**EECS 844 – Fall 2017**

Exam 2 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.

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 the dataset P1.mat are two signals, the input $x(n)$ and desired response $d(n)$ from an unknown system we wish to identify. Construct the Wiener filter and determine the resulting MSE for filter lengths of $M = 0, 1, 2, \ldots, 40$. ($M = 0$ means no filter)

a) Plot the MSE (in dB) as a function of length $M$, using both direct calculation (via average of $|e(n)|^2$) and the analytical MSE (recall the 'error performance surface'). Discuss what you observe.
b) Plot the magnitude and phase of the filter coefficients for the $M = 40$ case.
c) Use the 'freqz' command to plot the frequency response of the $M = 40$ filter. Use the modifier "whole" (see 'help freqz' for details) to plot the entire $2\pi$ digital frequency interval.
d) If we assume the unknown system is a MA model, what would you estimate to be length of the unknown MA model?

Solution:

Figure 1.1 shows the MSE as a function of filter length $M$ using three different ways to determine the MSE, with all three forms in agreement (as theoretically expected). It is observed that the MSE decreases monotonically with increasing $M$. Over the 40 coefficients, the MSE never flattens out, so if the unknown system is a MA model we can only say that the length of the model is greater than 40. This result can likewise be inferred from the filter coefficient magnitudes in Fig. 1.2, which show a very slow decrease towards 0, but still having a magnitude greater than 1 at filter coefficient 40. Based on the frequency content in Fig. 1.3, we observe bandpass characteristics with a mainlobe at normalized frequency of 0.25.

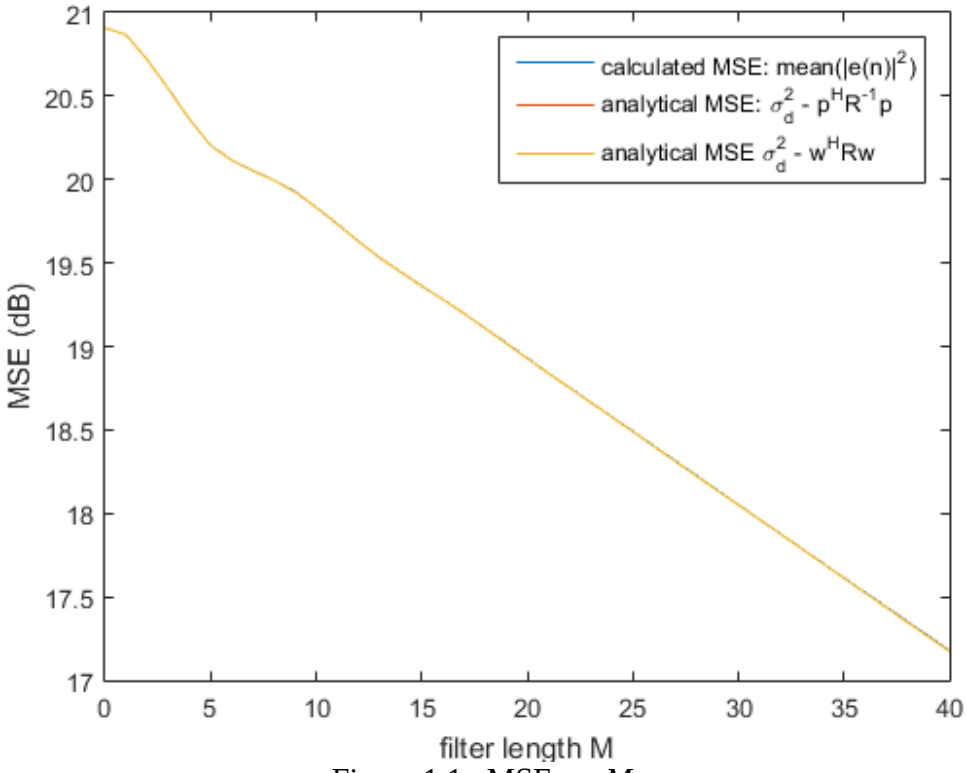



Figure 1.1. MSE vs. $M$

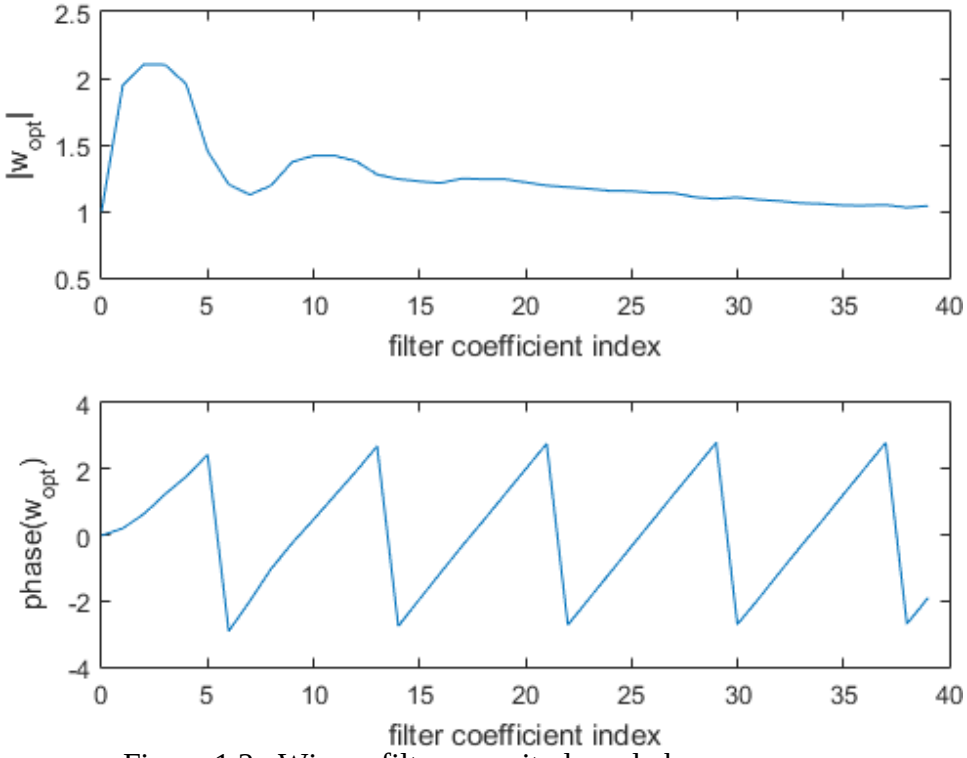



Figure 1.2. Wiener filter magnitude and phase response

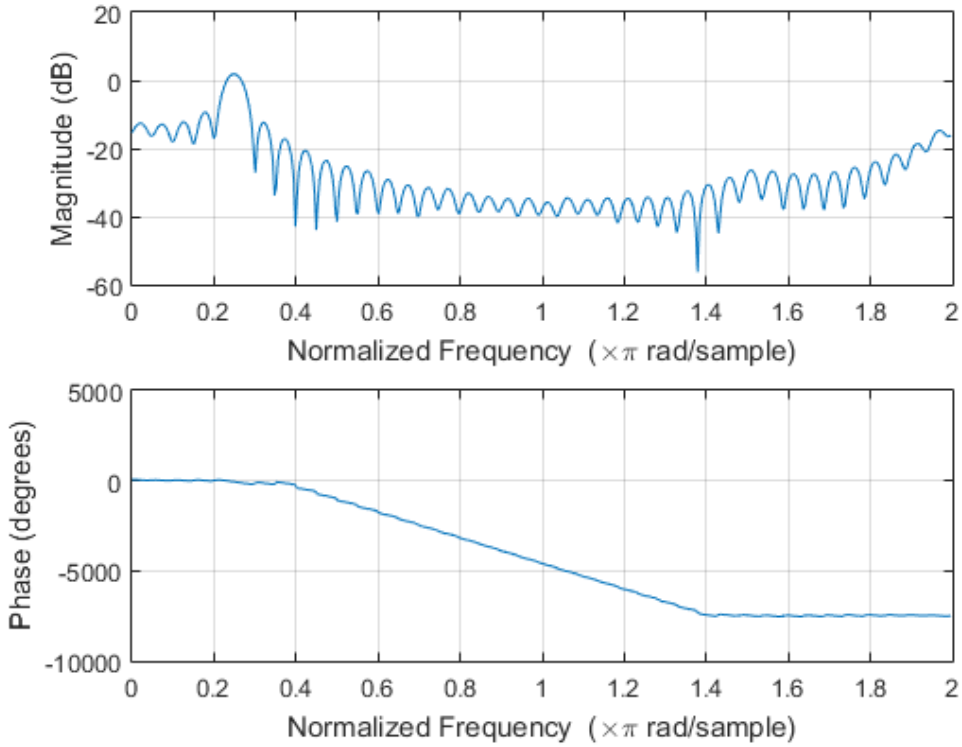



Figure 1.3. Wiener filter frequency response

# Matlab Code for Problem 1

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

M = 40;

