EECS 844 – Fall 2016

Exam 3 Cover page*

Each student is expected to complete the exam individually using only course notes, the book, and technical literature, and without aid from outside sources.

Aside from the most general conversation of the exam material, I assert that I have neither provided help nor accepted help from another student in completing this exam. As such, the work herein is mine and mine alone.

_____ _____

Signature Date

MY SOLUTIONS

_____ _____

Name (printed) Student ID #

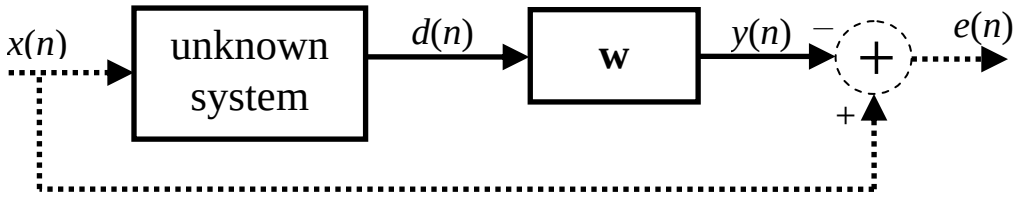
*** Attach as cover page to completed exam.**

1. In P1.mat the output of a hypothesized unknown linear system is the observed signal $d(n)$, to which we shall apply a whitening filter via linear prediction (these are the solid lines in the figure). In contrast, the dashed lines represent the inverse filtering formulation that we could perform if the hypothesized "driving signal" $x(n)$ were actually known (in practice it is not).

a) Determine **w** as the linear prediction error (LPE) filter using only $d(n)$ when the linear prediction component is $M = 29$ (so LPE filter is length 30).

b) Determine **w** as the inverse system identification Wiener filter using both $d(n)$ and $x(n)$ for $M = 30$.

c) Plot the time-domain magnitude/phase and frequency response of each filter. Comment on how they are related.



Solution:

Figure 1.1 contains the magnitude (in dB) and phase of the LPE filter $\mathbf{w}_{\text{LPE}} = [1 \;\; -\mathbf{w}_{\text{LP}}^T]^T$, where $\mathbf{w}_{\text{LP}}$ is the linear prediction filter, and the inverse system identification version of the Wiener filter. We observe that for the large coefficient values (the first few) the magnitude and phase of these two filters are nearly identical. As a result, the frequency responses shown in Fig. 1.2 are also nearly identical.

While it may not be obvious from these solutions, the Wiener filter implementation is more accurate because it uses both the input $x(n)$ and the output $d(n)$ of the unknown system. However, for many applications only $d(n)$ is available (e.g. speech).

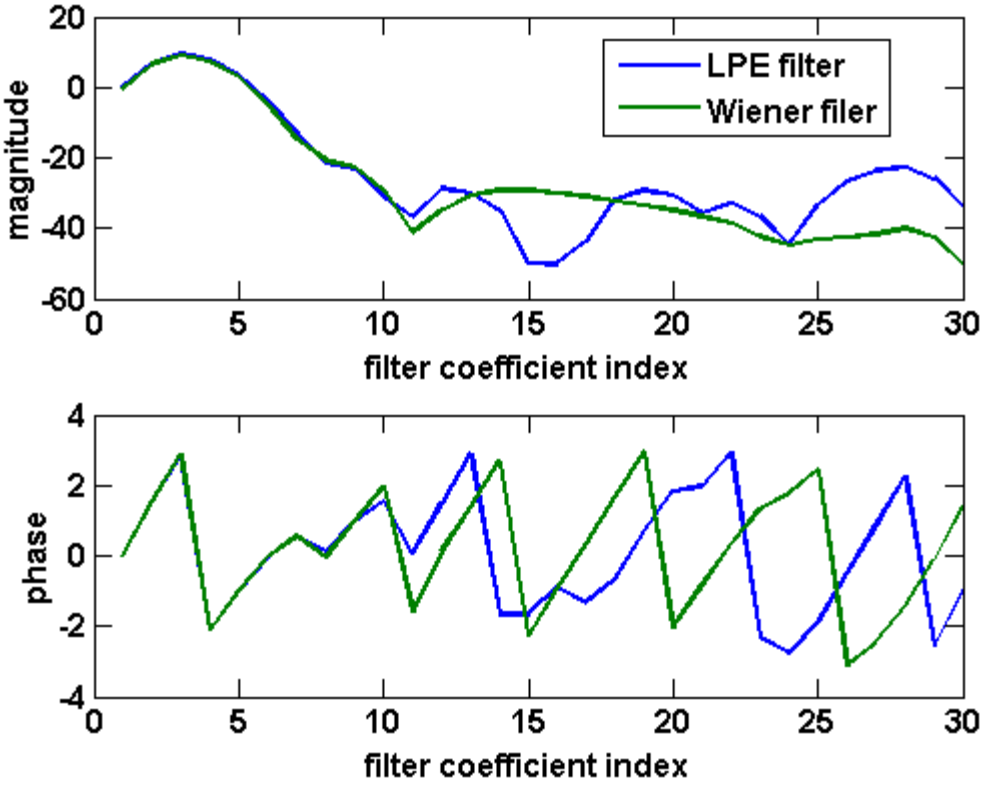


Figure 1.1. Magnitude and phase of the LPE filter and inverse system ID Wiener filter

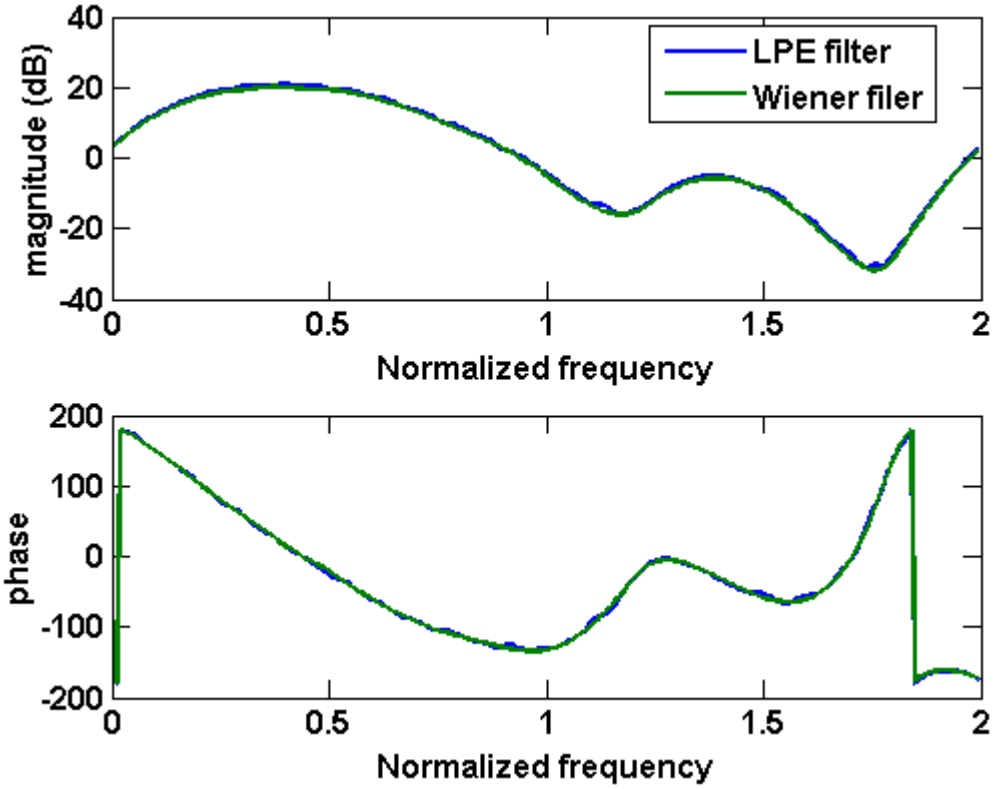


Figure 1.2. Frequency response of the LPE filter and the inverse system ID Wiener filter

