Digital Communications and Laboratory
Second Homework

Faccin Dario, Santi Giovanni

MATLAB code

```
clc; close all; clear global; clearvars;

N = [1:1:20];
L = [31 63 127 255 511 1023];
sigdB = -8;
sigmaw = 10^(sigdB/10);
a1 = -0.9635;
a2 = 0.4642;
noise = wgn(4*max(L),1,sigdB);
% save('good_noise3','noise')
load good_noise3

%%
index=0;
for l=1:length(L)
    index = index+1;
for n=1:length(N)
    w = noise(1:4*L(l));
    w_0 = w([1:2:end]);
    w_1 = w([2:2:end]);
    PN = PNSeq(L(l));          % ML sequence repeated once
    x=[PN ; PN];

    h = impz(1, [1 a1 a2]);   % Analytical h
   [h_even,h_odd] = polyphase(h,length(h));

    % scheme pag 239
    z_0=filter(h_even, 1, x);
    z_1=filter(h_odd, 1, x);
    d_0 = z_0 + w_0;
    d_1 = z_1 + w_1;
    d = PS(d_0, d_1);
```

```
    % Correlation method
    h0_cor=corr_method(x, d_0);
    h1_cor=corr_method(x, d_1);
    if N(n)<L(l)
        h0_cor=h0_cor(1:ceil(N(n)/2));
        h1_cor=h1_cor(1:floor(N(n)/2));
    end
    h_cor = PS(h0_cor, h1_cor);

    % Cost function
    d0_hat = filter(h0_cor,1,x);
    d1_hat = filter(h1_cor,1,x);
    d_hat_cor = PS(d0_hat, d1_hat);
    error_cor = d - d_hat_cor;
    E_cor = sum(error_cor(L(l):2*L(l)).^2);
    E_L_cor(1,n) = 10*log10(E_cor/L(l));

    % ls method
    h0_ls=LS(x, d_0, L(l));
    h1_ls=LS(x, d_1, L(l));
    if N(n)<L(l)
        h0_ls=h0_ls(1:ceil(N(n)/2));
        h1_ls=h1_ls(1:floor(N(n)/2));
    end
    h_ls = PS(h0_ls, h1_ls);

    % Cost Function
    d0_hat = filter(h0_ls,1,x);
    d1_hat = filter(h1_ls,1,x);
    d_hat_ls = PS(d0_hat, d1_hat);
    error_ls = d - d_hat_ls;
    E_ls = sum(error_ls(L(l):2*L(l)).^2);
    E_L_ls(1,n) = 10*log10(E_ls/L(l));

end
Cost_cor(:,index) = E_L_cor;
Cost_ls(:,index) = E_L_ls;

end

plot_est(Cost_cor,Cost_ls,sigdB);
```

```
clc; close all; clear global; clearvars;
```

```
L=63;                    % length of PN sequence
Nh=6;                  % Bound on the length of h
% Additive noise
sigdB = -8;
sigmaw = 10^(sigdB/10);
load good_noise3.mat
w = noise(1:4*L)
% w = wgn(4*L,1,sigdB);
w_0 = w([1:2:end-1]);
w_1 = w([2:2:end]);

PN = PNSeq(L);          % ML sequence repeated once
x=[PN ; PN];

%% POLYPHASE REALIZATION
a1 = -0.9635;
a2 = 0.4642;
h = impz(1, [1 a1 a2]);
[h_even,h_odd] = polyphase(h,Nh);

%% ESTIMATE OF h0 WITH THE CORRELATION METHOD
z_0=filter(h_even, 1, x);
d_0 = z_0 + w_0;
h0_cor=corr_method(x, d_0);       % correlation method
var_h0_cor = sigmaw/(L/2);            % variance of the
      estimate

%% ESTIMATE OF h1 WITH THE CORRELATION METHOD
z_1=filter(h_odd, 1, x);
d_1 = z_1 + w_1;
h1_cor=corr_method(x, d_1);           % correlation method
var_h1_cor = sigmaw/(L/2);            % variance of the
      estimate

% Estimated impulse response
h_cor=zeros(Nh,1);
h0_spaced=zeros(Nh,1);
h1_spaced=zeros(Nh,1);
for i=1:Nh
     if (i<=L)
     h_cor(2*i-1)=h0_cor(i);
     h_cor(2*i)=h1_cor(i);
     h0_spaced(2*i-1) = h0_cor(i);
