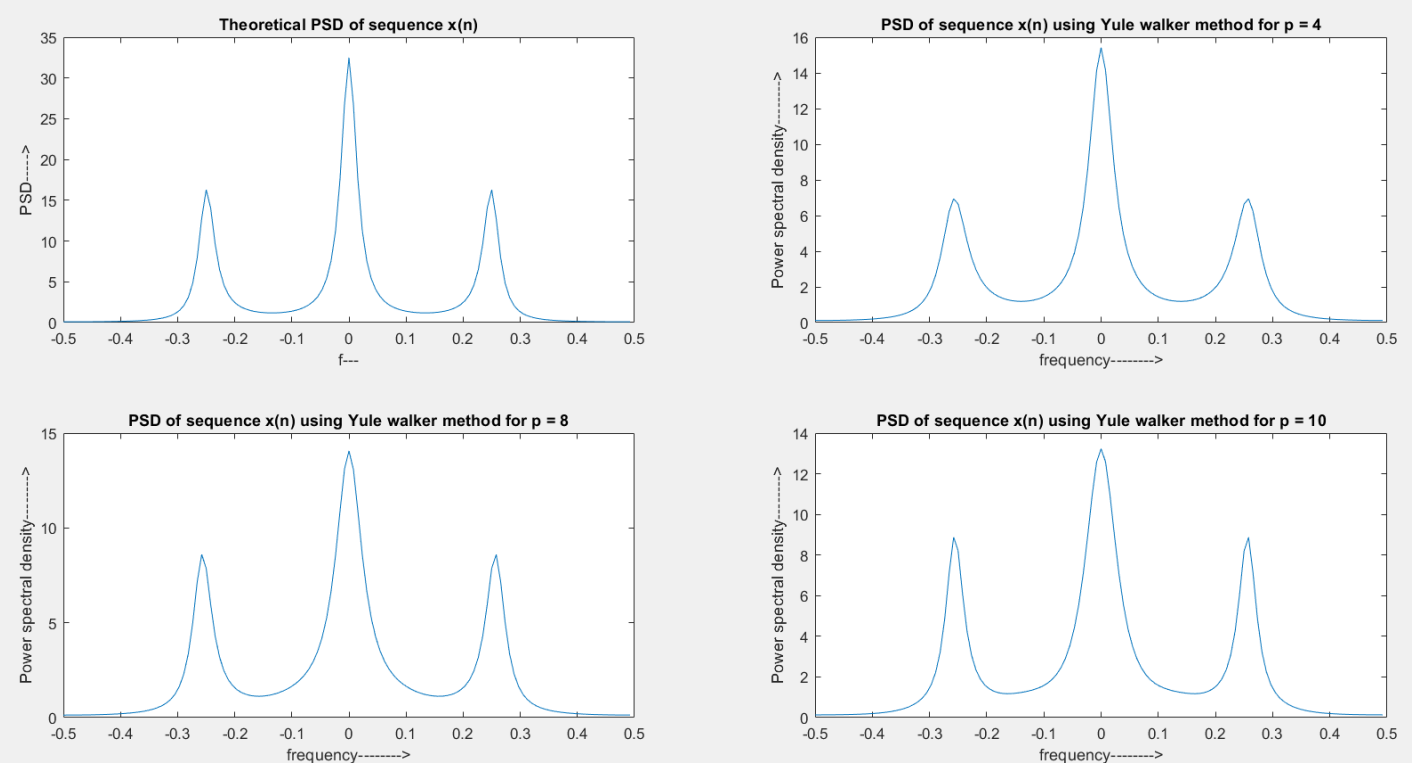








b) Yule walker AR model:



Discussion & Observations:

- In this experiment, we have observed the power spectral estimation using two different methods(parametric and non-parametric methods).
- The power spectrum of the signal gives the distribution of the signal power among various frequencies. The power spectrum is the Fourier transform of the correlation function that describes the characteristics over time series in the frequency domain. So the power spectrum represents variance or power as a function of frequency in the process and tells us where the energy is distributed
- The Power Spectral Estimation method is to obtain an approximate estimation of the power spectral density of a given real random process

- In Welch's method we have used a non-rectangular window, instead if we use a rectangular window, we get the periodogram method of PSD estimation.
- In the Yule-walker AR model, we haven't used windows and this is particularly used when the short data segments are available.
- In this experiment, we have used a hamming window to smoothen the spectral estimation curve.
- In welch's method, the variance of the periodogram estimate does not decay to zero as M→ ∞. This drawback can be recovered by segmenting the data x(n) and calculating the average of the periodogram of each segment.