Matlab Code for Problem 1

```
clear all
load P1
N = length(x);

M = 30;
MLP = M-1;

%% linear prediction
DLP = zeros(MLP,N-MLP+1);
for jm = 1:MLP
    DLP(MLP-jm+1,:) = d(jm:jm+N-MLP).';
end;
DLP2 = DLP(:,1:N-MLP);
RLP_d = (1./(N-(MLP+1)+1)).*DLP*DLP';
DLP_d = repmat(d(MLP+1:N)',MLP,1);
pLP_dd = sum(DLP2.*DLP_d,2)./(N-MLP);
LP_opt = [1; -inv(RLP_d)*pLP_dd];

% Wiener filter
D = zeros(M,N-M+1);
for jm = 1:M
    D(M-jm+1,:) = d(jm:jm+N-M).';
end;
R_d = (1./(N-M+1)).*D*D';
X_x = repmat(x(M:N)',M,1);
p_dx = sum(D.*X_x ,2)./(N-M+1);
WF_opt = [inv(R_d)*p_dx];

figure(1)
subplot(2,1,1)
plot(1:M,20*log10(abs(LP_opt)),1:M,20*log10(abs(WF_opt)));
ylabel('magnitude')
xlabel('filter coefficient index')
legend('LPE filter','Wiener filer')
subplot(2,1,2)
plot(1:M,angle(LP_opt),1:M,angle(WF_opt));
ylabel('phase')
xlabel('filter coefficient index')

[Hwf,Wwf] = freqz(WF_opt,1,10*M,'whole');
[Hlp,Wlp] = freqz(LP_opt,1,10*M,'whole');

figure(2)
subplot(2,1,1)
plot(Wlp./pi,20*log10(abs(Hlp)),Wwf./pi,20*log10(abs(Hwf)))
xlabel('Normalized frequency')
ylabel('magnitude (dB)')
legend('LPE filter','Wiener filer')
subplot(2,1,2)
plot(Wlp./pi,(180/pi).*angle(Hlp),Wwf./pi,(180/pi).*angle(Hwf))
xlabel('Normalized frequency')
ylabel('phase')
```

2. Derive and implement the steepest-descent algorithm for the cost function

$$J(\mathbf{w}) = \mathbf{w}^H \mathbf{R}\, \mathbf{w} + \left|\mathbf{w}^H \mathbf{s}(\omega_1) - 1\right|^2 + \left|\mathbf{w}^H \mathbf{s}(\omega_2) - 0.5\right|^2$$

where $\mathbf{w}$ is a time-domain filter, $\mathbf{R}$ is a correlation matrix, and $\mathbf{s}(\omega_1)$ and $\mathbf{s}(\omega_2)$ are steering vectors for particular frequencies. For $N = 40$, use the $N \times N$ correlation matrix and $N \times 1$ steering vectors provided in P2.mat and initializing the $N \times 1$ filter $\mathbf{w}(0)$ with all zeros, generate 2000 steepest-descent iterations using step-sizes of $\mu = 1/(10N)$, $\mu = 1/(50N)$, and $\mu = 1/(300N)$. Plot the convergence curve ($J(\mathbf{w}(n))$ vs. n) for each case. For each of these three cases, plot the magnitude frequency responses (in dB) of $\mathbf{w}(n=20)$, $\mathbf{w}(n=200)$, and $\mathbf{w}(n=2000)$. Discuss what you observe.

Solution:

Expand the cost function as

$$J(\mathbf{w}) = \mathbf{w}^H \mathbf{R}\, \mathbf{w} + \left(\mathbf{w}^H \mathbf{s}(\omega_1) - 1\right)\left(\mathbf{w}^H \mathbf{s}(\omega_1) - 1\right)^* + \left(\mathbf{w}^H \mathbf{s}(\omega_2) - 0.5\right)\left(\mathbf{w}^H \mathbf{s}(\omega_2) - 0.5\right)^*$$

and then determine the gradient of the cost function as

$$\begin{aligned} \nabla J(\mathbf{w}) &= 2\mathbf{R}\mathbf{w} + 2\mathbf{s}(\omega_1)\mathbf{s}^H(\omega_1)\mathbf{w} - 2\mathbf{s}(\omega_1) + 2\mathbf{s}(\omega_2)\mathbf{s}^H(\omega_2)\mathbf{w} - 2(0.5)\mathbf{s}(\omega_2) \\ &= 2[\mathbf{R} + \mathbf{s}(\omega_1)\mathbf{s}^H(\omega_1) + \mathbf{s}(\omega_2)\mathbf{s}^H(\omega_2)]\mathbf{w} - 2[\mathbf{s}(\omega_1) + (0.5)\mathbf{s}(\omega_2)] \\ &\doteq \mathbf{g} \end{aligned} \quad .$$

The steepest-descent update, as a function of discrete-time index n, is thus

$$\begin{aligned} \mathbf{w}(n+1) &= \mathbf{w}(n) - 0.5\mu\mathbf{g}(n) \\ &= \mathbf{w}(n) - \mu\left[\left(\mathbf{R} + \mathbf{s}(\omega_1)\mathbf{s}^H(\omega_1) + \mathbf{s}(\omega_2)\mathbf{s}^H(\omega_2)\right)\mathbf{w}(n) - \left(\mathbf{s}(\omega_1) + (0.5)\mathbf{s}(\omega_2)\right)\right]. \end{aligned}$$

Figure 2.1 plots the convergence of this cost function versus iteration index n for the three different step-sizes. As expected, the smaller the step-size, the longer the algorithm requires to converge. For these steering vectors and correlation matrix, if a larger step-size is used (e.g. 1/(9N)) then we would observe that it diverges.

It is interesting to examine the frequency responses of the filters at different stages of convergence. After 20 iterations (Fig. 2.2) the filter using the largest step-size (1/10N) is shows the much higher peaks corresponding to the steering vector constraints. After 200 iterations (Fig. 2.3) we find that the middle step-size (1/50N) has largely caught up, though it does exhibit lower surrounding sidelobes. Finally, after 2000 iterations (Fig. 2.4) it appears that all three filters have converged, with the smallest step-size case (1/300N) realizing the lowest sidelobes. Thus, here we find an example of slower convergence also eventually resulting in a better solution.

