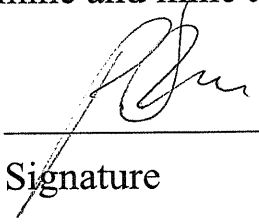
EECS 844 – Fall 2016

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 other 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

11/10/16

Date

RICH SIMEON

Name (printed)

Student ID #

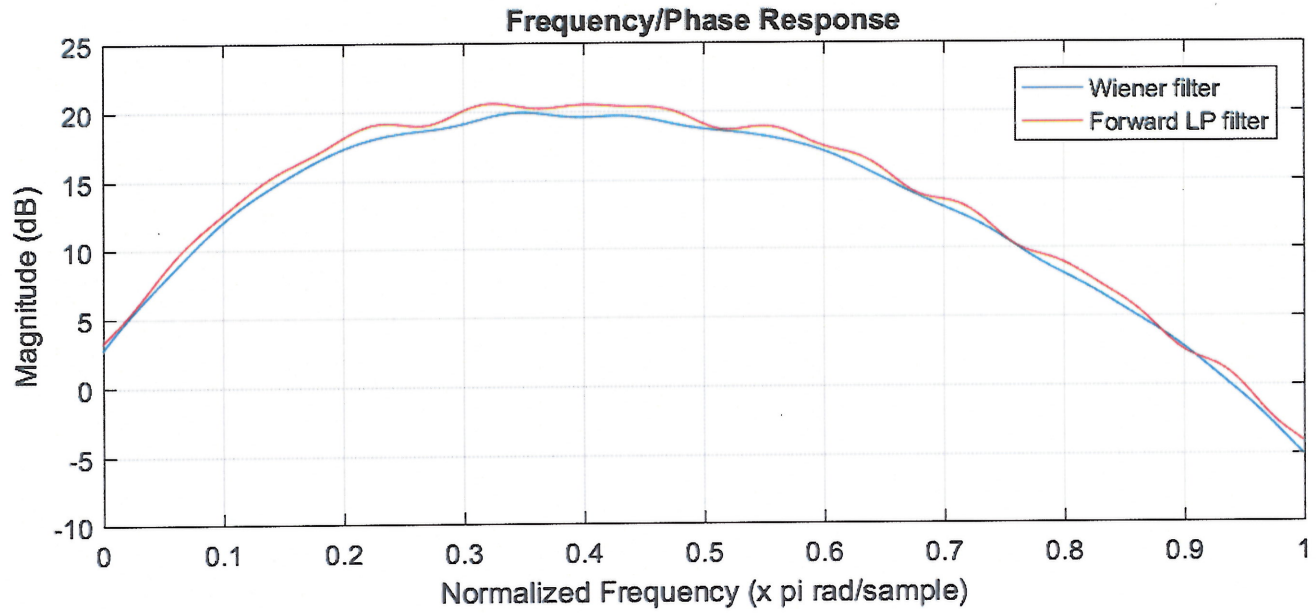
*** Attach as cover page to completed exam.**