MSE(1) = d'*d./N; % MSE for no filter applied

for jM = 1:M

    %%% estimate w_opt
    X = zeros(jM,N-jM+1);
    for jm = 1:jM
        X(jM-jm+1,:) = x(jm:jm+N-jM).';
    end;

    R_x = (1./(N-jM+1)).*X*X';
    R_xi = inv(R_x);

    D_d = repmat(d(jM:N)',jM,1);
    p_xd = sum(X.*D_d ,2)./(N-jM+1);

    wxd_opt = R_xi*p_xd;
    d_opt = filter(conj(wxd_opt),1,x);
    e_d = d-d_opt;
    MSE(jM+1) = (e_d'*e_d)./N;
    MSE2(jM+1) = MSE(1) - p_xd'*R_xi*p_xd;
    MSE3(jM+1) = MSE(1) - wxd_opt'*R_x*wxd_opt;
end;

figure(1)
plot(0:M,10*log10(MSE),0:M,10*log10(MSE2),0:M,10*log10(MSE3))
xlabel('filter length M')
ylabel('MSE (dB)')
legend('calculated MSE: mean(|e(n)|^{2})','analytical MSE:
\sigma_{d}^{2} - p^{H}R^{-1}p','analytical MSE \sigma_{d}^{2} -
w^{H}Rw')

figure(2)
subplot(2,1,1)
plot(0:M-1,abs(wxd_opt));
ylabel('|w_{opt}|')
xlabel('filter coefficient index')
subplot(2,1,2)
plot(0:M-1,angle(wxd_opt));
ylabel('phase(w_{opt})')
xlabel('filter coefficient index')

figure(3)
freqz(wxd_opt,length(wxd_opt),'whole')
```

**2**. Now repeat problem 1, but let $d(n)$ be the input and $x(n)$ be the desired response. Can we say anything more about the unknown system when the problem is posed this way?

Solution:

Figure 2.1 shows the MSE as a function of filter length $M$, again using the three different ways to determine the MSE, with all three forms in agreement. What becomes immediately obvious for this "inverse system identification" implementation is that the MSE drops to a flat level for $M > 5$ filter coefficients. Likewise from examining the length $M = 40$ filter in Fig. 2.2, we find that beyond $M = 5$ all the filter coefficients are rather small (actually would be zero if we had an infinite amount of data). We can therefore infer that the unknown system is an AR model instead of a MA model (would require an infinite number of filter coefficients to determine the actual MA model per the Wold decomposition). Finally, we see from the frequency response in Fig. 2.3 that we essentially obtain the inverse of that obtained previously in Fig. 1.3, which is to be expected.

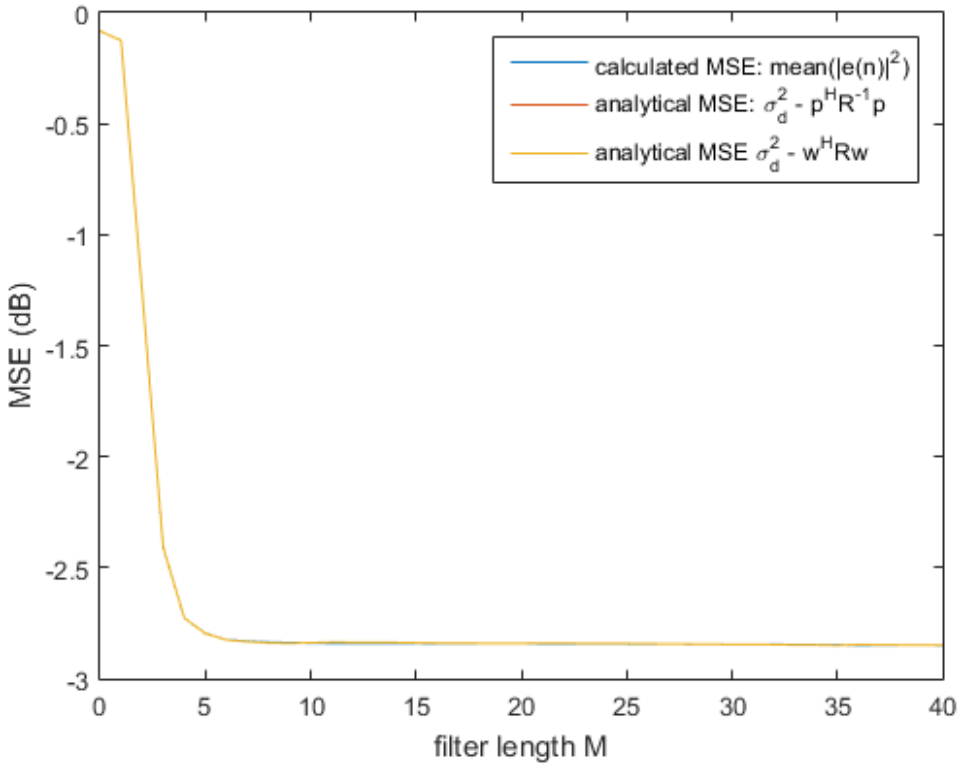



Figure 2.1. MSE vs. $M$

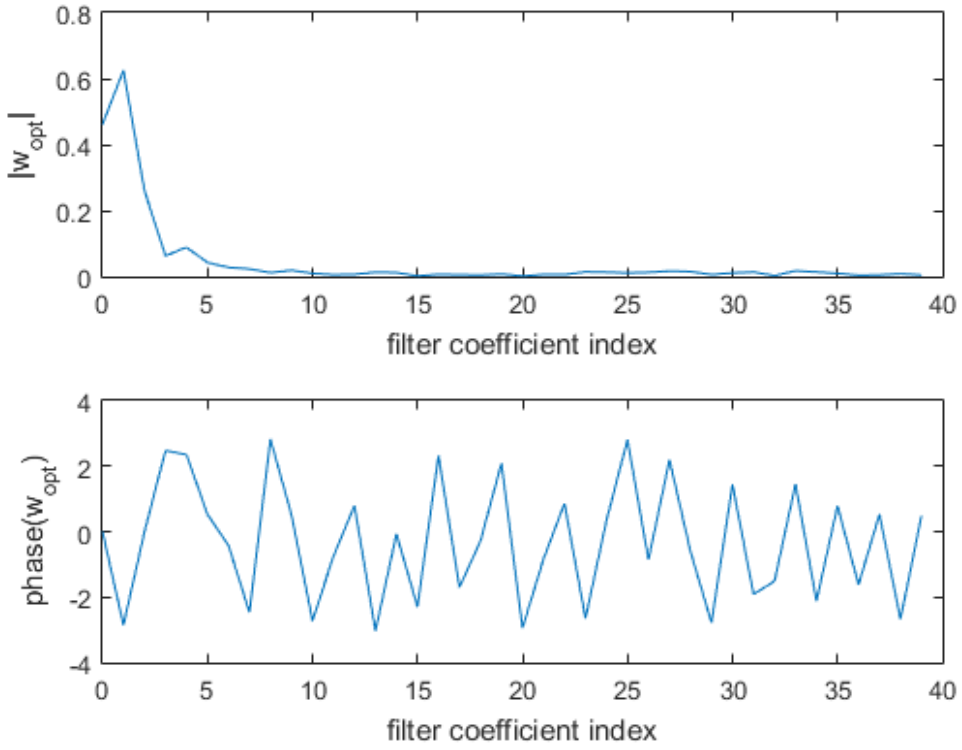



Figure 2.2. Wiener filter magnitude and phase response

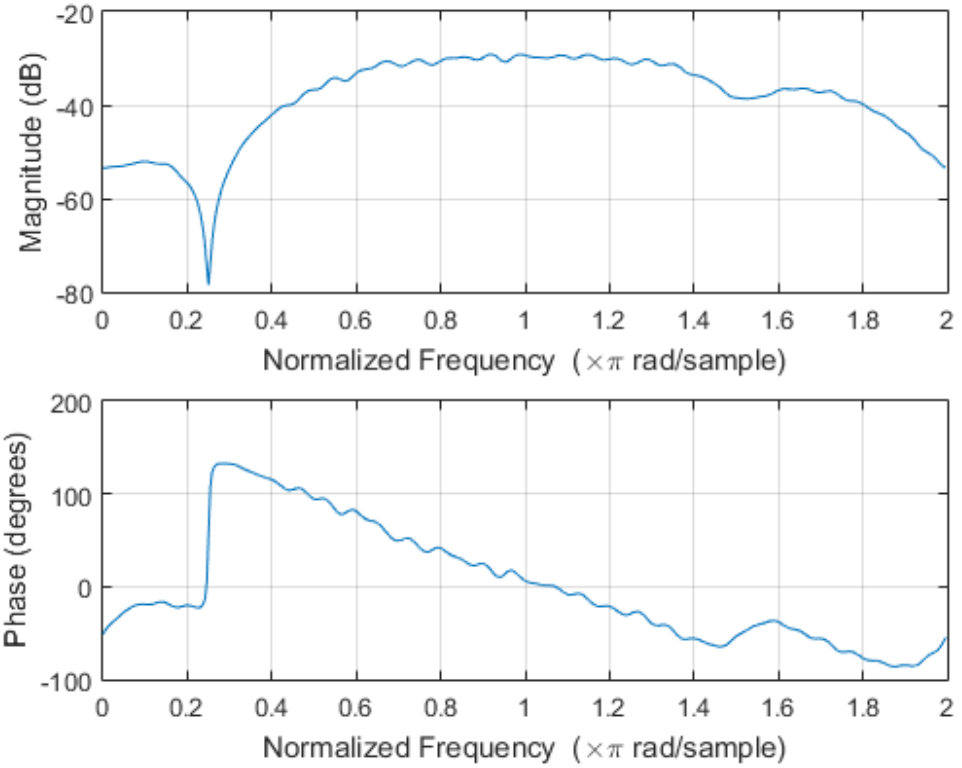



Figure 2.3. Wiener filter frequency response

# Matlab Code for Problem 2

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

M = 40;
x2 = d;
d2 = x;

MSE2(1) = d2'*d2./N; % MSE for no filter applied
MSEaltA2(1) = MSE2(1);
MSEaltB2(1) = MSE2(1);

for jM = 1:M

    %%% estimate w_opt2
    X2 = zeros(jM,N-jM+1);
    for jm = 1:jM
        X2(jM-jm+1,:) = x2(jm:jm+N-jM).';
    end;

    R_x2 = (1./(N-jM+1)).*X2*X2';
    R_xi2 = inv(R_x2);

    D_d2 = repmat(d2(jM:N)',jM,1);
    p_xd2 = sum(X2.*D_d2 ,2)./(N-jM+1);

    wxd_opt2 = R_xi2*p_xd2;
    d_opt2 = filter(conj(wxd_opt2),1,x2);
    e_d2 = d2-d_opt2;
    MSE2(jM+1) = (e_d2'*e_d2)./N;
    MSEaltA2(jM+1) = MSE2(1) - p_xd2'*R_xi2*p_xd2;
    MSEaltB2(jM+1) = MSE2(1) - wxd_opt2'*R_x2*wxd_opt2;
end;

figure(4)
plot(0:M,10*log10(MSE2),0:M,10*log10(MSEaltA2),0:M,10*log10(MSEaltB2))
xlabel('filter length M')
ylabel('MSE (dB)')
legend('calculated MSE: mean(|e(n)|^{2})','analytical MSE:
\sigma_{d}^{2} - p^{H}R^{-1}p','analytical MSE \sigma_{d}^{2} -
w^{H}Rw')

figure(5)
subplot(2,1,1)
plot(0:M-1,abs(wxd_opt2));
ylabel('|w_{opt}|')
xlabel('filter coefficient index')
subplot(2,1,2)
plot(0:M-1,angle(wxd_opt2));
ylabel('phase(w_{opt})')
xlabel('filter coefficient index')

figure(6)
freqz(wxd_opt2,length(wxd_opt2),'whole')
```