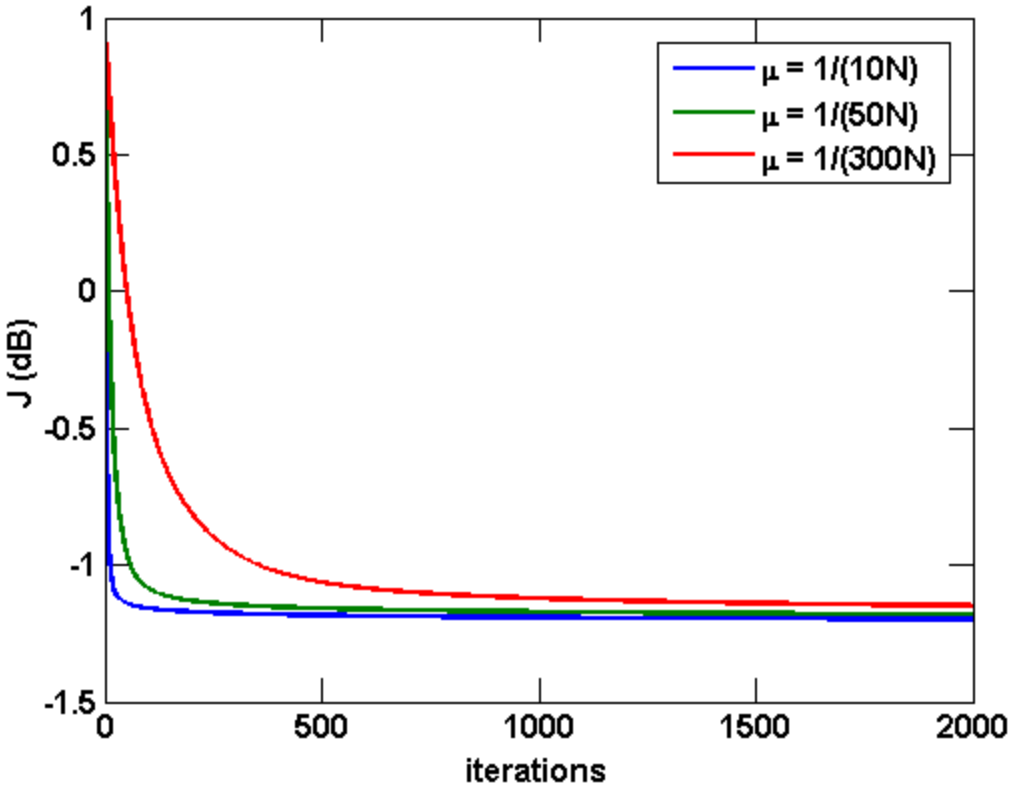


Figure 2.1. Convergence of cost function versus iteration index

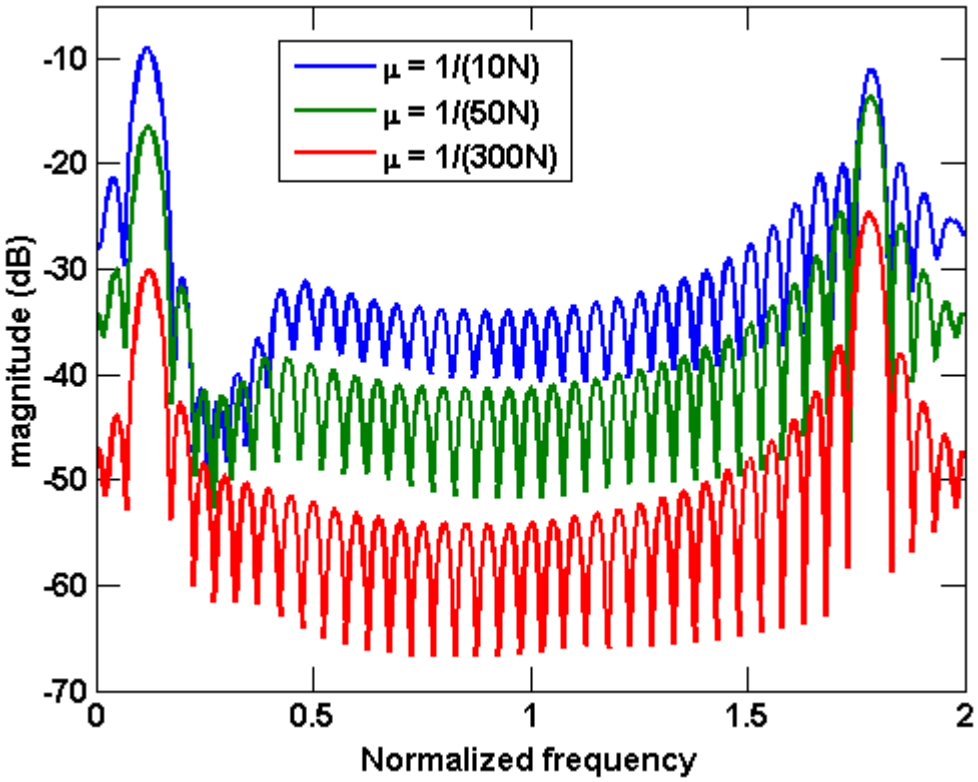


Figure 2.2. Frequency response of filters after 20 iterations

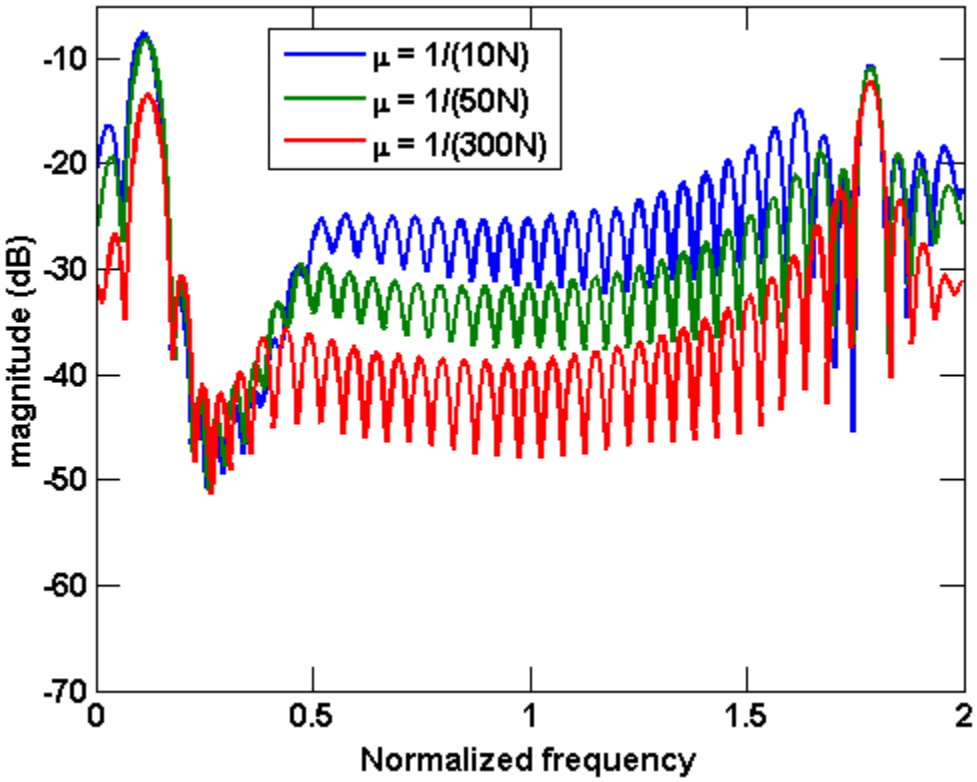


Figure 2.3. Frequency response of filters after 200 iterations

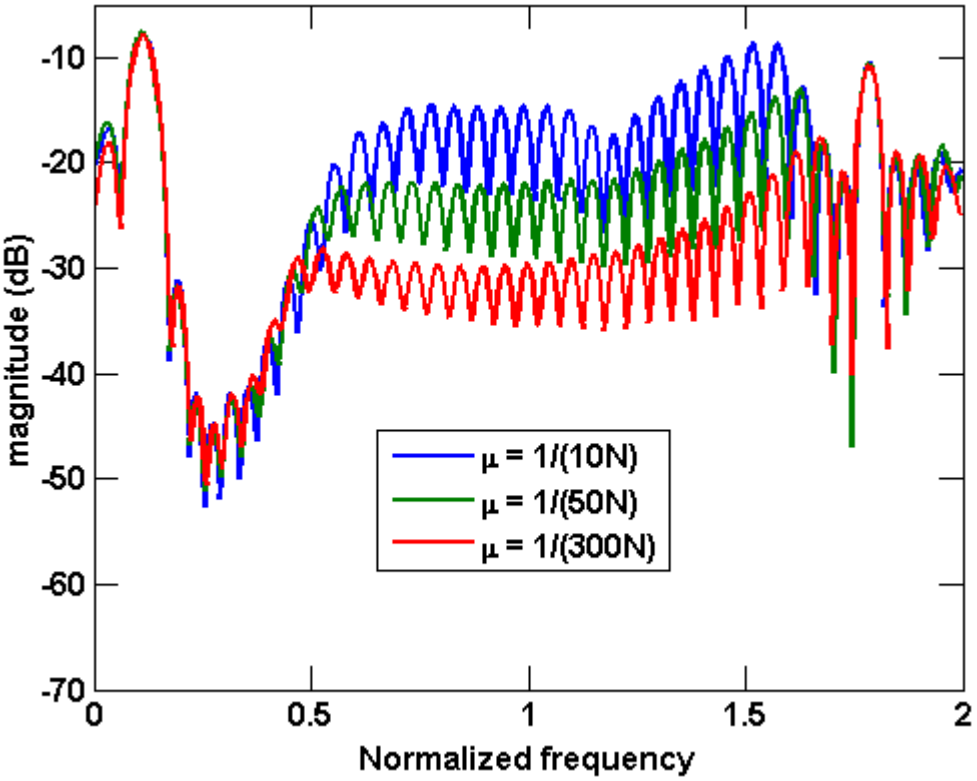


Figure 2.4. Frequency response of filters after 2000 iterations

Matlab Code for Problem 2

```
clear all
load P3
N = length(s1);
iterations = 2000;

w = zeros(N,1);
Iss = R + s1*s1' + s2*s2';
s12 = s1 + 0.5.*s2;
J0 = w'*R*w + abs(w'*s1 - 1)^2 + abs(w'*s2 - 0.5)^2;

%%% mu = 1/(10N)
w_out10 = zeros(N,iterations);
w10 = zeros(N,1);
mu10 = 1/(10*N);
for jj = 1:iterations
    g10 = 2.*(Iss*w10 - s12);
    w10 = w10 - 0.5.*mu10.*g10;
    w_out10(:,jj) = w10;
    J10(jj) = w10'*R*w10 + abs(w10'*s1 - 1)^2  + abs(w10'*s2 - 0.5) ^2;
end;

%%% mu = 1/(50N)
w_out50 = zeros(N,iterations);
w50 = zeros(N,1);
mu50 = 1/(50*N);
for jj = 1:iterations
    g50 = 2.*(Iss*w50 - s12);
    w50 = w50 - 0.5.*mu50.*g50;
    w_out50(:,jj) = w50;
    J50(jj) = w50'*R*w50 + abs(w50'*s1 - 1)^2 + abs(w50'*s2 - 0.5)^2;
end;

%%% mu = 1/(300N)
w_out300 = zeros(N,iterations);
w300 = zeros(N,1);
mu300 = 1/(300*N);
for jj = 1:iterations
    g300 = 2.*(Iss*w300 - s12);
    w300 = w300 - 0.5.*mu300.*g300;
    w_out300(:,jj) = w300;
    J300(jj) = w300'*R*w300 + abs(w300'*s1 - 1)^2 ...
      + abs(w300'*s2 -   0.5)^2;
end;

figure(1)
plot(0:iterations,10*log10([J0 J10]),0:iterations,10*log10([J0
J50]),0:iterations,10*log10([J0 J300]))
xlabel('iterations')
ylabel('J (dB)')
legend('\mu = 1/(10N)','\mu = 1/(50N)','\mu = 1/(300N)')

[Hf10_20,Wf10_20] = freqz(w_out10(:,20),1,10*N,'whole');
[Hf10_200,Wf10_200] = freqz(w_out10(:,200),1,10*N,'whole');
[Hf10_2000,Wf10_2000] = freqz(w_out10(:,2000),1,10*N,'whole');
```

```
[Hf50_20,Wf50_20] = freqz(w_out50(:,20),1,10*N,'whole');
[Hf50_200,Wf50_200] = freqz(w_out50(:,200),1,10*N,'whole');
[Hf50_2000,Wf50_2000] = freqz(w_out50(:,2000),1,10*N,'whole');
[Hf300_20,Wf300_20] = freqz(w_out300(:,20),1,10*N,'whole');
[Hf300_200,Wf300_200] = freqz(w_out300(:,200),1,10*N,'whole');
[Hf300_2000,Wf300_2000] = freqz(w_out300(:,2000),1,10*N,'whole');

figure(2)
plot(Wf10_20./pi,20*log10(abs(Hf10_20)),Wf50_20./pi,20*log10(abs(Hf50_2
0)),Wf300_20./pi,20*log10(abs(Hf300_20)))
xlabel('Normalized frequency')
ylabel('magnitude (dB)')
legend('\mu = 1/(10N)','\mu = 1/(50N)','\mu = 1/(300N)')
axis([0 2 -70 -5])

figure(3)
plot(Wf10_200./pi,20*log10(abs(Hf10_200)),Wf50_200./pi,20*log10(abs(Hf5
0_200)),Wf300_200./pi,20*log10(abs(Hf300_200)))
xlabel('Normalized frequency')
ylabel('magnitude (dB)')
legend('\mu = 1/(10N)','\mu = 1/(50N)','\mu = 1/(300N)')
axis([0 2 -70 -5])

figure(4)
plot(Wf10_2000./pi,20*log10(abs(Hf10_2000)),Wf50_2000./pi,20*log10(abs(
Hf50_2000)),Wf300_2000./pi,20*log10(abs(Hf300_2000)))
xlabel('Normalized frequency')
ylabel('magnitude (dB)')