     h1_spaced(2*i) = h1_cor(i);
     end
end
```

```
h_cor=h_cor(1:Nh);
var_cor = var_h0_cor + var_h1_cor;
h0_spaced=h0_spaced(1:Nh);
h1_spaced=h1_spaced(1:Nh);

% Plot
figure,
stem(0:Nh-1,h0_spaced,'ro'), hold on
stem(0:Nh-1,h1_spaced,'ko'), hold on
plot([0:Nh-1],h(1:Nh),'bx','LineWidth',1.5);
legend('h_0 component','h_1 component','Analytic impulse
      response');
xlabel('n'), ylim([-0.4 1.2]);
title('Polyphase components with the Correlation method')
grid on

%% ESTIMATE WITH THE LS METHOD
h0_ls=LS(x, d_0, L);
h1_ls=LS(x, d_1, L);

h_ls=zeros(Nh,1);
h0_spaced_ls=zeros(Nh,1);
h1_spaced_ls=zeros(Nh,1);
for i=1:Nh
    if (i<=L)
    h_ls(2*i-1)=h0_ls(i);
    h_ls(2*i)=h1_ls(i);
    h0_spaced_ls(2*i-1) = h0_ls(i);
    h1_spaced_ls(2*i) = h1_ls(i);
    end
end
h_ls=h_ls(1:Nh);
h0_spaced_ls=h0_spaced_ls(1:Nh);
h1_spaced_ls=h1_spaced_ls(1:Nh);

% Plot
figure,
stem(0:Nh-1,h0_spaced_ls,'ro'), hold on
stem(0:Nh-1,h1_spaced_ls,'ko'), hold on
plot([0:Nh-1],h(1:Nh),'bx','LineWidth',1.5);
legend('h_0 component','h_1 component','Analytic impulse
      response');
xlabel('n'), ylim([-0.4 1.2]);
title('Polyphase components with the LS method')
grid on
```

```
%% comparison cor vs ls
figure,
stem(0:Nh-1,h_cor,'bo')
hold on
stem(0:Nh-1,h_ls,'ro')
grid on
plot([0:Nh-1],h,'bx','LineWidth',1.5);
legend('h_cor','h_ls','Analytic impulse response');

%% VALUES FOR THE TABLE
table = zeros(Nh,3)
for i=1:Nh
    table(i,:) = [h(i) h_cor(i) h_ls(i)];
end
table'
```

```
function [rx]=autocorrelation_Unb(x)
% Unbiased autocorrelation estimator
%INPUT: r.p. x, length of the autocorrelation Lcorr
%OUTPUT: autocorrelation estimate vector rx of length K=length(x)
%every index is augmented by 1 because matlab starts from 1 and not 0
K=length(x);
rx=zeros(K, 1);
   for n=1:K
      %first x that has k as argument
      xnk=x(n:K);
      %second x that has k-n as argument and the conjugate
      xconj=conj(x(1:(K-n+1)));
      rx(n)=(xnk.'*xconj)/(K-n+1);
   end
end
```

```
function [rx]=corr_method(x,d)
% INPUT
% x the input sequence of length 2*L
% r the output of the filter
% OUTPUT
% rx the cross correlation of d and x of length L=length(x)/2

L=length(x)/2;
```

```
rx=zeros(L, 1);
for m=1:L-1
    rtemp=zeros(L,1);
    for k=1:L
        %starts using the samples of d after a transient
            of ength L-1
        rtemp(k)=d(L-2+k)*conj(x(L-1+k-m));
    end
    rx(m)=sum(rtemp)/L;
end
end
```

```
function [h_ls]=LS(x,d,L)
% INPUT
% x the input seq
% d filter output
% L half length PN seq
% OUTPUT
% h_ls the least squares estimate of h

%build matrix Phi and theta by I and o (page 246)
I=zeros(L);
for k=1:L
    I(:,k)=x(L-k+1:(2*L-k));
end
o=d(L:2*L-1);
Phi=I'*I;
theta=I'*o;

h_ls=inv(Phi)*theta;
%h_ls=h_ls(1:N);
end
```

```
function [pn] = PNSeq(L)

r = log2(L+1);
pn = zeros(L,1);

% Initial conditions (set to one, arbitrary)
% Must not be ALL zeros
pn(1:r) = ones(1,r).';

for l=r+1:L