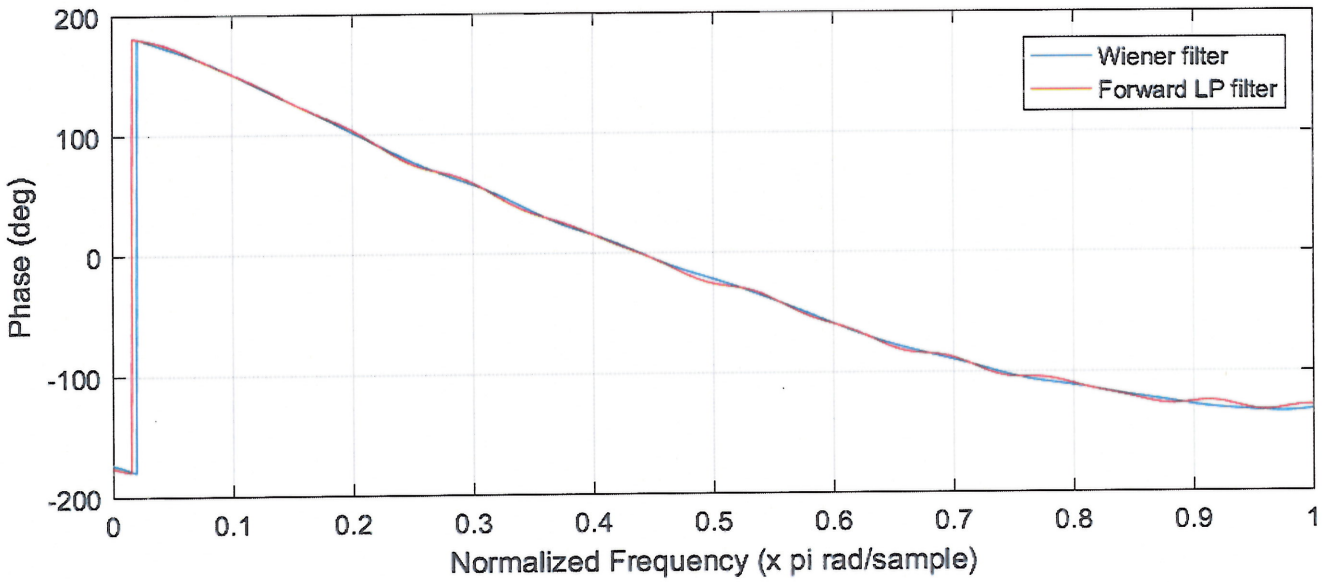
Problem 1

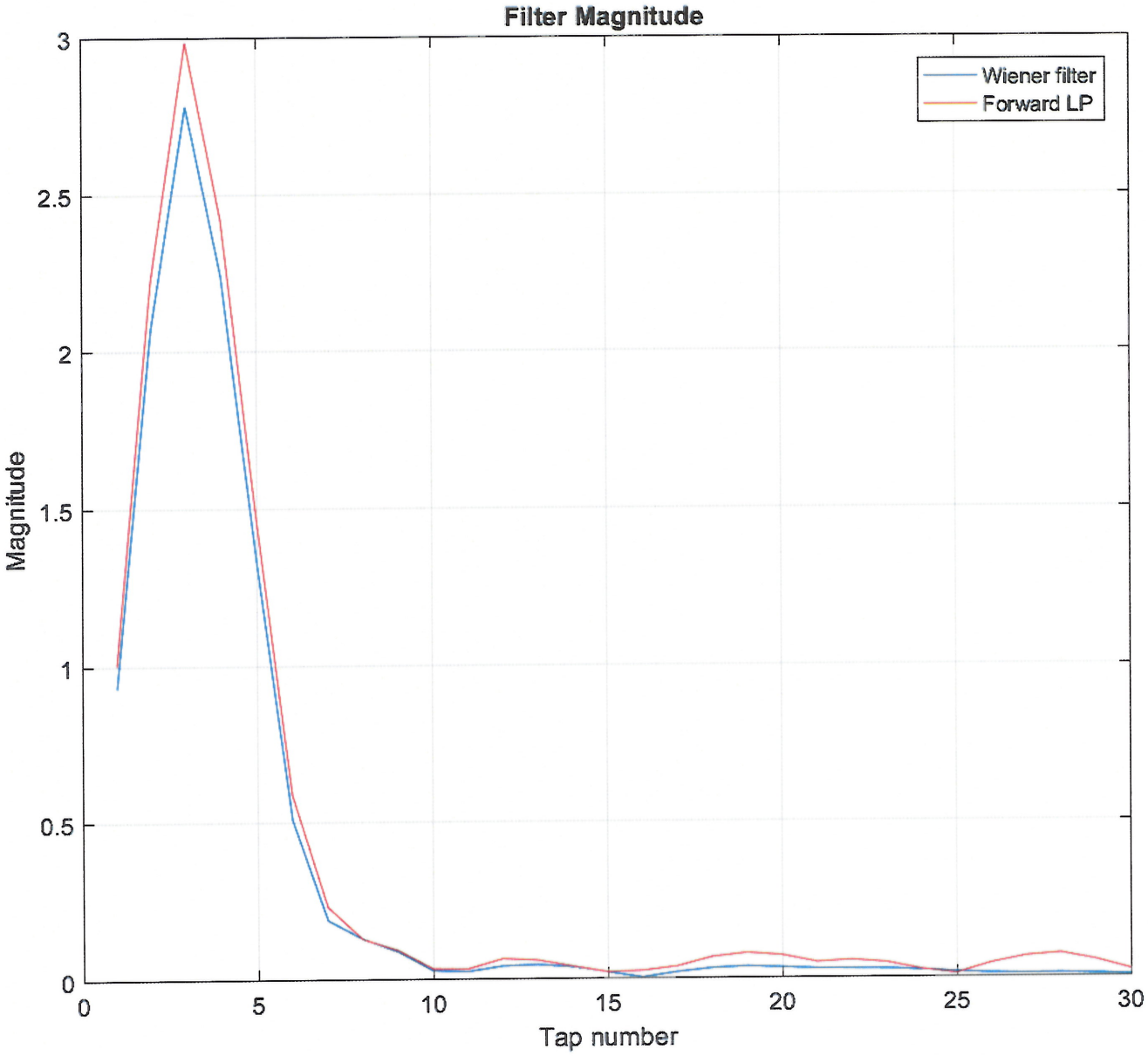
Observation of the Wiener weight solution and the forward linear predictor solution in terms of frequency response shows almost identical responses. The forward LP does exhibit some "lumpiness" in the magnitude response, presumably due to the fact that the LP is using previous samples to form the equivalent cross-correlation matrix (r), as opposed to the Wiener cross-correlation filter (P) which is formed with the desired signal x. The "quality" of r is dependent on the whiteness of the excitation signal x that feeds into the unknown system (which, by inspection of the autocorrelation of x, was shown to be white during the course of solving this problem).

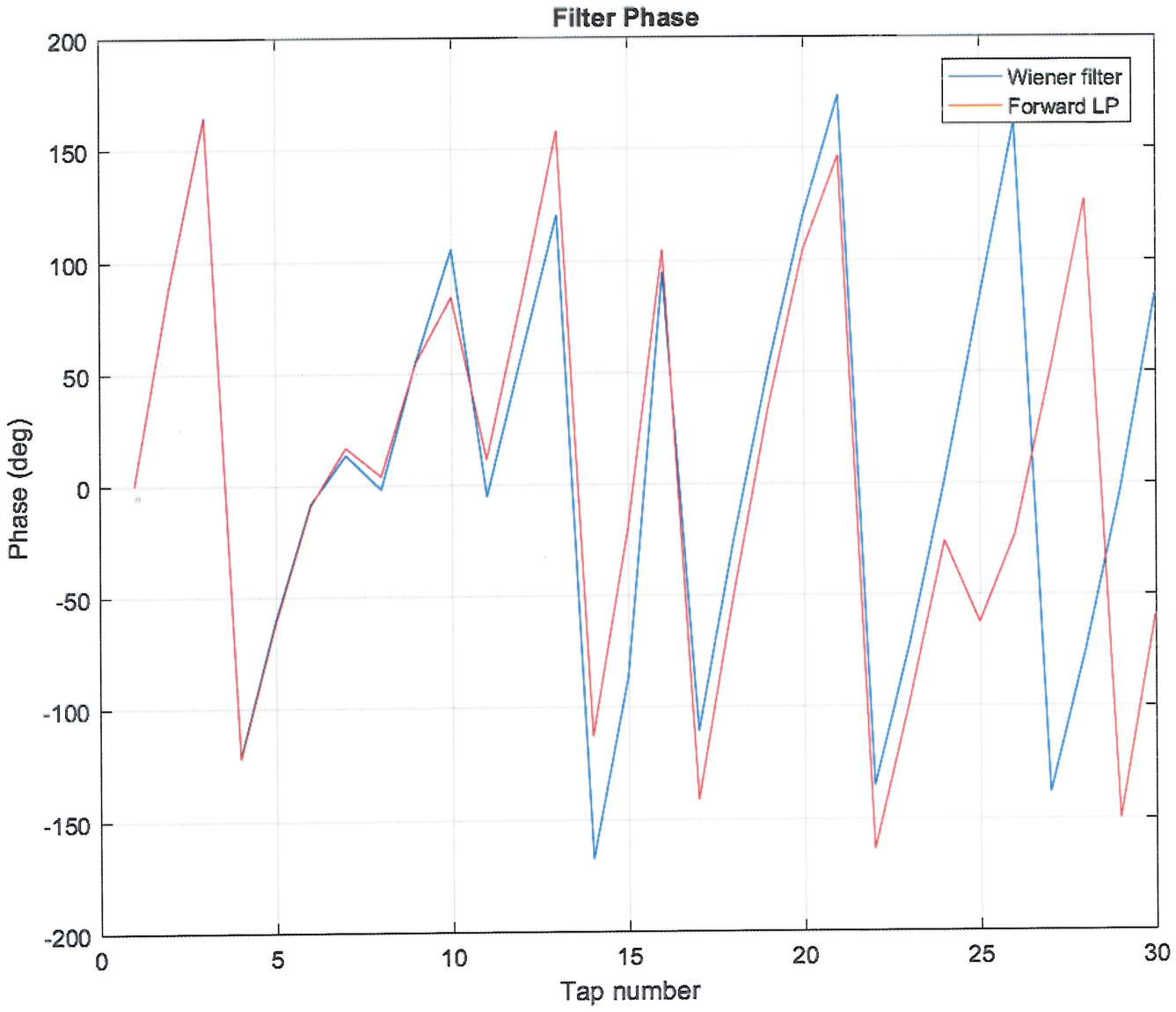
Since x is a white stochastic zero-mean process, the Wiener solution and the forward LP solution both represent the whitening filter that "cancels out" the unknown system's time correlation of the data (or alternatively, flattens the frequency response of the unknown system).

