Noise Cancellation Using Wiener Filtering

1) Objective:

The objective of the Wiener filter is to compute a statistical estimate of an unknown signal using a related signal as an input and filtering that known signal to produce the estimate as an output.

2) Abstract:

The Wiener filter is a filter used to produce an estimate of a desired or target random process by linear time-invariant (LTI) filtering of an observed noisy process, assuming known stationary signal and noise spectra, and additive noise. The Wiener filter minimizes the mean square error between the estimated random process and the desired process. The Wiener filter is based on a statistical approach, and a more statistical account of the theory is given in the minimum mean square error (MMSE) estimator article.

3) Introduction:

The Wiener filter can be used to filter out the noise from the corrupted signal to provide an estimate of the underlying signal of interest. The Wiener filter is based on a statistical approach, and a more statistical account of the theory is given in the minimum mean square error (MMSE) estimator article. Typical deterministic filters are designed for a desired frequency response. However, the design of the Wiener filter takes a different approach. One is assumed to have knowledge of the spectral properties of the original signal and the noise, and one seeks the linear time-invariant filter whose output would come as close to the original signal as possible. Wiener filters are characterized by the following. The assumptions include signal and (additive) noise are stationary linear stochastic processes with known spectral characteristics or known autocorrelation and cross-correlation.

4) Hardware and Software Required:

MATLAB

5) Concept or Working Principle:

Wiener filters are characterized by the following. The assumptions include signal and (additive) noise are stationary linear stochastic processes with known spectral characteristics or known autocorrelation and cross-correlation. The requirement: the filter must be physically realizable/causal.

Now we go for mathematical intuition for FIR Filters.

The causal finite impulse response (FIR) Wiener filter, instead of using some given data matrix X and output vector Y, finds optimal tap weights by using the statistics of the input and output signals. It populates the input matrix X with estimates of the autocorrelation of the input signal (T) and populates the output vector Y with estimates of the cross-correlation between the output and input signals (V).

In order to derive the coefficients of the Wiener filter, consider the signal $w[n]$ being fed to a Wiener filter of order (number of past taps) N and with coefficients {a0,a1,a2,...an}.

The output of the filter is denoted $x[n]$ which is given by the expression:

$$x[n] = \sum_{i=0}^{N} a_i w[n-i].$$

The residual error is denoted $e[n]$ and is defined as $e[n] = x[n] - s[n]$ (see the corresponding block diagram). The Wiener filter is designed so as to minimize the mean square error which can be stated concisely as follows:

$$a_i = \arg\min E\left[e^2[n]\right],$$

where {a0,a1,a2,...an} denotes the expectation operator. For simplicity, the following considers only the case where all these quantities are real. The mean square error (MSE) may be rewritten as:

$$
\begin{aligned}
E\left[e^2[n]\right] &= E\left[(x[n]-s[n])^2\right] \\
&= E\left[x^2[n]\right] + E\left[s^2[n]\right] - 2E[x[n]s[n]] \\
&= E\left[\left(\sum_{i=0}^{N} a_i w[n-i]\right)^2\right] + E\left[s^2[n]\right] - 2E\left[\sum_{i=0}^{N} a_i w[n-i]s[n]\right]
\end{aligned}
$$

To find the vector {a0,a1,a2,...an} which minimizes the expression above, calculate its derivative with respect to each term in the above vector

$$
\begin{aligned}
\frac{\partial}{\partial a_i} E\left[e^2[n]\right] &= \frac{\partial}{\partial a_i}\left\{E\left[\left(\sum_{i=0}^{N} a_i w[n-i]\right)^2\right] + E\left[s^2[n]\right] - 2E\left[\sum_{i=0}^{N} a_i w[n-i]s[n]\right]\right\} \\
&= 2E\left[\left(\sum_{j=0}^{N} a_j w[n-j]\right) w[n-i]\right] - 2E[w[n-i]s[n]] \\
&= 2\left(\sum_{j=0}^{N} E[w[n-j]w[n-i]]a_j\right) - 2E[w[n-i]s[n]]
\end{aligned}
$$