**3.** Dataset P3.mat contains 50 time snapshots (in the columns) from an $M = 12$ element uniform linear array (ULA) with half-wavelength element spacing (so $d = 0.5\lambda$).

*a)* Plot the MVDR power (spatial) spectrum estimate in dB in terms of spatial angle $\phi$. Discuss what you observe (e.g. how many signals do there appear to be?).
*b)* Plot the non-adaptive (spatial) spectrum estimate defined in Appendix A in terms of $\phi$. Describe how this result relates to the adaptive power spectrum.
*c)* Plot the MVDR beampattern for the spatial directions $\phi = -30°$, 0°, and +30° (in terms of $\phi$), and explain what you observe relative to the MVDR power spectrum from part *a*).

Solution:
Figure 3.1 illustrates the MVDR power spectrum where either 6 or 7 signals appear to be present. In Fig. 3.2 the eigenvalues of the correlation matrix **R** are plotted which also seems to indicate roughly 7 or 8. There are actually 7 signals at spatial angles of –40.0°, –32.9°, –3.4°, +3.7°, +28.1°, +41.8°, +56.0°.

Comparing the MVDR power spectrum in Fig. 3.1 to the non-adaptive spatial spectrum in Fig. 3.3 shows that the former certainly provides sharper peaks. In fact, the non-adaptive response appears to only show 2 signals present, instead of the 7 that are actually there.

Finally, for the beampatterns in Fig. 3.4 we obtain a unity gain in each specified "look" direction but the sidelobe responses change due to the proximity of other strong signals, for which there are discernible nulls in the beampatterns that correspond to the peaks in Fig. 3.1. It is important to understand the distinction that these beampattern results are "how the particular spatial filter responds as a function of electrical angle" while the adaptive and non-adaptive power spectra in Figs. 3.1 and 3.3, respectively, provide information about the directions from which actual signals are arriving. Also note that the unity gain does not necessary mean the beampattern will have a peak in that direction (as evidenced by the beampatterns for gain constraints in the spatial directions of –30° and +30°.