legend('\mu = 1/(10N)','\mu = 1/(50N)','\mu = 1/(300N)')
axis([0 2 -70 -5])
```

3. Using the same notation developed in class, analytically show that the GSC, given the constraint matrix $\mathbf{C} = \mathbf{s}(\omega_o)$ with associated constraint gain $g = 1$, is equivalent to the MVDR beamformer.

Hints: a) See Section II of the Breed & Strauss paper and Section II (the two paragraphs above equation 2) of the Apolinario, *et al* paper.
b) For matrix **A** full-rank and square, pseudo-inverse(**A**) = inverse(**A**) where pseudo-inverse(**A**) = $(\mathbf{A}^H\mathbf{A})^{-1}\mathbf{A}^H$ by Least-Squares.
c) Use the identity for the inverse of a partitioned matrix.

Solution:

For GSC we have

$$\begin{aligned}\mathbf{w} &= \mathbf{w}_q - \mathbf{C}_a\mathbf{w}_{ao} \\ &= \mathbf{C}\left(\mathbf{C}^H\mathbf{C}\right)^{-1}\mathbf{g} - \mathbf{C}_a\left(\mathbf{C}_a^H\mathbf{R}\mathbf{C}_a\right)^{-1}\mathbf{C}_a^H\mathbf{R}\mathbf{C}\left(\mathbf{C}^H\mathbf{C}\right)^{-1}\mathbf{g}\end{aligned}$$

which, for $\mathbf{C} = \mathbf{s}(\theta)$ and $\mathbf{g} = 1$, simplifies to

$$\mathbf{w} = \frac{1}{M}\left[\mathbf{I} - \mathbf{C}_a\left(\mathbf{C}_a^H\mathbf{R}\mathbf{C}_a\right)^{-1}\mathbf{C}_a^H\mathbf{R}\right]\mathbf{s}(\theta) \tag{1}$$

where we note that the term in the bracket looks similar to a projection. For the partitioned matrix

$$\mathbf{U} = \left[\mathbf{C} \vdots \mathbf{C}_a\right]$$

with $\mathbf{C}$ and $\mathbf{C}_a$ the orthogonal complement of one another, we can express the inverse of $\mathbf{U}$ as

$$\begin{aligned}\mathbf{U}^{-1} &= \left(\mathbf{U}^H\mathbf{U}\right)^{-1}\mathbf{U}^H \\ &= \left(\begin{bmatrix}\mathbf{C}^H \\ \cdots \\ \mathbf{C}_a^H\end{bmatrix}\left[\mathbf{C} \vdots \mathbf{C}_a\right]\right)^{-1}\begin{bmatrix}\mathbf{C}^H \\ \cdots \\ \mathbf{C}_a^H\end{bmatrix} \\ &= \left(\begin{bmatrix}\mathbf{C}^H\mathbf{C} & \mathbf{C}^H\mathbf{C}_a \\ \mathbf{C}_a^H\mathbf{C} & \mathbf{C}_a^H\mathbf{C}_a\end{bmatrix}\right)^{-1}\begin{bmatrix}\mathbf{C}^H \\ \cdots \\ \mathbf{C}_a^H\end{bmatrix} \\ &= \left(\begin{bmatrix}\mathbf{C}^H\mathbf{C} & \mathbf{0} \\ \mathbf{0} & \mathbf{C}_a^H\mathbf{C}_a\end{bmatrix}\right)^{-1}\begin{bmatrix}\mathbf{C}^H \\ \cdots \\ \mathbf{C}_a^H\end{bmatrix}\end{aligned}$$

where the last line is due to the orthogonal complement relationship. The last line also shows that the matrix to be inverted is block diagonal. The inverse of a block diagonal matrix is the inverse of the individual blocks as

$$\mathbf{U}^{-1} = \left(\begin{bmatrix} \mathbf{C}^H\mathbf{C} & \mathbf{0} \\ \mathbf{0} & \mathbf{C}_\mathrm{a}^H\mathbf{C}_\mathrm{a} \end{bmatrix} \right)^{-1} \begin{bmatrix} \mathbf{C}^H \\ \cdots \\ \mathbf{C}_\mathrm{a}^H \end{bmatrix}$$
$$= \begin{pmatrix} [\mathbf{C}^H\mathbf{C}]^{-1} & \mathbf{0} \\ \mathbf{0} & [\mathbf{C}_\mathrm{a}^H\mathbf{C}_\mathrm{a}]^{-1} \end{pmatrix} \begin{bmatrix} \mathbf{C}^H \\ \cdots \\ \mathbf{C}_\mathrm{a}^H \end{bmatrix}.$$