Assuming that w[n] and s[n] are each stationary and jointly stationary, the sequences Rw[m] and Rws[m]known respectively as the autocorrelation of w[n] and the cross-correlation between w[n] and s[n] can be defined as follows:

$$
\begin{aligned}
R_w[m] &= E\{w[n]w[n+m]\} \\
R_{ws}[m] &= E\{w[n]s[n+m]\}
\end{aligned}
$$

The derivative of the MSE may therefore be rewritten as:

$$
\frac{\partial}{\partial a_i} E\left[e^2[n]\right] = 2\left(\sum_{j=0}^{N} R_w[j-i]a_j\right) - 2R_{ws}[i] \qquad i = 0, \cdots, N.
$$

Note that for real w[n], the autocorrelation is symmetric:

$$
R_w[j-i] = R_w[i-j]
$$

Letting the derivative be equal to zero results in:

$$\sum_{j=0}^{N} R_w[j-i]a_j = R_{ws}[i] \qquad i = 0, \cdots, N.$$

which can be rewritten (using the above symmetric property) in matrix form

$$\underbrace{\begin{bmatrix} R_w[0] & R_w[1] & \cdots & R_w[N] \\ R_w[1] & R_w[0] & \cdots & R_w[N-1] \\ \vdots & \vdots & \ddots & \vdots \\ R_w[N] & R_w[N-1] & \cdots & R_w[0] \end{bmatrix}}_{\mathbf{T}} \underbrace{\begin{bmatrix} a_0 \\ a_1 \\ \vdots \\ a_N \end{bmatrix}}_{\mathbf{a}} = \underbrace{\begin{bmatrix} R_{ws}[0] \\ R_{ws}[1] \\ \vdots \\ R_{ws}[N] \end{bmatrix}}_{\mathbf{v}}$$

These equations are known as the Wiener–Hopf equations. The matrix T appearing in the equation is a symmetric Toeplitz matrix. Under suitable conditions on {\displaystyle R} R, these matrices are known to be positive definite and therefore non-singular yielding a unique solution to the determination of the Wiener filter coefficient vector, a=inv(T)*v. Furthermore, there exists an efficient algorithm to solve such Wiener–Hopf equations known as the Levinson-Durbin algorithm so an explicit inversion of T is not required.

Source: Wikipedia

In this problem, there are two sensors- primary and secondary. The primary sensor receives the signal s[n],corrupted with noise v1[n]. While the secondary sensor receives v2[n], a signal that is correlated with the noise v1[n]. The aim is to estimate s[n].

6) Program:

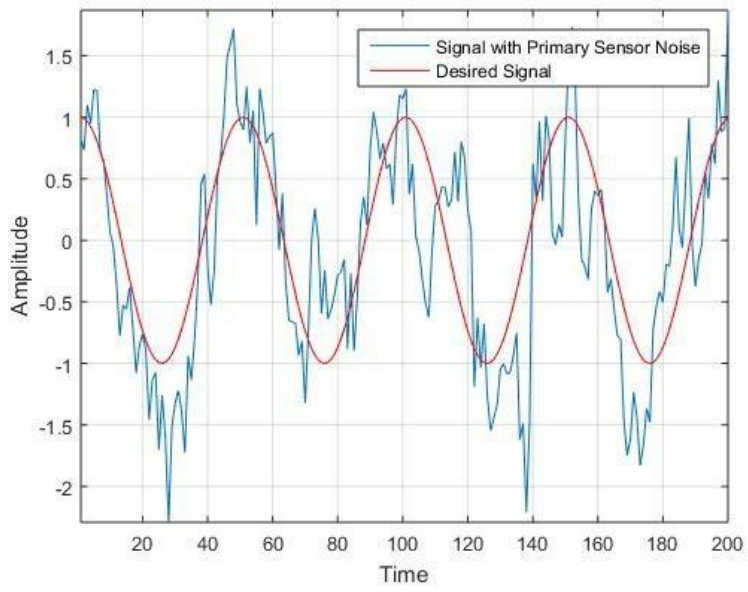
```
clear;
close all;

% Given wave characteristics

theta = 2*pi*rand-pi; %varying theta from -pi to pi
w0 = 0.04*pi;
N = 100000;
n = 1:N;

dn=sin(n*w0+theta);

vn=0.4*randn(1,N);

v1(1)=vn(1);
v2(1)=vn(1);
for i=2:N
v1(i)=0.8*v1(i-1)+vn(i); %primary sensor noise
v2(i)=-0.6*v2(i-1)+vn(i); %secondary sensor noise
%both noises are correlated
end

xn=dn+v1; % Recieved Signal

figure()
plot(1:200,xn(1:200));
hold on;
plot(1:200,dn(1:200),'r');
xlabel('Time'); ylabel('Amplitude'); grid on; axis tight;
legend('Signal with Primary Sensor Noise','Desired Signal');

figure();
plot(1:200,v2(1:200));
title('Plot of Secondary Sensor Noise');

rv2=AutoCorr(v2(1:200));
```

```
stem(rv2(20:39),'xr');
xlabel('Index');ylabel('Amplhitude');grid on; axis tight;
title('Autocorrelation of Secondary Sensor Noise Signal');

rxv2=CrossCorr(xn(1:200),v2(1:200));

figure();
stem(rxv2);
xlabel('Index');
title('Cross-correlation between Received Signal and Secondary Noise');grid on;
axis tight;

% Order 8 Wiener Filter

order = 8;
Rv2 = zeros(order,order);
for i = 1: order
count = i;
for j = 1 : order
Rv2(i,j) = rv2(count);
if (j<i)
count = count -1;
else
count = count +1;
end

end
end

wiener2 = Rv2\rxv2(1:order);

v1_est = conv(wiener2,v2);

d_est = xn - v1_est(1:N);

figure();
plot(d_est(1:200));
```

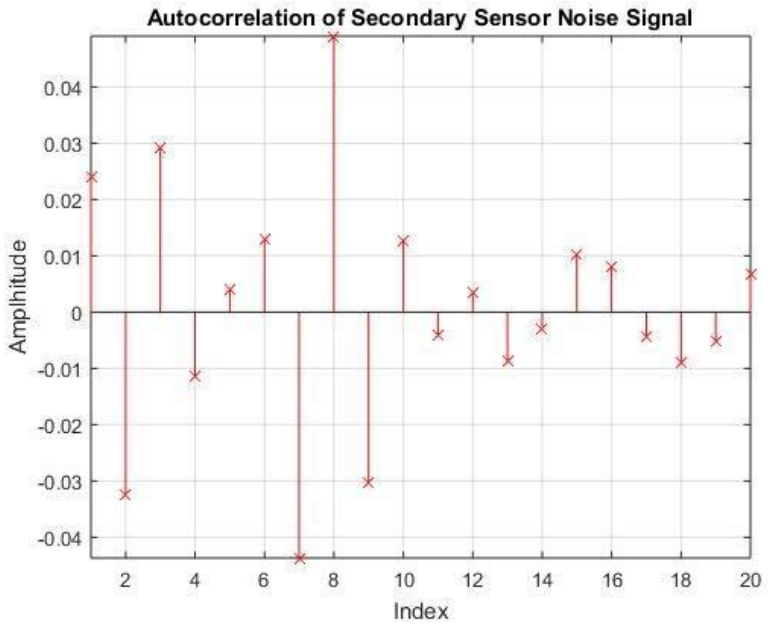
```
hold on
plot(dn(1:200),'r');
hold off;
title('Estimated Signal and Desired Signal for Order = 8');
xlabel('Time'); ylabel('Amplitude'); grid on; axis tight;
legend('Estimated Signal', 'Original Signal');
MSE_d_8 = mse(d_est,dn);
MSE_v1_8 = mse(v1_est,v1);
display(MSE_d_8);
display(MSE_v1_8);

% Order 64 Wiener Filter

order = 64;
Rv2 = zeros(order,order);
for i = 1: order
count = i;
for j = 1 : order
Rv2(i,j) = rv2(count);
if (j<i)
count = count -1;
else
count = count +1;
end

end
end

wiener2 = Rv2\rxv2(1:order);

v1_est = conv(wiener2,v2);

d_est = xn - v1_est(1:N);

figure();
```

```
plot(d_est(1:200));
hold on
plot(dn(1:200),'r');
hold off;
title('Estimated Signal and Desired Signal for order 64');
xlabel('Time'); ylabel('Amplitude'); grid on; axis tight;
legend('Estimated Signal', 'Original Signal');
MSE_d_64 = mse(d_est,dn);
MSE_v1_64 = mse(v1_est,v1);
display(MSE_d_64);
display(MSE_v1_64);
```