Appendix

a) welch's Non-parametric method

Code :

```
clear;

a=0.9;
N = 128;
Nfft =128;

r =wgn(1,N,0.1); %random noise
v = var(r);
disp(v);


H = myimpulse(a,Nfft); %impulse function
sq_H = H.^2;
```

```
R = fftshift(fft(r,Nfft));
X = H.*R;

w = -pi:2*pi/(Nfft):(Nfft-2)*pi/(Nfft);help
f = w/(2*pi); %frequency points
z = exp(-1i*2*pi*f);

x = real(ifft(X,N)); %x(n) function

figure(1)
subplot(321);
plot(f,abs(sq_H).*v); %theoretical PSD plot
grid on;
title("Theoretical PSD of sequence x(n)");
xlabel("frequency------->");
ylabel("PSD-------->");
%%
percentage =-25;
L=8;
%% No. of overlapping elements
for count=1:5
   percentage =percentage+25;
   overlap=percentage*0.01;
   M=ceil(N/(1+(1-overlap)*(L-1)));
   D=ceil((1-overlap)*M);
   x_blocks = zeros(L,M);
   n =0;
   %%dividing x into L blocks
   for i = 1:L
       for j =1:M
           n = n+1;
           if (n<=N)
               x_blocks(i,j) = x(n);
           end
       end
       n =i*D;
   end
```

```
    %%window function
    U = 0;
    w = zeros(1,M);

    for i=1:M
        w(i)= 0.54 -0.46*cos(2*pi*(i-1)/(M-1));
        U = U +w(i)*w(i);
    end
    U =U/M ;
    %%PSD of the blocks
    p = zeros(L,Nfft);
    for i = 1:L
        xw = x_blocks(i,:).*w;
        XW = fft(xw,Nfft);
        p(i,:) = abs(XW).*abs(XW);
        p(i,:) = p(i,:)./(M*U);
    end
    %%estimated PSD
    px = zeros(1,Nfft);
    for j =1:Nfft
        for i = 1:L
            px(j) = px(j) + p(i,j);
        end
        px(j) = px(j)/L;
    end
    k = percentage/25;
    subplot(3,2,k+2); %% PSD plot
    plot(f,px);
    title(['PSD of sequence x(n) with ',num2str(percentage),'% overlapping']);
    xlabel('Frequency ------>');
    ylabel('PSD ------>');
end
  %%



%%transfer function
function [H] = myimpulse(a,N)
```

```
        f = -0.5:1/N:0.5*(1-1/N);
        z = exp(-1i*2*pi*f);
        H = zeros(1,N);

        for i = 1:N
              H(i) = 1/((1-a*z(i))*(1-a*1i*z(i))*(1+a*1i*z(i)));
        end

end
```

b) Yule-walker AR model:

Code:

```
clear all;

N =128;
L =8;
M =16;
D =0;
Nfft = 128;

r = normrnd(0,1,1,N);                          %%random function
v = var(r);
R = fftshift(fft(r,Nfft));

H = myimpulse(0.9,Nfft);                       %%impulse function

w = -pi/2:pi/(Nfft):(Nfft-1)*pi/(2*Nfft);
f = w/pi;                                     %%frequency initialization



X = R.*H;
X = fftshift(X);
```

```
x = real(ifft(X,N));                          %%x(n) using ifft

rx = zeros(1,N);

for i =1:N

for j = 1:N-i+1
      rx(i) = rx(i) +x(j)*x(j+i-1);        %%Auto correlation estimate of x(n)
end

rx(i) = rx(i)/N;

end

figure()
plot(f,abs(H.*H)*v);                          %%theoretical PSD plot
title("Theoretical PSD of sequence x(n)");
xlabel("f---");
ylabel("PSD----->");


for p =[4,8,10]
      Rx = zeros(p,p);                        %%Rx is rxx matrix
      B = zeros (1,p);
      Rx(1,:) = rx(1:p);

      B(1) =rx(2);
      for i =2:p

      for j = 1:i-1
      Rx(i,j) = rx(i-j+1);                    %%assigning values to Rx matrix
      end

      Rx(i,i:p) = rx(1:p-i+1);
      B(i) = rx(i+1);

      end
```

```
      B = B';
      A = Rx\(-B);                                  %%AR model parameters
      A = transpose(A);

     variance =rx(1)+A*B;                            %%estimated variance


     H1 = Yuleimpulse(A,Nfft);                        %%Yule impulse function
     H1_sq = abs(H1).*abs(H1);

     PSD1 = H1_sq*abs(variance);                      %%PSD using yule method


     figure()
     plot(f,PSD1);                                 %%PSD using yule method plot
     title(['PSD of sequence x(n) using Yule walker method for p = ',num2str(p)]);
     xlabel('frequency-------->');
     ylabel('Power spectral density-------->');

  end

%%transfer function
function [H] = myimpulse(a,N)
f = -0.5:1/N:0.5*(1-1/N);
z = exp(-1i*2*pi*f);
H = zeros(1,N);

for i = 1:N
   H(i) = 1/((1-a*z(i))*(1-a*1i*z(i))*(1+a*1i*z(i)));
end

end



%%yule transfer function
function [H] = Yuleimpulse(A,N)
```

```
f = -0.5:1/N:0.5*(1-1/N);
z = exp(-1i*2*pi*f);
H = zeros(1,N);
n = length(A);
for i = 1:N
   H(i) =1;
   for j = 1 :n
       H(i) = H(i) +A(j)*(z(i))^j;
   end
   H(i) = 1/H(i);
end

end
```