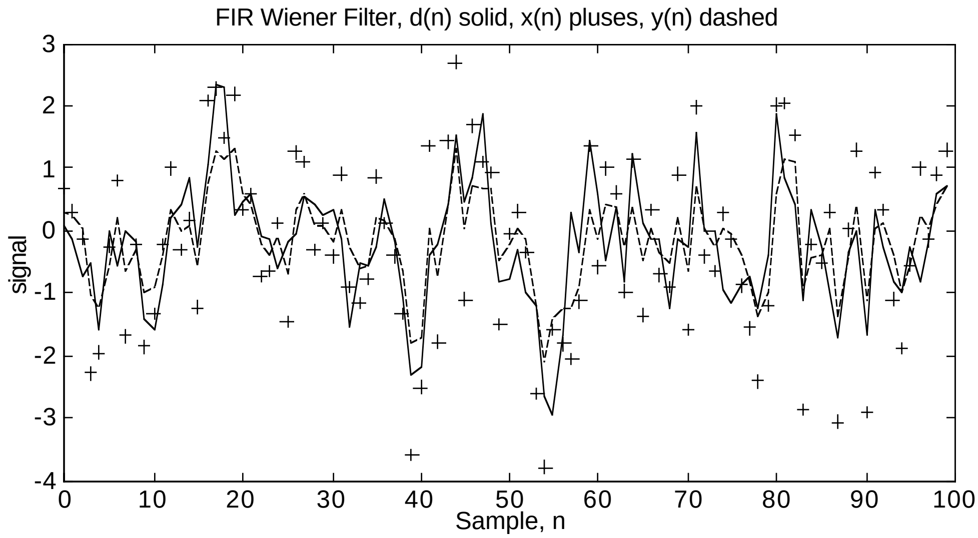
```
% EE254                                                         Prof. Essam Marouf
% DSP II                                                        Matlab Handout #3
% -------------------------------------------------------------------------------------------------------
% Wiener Filters
% -----------------
% 1- FIR Wiener Filter
% -------------------------
% (Analytic solution is done in class).
%
clear, close all
% Generate AR(1) desired signal d(n).
sigma_sq_w=0.64;
b=[1];
a=[1 -0.6];
L1=100;
L2=200;
d=arma(sigma_sq_w, a,b,L1,L2);                 % desired signal; see function listing below
d=d';
n=[0:L1-1]';                                   % Sample number
%
% Add a WGN vector v to d
sigma_sq_v=1;
v=sqrt(sigma_sq_v)*randn(size(d));             % WGN of variance sigma_sq_v
x=d+v;                                         % "Measured" noisy signal
%
% Experiment with filter order M = 1, 2, 3, ...etc.
M=2;
% Compute correlation matrix R and cross-correlation vector rxd.
k=[0:M]';
rdd=(0.6).^k;                                  % see lecture results for rdd(k)
rxd=rdd;                                       % d and v are uncorrelated
rxx(1)=rdd(1)+sigma_sq_v;                      % rxx(0)=rdd(0)+sigma_sq_v
rxx(2:M+1)=rdd(2:M+1);                         % rxx(k)=rdd(k) for all k>0
R=toeplitz(rxx);                               % when in Matlab, type help Toeplitz
ho=R\rxd                                       % solve for optimal filter coefficients
    ho =
         0.4451
         0.1500
         0.0549

y=filter(ho,1,x);                              % filter noisy signal
e=d-y;                                         % error signal
J_hat=std(e)^2                                 % estimate the MSE
J_hat =
    0.3591                                     % random value around J_min

J_min=rdd(1)-ho'*rxd                           % class result for min MSE
J_min =
    0.4451                                     % statistical minimum
%
% plot the results
plot(n,d,'-',n,x,'+',n,y,'--')
xlabel('Sample, n'), ylabel('signal'),
```