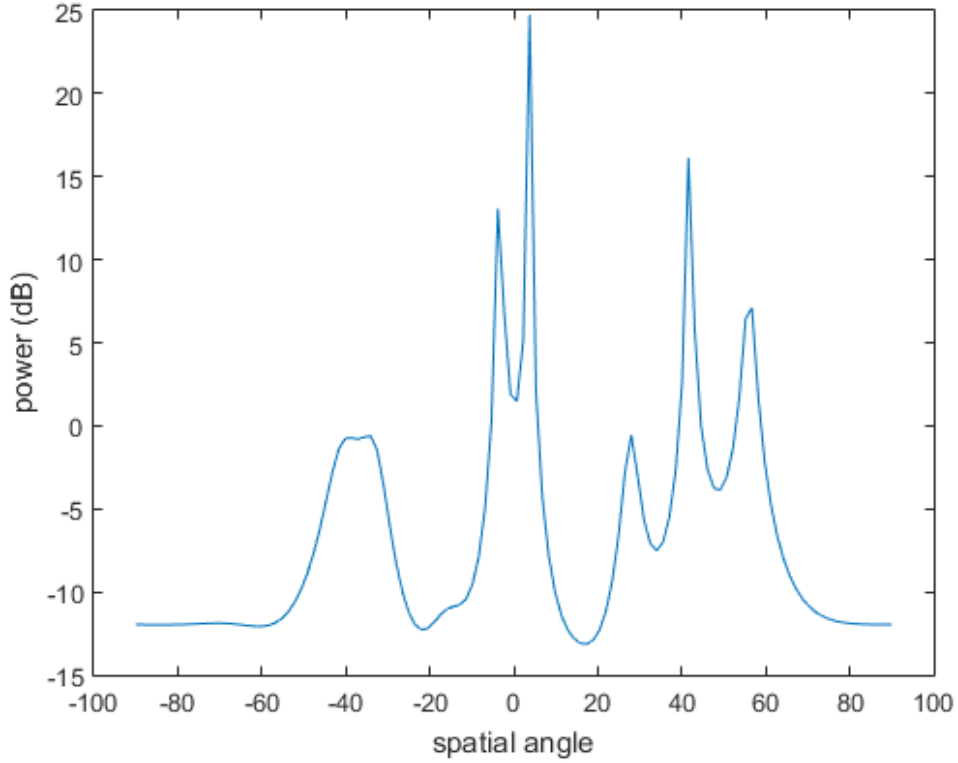



Figure 3.1. MVDR spatial power spectrum

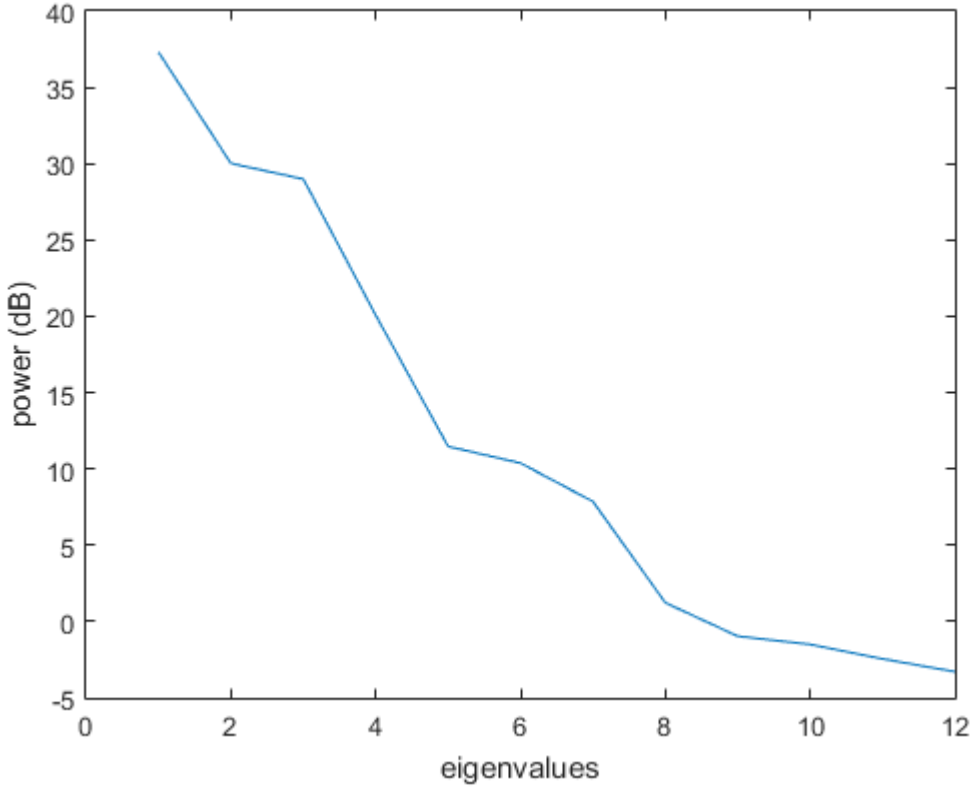



Figure 3.2. Eigenvalues of correlation matrix

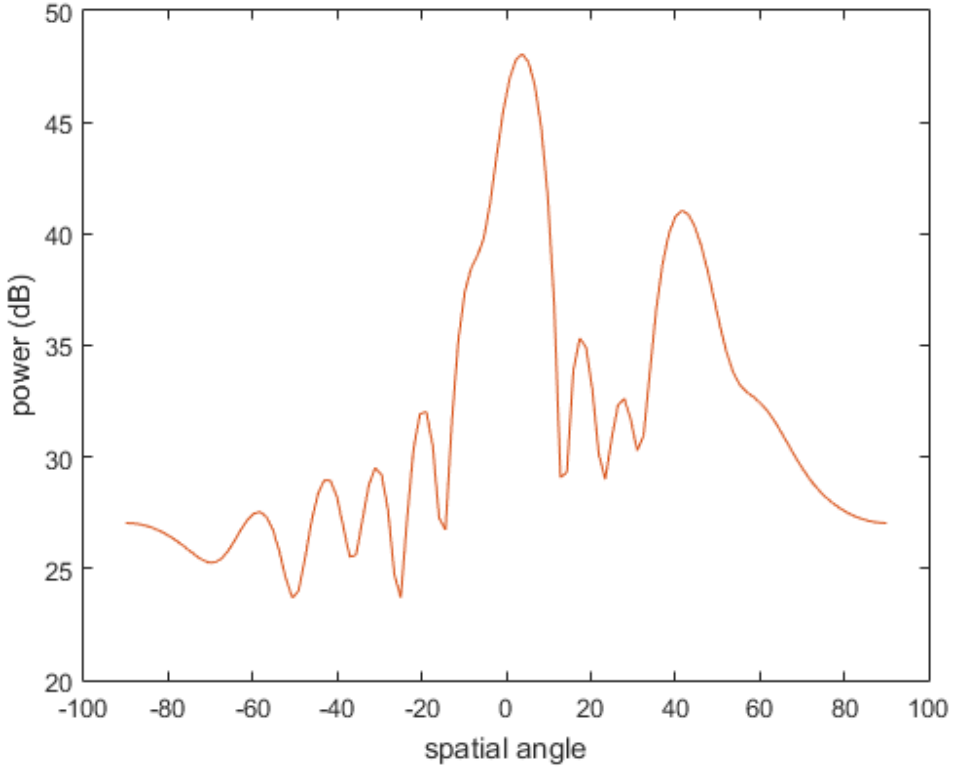



Figure 3.3. Non-adaptive spatial power spectrum

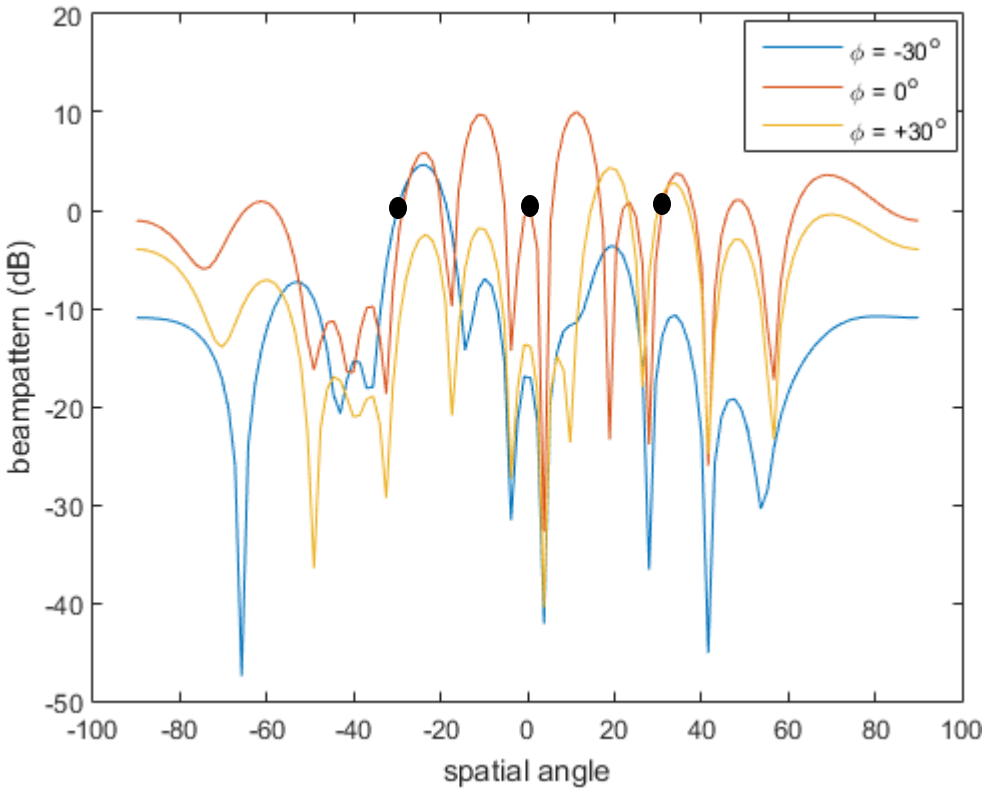



Figure 3.4. MVDR beampatterns for spatial steering directions of –30°, 0°, and +30°

# Matlab Code for Problem 3

```
load P3
[M K] = size(X);

R = X*X'./K;
Ri = inv(R);

d = 0.5; %units of lambda

phi_look1 = -45*pi/180;
phi_look2 = 0*pi/180;
phi_look3 = 45*pi/180;

thet_look1 = 2*pi*d*sin(phi_look1);
thet_look2 = 2*pi*d*sin(phi_look2);
thet_look3 = 2*pi*d*sin(phi_look3);

s_look1 = exp(j.*thet_look1.*[0:M-1]).';
s_look2 = exp(j.*thet_look2.*[0:M-1]).';
s_look3 = exp(j.*thet_look3.*[0:M-1]).';

w_MVDR1 = Ri*s_look1./(s_look1'*Ri*s_look1);
w_MVDR2 = Ri*s_look2./(s_look2'*Ri*s_look2);
w_MVDR3 = Ri*s_look3./(s_look3'*Ri*s_look3);

LSP = linspace(-90,90,10*M);
for theta_i = 1:length(LSP)
    Ct(1:M,theta_i) = (exp(-j.*2*pi*d*sin(LSP(theta_i)*pi/180).*
         [0:M-1]).');
    MVDR_pow(theta_i) = 1./(Ct(:,theta_i)'*Ri*Ct(:,theta_i));
    beam_MVDR1(theta_i) = Ct(:,theta_i)'*w_MVDR1;
    beam_MVDR2(theta_i) = Ct(:,theta_i)'*w_MVDR2;
    beam_MVDR3(theta_i) = Ct(:,theta_i)'*w_MVDR3;

end;

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

figure(1)
plot(LSP,10*log10(MVDR_pow));
xlabel('spatial angle')
ylabel('power (dB)')

figure(2)
plot(LSP,10*log10(pow_nonadap));
max(real(10*log10(pow_nonadap)))])
xlabel('spatial angle')
ylabel('power (dB)')

figure(3)
plot(LSP,20*log10(beam_MVDR1),LSP,20*log10(beam_MVDR2),…
   LSP,20*log10(beam_MVDR3));
xlabel('spatial angle')
ylabel('beampattern (dB)')
```

**4.** Dataset P4.mat contains the discrete-time signal $x(n)$ for which we wish to isolate the individual frequency components (recall that $f \in [-0.5,+0.5]$). For filter lengths of $M$ = 10, 20, 40, and 80 plot the associated MVDR power spectrum for each (in dB). Discuss what you observe.

Solution:

Figure 4.1 shows the MVDR (temporal) power spectrum for the signal $x(n)$ that was composed of 9 sinusoids in noise. For the $M$ = 80 case (purple) all 9 signals are visible, with a decreasing number visible for smaller values of $M$. The $M$ = 10 case (blue) only revealed 4 signals, despite the fact that $M$ = 10 is greater than 9.