Problem 1


Frequency/Phase Response
Wiener filter
Forward LP filter
Magnitude (dB)
25
20
15
10
5
0
-5
-10
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1
Normalized Frequency (x pi rad/sample)


Wiener filter
Forward LP filter
Phase (deg)
200
100
0
-100
-200
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1
Normalized Frequency (x pi rad/sample)


Filter Magnitude
Wiener filter
Forward LP
Magnitude
Tap number
0
0.5
1
1.5
2
2.5
3
0
5
10
15
20
25
30


Filter Phase
Wiener filter
Forward LP
Phase (deg)
Tap number
200
150
100
50
0
-50
-100
-150
-200
0
5
10
15
20
25
30

```
% Exam 3 Problem 1

clear all;
close all;
hold off;

load p1.mat; % loads x,d

K = length(x);
M_lp = 29; % Forward linear prediction filter order
M_w  = 30; % Wiener filter order

rc = xcorr(d,M_w-1,'unbiased');
Rdd = toeplitz(conj(flipud(rc(1:M_w))));
r = Rdd(2:M_lp+1,1);

% Calculate LP filter (w_lp)
w_lp = inv(Rdd(1:M_lp,1:M_lp))*r;
a = zeros(M_lp+1,1);
a(1) = 1;
a(2:M_lp+1) = -w_lp;

% Calculate P
P = zeros(M_w,1);
for k=M_w:length(x)
    P = P + (flipud(d(k-M_w+1:k)) * conj(x(k)));
end
P = P / (length(x)-M_w+1);

% Calculate Wiener filter (ww)
ww = inv(Rdd)*P;

% Run d through LP filter
y_lp = zeros(10000,1);
for n=M_lp+1+1:10000
    sum = 0;
    for k=1:M_lp+1
        sum = sum + conj(a(k))*d(n-k+1);
    end
    y_lp(n) = sum;
end

% Run d through Wiener filter
y = zeros(10000,1);
for n=M_w+1:10000
    sum = 0;
    for k=1:M_w
        sum = sum + conj(ww(k))*d(n-k+1);
    end
    y(n) = sum;
end
```

```
%% Plots--------------

figure(1);
[H1,W] = freqz(ww);
[H2,W] = freqz(a);
W = linspace(0,1,length(H1));
h0(1)=subplot(2,1,1);
plot(W,20*log10(abs(H1)));
grid on;
hold on;
plot(W,20*log10(abs(H2)),'color','red');
title('Frequency/Phase Response');
ylabel('Magnitude (dB)');
xlabel('Normalized Frequency (x pi rad/sample)');
legend({'Wiener filter','Forward LP filter'});
h0(2)=subplot(2,1,2);
plot(W,angle(H1)*180/pi);
grid on;
hold on;
plot(W,angle(H2)*180/pi,'color','red');
ylabel('Phase (deg)');
xlabel('Normalized Frequency (x pi rad/sample)');
linkaxes(h0,'x');
legend({'Wiener filter','Forward LP filter'});

figure(2);
plot(abs(ww),'color','blue');
grid on;
hold on;
plot(abs(a),'color','red');
title('Filter Magnitude');
ylabel('Magnitude');
xlabel('Tap number');
legend({'Wiener filter','Forward LP'});

figure(3);
plot(angle(ww)*180/pi,'color','blue');
grid on;
hold on;
plot(angle(a)*180/pi,'color','red');
title('Filter Phase');
ylabel('Phase (deg)');
xlabel('Tap number');
legend({'Wiener filter','Forward LP'});
```

Problem 2

As expected, the convergence time for smaller values of mu was much faster than the convergence time of higher values of mu. All three values eventually converge very close to the same minimum cost value of 0.76.

Direct observation of the steering vectors s1 ands2 show that the steering angles are at 39.6 degrees and -21.6 degrees. Examination of the adaptive filter responses show that the filter does in fact converge to have peaks at 39.0 degrees and -20.5 degrees. However, the larger mu values develop large sidelobes outside the constraints; in fact, at mu=1/10N, the sidelobes are higher than the response of the steering angle responses. At mu=1/300N, the sidelobes are much better behaved, at about 8dB below the constrained steering angles.

Derivation: $J(\vec{w}) = \vec{w}^H R \vec{w} + |\vec{w}^H \vec{s}(w_1) - 1|^2 + |\vec{w}^H \vec{s}(w_2) - 0.5|^2$

$\nabla_w \vec{w}^H R \vec{w} = 2R\vec{w}$

$\nabla_w |\vec{w}^H \vec{s}(w_1) - 1|^2 = \nabla_w [Re(\vec{w}^H \vec{s}(w_1) - 1) + j\,Im(\vec{w}^H \vec{s}(w_1))][Re(\vec{w}^H \vec{s}(w_1) - 1) - j\,Im(\vec{w}^H \vec{s}(w_1))]$

$= \nabla_w (Re(\vec{w}^H \vec{s}(w_1) - 1)^2 + Im(\vec{w}^H \vec{s}(w_1))^2$

$= 2(Re(\vec{w}^H \vec{s}(w_1) - 1)\vec{s}(w_1) + 2\,Im(\vec{w}^H \vec{s}(w_1))\vec{s}(w_1)$

