Digital Signal Processing Lab

Experiment 4 & 5

By Hardik Tibrewal (18EC10020)

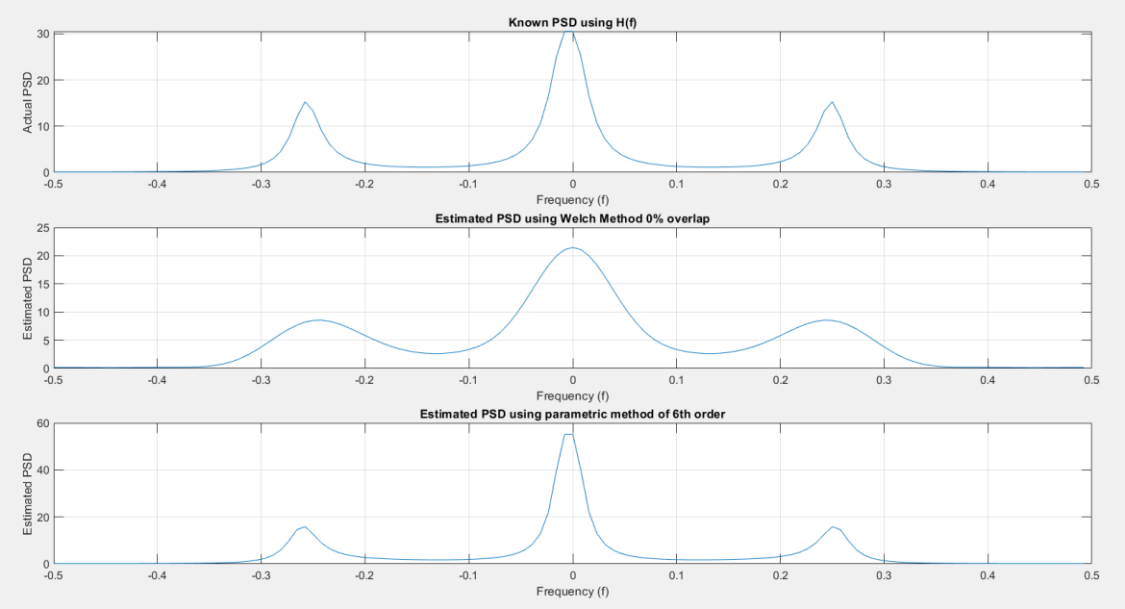
Aim:

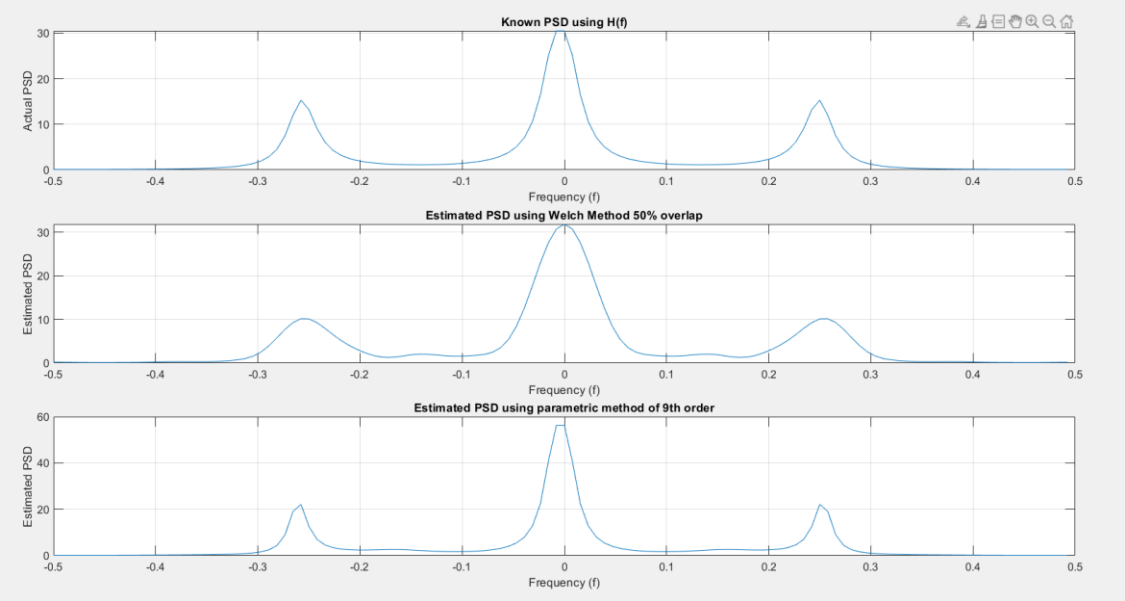
Power Spectrum Density estimation using Welch, and Yule-Walker methods.

Designing an Adaptive Line Enhancer.