Now applying $\mathbf{U}\mathbf{U}^{-1} = \mathbf{I}$ we obtain

$$\mathbf{U}\mathbf{U}^{-1} = \begin{bmatrix} \mathbf{C} \vdots \mathbf{C}_\mathrm{a} \end{bmatrix} \begin{pmatrix} [\mathbf{C}^H\mathbf{C}]^{-1} & \mathbf{0} \\ \mathbf{0} & [\mathbf{C}_\mathrm{a}^H\mathbf{C}_\mathrm{a}]^{-1} \end{pmatrix} \begin{bmatrix} \mathbf{C}^H \\ \cdots \\ \mathbf{C}_\mathrm{a}^H \end{bmatrix} = \mathbf{I}$$

which simplifies as

$$\mathbf{C}[\mathbf{C}^H\mathbf{C}]^{-1}\mathbf{C}^H + \mathbf{C}_\mathrm{a}[\mathbf{C}_\mathrm{a}^H\mathbf{C}_\mathrm{a}]^{-1}\mathbf{C}_\mathrm{a}^H = \mathbf{I}$$

and can be rearranged as the projection

$$\mathbf{I} - \mathbf{C}_\mathrm{a}[\mathbf{C}_\mathrm{a}^H\mathbf{C}_\mathrm{a}]^{-1}\mathbf{C}_\mathrm{a}^H = \mathbf{C}[\mathbf{C}^H\mathbf{C}]^{-1}\mathbf{C}^H. \qquad (2)$$

Based on the approach in the Breed & Strauss paper (beginning of Section II), now define

$$\mathbf{D} = \mathbf{R}^{-1/2}\mathbf{C}$$

and

$$\mathbf{D}_\mathrm{a} = \mathbf{R}^{+1/2}\mathbf{C}_\mathrm{a}$$

so that

$$\mathbf{D}^H\mathbf{D}_\mathrm{a} = \mathbf{C}^H\mathbf{R}^{-1/2}\mathbf{R}^{+1/2}\mathbf{C}_\mathrm{a} = \mathbf{C}^H\mathbf{I}\mathbf{C}_\mathrm{a} = \mathbf{C}^H\mathbf{C}_\mathrm{a} = \mathbf{0}$$

where we have assumed that $\mathbf{R}^{-1/2}$ is a Hermitian matrix (which it would be if $\mathbf{R}$ is Hermitian). Because $\mathbf{D}$ and $\mathbf{D}_\mathrm{a}$ are likewise shown to be orthogonal complements, we can write (2) as

$$\mathbf{I} - \mathbf{D}_\mathrm{a}[\mathbf{D}_\mathrm{a}^H\mathbf{D}_\mathrm{a}]^{-1}\mathbf{D}_\mathrm{a}^H = \mathbf{D}[\mathbf{D}^H\mathbf{D}]^{-1}\mathbf{D}^H$$

and subsequently substitute in for $\mathbf{D}$ and $\mathbf{D}_{\mathrm{a}}$ as

$$\mathbf{I}-\mathbf{R}^{+1/2}\mathbf{C}_{\mathrm{a}}[\mathbf{C}_{\mathrm{a}}^{H}\mathbf{R}\mathbf{C}_{\mathrm{a}}]^{-1}\mathbf{C}_{\mathrm{a}}^{H}\mathbf{R}^{+1/2}=\mathbf{R}^{-1/2}\mathbf{C}[\mathbf{C}^{H}\mathbf{R}^{-1}\mathbf{C}]^{-1}\mathbf{C}^{H}\mathbf{R}^{-1/2}. \tag{3}$$

Now rearrange (1) as

$$\begin{aligned}\mathbf{w} &= \frac{1}{M}\left[\mathbf{I}-\mathbf{C}_{\mathrm{a}}\left(\mathbf{C}_{\mathrm{a}}^{H}\mathbf{R}\mathbf{C}_{\mathrm{a}}\right)^{-1}\mathbf{C}_{\mathrm{a}}^{H}\mathbf{R}\right]\mathbf{s}(\theta)\\ &= \frac{1}{M}\left[\mathbf{R}^{-1/2}\mathbf{R}^{+1/2}-\mathbf{R}^{-1/2}\mathbf{R}^{+1/2}\mathbf{C}_{\mathrm{a}}\left(\mathbf{C}_{\mathrm{a}}^{H}\mathbf{R}\mathbf{C}_{\mathrm{a}}\right)^{-1}\mathbf{C}_{\mathrm{a}}^{H}\mathbf{R}^{+1/2}\mathbf{R}^{+1/2}\right]\mathbf{s}(\theta)\\ &= \frac{1}{M}\mathbf{R}^{-1/2}\left[\mathbf{I}-\mathbf{R}^{+1/2}\mathbf{C}_{\mathrm{a}}\left(\mathbf{C}_{\mathrm{a}}^{H}\mathbf{R}\mathbf{C}_{\mathrm{a}}\right)^{-1}\mathbf{C}_{\mathrm{a}}^{H}\mathbf{R}^{+1/2}\right]\mathbf{R}^{+1/2}\mathbf{s}(\theta)\end{aligned}$$

which, upon substituting (3) into the term in the brackets in the last step, yields

$$\begin{aligned}\mathbf{w} &= \frac{1}{M}\mathbf{R}^{-1/2}\left[\mathbf{I}-\mathbf{R}^{+1/2}\mathbf{C}_{\mathrm{a}}\left(\mathbf{C}_{\mathrm{a}}^{H}\mathbf{R}\mathbf{C}_{\mathrm{a}}\right)^{-1}\mathbf{C}_{\mathrm{a}}^{H}\mathbf{R}^{+1/2}\right]\mathbf{R}^{+1/2}\mathbf{s}(\theta)\\ &= \frac{1}{M}\mathbf{R}^{-1/2}\left[\mathbf{R}^{-1/2}\mathbf{C}\left(\mathbf{C}^{H}\mathbf{R}^{-1}\mathbf{C}\right)^{-1}\mathbf{C}^{H}\mathbf{R}^{-1/2}\right]\mathbf{R}^{+1/2}\mathbf{s}(\theta)\\ &= \frac{1}{M}\mathbf{R}^{-1}\mathbf{C}\left(\mathbf{C}^{H}\mathbf{R}^{-1}\mathbf{C}\right)^{-1}\mathbf{C}^{H}\mathbf{s}(\theta)\end{aligned}$$

with the last resulting from correlation matrix terms either cancelling or combining. Since we have $\mathbf{C}=\mathbf{s}(\theta)$, the above result simplifies to

$$\begin{aligned}\mathbf{w} &= \frac{1}{M}\mathbf{R}^{-1}\mathbf{C}\left(\mathbf{C}^{H}\mathbf{R}^{-1}\mathbf{C}\right)^{-1}\mathbf{C}^{H}\mathbf{s}(\theta)\\ &= \frac{1}{M}\mathbf{R}^{-1}\mathbf{s}(\theta)\left(\mathbf{s}^{H}(\theta)\mathbf{R}^{-1}\mathbf{s}(\theta)\right)^{-1}\mathbf{s}^{H}(\theta)\,\mathbf{s}(\theta)\\ &= \frac{\mathbf{R}^{-1}\mathbf{s}(\theta)}{\left(\mathbf{s}^{H}(\theta)\mathbf{R}^{-1}\mathbf{s}(\theta)\right)}\end{aligned}$$

because $\mathbf{s}^{H}(\theta)\,\mathbf{s}(\theta)=M$. The final result is the MVDR filter.