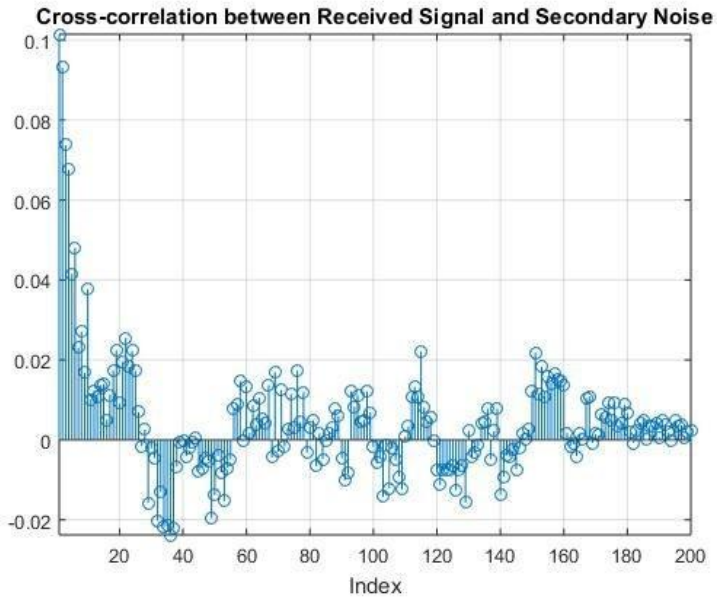
7)Output Results :