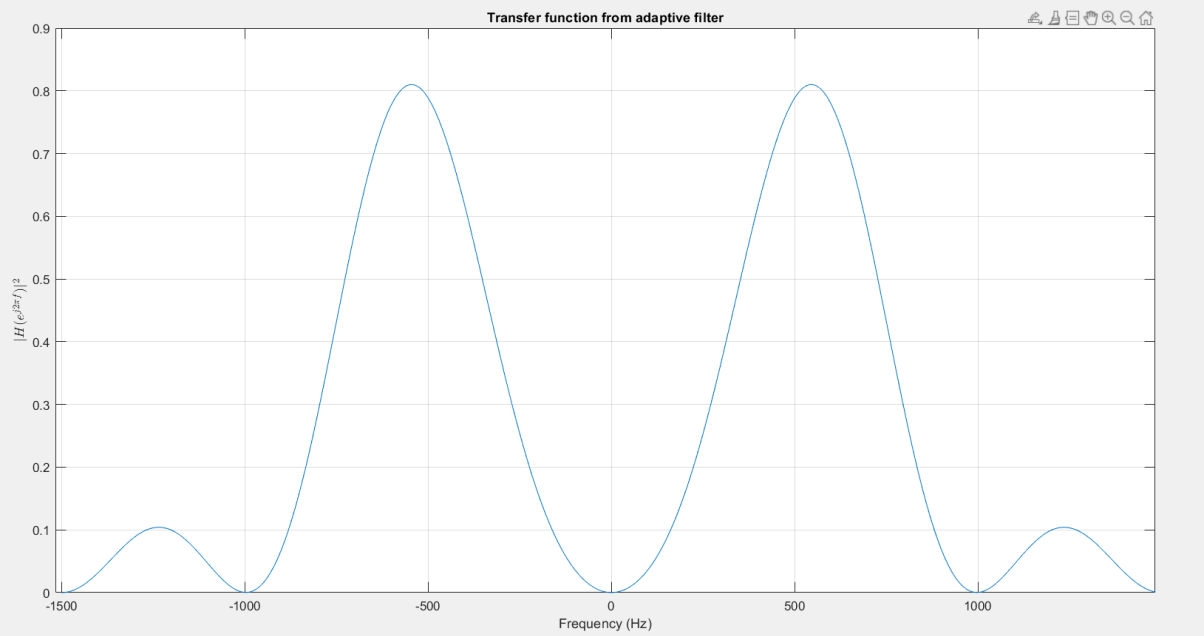
Plots: (Denominator of transfer function is $1z^{0} - 0.9z^{-1} + 0.81z^{-2} - 0.729z^{-3}$)

Exp 4:





Exp 5:



Code:

Exp 4:

```
clc
clear all
close all

mean = 0;
std_dev = 1;
N = 128;
rng('default');
noise = std_dev.*randn(N,1) + mean;
denom = 0.9;
A = [1, -denom, denom^2, -denom^3];

X = filter(1, A, noise);
L = 8;
ov_lap = 0; %fractional overlap
M = round(N/(L*(1-ov_lap)+ov_lap));
D = round(ov_lap*M);

window = hamming(M);
U = (1/M)*sum(window.^2);

X_divs = zeros(M, L);
l=1; r=M;

for ii = 1:L
    X_divs(:,ii) = X(l:r);
    l = r-D+1;
    r = l+M-1;
end
```

```
f = -0.5:1/N:(0.5-(1/N));
P_xx = zeros(N, 1);

for ii = 1:L
   X_divs(:,ii) = X_divs(:,ii).*window;
   P_xx(:) = P_xx(:) + (abs(fft(X_divs(:,ii),N)).^2)/(L*M*U);
end

[H, W] = freqz(1,A,N/2);
H = (abs(H).^2).*std_dev^2;
H1 = [flip(H); H];
figure();
subplot(311);
plot(f, H1);
grid on;
xlabel('Frequency (f)');ylabel("Actual PSD");title("Known PSD using H(f)");

subplot(312);
plot(f, fftshift(P_xx));
grid on;
xlabel('Frequency (f)');ylabel("Estimated PSD");title("Estimated PSD using Welch Method
"+num2str(100*ov_lap)+"% overlap");

%%% Parametric Method %%%
p = 6;
R = xcorr(X)/N;
r = R(N:N+p);
mat = zeros(p,p);
mat2 = -1*(r(2:p+1));

for ii = 1:p
   for jj = 1:p
      mat(ii,jj) = r(abs(ii-jj)+1);
   end
end

coeff_a = [1; mat\mat2];
new_var = sum(coeff_a.*r);

[h_new, w_new] = freqz(1, coeff_a(:,1), N/2);
h_new =  (abs(h_new).^2)*(new_var);
l_new = [flip(h_new); h_new];
subplot(313);
plot(f,l_new);
grid on;
xlabel('Frequency (f)');ylabel("Estimated PSD");title("Estimated PSD using parametric method of "+p+"th
order");
```

Exp 5:

```
clc
clear all
close all

f = 1000;
fs = 3*f;
t = 0:1/fs:0.1-1/fs;
N = length(t);
f_range = -fs/2:fs/N:fs/2-fs/N;
m = 6; mu = 1e-4; epsilon = 1e-3;

x = 2*sin(2*pi*f*t)';
h = randn(m,1)*0.01;

x_n = buffer(x, m, m-1);
x_n = flip(x_n, 1);

y = x_n'*h;
diff = (x-y);
error = zeros(m, N);
for ii = 1:N
    error(:, ii) = x_n(:, ii)*diff(ii);
end
update = sum(error,2);
h_new = h + mu*update;
change = sum((h_new-h).^2)/sum(h.^2);

while change >= epsilon
    h = h_new;
    y = x_n'*h;
    diff = (x-y);
    for ii = 1:size(x_n,2)
        error(:, ii) = x_n(:, ii)*diff(ii);
    end
    update = sum(error,2);
    h_new = h + mu*update;
    change = sum((h_new-h).^2)/sum(h.^2);
end

figure();
plot(f_range, (abs(fft(h_new, N)).^2));
grid on;
xlabel('Frequency (Hz)'); title('Transfer function from adaptive filter')
ylabel('$|H(e^{j2\pi f})|^2$', 'Interpreter', 'latex');
```