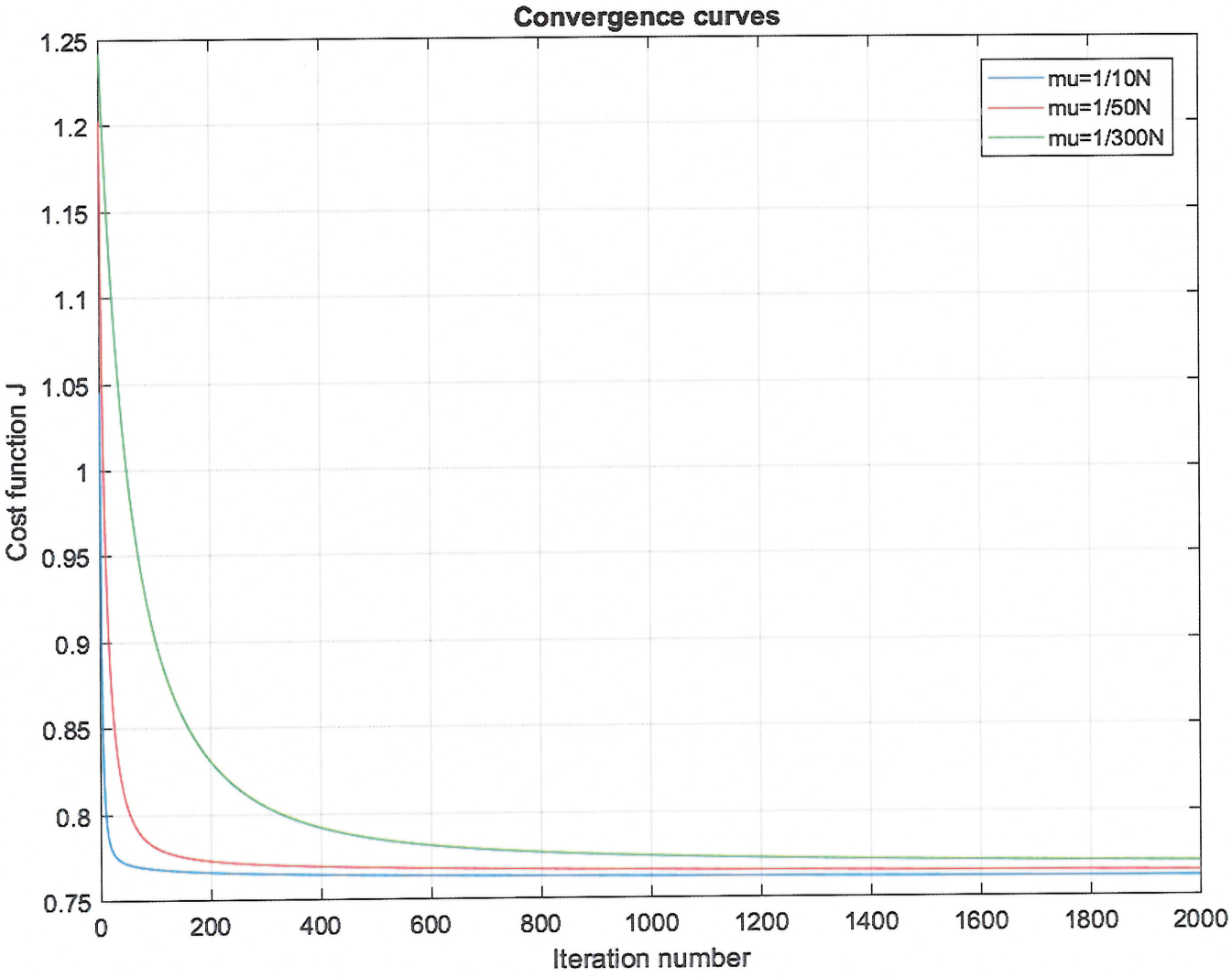
$= 2\vec{s}(w_1)[Re(\vec{w}^H \vec{s}(w_1)) + Im(\vec{w}^H \vec{s}(w_1)) - 1]$

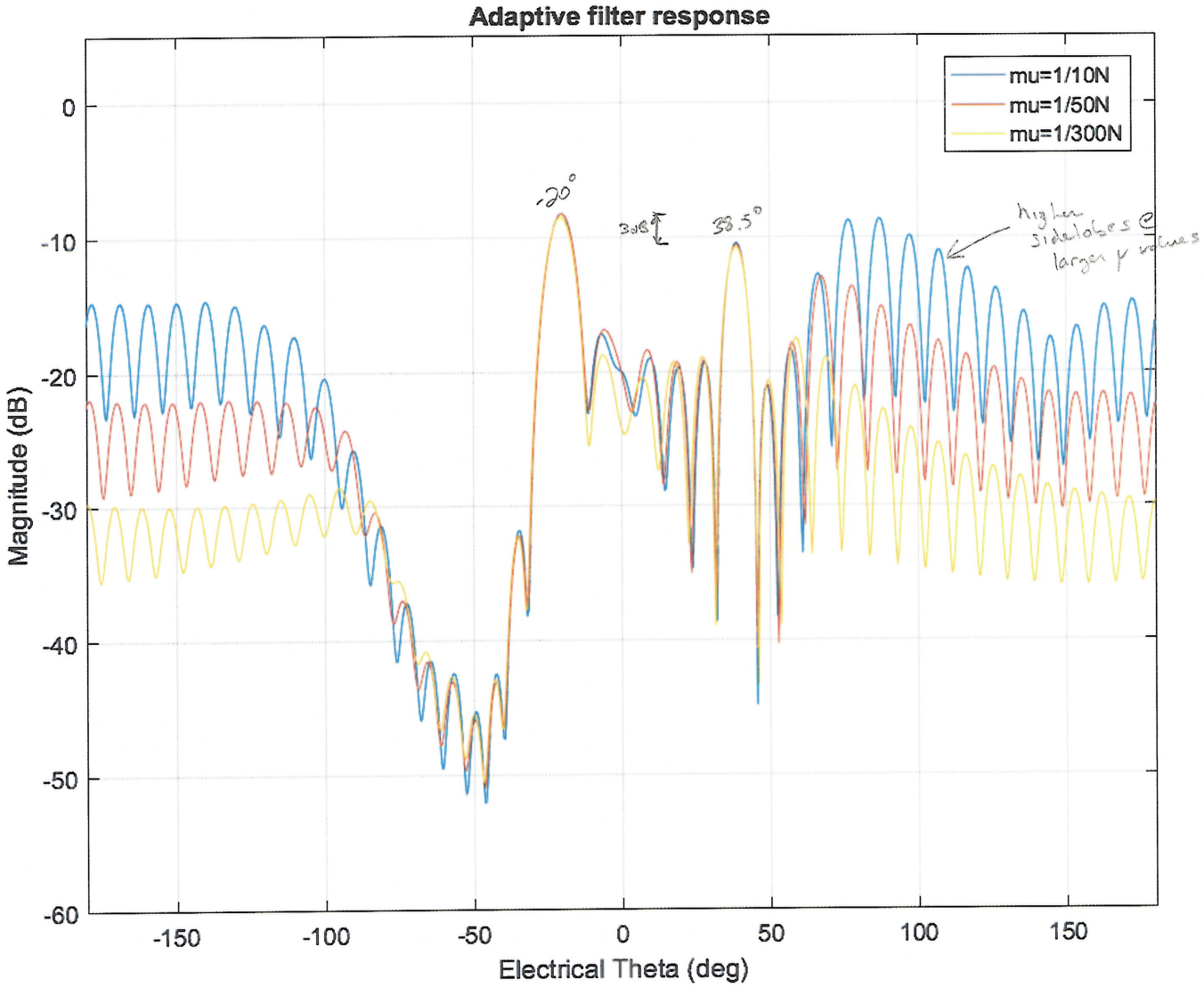
$\nabla_w |\vec{w}^H \vec{s}(w_2) - 0.5|^2 = 2\vec{s}(w_2)[Re(\vec{w}^H \vec{s}(w_2)) + Im(\vec{w}^H \vec{s}(w_2)) - 0.5]$