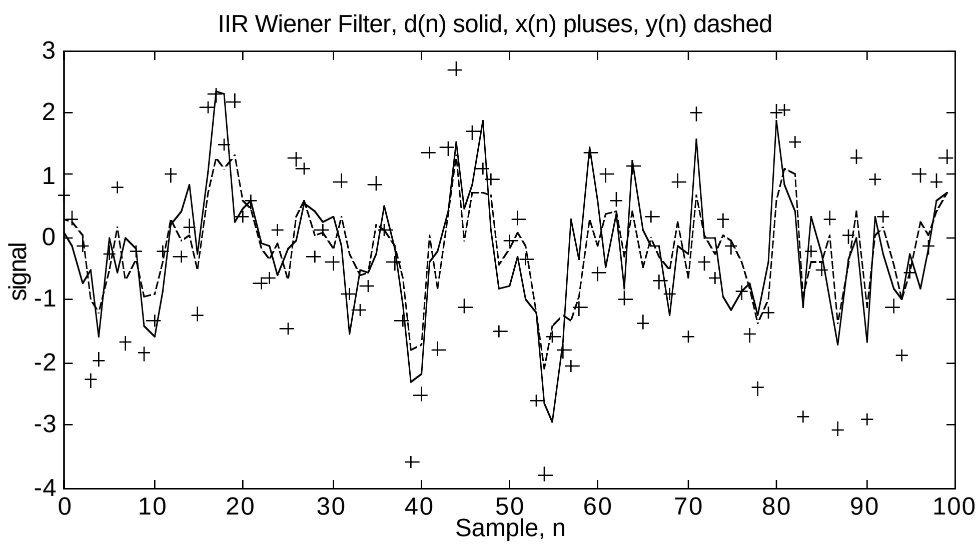
```
title('FIR Wiener Filter, d(n) solid, x(n) pluses, y(n) dashed'), pause
%
% Causal IIR Wiener Filter
% ------------------------
aa=[1 -1/3];                                    % From class results:
bb=[4/9];
yy=filter(bb,aa,x);                             % apply Wiener filter
ee=d-yy;                                        % corresponding error signal
J_hat=std(ee)^2                                 % estimate of MSE
J_hat =
        0.3626

% J_min=0.444;                                  % statistical min; see class note
% plot the results
plot(n,d,'-',n,x,'+',n,yy,'--')
xlabel('Sample, n'), ylabel('signal'),
title('IIR Wiener Filter, d(n) solid, x(n) pluses, y(n) dashed')
```

```
function x=arma(sigma_sq, a, b, L1, L2)
% ARMA Autoregressive-Moving Average Signal Model
% arma(sigma_sq, a, b) returns an L x 1 arma vector x
% by filtering white gauassian noise of variance sigma_sq
% through a digital filter of numerator coefficients "b"
% and denomenator coefficients "a". The first 100 samples
% of the generated vector are discarded to minimize transient
% effects
% Input: sigma_sq   variance of white gaussian noise input
%        a          row vector; denomenator filter coefficients
%        b          col vector; numerator filter coefficients
%        L1         Length of returned ARMA vector
%        L2         Number of samples to be discarded
%
L=L1+L2;
x=sqrt(sigma_sq)*randn(size([1:L]));            % WGN variance = sigma_sq
x=filter(b,a,x);                                % type help filter
x=x(L2+1:L);                                    % Discard the first L1 samples
```

% 1- FIR Wiener Filter
% ------------------------



% 2- Causal IIR Wiener Filter
% -----------------------------



Example (adapted from Example 7.2.6 of Hayes)

$$d(n) = \sin(n\omega_0 + \phi) + v_1(n) \quad , \omega_0 = 0.05\pi$$

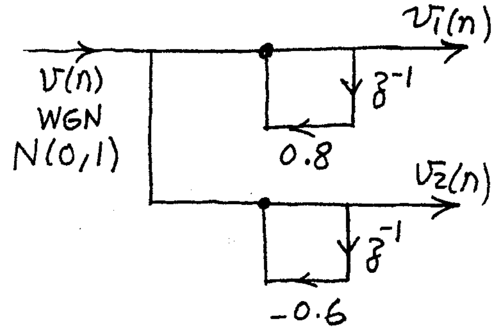
$$x(n) = v_2(n)$$

where $v_1(n)$ and $v_2(n)$ are correlated noise (generated by the same source). Here $v_1(n)$ & $v_2(n)$ are modeled as AR(1) random signals generated by the same white noise input $v(n)$