    switch r
        case 1
```

```
            pn(l) = pn(l-1);
        case 2
            pn(l) = xor(pn(l-1), pn(l-2));
        case 3
            pn(l) = xor(pn(l-2), pn(l-3));
        case 4
            pn(l) = xor(pn(l-3), pn(l-4));
        case 5
            pn(l) = xor(pn(l-3), pn(l-5));
        case 6
            pn(l) = xor(pn(l-5), pn(l-6));
        case 7
            pn(l) = xor(pn(l-6), pn(l-7));
        case 8
            pn(l) = xor(xor(pn(l-2),pn(l-3)),xor(pn(l-4),
                pn(l-8)));
        case 9
            pn(l) = xor(pn(l-5), pn(l-9));
        case 10
            pn(l) = xor(pn(l-7), pn(l-10));



    end
end

% Bits are {-1, 1}
pn = 2*pn -1;

end
```

```
function [h_even h_odd] = polyphase(h,Nlim)

% Even samples
h_even=zeros(ceil(Nlim/2),1);
for k=1:(Nlim/2)
    h_even(k)=h(2*k-1);
end

% Odd samples
h_odd=zeros(floor(Nlim/2),1);
for k=1:Nlim/2
    h_odd(k)=h(2*k);
end

```

```
end
```

```
function [h] = PS(h0, h1)

temp = length(h0)+length(h1);
h=zeros(temp,1);
if mod(temp,2)==0
    for i=1:temp/2
        h(2*i-1)=h0(i);
        h(2*i)=h1(i);
    end
elseif length(h0)==1
        h(1)=h0(1);
else
    for i=1:length(h0)-1
        h(2*i-1)=h0(i);
        h(2*i)=h1(i);
    end
    h(2*(i+1)-1) = h0(i+1);
end

end
```

```
function plot_est(cor, ls, sigdB)

set(0,'defaultTextInterpreter','latex')
% figure()
scrsz = get(0,'ScreenSize');
figure('Position',[15 scrsz(4)/5 scrsz(3)/1.5 scrsz(4)
    /1.5])
[N,lengL] = size(cor);
N = [1:1:N];
a = sigdB*ones(1,20);

plot(N,ls(:,1),'b-*')
hold on, plot(N,ls(:,2),'c*-')
hold on, plot(N,ls(:,3),'g-*')
hold on, plot(N,ls(:,4),'y-*')
hold on, plot(N,ls(:,5),'m-*')
hold on, plot(N,ls(:,6),'r-*')
hold on, plot(N,cor(:,1),'bo--')
hold on, plot(N,cor(:,2),'co--')
hold on, plot(N,cor(:,3),'go--')
hold on, plot(N,cor(:,4),'yo--')
hold on, plot(N,cor(:,5),'mo--')
```

```
hold on, plot(N,cor(:,6),'ro--')

hold on, plot(a,'b--','LineWidth',2);
text(2,-7.7,'$\sigma_w^2$','FontSize',16,'Color','blue');

xlabel('$N_h$');
ylabel('$\mathcal{E}/L$ [dB]')
xlim([1 20]);
legend('L31','L63','L127','L255','L511','L1023')
set(gca,'FontSize',15);
grid on

end
```

```
clc
clearvars
close all
set(0,'defaultTextInterpreter','latex')    % latex format

% Given Parameters
Tc = 1;
fd = (40*10^-5)/Tc;                % Doppler spread given
Tp = 1/10*(1/fd);
N_h0 = 7500;                       % samples first plot
N_t = 80000;                       % samples of second plot

K_dB = 2;                          % Rice Factor in dB
K = 10^(K_dB/10);                  % Rice Factor in linear units
C = sqrt(K/(K+1));

[a_ds, b_ds] = ClassicalDS();       % Parameters of the IIR filter which implement
                                    % the classical Doppler Spectrum (page 317)
h_dopp = impz(b_ds, a_ds);          % Impulse response of DS
E_d = sum(h_dopp.^2);               % Energy of the impulse response
b_ds = b_ds/sqrt(E_d);