$\nabla J = 2[R\vec{w} + \vec{s}(w_1)(Re(\vec{w}^H \vec{s}(w_1)) + Im(\vec{w}^H \vec{s}(w_1)) - 1) + \vec{s}(w_2)(Re(\vec{w}^H \vec{s}(w_2)) + Im(\vec{w}^H \vec{s}(w_2)) - 0.5)$

$\vec{w}_{n+1} = \vec{w}_n - 0.5\mu \nabla J$

Problem 2


Convergence curves
mu=1/10N
mu=1/50N
mu=1/300N
Cost function J
Iteration number
1.25
1.2
1.15
1.1
1.05
1
0.95
0.9
0.85
0.8
0.75
0
200
400
600
800
1000
1200
1400
1600
1800
2000


Adaptive filter response
mu=1/10N
mu=1/50N
mu=1/300N
-20°
3dB
33.5°
Higher sidelobes @ larger μ values
Magnitude (dB)
Electrical Theta (deg)
0
-10
-20
-30
-40
-50
-60
-150
-100
-50
0
50
100
150

```
% Exam 3 Problem 2

clear all;
close all;
hold off;

load p2.mat; % loads R,s1,s2

M = 2000;           % number of iterations
N = length(s1);
mu = [1/(10*N) 1/(50*N) 1/(300*N)];     % step size array
J = zeros(M,length(mu));   % cost function for different step sizes

numsamps = 20*N;
beampattern = zeros(numsamps,length(mu));% Frequency responses for converged filters
theta = linspace(-180,180,numsamps);

%% Run steepest-descent algorithm over all step sizes
for step_size_idx=1:length(mu)
    % Initialize
    w = zeros(N,M);     % weight vector, init to zeros

    % Iterate
    for idx=1:M
        del_J = 2*R*w(:,idx) +...
                2*s1*(real(ctranspose(w(:,idx))*s1) + imag(ctranspose(w(:,idx))*s1) - 1)
+ ...
                2*s2*(real(ctranspose(w(:,idx))*s2) + imag(ctranspose(w(:,idx))*s2)
0.5);

        w(:,idx+1) = w(:,idx) - 0.5*mu(step_size_idx)*del_J;
        J(idx,step_size_idx) = ctranspose(w(:,idx+1))*R*w(:,idx+1) +...
                 abs(ctranspose(w(:,idx+1))*s1 - 1)^2 +...
                 abs(ctranspose(w(:,idx+1))*s2 - 0.5)^2;
    end

    for idx=1:numsamps
        % Compute beampattern vs electrical angle
        s_e = transpose(exp(-j*theta(idx)*(pi/180)*linspace(0,N-1,N)));
        beampattern(idx,step_size_idx) = ctranspose(s_e)*w(:,2000);
    end
end
%% Plots------------------------
figure(1);
plot(real(J(:,1)));
hold on;
plot(real(J(:,2)),'color','red');
plot(real(J(:,3)),'color','green');
title('Convergence curves');
legend({'mu=1/10N','mu=1/50N','mu=1/300N'});
ylabel('Cost function J');
```

```
xlabel('Iteration number');
grid on;

figure(2);
plot(theta,20*log10(abs(beampattern(:,1))));
title('Adaptive filter response');
xlabel('Electrical Theta (deg)');
ylabel('Magnitude (dB)');
axis([-180 180 -60 5]);
hold on;
plot(theta,20*log10(abs(beampattern(:,2))));
plot(theta,20*log10(abs(beampattern(:,3))));
grid on;
legend({'mu=1/10N','mu=1/50N','mu=1/300N'});
```

Problem 3

The MVDR weight vector $\vec{w}_o = \dfrac{R^{-1}\vec{s}(\theta)}{s^H(\theta) R^{-1} s(\theta)}$

The GSC weight vector $\vec{w}_o = U\vec{q} = c\vec{v} - C_a \vec{w}_a$

$$= c\vec{v} - C_a \underbrace{\left(C_a^H R C_a\right)^{-1} C_a^H R c\vec{v}}_{\vec{w}_a}$$

$$= \left(I - C_a (C_a^H R C_a)^{-1} C_a^H R\right) c\vec{v}$$

From Apolinario,

$$C_a (C_a^T C_a)^{-1} C_a^T = I - C (C^T C)^{-1} C^T$$

for $C_a^T C = \vec{0}$ (C_a spans the null space of C)

From Breed & Strauss, (eq 6-7)

$$C^T (C^T R^{-1} C)^{-1} C R^{-1} = I - C_a (C_a^T R C_a)^{-1} C_a^T R$$

Hence

$$\vec{w}_o = \left(I - C_a (C_a^H R C_a)^{-1} C_a^H R\right) c\vec{v}$$

$$= c^H (C^H R^{-1} C)^{-1} C^H R^{-1} c\vec{v}$$

$$= c^H (C^H R^{-1} C)^{-1} C R^{-1} C \underbrace{(C^H C)^{-1} \vec{g}}_{\vec{v}}$$

Substituting $\vec{g} = 1$, $C = s(\theta)$

$$\vec{w}_o = \frac{s(\theta)^H s(\theta) \cdot R^{-1} s(\theta)(1)}{s(\theta)^H s(\theta)\; s(\theta)^H R^{-1} s(\theta)} = \frac{R^{-1} s(\theta)}{s(\theta)^H R^{-1} s(\theta)} = \text{MVDR } w_o$$

Problem 4

Mismatch filter	Mismatch loss (dB)
LS Mismatch Filter	5.8
LS Mismatch Filter, modified A	5.2
LS Mismatch Filter, diagonally loaded	0.14
LS Mismatch Filter, modified and diagonally loaded	2.2

The normalized matched filter is the most basic calculation for deriving a matched filter for x, but as shown in the convolution plot of the NMF with x, it has a peak only about 15-20dB high, and is slightly wider than an impulse.