4. Data set P4.mat contains a known signal **x** of length M = 100 that we wish to deconvolve from other unknown signals.

a) Generate the *normalized matched filter* as $\mathbf{h}_{\text{NMF}} = \mathbf{x}^{B*}/(\mathbf{x}^H\mathbf{x})$ and plot the convolution of this filter with the signal **x**. Note that a filter output is an amplitude when plotting in dB.

b) Implement the Least-Squares *mismatched filter* $\mathbf{h}_{\text{MMF}}$ with a length of 3M and place the '1' in the elementary vector in element m = 2M. Once determined, normalize the MMF as

$$\mathbf{h}_{\text{NMMF}} = \frac{\mathbf{h}_{\text{MMF}}}{(\mathbf{h}_{\text{MMF}}^H \mathbf{h}_{\text{MMF}})^{1/2}(\mathbf{x}^H\mathbf{x})^{1/2}}$$

to allow for determination of the loss in SNR. Plot (in dB) the convolution of the normalized MMF with the signal **x**. Comment on what you observe. *{Hint: the 'toeplitz' command is useful for constructing the matrix* ***A*** *comprised of delay shifted versions of* ***x****}*

c) Repeat part *b)* except modify the matrix **A** by replacing the values in the $(m-1)$th and $(m+1)$th rows with zeros. Comment on what you observe.

d) Repeat parts *b*) and *c*) except incorporate a diagonal load term that is 2% of the largest eigenvalue of the matrix $\mathbf{A}^H\mathbf{A}$.

e) For each of the four normalized mismatched filters obtained in the above steps, compute

$$\text{mismatch loss} = -20\log_{10}\left(\frac{\max|\mathbf{h}_{\text{NMMF}} * \mathbf{x}|}{\max|\mathbf{h}_{\text{NMF}} * \mathbf{x}|}\right)\text{dB},$$

for * representing convolution. Comment on what you observe.

Solution:

The matched filter (MF) and Least-Squares (LS) mismatched filter (MMF) responses to convolution with **x**, after aligning the peaks, are shown in Fig. 4.1. It is observed that the MMF cases produce lower sidelobes, thus providing a better approximation to an impulse for deconvolution. However, relative to the MF the different MMFs also exhibit mismatched losses as listed in the table below, with the case using zeroed rows and diagonal loading yielding the least mismatch loss (though it also provides the least sidelobe suppression among the MMFs).

Note that the zeroing of rows is a form of "beamspoiling" that actually degrades resolution slightly while provide some additional robustness to mismatch loss. These 3 factors: mismatch loss, sidelobe level, and resolution, can be traded off within the LS MMF framework by using these tools (and others).

LS MMF	Mismatch loss
normal	6.30 dB
zeroed rows	6.31 dB
diagonally loaded	2.68 dB
zeroes rows & diagonally loaded	2.08 dB

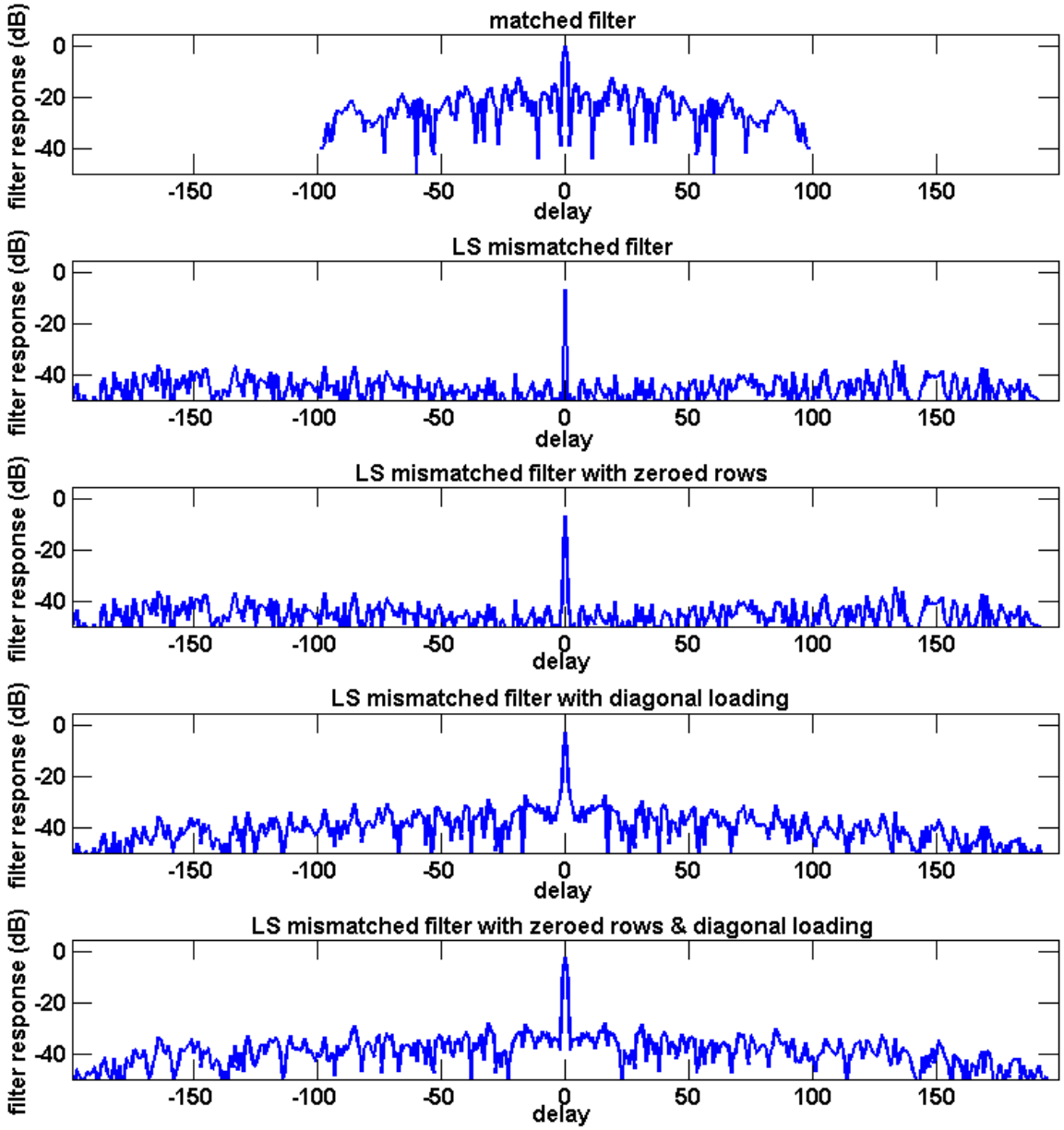


Figure 4.1. Matched filter and mismatched filter responses to convolution with **x**

Matlab Code for Problem 4

```
clear all
load P4
M = length(x);
mf = conj(flipud(x))./(x'*x);

K = 3;
mmfLen = M*K;
mmfOff = 2*M;

A = toeplitz([x; zeros(mmfLen-1,1)],[x(1) zeros(1,mmfLen-1)]);
Az = A;
Az(mmfOff-1,:) = 0;
Az(mmfOff+1,:) = 0;

[NA MA] = size(A);
[V D] = eig(A'*A);
lam_max = max(diag(D));

Phi = inv(A'*A)*A';
Phiz = inv(Az'*Az)*Az';
Phid = inv(A'*A + (0.02.*lam_max).*eye(MA))*A';
Phidz = inv(Az'*Az + (0.02.*lam_max).*eye(MA))*Az';

e = zeros(NA,1);
e(mmfOff) = 1;

mmf = Phi*e;
mmf = mmf./sqrt(mmf'*mmf)./sqrt(x'*x);
mmfz = Phiz*e;
mmfz = mmfz./sqrt(mmfz'*mmfz)./sqrt(x'*x);
mmfd = Phid*e;
mmfd = mmfd./sqrt(mmfd'*mmfd)./sqrt(x'*x);
mmfdz = Phidz*e;
mmfdz = mmfdz./sqrt(mmfdz'*mmfdz)./sqrt(x'*x);

mf_out = conv(mf,x);
mmf_out = conv(mmf,x);
mmfz_out = conv(mmfz,x);
mmfd_out = conv(mmfd,x);
mmfdz_out = conv(mmfdz,x);

MMloss = -20*log10(max(abs(mmf_out))/max(abs(mf_out)))
MMlossz = -20*log10(max(abs(mmfz_out))/max(abs(mf_out)))
MMlossd = -20*log10(max(abs(mmfd_out))/max(abs(mf_out)))
MMlossdz = -20*log10(max(abs(mmfdz_out))/max(abs(mf_out)))

yf_mf = conv(mf,y);
ymf_shft = [zeros(1*M-1,1); yf_mf];
yf_mmf = conv(mmf,y);
yf_mmfz = conv(mmfz,y);
yf_mmfd = conv(mmfd,y);
yf_mmfdz = conv(mmfdz,y);
```

```
len = length(y);

figure(1)
subplot(5,1,1)
plot(-(M-1):(M-1),20*log10(abs(mf_out)))
axis([-((K+1)*M/2-1) ((K+1)*M/2-1) -50 5])
grid on
title('matched filter')
xlabel('delay')
ylabel('filter response (dB)')

subplot(5,1,2)
plot(-((K+1)*M/2-1):((K+1)*M/2-1),20*log10(abs(mmf_out)))
axis([-((K+1)*M/2-1) ((K+1)*M/2-1) -50 5])
title('LS mismatched filter')
grid on
xlabel('delay')
ylabel('filter response (dB)')

subplot(5,1,3)
plot(-((K+1)*M/2-1):((K+1)*M/2-1),20*log10(abs(mmfz_out)))
axis([-((K+1)*M/2-1) ((K+1)*M/2-1) -50 5])
title('LS mismatched filter with zeroed rows')
grid on
xlabel('delay')
ylabel('filter response (dB)')

subplot(5,1,4)
plot(-((K+1)*M/2-1):((K+1)*M/2-1),20*log10(abs(mmfd_out)))
axis([-((K+1)*M/2-1) ((K+1)*M/2-1) -50 5])
title('LS mismatched filter with diagonal loading')
grid on
xlabel('delay')
ylabel('filter response (dB)')

subplot(5,1,5)
plot(-((K+1)*M/2-1):((K+1)*M/2-1),20*log10(abs(mmfdz_out)))
axis([-((K+1)*M/2-1) ((K+1)*M/2-1) -50 5])
title('LS mismatched filter with zeroed rows & diagonal loading')
grid on
xlabel('delay')
ylabel('filter response (dB)')
```