While it may be expected that both the $M$ = 20 and $M$ = 40 cases should have been able to identify all 9 signals since they possessed more than enough degrees of freedom, the length of the filter also dictates how well closely-spaced signals can be separated, which is related to the nominal resolution (which we shall discuss later in the semester). Indeed, those signals missed by the $M$ = 20 case were smaller (in terms of power) signals that had frequencies close to larger power signals and only the improved resolving ability of the $M$ = 80 case could identify them. Note that the increased filter size also provides greater coherent integration relative to noise, thus enhanced SNR.

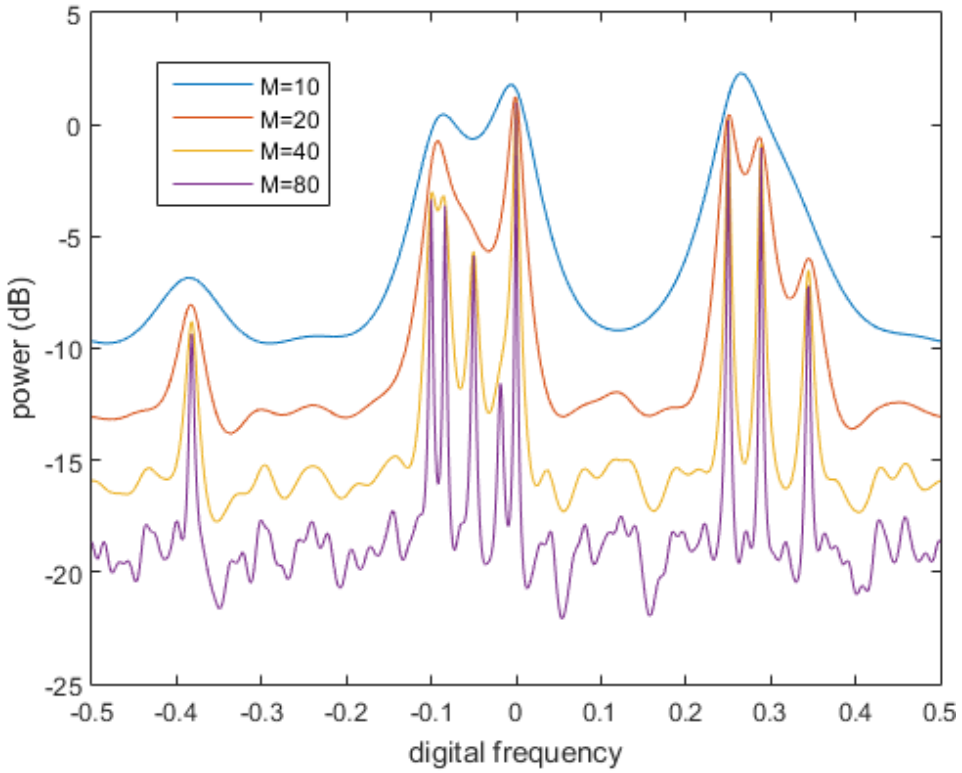



Figure 4.1. MVDR temporal power spectrum

# Matlab Code for Problem 4

```
load P4

N = length(x);
M10 = 10;
M20 = 20;
M40 = 40;
M80 = 80;

for jm = 1:M10
    X10(M10-jm+1,:) = x(jm:jm+N-M10).';
end;
R10 = (1./(N-M10+1)).*X10*X10';
Ri10 = inv(R10);

for jm = 1:M20
    X20(M20-jm+1,:) = x(jm:jm+N-M20).';
end;
R20 = (1./(N-M20+1)).*X20*X20';
Ri20 = inv(R20);

for jm = 1:M40
    X40(M40-jm+1,:) = x(jm:jm+N-M40).';
end;
R40 = (1./(N-M40+1)).*X40*X40';
Ri40 = inv(R40);

for jm = 1:M80
    X80(M80-jm+1,:) = x(jm:jm+N-M80).';
end;
R80 = (1./(N-M80+1)).*X80*X80';
Ri80 = inv(R80);

LSP = linspace(-0.5,0.5,M80*10+1);

for theta_i = 1:length(LSP)
    Ct10(1:M10,theta_i) = (exp(-j.*(LSP(theta_i)*2*pi).*[0:M10-1]).');
    MVDR_pow10(theta_i) = 1./(Ct10(:,theta_i)'*Ri10*Ct10(:,theta_i));
    Ct20(1:M20,theta_i) = (exp(-j.*(LSP(theta_i)*2*pi).*[0:M20-1]).');
    MVDR_pow20(theta_i) = 1./(Ct20(:,theta_i)'*Ri20*Ct20(:,theta_i));
    Ct40(1:M40,theta_i) = (exp(-j.*(LSP(theta_i)*2*pi).*[0:M40-1]).');
    MVDR_pow40(theta_i) = 1./(Ct40(:,theta_i)'*Ri40*Ct40(:,theta_i));
    Ct80(1:M80,theta_i) = (exp(-j.*(LSP(theta_i)*2*pi).*[0:M80-1]).');
    MVDR_pow80(theta_i) = 1./(Ct80(:,theta_i)'*Ri80*Ct80(:,theta_i));
end;

figure(1)
plot(LSP,10*log10(MVDR_pow10),LSP,10*log10(MVDR_pow20),LSP,10*log10(MVD
R_pow40),LSP,10*log10(MVDR_pow80));
%axis([-0.5 0.5 -20 25])
xlabel('digital frequency')
ylabel('power (dB)')
legend('M=10','M=20','M=40','M=80')
```

**5**. Using data set P4.mat, implement an $M = 40$ GSC having only a unity gain constraint at frequency $f = +0.25$. See Appendix B.

*a)* Compare the frequency response (implement same as the beampattern) of the resulting GSC filter with the frequency response of an MVDR filter having the same gain constraint. (Plot in dB)
*b)* Use eigenvector constraints to define a broad null over the spectral region from $f = -0.4$ to $f = -0.2$. (plot the eigenvalues in dB)
*c)* Implement the GSC with the same gain constraint and with 0, 10, 12, and 14 eigenvector null constraints. Plot the associated frequency responses and comment on what you observe.

Solution:
Figure 5.1 shows the GSC (with only a gain constraint) and MVDR. As expected, the frequency responses are identical.

To implement eigenvector null constraints, Fig. 5.2 shows the eigenvalues that result from forming a modeled correlation matrix based on steering vectors distributed over the interval $f = -0.4$ to $f = -0.2$. Roughly the last 7-8 appear to be the principal eigenvalues.

The frequency responses of the GSC filter obtained when 0, 10, 12, and 14 null constraints are used (in addition to a single gain constraint) are shown in Fig. 5.3. As the number of null constraints is increased, the null becomes considerably deeper and slightly wider. Given the expected limits of numerical precision in practice, as well as the desire to avoid "over-nulling" which impacts the surrounding frequencies, one could argue that the 10 null constraint case is best of those considered.