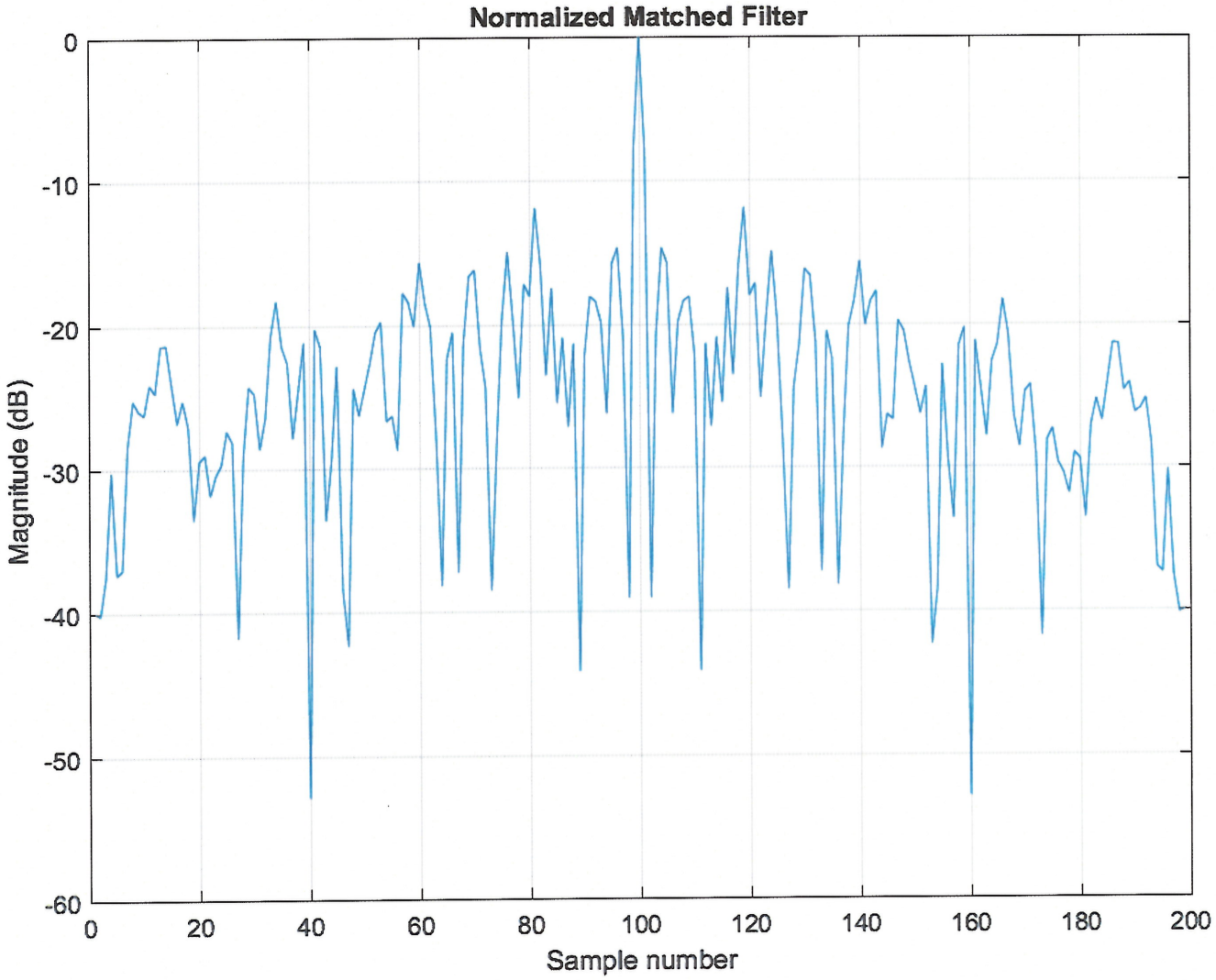
The LS Mismatch Filter has the sharpest peak out of all the different versions of the mismatched filter, but also the highest mismatch loss (and thus a lower SNR). The peak is about 40dB high, much higher and sharper than the NMF.

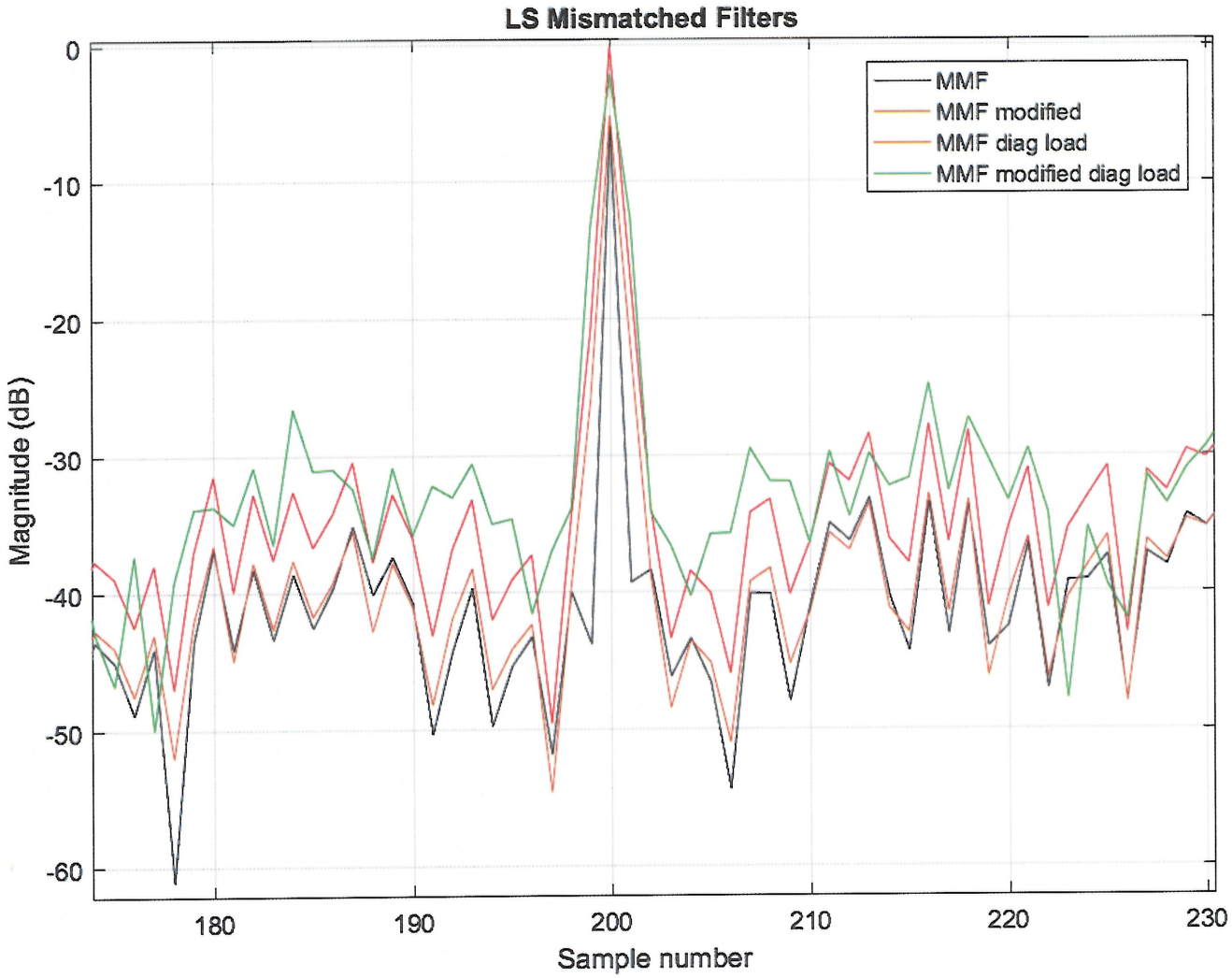
By modifying the A matrix to lose the surrounding rows with zeros, the mismatch loss is slightly smaller, but the peak is slightly wider, meaning that the mismatch filter performance is worse for detailed deconvolution. Losing the rows around the impulse in the elementary vector has the effect of widening the deconvolved signal peak slightly.