5. Data set P4.mat also contains the received signal $y(n)$ that is the result of convolving the known signal **x** from Prob. 4 with some unknown system. Apply each of the 5 filters (MF and 4 MMF) from Prob. 4 to perform deconvolution to estimate the unknown system. Plot the results (in dB) and comment on what you observe.

Solution:

The matched filter and Least-Squares (LS) mismatched filter responses to the received signal $y(n)$ are depicted in Fig. 5.1. Based on the matched filter response, there appears to be only 1 component to the unknown system. However, the set of MMF responses reveal that there are actually three additional smaller components. The degree to which these smaller components are visible depends on the degree to which the sidelobes are reduced (arguably the MMF with the worst mismatch loss provides the best sidelobe suppression as a trade-off).

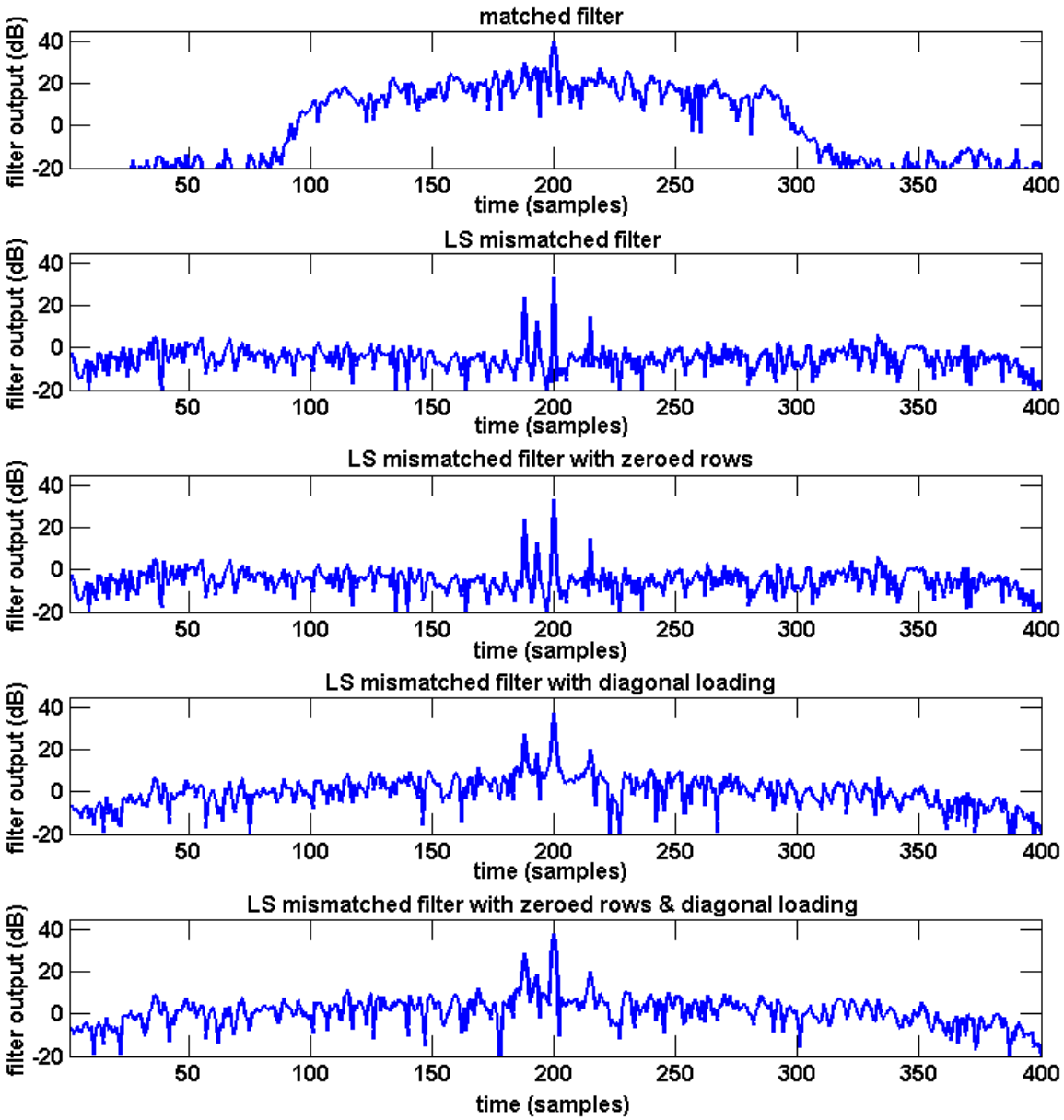


Figure 5.1. Matched filter and mismatched filter responses to $y(n)$

Matlab Code for Problem 5

```
clear all
load P4
M = length(x);
mf = conj(flipud(x))./(x'*x);

K = 3;
mmfLen = M*K;
mmfOff = 2*M;

A = toeplitz([x; zeros(mmfLen-1,1)],[x(1) zeros(1,mmfLen-1)]);
Az = A;
Az(mmfOff-1,:) = 0;
Az(mmfOff+1,:) = 0;

[NA MA] = size(A);
[V D] = eig(A'*A);
lam_max = max(diag(D));

Phi = inv(A'*A)*A';
Phiz = inv(Az'*Az)*Az';
Phid = inv(A'*A + (0.02.*lam_max).*eye(MA))*A';
Phidz = inv(Az'*Az + (0.02.*lam_max).*eye(MA))*Az';

e = zeros(NA,1);
e(mmfOff) = 1;

mmf = Phi*e;
mmf = mmf./sqrt(mmf'*mmf)./sqrt(x'*x);
mmfz = Phiz*e;
mmfz = mmfz./sqrt(mmfz'*mmfz)./sqrt(x'*x);
mmfd = Phid*e;
mmfd = mmfd./sqrt(mmfd'*mmfd)./sqrt(x'*x);
mmfdz = Phidz*e;
mmfdz = mmfdz./sqrt(mmfdz'*mmfdz)./sqrt(x'*x);

mf_out = conv(mf,x);
mmf_out = conv(mmf,x);
mmfz_out = conv(mmfz,x);
mmfd_out = conv(mmfd,x);
mmfdz_out = conv(mmfdz,x);

yf_mf = conv(mf,y);
ymf_shft = [zeros(1*M-1,1); yf_mf];
yf_mmf = conv(mmf,y);
yf_mmfz = conv(mmfz,y);
yf_mmfd = conv(mmfd,y);
yf_mmfdz = conv(mmfdz,y);

len = length(y);

figure(1)
subplot(5,1,1)
plot(1:len,20*log10(abs(ymf_shft(M:len+M-1))));
```

```
axis([1 len -20 45])
title('matched filter')
grid on
xlabel('time (samples)')
ylabel('filter output (dB)')

subplot(5,1,2)
plot(1:len,20*log10(abs(yf_mmf(M+1:len+M))));
axis([1 len -20 45])
title('LS mismatched filter')
grid on
xlabel('time (samples)')
ylabel('filter output (dB)')

subplot(5,1,3)
plot(1:len,20*log10(abs(yf_mmfz(M+1:len+M))));
axis([1 len -20 45])
title('LS mismatched filter with zeroed rows')
grid on
xlabel('time (samples)')
ylabel('filter output (dB)')

subplot(5,1,4)
plot(1:len,20*log10(abs(yf_mmfd(M+1:len+M))));
axis([1 len -20 45])
title('LS mismatched filter with diagonal loading')
grid on
xlabel('time (samples)')
ylabel('filter output (dB)')

subplot(5,1,5)
plot(1:len,20*log10(abs(yf_mmfdz(M+1:len+M))));
axis([1 len -20 45])
title('LS mismatched filter with zeroed rows & diagonal loading')
grid on
xlabel('time (samples)')
ylabel('filter output (dB)')
```