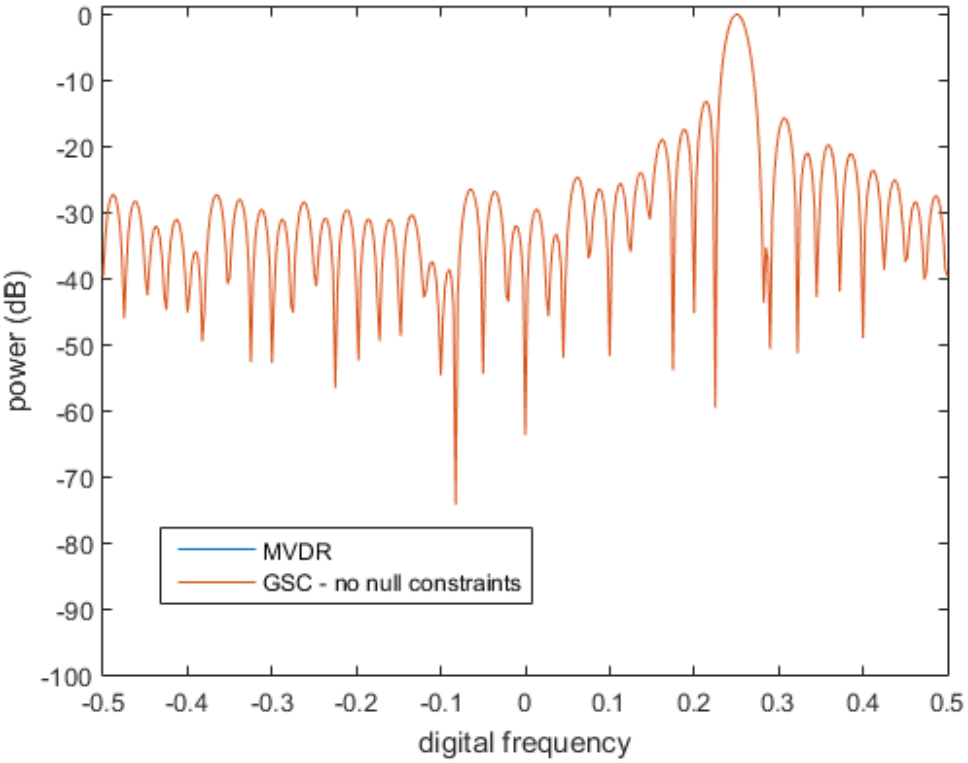



Figure 5.1. MVDR and GSC with only a gain constraint

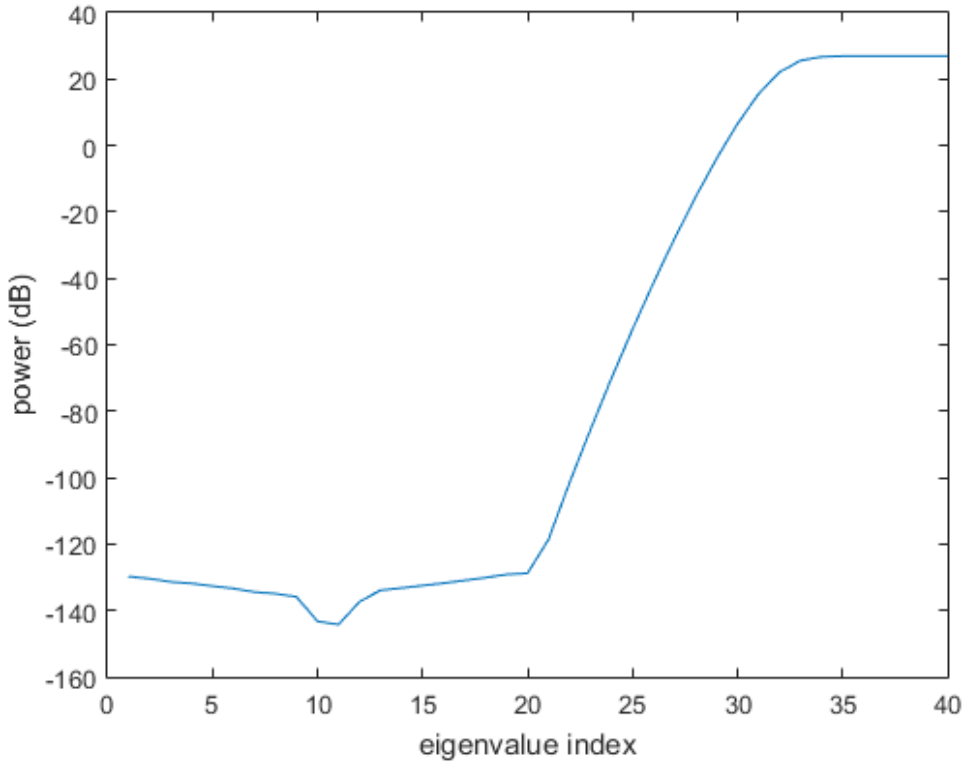



Figure 5.1. Modeled eigenvalues for broad frequency null.

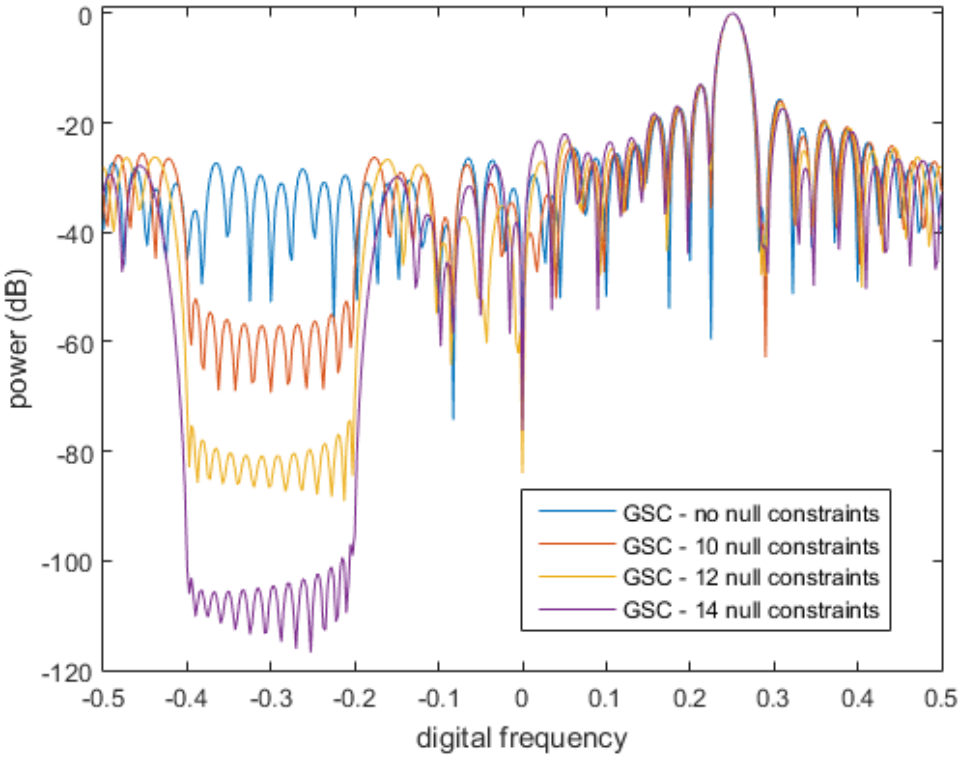



Figure 5.2. Frequency response for GSC with a gain constraint and 0, 10, 12 and 14 null constraints

# Matlab Code for Problem 5

```
clear all

load P4

N = length(x);
M40 = 40;

for jm = 1:M40
    X40(M40-jm+1,:) = x(jm:jm+N-M40).';
end;
R40 = (1./(N-M40+1)).*X40*X40' + eye(40);
Ri40 = inv(R40);

LSP = linspace(-0.5,0.5,M40*10+1);
for theta_i = 1:length(LSP)
    Ct40(1:M40,theta_i) = (exp(-j.*(LSP(theta_i)*2*pi).*[0:M40-1]).');
end;

s_look = exp(-j.*0.25*2*pi.*[0:M40-1]).';
wMVDR = Ri40*s_look./(s_look'*Ri40*s_look);
beam_MVDR = abs(Ct40'*wMVDR);

%%%%%%%%%%%%%%%% GSC - EV constraints
freq_null = linspace(-0.4,-0.2,100);
numfreqs = length(freq_null);
for jt = 1:numfreqs
    Cev(1:M40,jt) = (exp(-j.*(freq_null(jt)*2*pi).*[0:M40-1]).');
end;
Rnull = Cev*Cev';
[Vev,Dev] = eig(Rnull);

figure(1)
plot(1:M40,10*log10(flipud(diag(Dev))))
xlabel('eigenvalue index')
ylabel('power (dB)')

nullrank0 = 0;
C0 = [s_look Vev(:,(M40-nullrank0+1):M40)];
g0 = [1; zeros(nullrank0,1)];
wq0 = C0*inv(C0'*C0)*g0;
[U0,S0,V0] = svd(C0);
Ca0 = U0(:,(nullrank0+2):M40);
wa0 = inv(Ca0'*R40*Ca0)*Ca0'*R40*wq0;
wGSC0 = wq0 - Ca0*wa0;
beam_GSC0 = abs(Ct40'*wGSC0);

nullrank10 = 10;
C10 = [s_look Vev(:,(M40-nullrank10+1):M40)];
g10 = [1; zeros(nullrank10,1)];
wq10 = C10*inv(C10'*C10)*g10;
[U10,S10,V10] = svd(C10);
Ca10 = U10(:,(nullrank10+2):M40);
wa10 = inv(Ca10'*R40*Ca10)*Ca10'*R40*wq10;
wGSC10 = wq10 - Ca10*wa10;
```

```
beam_GSC10 = abs(Ct40'*wGSC10);

nullrank12 = 12;
C12 = [s_look Vev(:,(M40-nullrank12+1):M40)];
g12 = [1; zeros(nullrank12,1)];
wq12 = C12*inv(C12'*C12)*g12;
[U12,S12,V12] = svd(C12);
Ca12 = U12(:,(nullrank12+2):M40);
wa12 = inv(Ca12'*R40*Ca12)*Ca12'*R40*wq12;
wGSC12 = wq12 - Ca12*wa12;
beam_GSC12 = abs(Ct40'*wGSC12);

nullrank14 = 14;
C14 = [s_look Vev(:,(M40-nullrank14+1):M40)];
g14 = [1; zeros(nullrank14,1)];
wq14 = C14*inv(C14'*C14)*g14;
[U14,S14,V14] = svd(C14);
Ca14 = U14(:,(nullrank14+2):M40);
wa14 = inv(Ca14'*R40*Ca14)*Ca14'*R40*wq14;
wGSC14 = wq14 - Ca14*wa14;
beam_GSC14 = abs(Ct40'*wGSC14);

figure(2)
plot(LSP,20*log10(beam_MVDR),LSP,20*log10(beam_GSC0));
axis([-0.5 0.5 -100 1])
legend('MVDR','GSC - no null constraints')
xlabel('digital frequency')
ylabel('power (dB)')

figure(3)
plot(LSP,20*log10(beam_GSC0),LSP,20*log10(beam_GSC10),LSP,20*log10(beam
_GSC12),LSP,20*log10(beam_GSC14));
axis([-0.5 0.5 -120 1])
legend('GSC - no null constraints','GSC - 10 null constraints','GSC -
12 null constraints','GSC - 14 null constraints')
xlabel('digital frequency')
ylabel('power (dB)')
```

**6**. Dataset P6.mat contains four signals $x_1(n)$, $x_2(n)$, $x_3(n)$, and $x_4(n)$ that are related through unknown LTI systems (MA models of order no more than 30). Sketch the signal flow network, labeling where each signal exists and each branch with the number of significant non-zero coefficient in the associated LTI MA system. Discuss/show how these results were obtained.