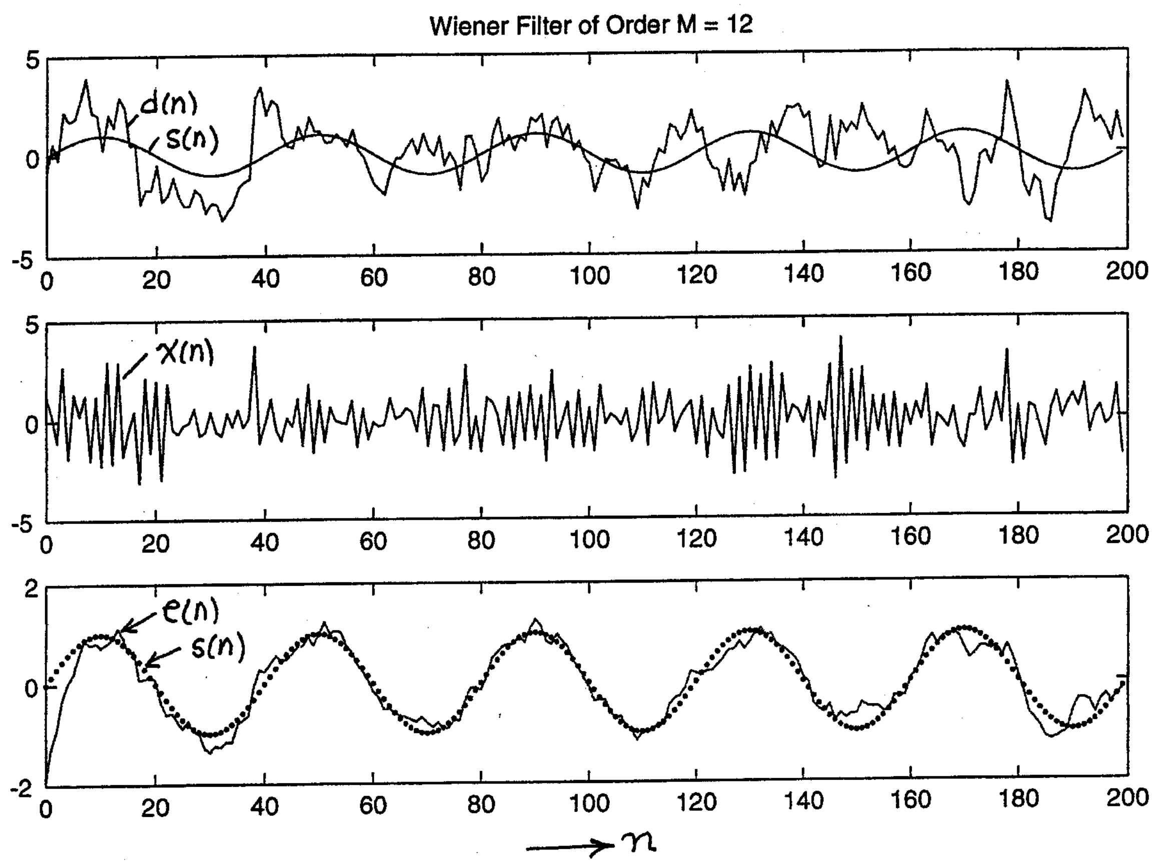
$$v_1(n) = 0.8\, v_1(n-1) + v(n)$$

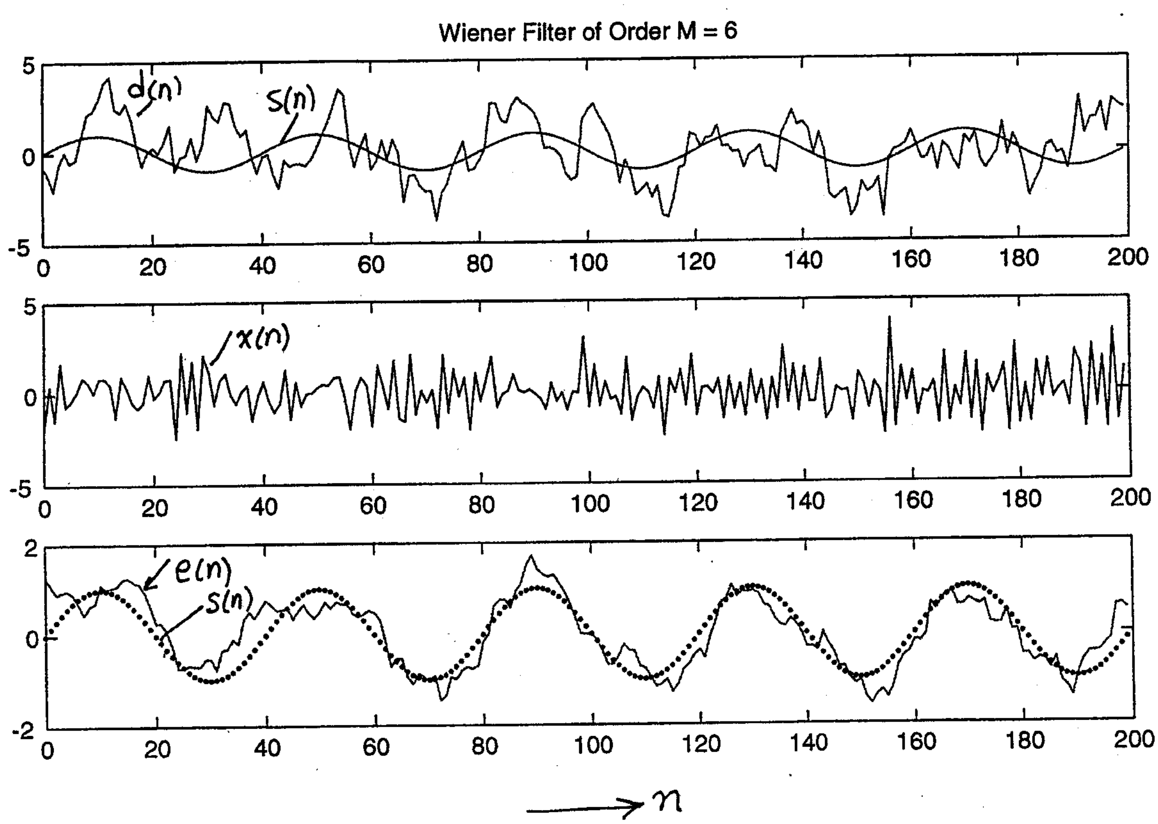
$$v_2(n) = -0.6\, v_2(n-1) + v(n)$$

$v(n)$ is zero-mean unit-variance




```
% Hayes Example 7.2.6
% Noise cancellation using Wiener Filter
% ---------------------------------------
close all, clear
n1=[0:500]';
vn=randn(size(n1));            % white guassian noise
tmp=filter(1,[1 -0.8],vn);     % generate 501 samples
v1n=tmp(301:500);          % keep only the last 200 (avoids transients)
tmp=filter(1,[1  0.6],vn);     % same for v2n
v2n=tmp(301:500);
%
w0=0.05*pi;
n=[0:199]';
sn= sin(w0*n);                 % phi is any value between 0 anf 2pi;
                          % phi=0 is chosen for convenience
dn=sn+v1n;                     % noisy sinusoid; primary signal
xn=v2n;                   % correlated noise; secondary signal

% estimate from the data rxx and rxd
N=length(xn);