Diagonal loading of A^HA using 2% of the largest eigenvalue of A^HA transforms A^HA very close into the identity matrix, meaning the LS mismatched filter approximates the normalized matched filter. This is supported by the fact that the mismatch loss is almost 0dB. While this maximizes the SNR, it is not ideal for deconvolution. However, the noise levels (levels outside of the peak) are comparable to that of the modified A^HA matrix filter.

The combination of diagonal loading and modification of A^HA through row removal shows a mismatch loss better than the LS mismatched filter but slightly worse than the diagonally-loaded mismatched filter. This combination seems to have the widest peak of all the mismatched filter (worst deconvolution performance) but preserves the most SNR.

Problem 4


Normalized Matched Filter
Magnitude (dB)
0
-10
-20
-30
-40
-50
-60
0
20
40
60
80
100
120
140
160
180
200
Sample number


LS Mismatched Filters
MMF
MMF modified
MMF diag load
MMF modified diag load
Magnitude (dB)
0
-10
-20
-30
-40
-50
-60
180
190
200
210
220
230
Sample number

Problem 5

Deconvolution of the received signal y(n) with the known signal x(n) using the previously-generated mismatch filters shows the effects of estimating the unknown system using filters with different resolutions (via peak impulse width) vs. different SNRs (via peak impulse height).

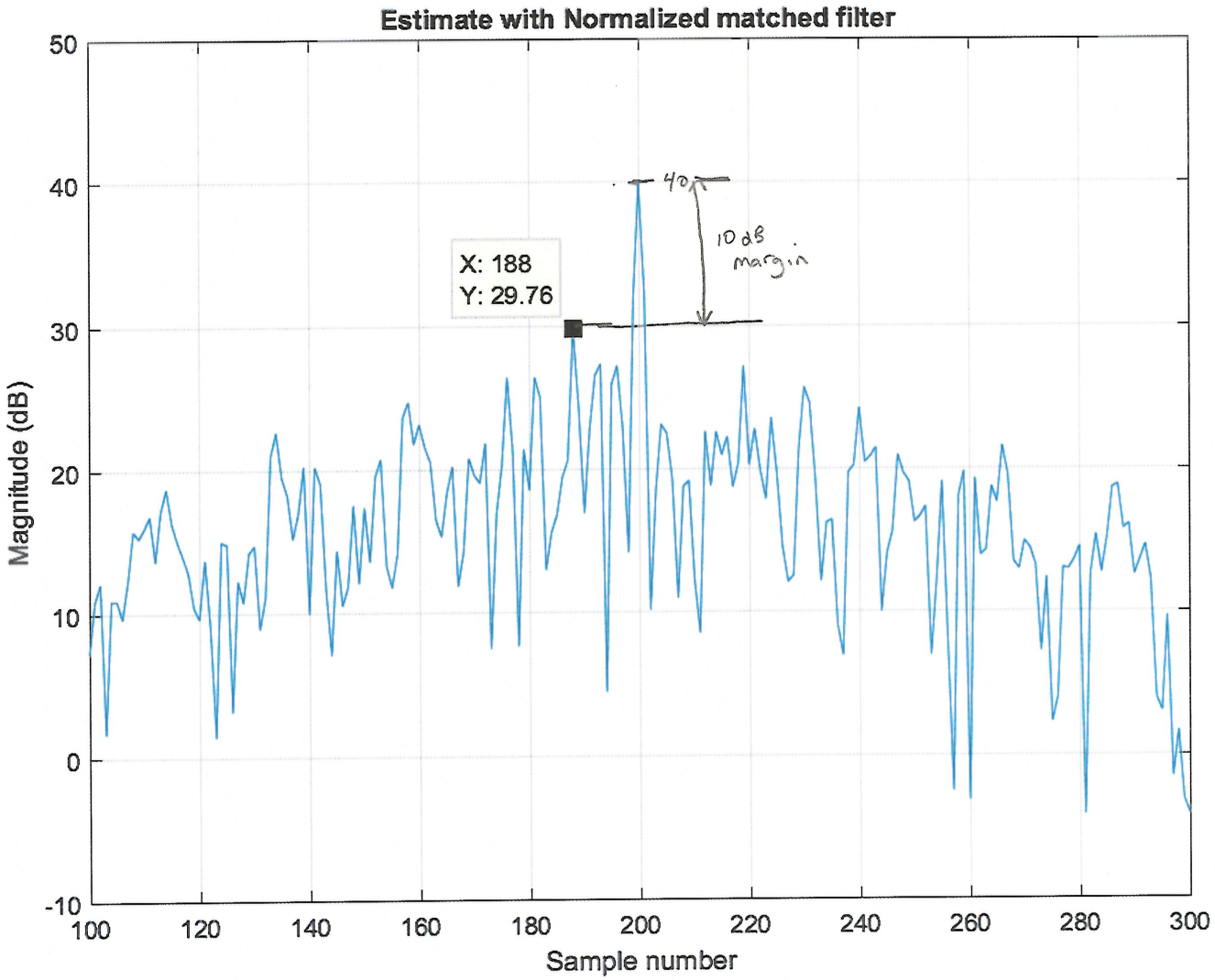
All plots are zoomed in near detected correlation peaks to show details.