Solution:

By computing the Wiener filter in a system identification manner between each pair of signals we can determine how the signals are connected through LTI systems. Figure 6.1 depicts the estimated systems when $x_1(n)$ is assumed to be the input and $x_2(n)$, $x_3(n)$, and $x_4(n)$ are assumed to be the resulting output (the desired response). Likewise, Figures 6.2, 6.3, and 6.4 illustrate the estimated systems when $x_2(n)$, $x_3(n)$, and $x_4(n)$ are respectively assumed to be the input and each of the other three signals the resulting output.

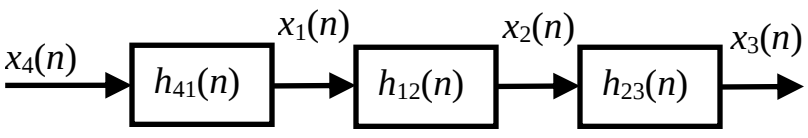
Using the fact that we know the unknown systems are moving average (MA) of order no more than 30, we can determine by inspection of these results that the filter lengths are as follows.

| | $x_1(n)$ output | $x_2(n)$ output | $x_3(n)$ output | $x_4(n)$ output |
|---|---|---|---|---|
| $x_1(n)$ input | N/A | $M = 8$ | $M = 12$ | Not clear |
| $x_2(n)$ input | Not clear | N/A | $M = 5$ | Not clear |
| $x_3(n)$ input | > 30 | > 30 | N/A | Not clear |
| $x_4(n)$ input | $M = 16$ | $M = 23$ | $M = 27$ | N/A |

Therefore, we can make the following observations:

- $x_2(n)$ is input to 5 MA coefs. producing $x_3(n) \Rightarrow h_{23}(n)$
- $x_1(n)$ is input to 8 MA coefs. producing $x_2(n) \Rightarrow h_{12}(n)$
- $x_1(n)$ is input to 12 (=5+8-1) MA coefs. producing $x_3(n) \Rightarrow h_{12}(n) * h_{23}(n)$
- $x_4(n)$ is input to 16 MA coefs. producing $x_1(n) \Rightarrow h_{41}(n)$
- $x_4(n)$ is input to 23 (=16+8-1) MA coefficients producing $x_2(n) \Rightarrow h_{41}(n) * h_{12}(n)$
- $x_4(n)$ is input to 27 (=23+5-1) MA coefficients producing $x_3(n)$
  $\Rightarrow h_{41}(n) * h_{12}(n) * h_{23}(n)$

Thus the network is:



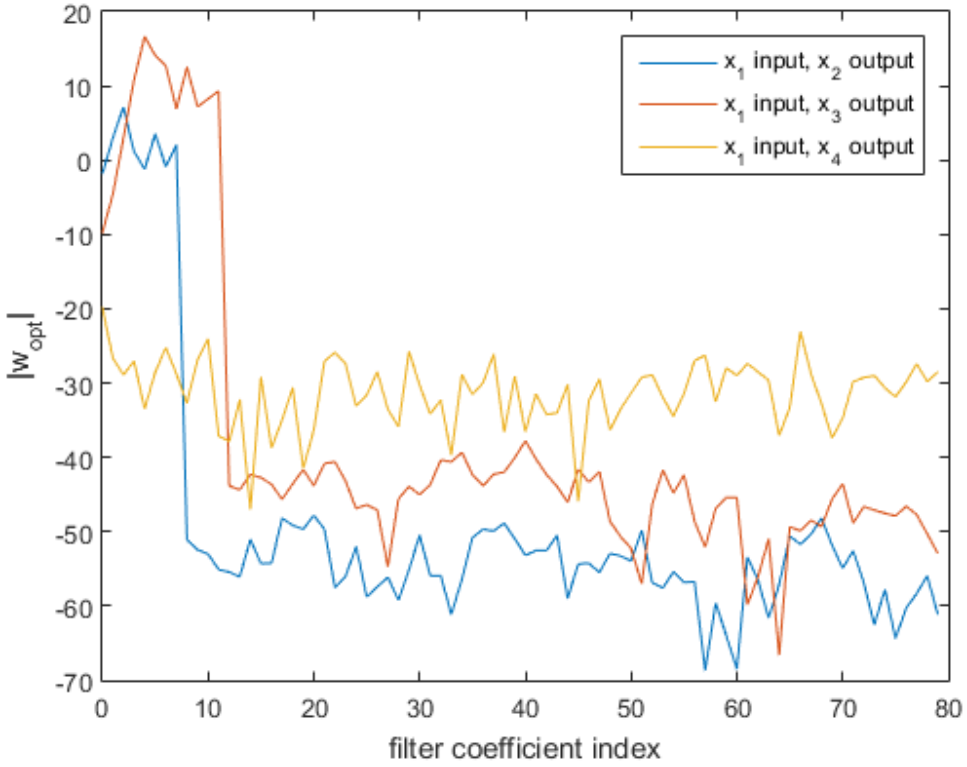



Figure 6.1. Wiener filters estimated if $x_1(n)$ is input and $x_2(n)$, $x_3(n)$, or $x_4(n)$ are output

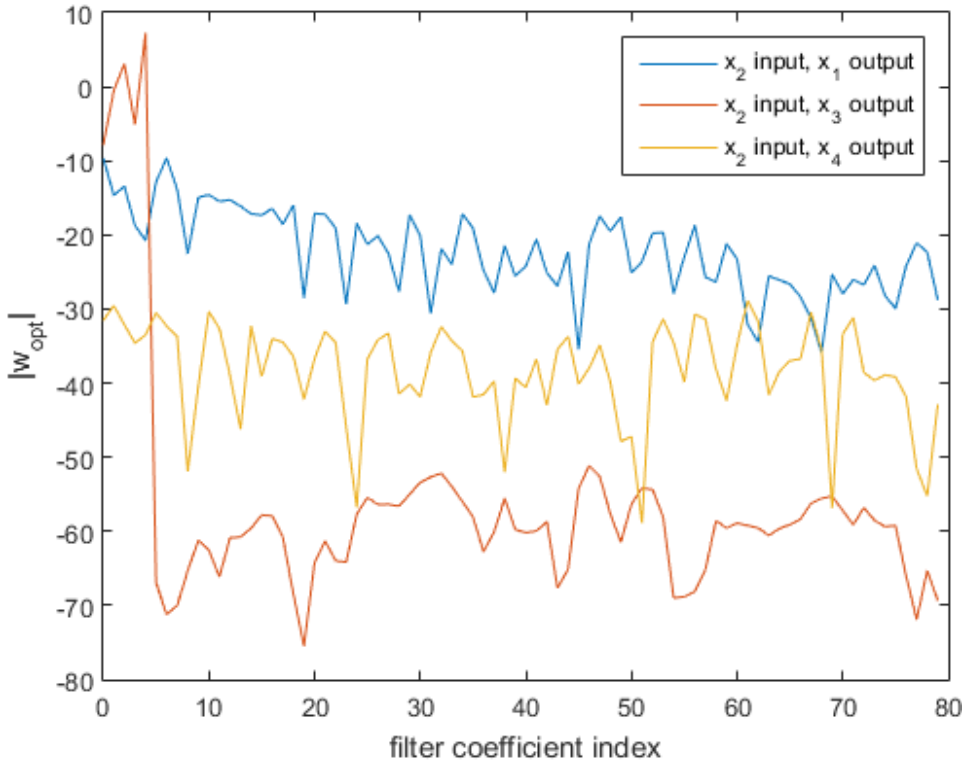



Figure 6.2. Wiener filters estimated if $x_2(n)$ is input and $x_1(n)$, $x_3(n)$, or $x_4(n)$ are output