rxx=xcorr(xn,xn)/N;   % biased correlation estimator
rxd=xcorr(xn,dn)/N;   % cross-correlation estimator
%
M = 6;                % Wiener filter order
rxx=rxx(N: N+M);      % extract rxx(0), rxx(1), ...., rxx(M)
rxd=rxd(N: N+M);      % extract rxd(0), rxd(1), ...., rxd(M)
R=toeplitz (rxx);              % correlation matrix
ho=R\rxd;                      % solve for the Wiener filter coeff's
%
yn=filter(ho,1,xn);   % filter xn through ho
en=dn-yn;             % en is the cleaned dn here
%
% Plot results
subplot(3,1,1), plot(n,sn,'-b',n,dn,'-r')
subplot(3,1,2), plot(n,xn,'-b')
subplot(3,1,1), plot(n,sn,'-b',n,dn,'-r'),
axis([0 200 -5 5])
title(['Wiener Filter of Order M = ' num2str(M)])
subplot(3,1,2), plot(n,xn,'-b')
axis([0 200 -5 5])
subplot(3,1,3), plot(n,sn,'.b',n,en,'-r')
axis([0 200 -2 2])
subplot
```


Wiener Filter of Order M = 12
d(n)
s(n)
x(n)
e(n)
s(n)
n


Wiener Filter of Order M = 6
d(n)
s(n)
x(n)
e(n)
s(n)
n