Md = 1-C^2;                         % normalization of the statistical power

% Doppler spectrum
[H_dopp,w]=freqz(h_dopp,1,1024,'whole',1/Tp);
DS = abs(H_dopp).^2;

% figure
subplot(121), plot(h_dopp,'r'), ylabel('$|h_{ds}|$'), hold on
stem(1:length(h_dopp),real(h_dopp));
axis([0 Tp -0.15 0.25]), grid on
legend('continuous |h_{ds}|','sampled |h_{ds}|');
title('Impulse response of the IIR filter');
subplot(122), plot(w,10*log10(DS)), ylabel('$|D(f)|$'), grid on;
hold on, plot([fd fd], [-60 20], 'r--'), text(4.1e-4, 10, '$f_d$');
```

```
xlim([0 3*fd]), xlabel('f');
ylim([-60 20]);
legend('Doppler Spectrum');
title('Doppler Spectrum')

% Transient is determined by the pole closest to the unit circle
poles = abs(roots(a_ds));                  % poles' magnitude
most_imp = max(poles);
tr = 5*Tp*ceil(-1/log(most_imp));          % transient as 5*Neq*Tp

h_samples_needed = N_t+tr;                 % total length including the transient
w_samples_needed = ceil(h_samples_needed/Tp);
% w = wgn(w_samples_needed,1,0,'complex');        % w ~ CN(0,1)
load ex2_noise
hprime = filter(b_ds, a_ds, w);

t = 1:length(hprime);                      % interpolation to Tq
Tq = Tc/Tp;
t_fine = Tq:Tq:length(hprime);
h_fine = interp1(t, hprime, t_fine, 'spline');
sigma = sqrt(Md);
h_fine = h_fine*sigma;                     % impose the desired power delay profile
h0 = h_fine(tr+1:end)+C;                   % remove the transient and add C

% figure, plot(abs(h0(1:N_h0)))
% xlabel('$nT_c$')
% ylabel('$|h_0(nT_c)|$')
% xlim([1 N_h0]), grid on
% title('Impulse response of the channel')

% ESTIMATE OF THE PDF OF H_p=|h0|/sqrt(M)
h_p = h0/sqrt(C^2+Md);           % the normalization here does not make much sense
                                 % as M_h0=1-C^2, but it's to keep the formulas as in the book
abs_h = abs(h_p);                % magnitude
a=linspace(0,10,3000);
% Rice distribution
```

```
th_pdf = 2*(1+K).*a.*exp(-K-(1+K).*a.^2).*besseli(0,2.*a*
      sqrt(K*(1+K)));
% Estimate of the pdf
[y,t] = hist(abs_h,30);
est_pdf = y/max(y);

figure
bar(t,est_pdf,'g'), hold on,plot(a,th_pdf,'r--','
      LineWidth',2);
ylabel('Number of samples')
xlabel('Value')
title('Histogram of $\frac{|h_0|}{\sqrt{M_{|h_0|}}}$')
legend('estimate pdf','theoretical pdf');
axis([0 3 0 1.3]);
grid on

%% SPECTRUM ESTIMATION
% Theoretical PSD
Npoints=N_t*1/Tp;
[H_dopp,w]=freqz(h_dopp,1,Npoints,'whole');
H_dopp=(1/Npoints)*abs(H_dopp).^2;
DS = fftshift(H_dopp);

% Welch estimator
D = ceil(N_t/2);          % window length
D = 40000;
S = D/2;                      % overlap
w_welch=window(@bartlett,D);
[Welch_P, N] = welchPSD(h0', w_welch, S);
Welch_P = Welch_P/N;
Welch_cent=fftshift(Welch_P);

C_comp = 10*log10(C^2);
PSD_theo = 10*log10(Md*DS);
PSD_theo(length(PSD_theo)/2) = C_comp;


f1=[-N/2+1:N/2];
f2=[-Npoints/2+1:Npoints/2];

figure,
plot(f1-1, 10*log10(Welch_cent)) , hold on, plot(f2,
      PSD_theo,'r')
ylim([-40 0])