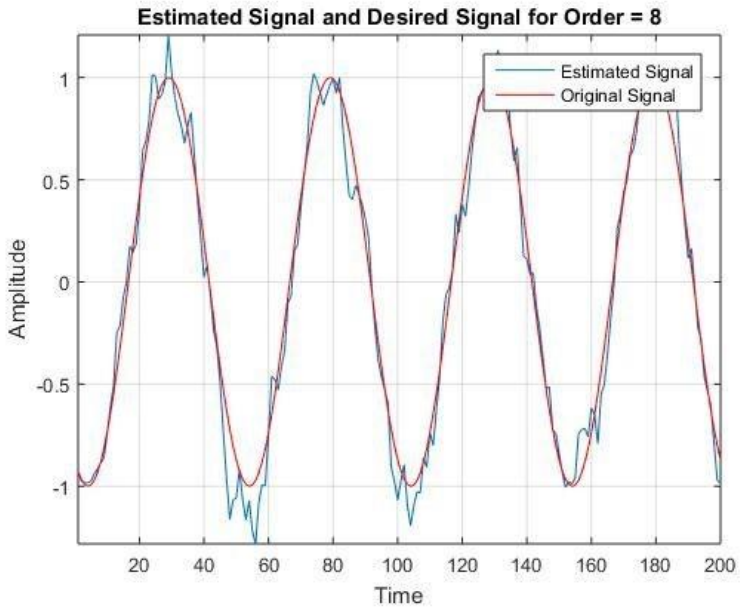
Signal With Noise and Original Signal



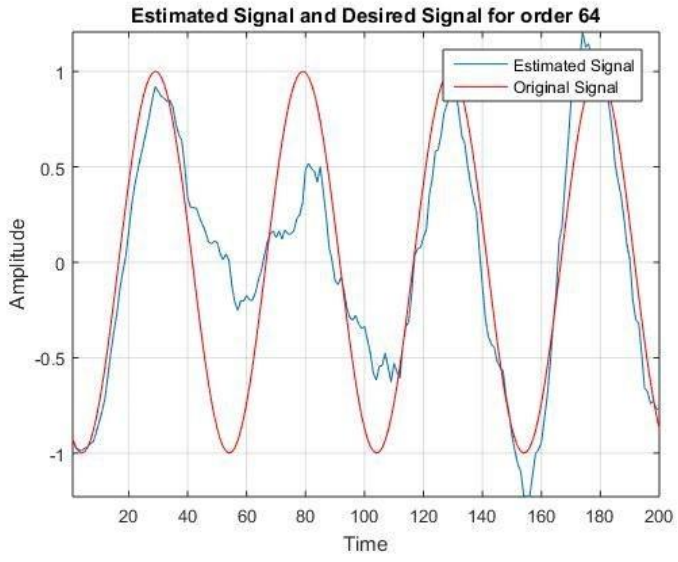
Autocorrelation Values for Secondary Noise Signal



Cross Correlation Values for Received Signal and Secondary Noise Signal



Estimated and Original Signal For Filter Order 8



Estimated and Original Signal For Filter Order 64

For Order 8:

MSE of the estimated signal: 0.0649

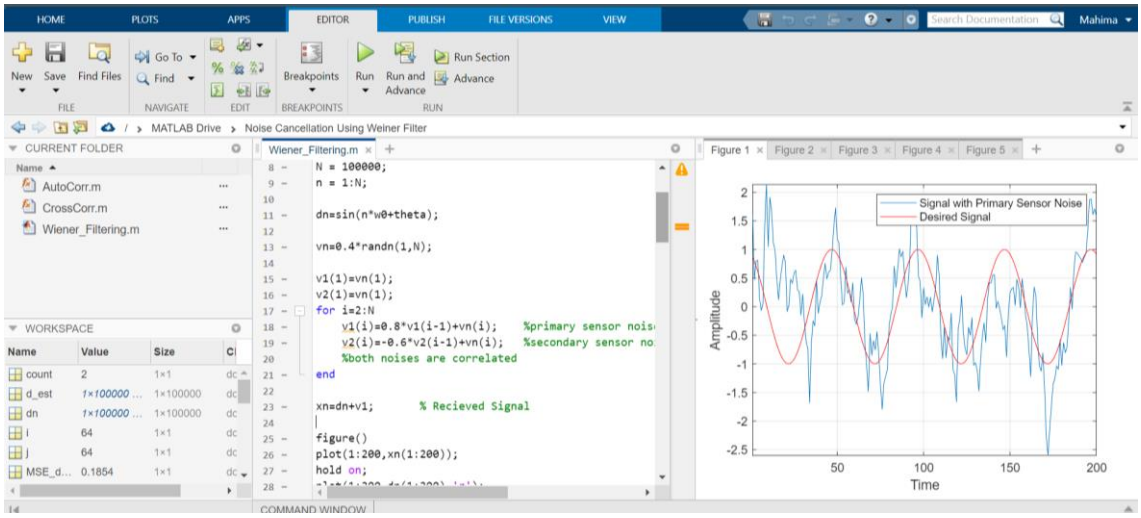
MSE of the estimated noise: 0.2198

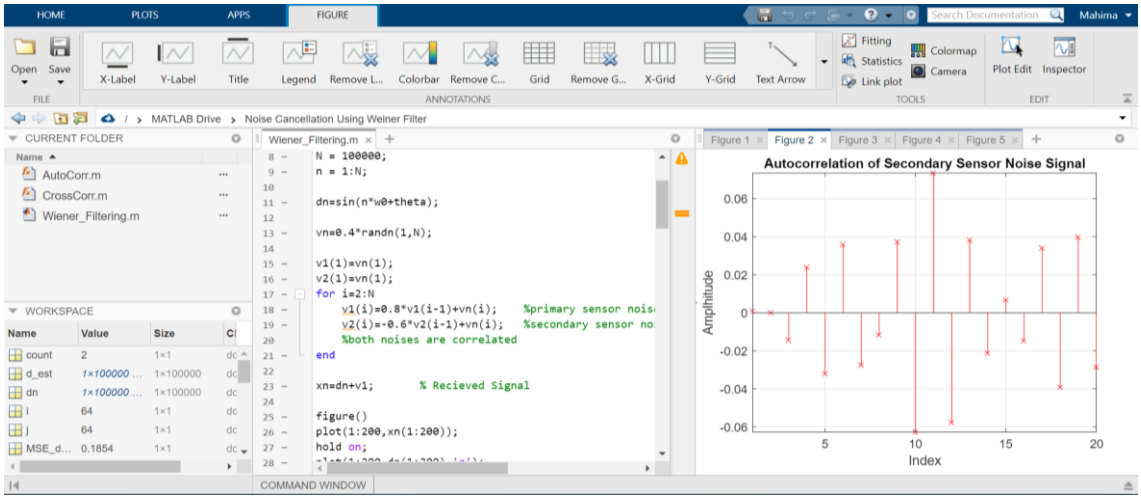
For Order 64:

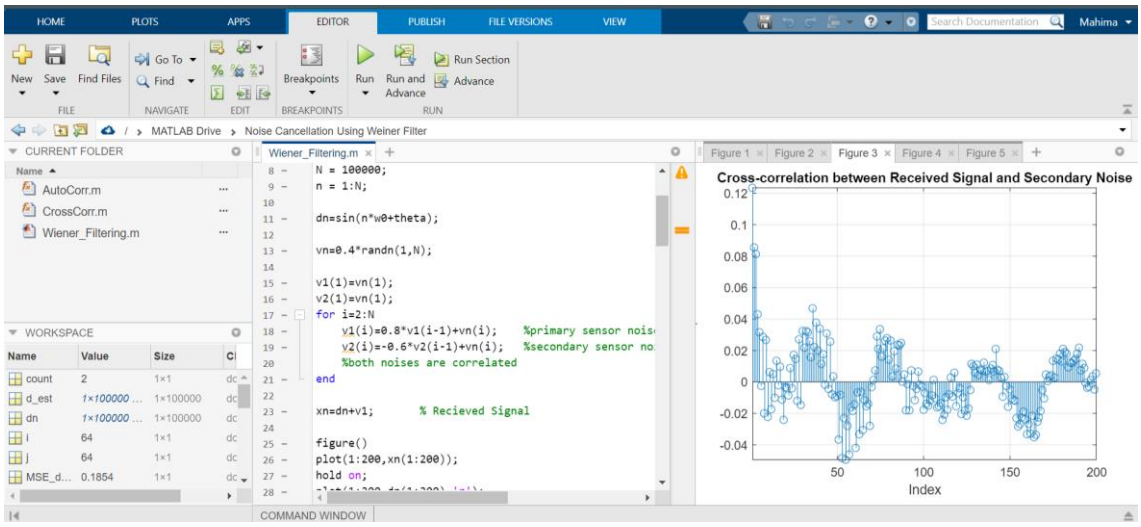
MSE of the estimated signal: 0.3748

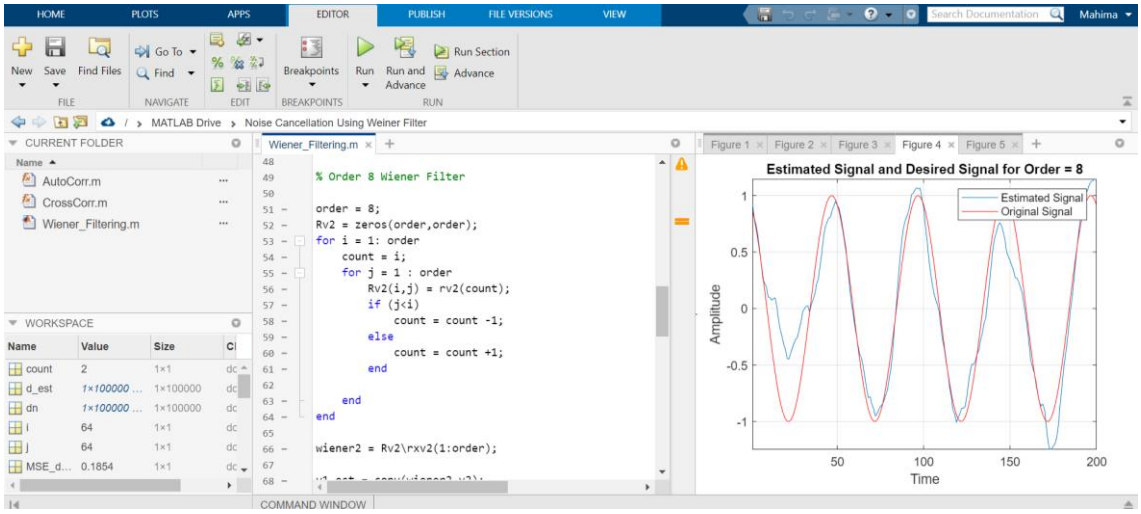
MSE of the estimated noise: 0.4705

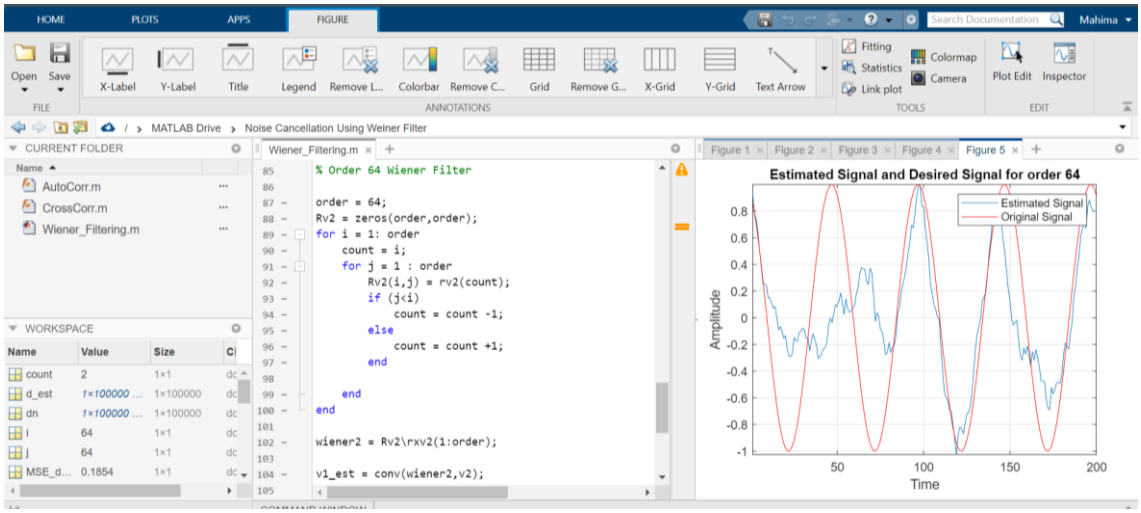
OUTPUT-SCREENSHOTS:






HOME
PLOTS
APPS
EDITOR
PUBLISH
FILE VERSIONS
VIEW
Search Documentation
Mahima
New
Save
Find Files
Go To
Find
Breakpoints
Run
Run and Advance
Run Section
Advance
FILE
NAVIGATE
EDIT
BREAKPOINTS
RUN
MATLAB Drive
Noise Cancellation Using Weiner Filter
CURRENT FOLDER
Name
AutoCorr.m
CrossCorr.m
Wiener_Filtering.m
WORKSPACE
Name
Value
Size
count 2 1×1
d_est 1×100000 ... 1×100000
dn 1×100000 ... 1×100000
i 64 1×1
j 64 1×1
MSE_d... 0.1854 1×1
Wiener_Filtering.m
N = 100000;
n = 1:N;
dn=sin(n*w0+theta);
vn=0.4*randn(1,N);
v1(1)=vn(1);
v2(1)=vn(1);
for i=2:N
v1(i)=0.8*v1(i-1)+vn(i);
v2(i)=-0.6*v2(i-1)+vn(i);
%both noises are correlated
end
xn=dn+v1;
% Recieved Signal
figure()
plot(1:200,xn(1:200));
hold on;
Figure 1
Figure 2
Figure 3
Figure 4
Figure 5
Cross-correlation between Received Signal and Secondary Noise
0.12
0.1
0.08
0.06
0.04
0.02
0
-0.02
-0.04
50
100
150
200
Index
COMMAND WINDOW


HOME
PLOTS
APPS
EDITOR
PUBLISH
FILE VERSIONS
VIEW
Search Documentation
Mahima
New
Save
Find Files
Go To
Find
Breakpoints
Run
Run and Advance
Run Section
Advance
FILE
NAVIGATE
EDIT
BREAKPOINTS
RUN
MATLAB Drive
Noise Cancellation Using Weiner Filter
CURRENT FOLDER
AutoCorr.m
CrossCorr.m
Wiener_Filtering.m
WORKSPACE
count 2 1×1
d_est 1×100000 ... 1×100000
dn 1×100000 ... 1×100000
i 64 1×1
j 64 1×1
MSE_d... 0.1854 1×1
Wiener_Filtering.m
% Order 8 Wiener Filter
order = 8;
Rv2 = zeros(order,order);
for i = 1: order
count = i;
for j = 1 : order
Rv2(i,j) = rv2(count);
if (j<i)
count = count -1;
else
count = count +1;
end
end
end
wiener2 = Rv2\rxv2(1:order);
Figure 1
Figure 2
Figure 3
Figure 4
Figure 5
Estimated Signal and Desired Signal for Order = 8
Estimated Signal
Original Signal
Amplitude
1
0.5
0
-0.5
-1
50
100
150
200
Time
COMMAND WINDOW


HOME
PLOTS
APPS
FIGURE
Search Documentation
Mahima
Open
Save
X-Label
Y-Label
Title
Legend
Remove L...
Colorbar
Remove C...
Grid
Remove G...
X-Grid
Y-Grid
Text Arrow
Fitting
Statistics
Link plot
Colormap
Camera
Plot Edit
Inspector
FILE
ANNOTATIONS
TOOLS
EDIT
MATLAB Drive
Noise Cancellation Using Weiner Filter
CURRENT FOLDER
AutoCorr.m
CrossCorr.m
Wiener_Filtering.m
WORKSPACE
count 2 1×1
d_est 1×100000 ... 1×100000
dn 1×100000 ... 1×100000
i 64 1×1
j 64 1×1
MSE_d... 0.1854 1×1
Wiener_Filtering.m
% Order 64 Wiener Filter
order = 64;
Rv2 = zeros(order,order);
for i = 1: order
count = i;
for j = 1 : order
Rv2(i,j) = rv2(count);
if (j<i)
count = count -1;
else
count = count +1;
end
end
end
wiener2 = Rv2\rxv2(1:order);
v1_est = conv(wiener2,v2);
Figure 1
Figure 2
Figure 3
Figure 4
Figure 5
Estimated Signal and Desired Signal for order 64
Estimated Signal
Original Signal
Amplitude
0.8
0.6
0.4
0.2
0
-0.2
-0.4
-0.6
-0.8
-1
50
100
150
200
Time

8) Conclusion:

For the purpose of wiener filtering, we require the process to be stationary and ergodic, as we use the values of sample cross-correlation. For this purpose, we have used a harmonic process as the desired signal. For noise cancellation, the primary sensor senses the channel output and the secondary sensor senses the ambient surroundings. The naïve method still gives us a decent result. This is as the constant ‘a’ is such that there is a strong correlation between v1 and v2.

9) Reference:

Source: Wikipedia