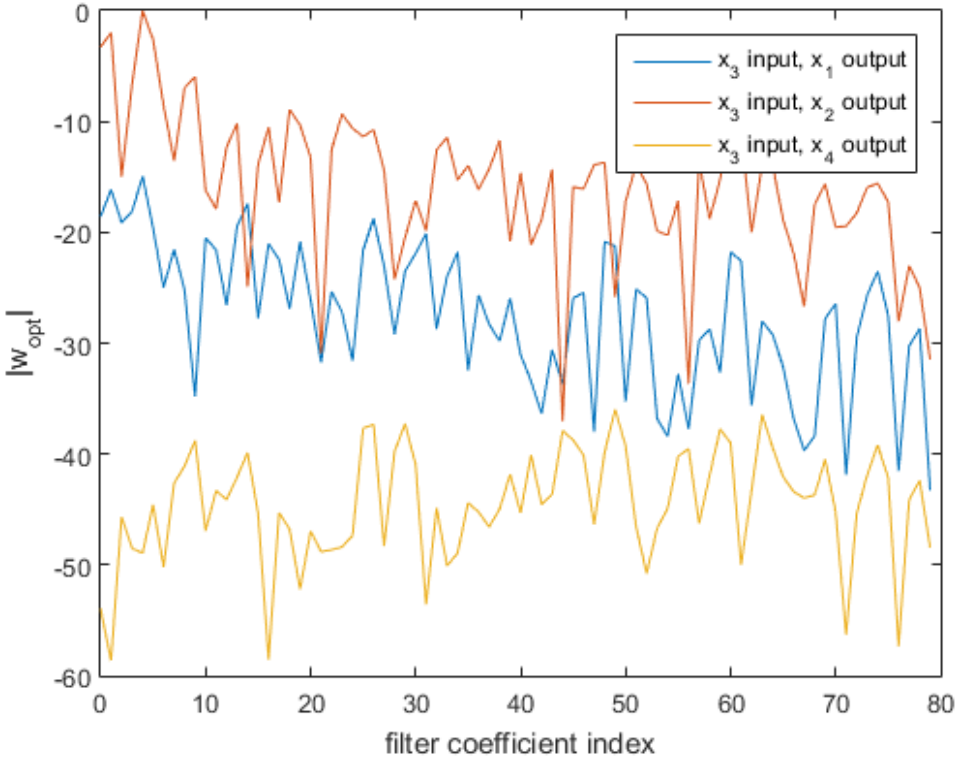



Figure 6.3. Wiener filters estimated if $x_3(n)$ is input and $x_1(n)$, $x_2(n)$, or $x_4(n)$ are output

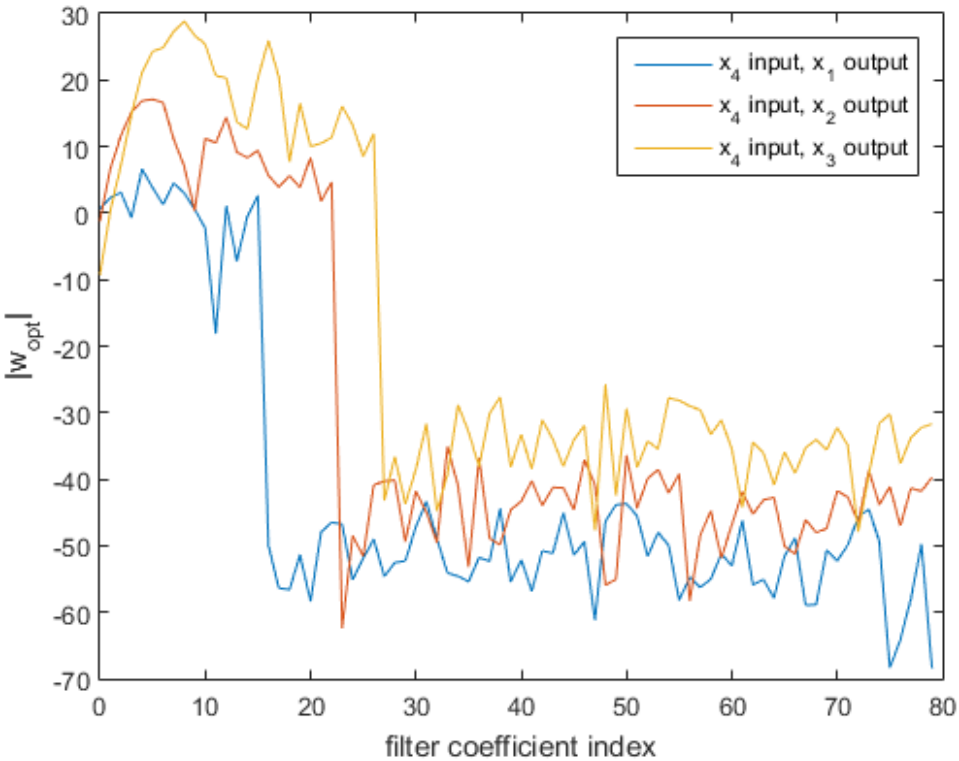



Figure 6.4. Wiener filters estimated if $x_4(n)$ is input and $x_1(n)$, $x_2(n)$, or $x_3(n)$ are output

# Matlab Code for Problem 6

```
clear all
load P6

N = length(x1);
M = 80;

X1 = zeros(M,N-M+1);
X2 = zeros(M,N-M+1);
X3 = zeros(M,N-M+1);
X4 = zeros(M,N-M+1);
for jm = 1:M
    X1(M-jm+1,:) = x1(jm:jm+N-M).';
    X2(M-jm+1,:) = x2(jm:jm+N-M).';
    X3(M-jm+1,:) = x3(jm:jm+N-M).';
    X4(M-jm+1,:) = x4(jm:jm+N-M).';
end;

R_x1 = (1./(N-M+1)).*X1*X1';
R_x1i = inv(R_x1);
R_x2 = (1./(N-M+1)).*X2*X2';
R_x2i = inv(R_x2);
R_x3 = (1./(N-M+1)).*X3*X3';
R_x3i = inv(R_x3);
R_x4 = (1./(N-M+1)).*X4*X4';
R_x4i = inv(R_x4);

D_d1 = repmat(x1(M:N)',M,1);
D_d2 = repmat(x2(M:N)',M,1);
D_d3 = repmat(x3(M:N)',M,1);
D_d4 = repmat(x4(M:N)',M,1);

p_x1d2 = sum(X1.*D_d2 ,2)./(N-M+1);
p_x1d3 = sum(X1.*D_d3 ,2)./(N-M+1);
p_x1d4 = sum(X1.*D_d4 ,2)./(N-M+1);
p_x2d1 = sum(X2.*D_d1 ,2)./(N-M+1);
p_x2d3 = sum(X2.*D_d3 ,2)./(N-M+1);
p_x2d4 = sum(X2.*D_d4 ,2)./(N-M+1);
p_x3d1 = sum(X3.*D_d1 ,2)./(N-M+1);
p_x3d2 = sum(X3.*D_d2 ,2)./(N-M+1);
p_x3d4 = sum(X3.*D_d4 ,2)./(N-M+1);
p_x4d1 = sum(X4.*D_d1 ,2)./(N-M+1);
p_x4d2 = sum(X4.*D_d2 ,2)./(N-M+1);
p_x4d3 = sum(X4.*D_d3 ,2)./(N-M+1);

wx1d2_opt = R_x1i*p_x1d2;
wx1d3_opt = R_x1i*p_x1d3;
wx1d4_opt = R_x1i*p_x1d4;
wx2d1_opt = R_x2i*p_x2d1;
wx2d3_opt = R_x2i*p_x2d3;
wx2d4_opt = R_x2i*p_x2d4;
wx3d1_opt = R_x3i*p_x3d1;
wx3d2_opt = R_x3i*p_x3d2;
wx3d4_opt = R_x3i*p_x3d4;
wx4d1_opt = R_x4i*p_x4d1;
```

```
wx4d2_opt = R_x4i*p_x4d2;
wx4d3_opt = R_x4i*p_x4d3;

figure(1)
plot(0:M-1,20*log10(abs(wx1d2_opt)),0:M-1,20*log10(abs(wx1d3_opt)),0:M-
1,20*log10(abs(wx1d4_opt)));
ylabel('|w_{opt}|')
xlabel('filter coefficient index')
legend('x_{1} input, x_{2} output','x_{1} input, x_{3} output','x_{1}
input, x_{4} output')

figure(2)
plot(0:M-1,20*log10(abs(wx2d1_opt)),0:M-1,20*log10(abs(wx2d3_opt)),0:M-
1,20*log10(abs(wx2d4_opt)));
ylabel('|w_{opt}|')
xlabel('filter coefficient index')
legend('x_{2} input, x_{1} output','x_{2} input, x_{3} output','x_{2}
input, x_{4} output')

figure(3)
plot(0:M-1,20*log10(abs(wx3d1_opt)),0:M-1,20*log10(abs(wx3d2_opt)),0:M-
1,20*log10(abs(wx3d4_opt)));
ylabel('|w_{opt}|')
xlabel('filter coefficient index')
legend('x_{3} input, x_{1} output','x_{3} input, x_{2} output','x_{3}
input, x_{4} output')

figure(4)
plot(0:M-1,20*log10(abs(wx4d1_opt)),0:M-1,20*log10(abs(wx4d2_opt)),0:M-
1,20*log10(abs(wx4d3_opt)));
ylabel('|w_{opt}|')
xlabel('filter coefficient index')
legend('x_{4} input, x_{1} output','x_{4} input, x_{2} output','x_{4}
input, x_{3} output')
```