xlim([-5*N*fd  5*N*fd]);
xticks([-5*N*fd -4*N*fd -3*N*fd -2*N*fd -1*N*fd 0 1*N*fd
```

```
    2*N*fd 3*N*fd 4*N*fd 5*N*fd])
xticklabels({'-5f_d','-4f_d','-3f_d','-2f_d','-f_d','0','f_d','2f_d','3f_d','4f_d','5f_d'});
ylabel('H(f) [dB]')
xlabel('f')
legend('Welch Periodogram','Theoretical PSD')
title('Spectrum Estimate')
grid on

%%
f1=[-N/2+1:N/2];
f2=[-Npoints/2+1:Npoints/2];

% Comparison of different S,D
D = [10000 15000 20000 40000];
S = D./2;
for i=1:length(S)
    w_welch=window(@hamming,D(i));
    [Welch_P(:,i), N] = welchPSD(h0', w_welch, S(i));
end
Welch_P = Welch_P/N;
Welch_cent=fftshift(Welch_P);

figure,
plot(f1, 10*log10(Welch_cent)), hold on
plot(f2, 10*log10(Md*DS),'r','LineWidth',1.5)
ylim([-50 0])
xlim([-5*N*fd  5*N*fd]);
xticks([-5*N*fd -4*N*fd -3*N*fd -2*N*fd -1*N*fd 0 1*N*fd
    2*N*fd 3*N*fd 4*N*fd 5*N*fd])
xticklabels({'-5f_d','-4f_d','-3f_d','-2f_d','-f_d','0','f_d','2f_d','3f_d','4f_d','5f_d'});
ylabel('H(f) [dB]')
xlabel('f')
legend(['D = ' int2str(D(1)) ' and S = ' int2str(S(1)) ],
    ['D = ' int2str(D(2)) ' and S = ' int2str(S(2)) ],[
    'D = ' int2str(DS(3)) ' and S = ' int2str(S(3)) ],['
    D = ' int2str(DS(4)) ' and S = ' int2str(S(4)) ]);
```

```
clc; close all; clear global; clearvars;

N = [1:1:20];
L = [31 63 127 255 511 1023];
sigdB = -8;
sigmaw = 10^(sigdB/10);
```

```
a1 = -0.9635;
a2 = 0.4642;
noise = wgn(4*max(L),1,sigdB);
% save('good_noise3','noise')
load good_noise3

%%
index=0;
for l=1:length(L)
    index = index+1;
for n=1:length(N)
    w = noise(1:4*L(l));
    w_0 = w([1:2:end]);
    w_1 = w([2:2:end]);
    PN = PNSeq(L(l));            % ML sequence repeated once
    x=[PN ; PN];

    h = impz(1, [1 a1 a2]);    % Analytical h
   [h_even,h_odd] = polyphase(h,length(h));

    % scheme pag 239
    z_0=filter(h_even, 1, x);
    z_1=filter(h_odd, 1, x);
    d_0 = z_0 + w_0;
    d_1 = z_1 + w_1;
    d = PS(d_0, d_1);

    % Correlation method
    h0_cor=corr_method(x, d_0);
    h1_cor=corr_method(x, d_1);
    if N(n)<L(l)
        h0_cor=h0_cor(1:ceil(N(n)/2));
        h1_cor=h1_cor(1:floor(N(n)/2));
    end
    h_cor = PS(h0_cor, h1_cor);

    % Cost function
    d0_hat = filter(h0_cor,1,x);
    d1_hat = filter(h1_cor,1,x);
    d_hat_cor = PS(d0_hat, d1_hat);
    error_cor = d - d_hat_cor;
    E_cor = sum(error_cor(L(l):2*L(l)).^2);
    E_L_cor(1,n) = 10*log10(E_cor/L(l));

    % ls method
    h0_ls=LS(x, d_0, L(l));
```

```
    h1_ls=LS(x, d_1, L(l));
    if N(n)<L(l)
        h0_ls=h0_ls(1:ceil(N(n)/2));
        h1_ls=h1_ls(1:floor(N(n)/2));
    end
    h_ls = PS(h0_ls, h1_ls);

    % Cost Function
    d0_hat = filter(h0_ls,1,x);
    d1_hat = filter(h1_ls,1,x);
    d_hat_ls = PS(d0_hat, d1_hat);
    error_ls = d - d_hat_ls;
    E_ls = sum(error_ls(L(l):2*L(l)).^2);
    E_L_ls(1,n) = 10*log10(E_ls/L(l));

end
Cost_cor(:,index) = E_L_cor;
Cost_ls(:,index) = E_L_ls;

end

plot_est(Cost_cor,Cost_ls,sigdB);
```

```