6. Data set P6.m contains time samples for an M = 30 element uniform linear array (ULA) with half-wavelength element spacing. Plot in dB and according to electrical angle:

a) the non-adaptive spatial power spectrum (see Appendix A; same as previous exam),

b) the MVDR spatial power spectrum using $\mathbf{R} = (1/L)\,\mathbf{X}\mathbf{X}^H$,

c) the MVDR spatial power spectrum using $\mathbf{R}$ obtained from forward/backward averaging via Appendix B,

d) and each of the above two MVDR cases diagonally loaded with $10\mathbf{I}$.

What do you observe from the various results (there are 5 cases)?

Solution:

Figure 6.1 shows the five different spatial power spectrum estimates. From the non-adaptive case it would appear that there are 3 significant signals present. Nearly all the other cases imply the same, albeit with sharper peaks. However, the forward-backward averaging case (without diagonal loading) reveals what could be a fourth signal due to the peak near –50° splitting.

In fact, there are four signals here at –50°, –43°, +18°, and +60°. The first two are so close together that most of these approaches cannot separate due to the sample support being rather low (L = 25) relative to the M = 30 antenna elements. Interestingly, the robustness provided by diagonal loading (notice that the sidelobes are lower for those two cases) is also found to somewhat hinder the sensitivity to discriminate these two nearby signals. The trade-off between high sensitivity/fidelity and robustness appears in numerous signal processing applications.

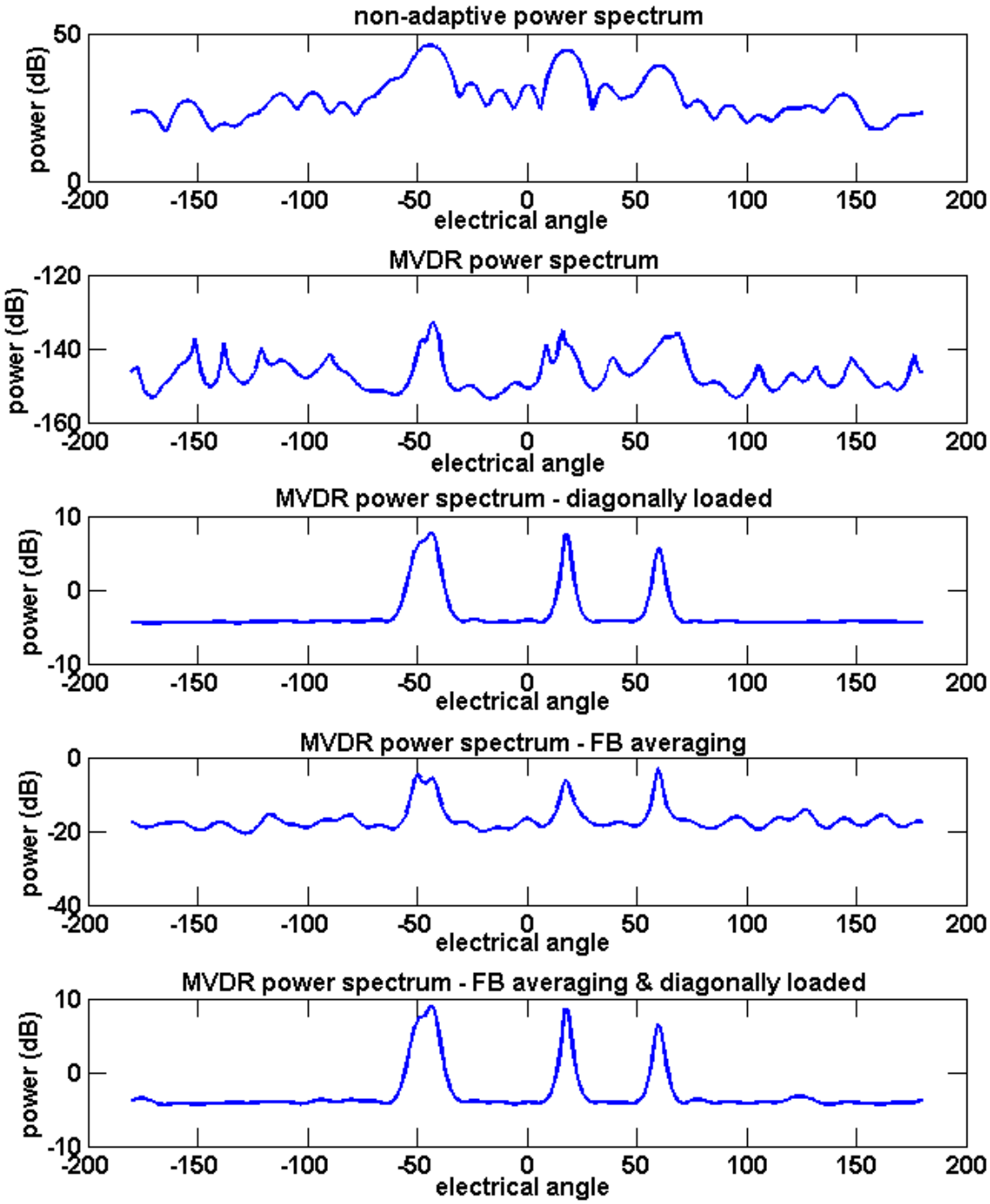


Figure 6.1. Spatial power spectrum estimates

Matlab Code for Problem 6

```
clear all
load P6
[M K] = size(X);

R = X*X'./K;
Ri = inv(R);
Rdi = inv(R + 10*eye(M,M));

Xb = conj(flipud(X)); % flips the array elements
Rfb = (X*X' + Xb*Xb')./(2*K);
Rfbi = inv(Rfb);
Rfbdi = inv(Rfb + 10*eye(M,M));

LSP = linspace(-180,180,M*10);

for theta_i = 1:length(LSP)
    Ct(1:M,theta_i) = (exp(j.*(LSP(theta_i)*pi/180).*[0:M-1]).');

    MVDR_pow(theta_i) = 1./(Ct(:,theta_i)'*Ri*Ct(:,theta_i));
    MVDR_powd(theta_i) = 1./(Ct(:,theta_i)'*Rdi*Ct(:,theta_i));

    MVDR_powfb(theta_i) = 1./(Ct(:,theta_i)'*Rfbi*Ct(:,theta_i));
    MVDR_powfbd(theta_i) = 1./(Ct(:,theta_i)'*Rfbdi*Ct(:,theta_i));
end;

pow_nonadap = mean(abs(Ct'*X).^2,2);

figure(1)
subplot(5,1,1)
plot(LSP,10*log10(pow_nonadap));
title('non-adaptive power spectrum')
xlabel('electrical angle')
ylabel('power (dB)')

subplot(5,1,2)
plot(LSP,10*log10(MVDR_pow));
title('MVDR power spectrum')
xlabel('electrical angle')
ylabel('power (dB)')

subplot(5,1,3)
plot(LSP,10*log10(MVDR_powd));
title('MVDR power spectrum - diagonally loaded')
xlabel('electrical angle')
ylabel('power (dB)')

subplot(5,1,4)
plot(LSP,10*log10(MVDR_powfb));
title('MVDR power spectrum - FB averaging')
xlabel('electrical angle')
ylabel('power (dB)')
```

```
subplot(5,1,5)
plot(LSP,10*log10(MVDR_powfbd));
title('MVDR power spectrum - FB averaging & diagonally loaded')
xlabel('electrical angle')
ylabel('power (dB)')
```