clc; close all; clear global; clearvars;

N = [1:1:20];
L = [31 63 127 255 511 1023];
sigdB = -8;
sigmaw = 10^(sigdB/10);
a1 = -0.9635;
a2 = 0.4642;
noise = wgn(4*max(L),1,sigdB);
% save('good_noise3','noise')
load good_noise3

%%
index=0;
for l=1:length(L)
    index = index+1;
for n=1:length(N)
    w = noise(1:4*L(l));
    w_0 = w([1:2:end]);
    w_1 = w([2:2:end]);
    PN = PNSeq(L(l));          % ML sequence repeated once
    x=[PN ; PN];
```

```
    h = impz(1, [1 a1 a2]);    % Analytical h
    [h_even,h_odd] = polyphase(h,length(h));

    % scheme pag 239
    z_0=filter(h_even, 1, x);
    z_1=filter(h_odd, 1, x);
    d_0 = z_0 + w_0;
    d_1 = z_1 + w_1;
    d = PS(d_0, d_1);

    % Correlation method
    h0_cor=corr_method(x, d_0);
    h1_cor=corr_method(x, d_1);
    if N(n)<L(l)
        h0_cor=h0_cor(1:ceil(N(n)/2));
        h1_cor=h1_cor(1:floor(N(n)/2));
    end
    h_cor = PS(h0_cor, h1_cor);

    % Cost function
    d0_hat = filter(h0_cor,1,x);
    d1_hat = filter(h1_cor,1,x);
    d_hat_cor = PS(d0_hat, d1_hat);
    error_cor = d - d_hat_cor;
    E_cor = sum(error_cor(L(l):2*L(l)).^2);
    E_L_cor(l,n) = 10*log10(E_cor/L(l));

    % ls method
    h0_ls=LS(x, d_0, L(l));
    h1_ls=LS(x, d_1, L(l));
    if N(n)<L(l)
        h0_ls=h0_ls(1:ceil(N(n)/2));
        h1_ls=h1_ls(1:floor(N(n)/2));
    end
    h_ls = PS(h0_ls, h1_ls);

    % Cost Function
    d0_hat = filter(h0_ls,1,x);
    d1_hat = filter(h1_ls,1,x);
    d_hat_ls = PS(d0_hat, d1_hat);
    error_ls = d - d_hat_ls;
    E_ls = sum(error_ls(L(l):2*L(l)).^2);
    E_L_ls(l,n) = 10*log10(E_ls/L(l));

end
```

```
Cost_cor(:,index) = E_L_cor;
Cost_ls(:,index) = E_L_ls;

end

plot_est(Cost_cor,Cost_ls,sigdB);
```

```
function [a,b] = ClassicalDS()

a = [1, −4.4153, 8.6283, −9.4592, 6.1051, −1.3542,
    −3.3622, 7.2390, −7.9361, 5.1221, −1.8401, 2.8706e
    −1];
b = [1.3651e−4, 8.1905e−4, 2.0476e−3, 2.7302e−3, 2.0476e
    −3, 9.0939e−4, 6.7852e−4, 1.3550e−3, 1.8076e−3,
    1.3550e−3, 5.3726e−4, 6.1818e−5, −7.1294e−5, ...
        −9.5058e−5, −7.1294e−5, −2.5505e−5, 1.3321e−5,
            4.5186e−5, 6.0248e−5, 4.5186e−5, 1.8074e−5,
            3.0124e−6];

end
```

```
function [welch_est, Ns] = welchPSD(inputsig, window,
    overlaps)
% REQUIRES COLUMN VECTOR FOR THE INPUT DATA

% Length of the window
D = length(window);
% Length of input signal
K = length(inputsig);
% Normalized energy of the window
Mw = sum(window .^ 2) * (1/D);
% Number of subsequences
N_s = floor((K−D)/(D−overlaps) + 1);
%Initialization of each periodogram
P_per = zeros(K, N_s);

for s = 0:(N_s−1)
    % Windowed input
    x_s = window .* inputsig(s*(D−overlaps)+1:s*(D−
        overlaps)+D);
    % DFT on K samples of windowed input
    X_s = fft(x_s, K);
    % Periodogram for the window
    P_per(:,s+1) = (abs(X_s)).^2 * (1/(D*Mw)); % Tc = 1;
end
```

```
% Sum of all periodograms
welch_est = sum(P_per, 2) * (1/N_s);
Ns = length(welch_est);
end
```