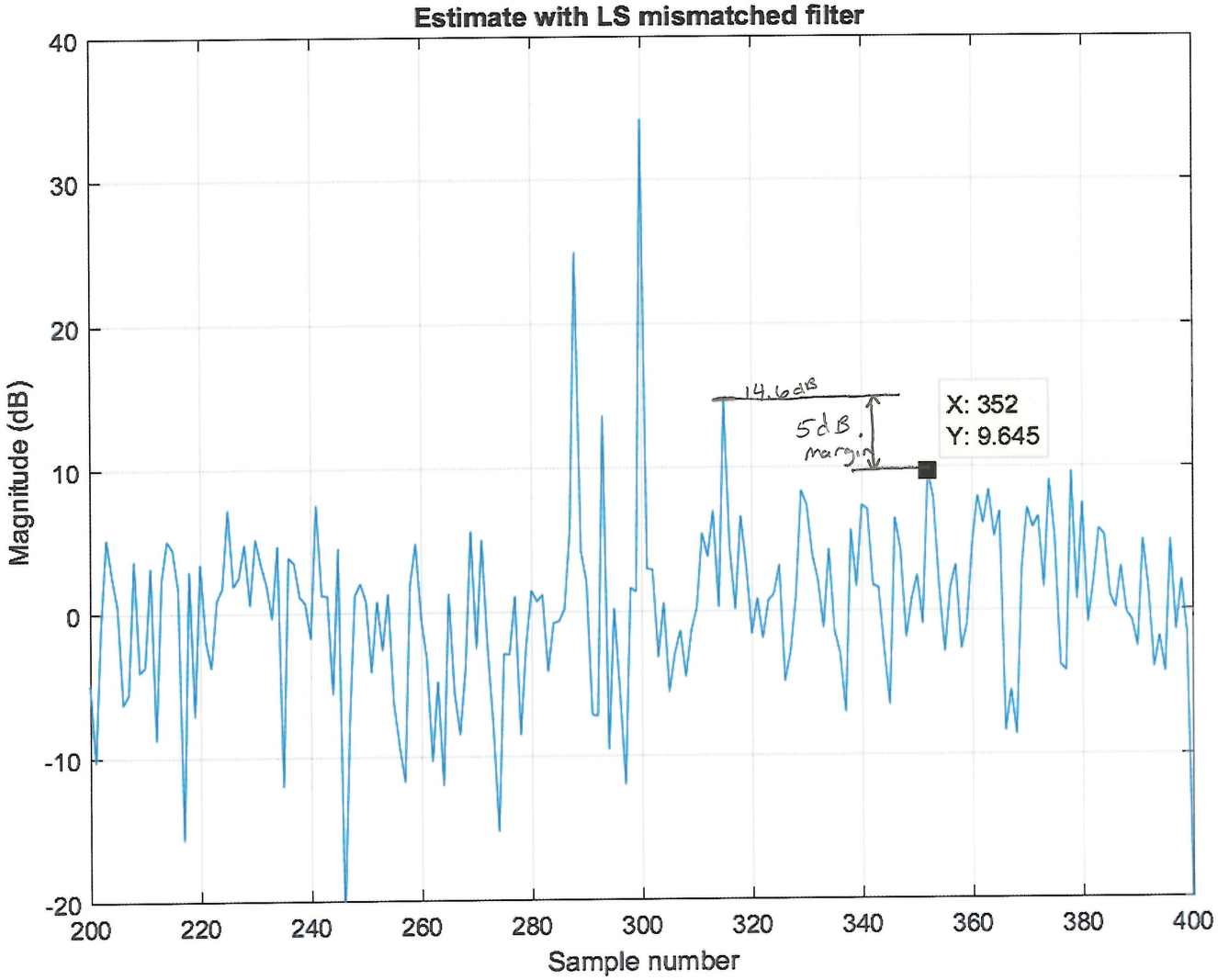
The convolution of the normalized matched filter with y(n) shows a single correlated peak at 200 samples, and the peak is unique enough that it is 10dB above the surrounding samples (noise floor), indicating good SNR performance.

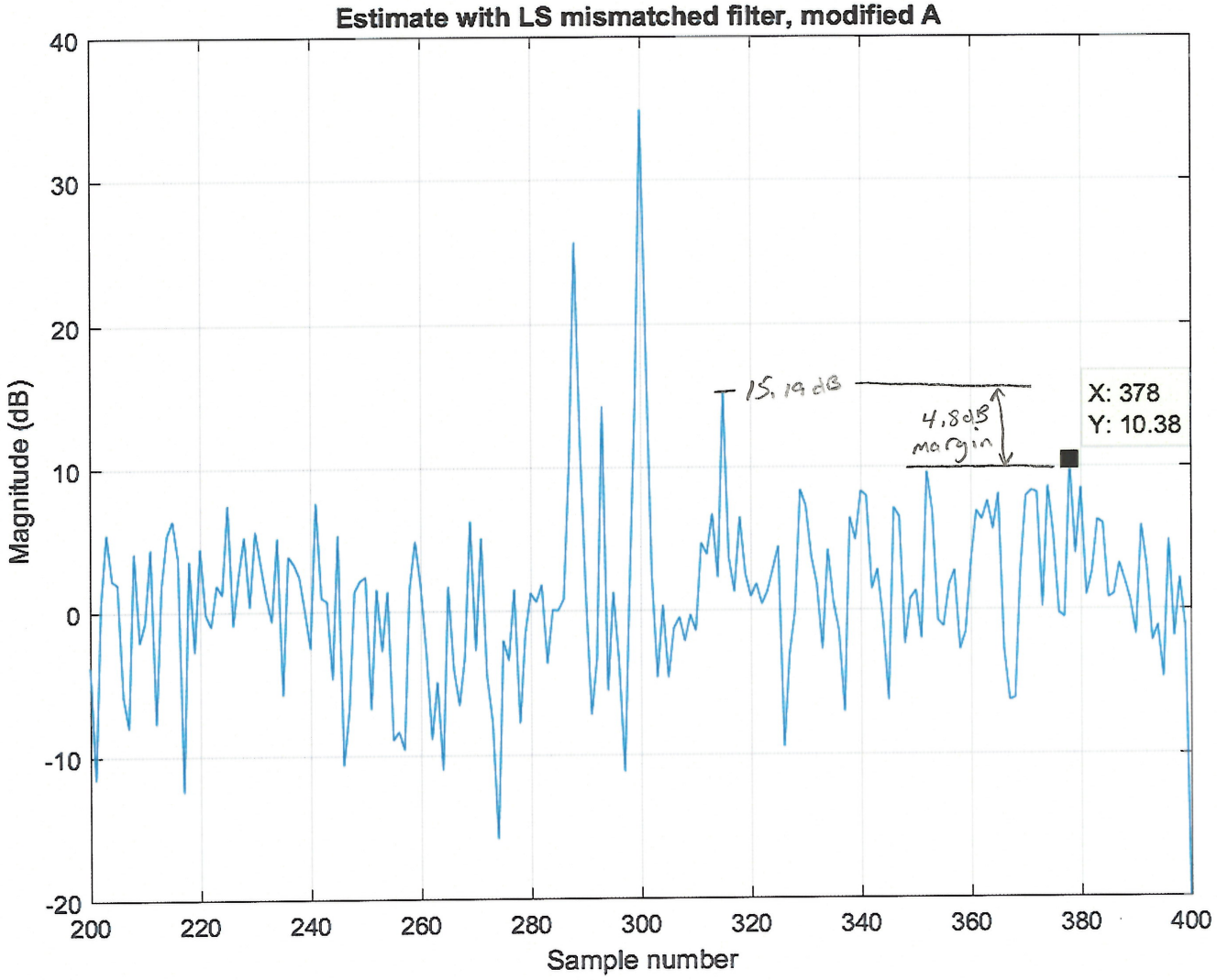
In contrast, the remaining four mismatches filters all show four unique peaks at n = {288, 293, 300, and 315}, indicating that the normalized matched filter's SNR was not good enough to discern the other peaks that the mismatched filters caught. While they all showed more peaks, the peaks were lower relative to the noise floor (not as high as the normalized matched filter's peak).

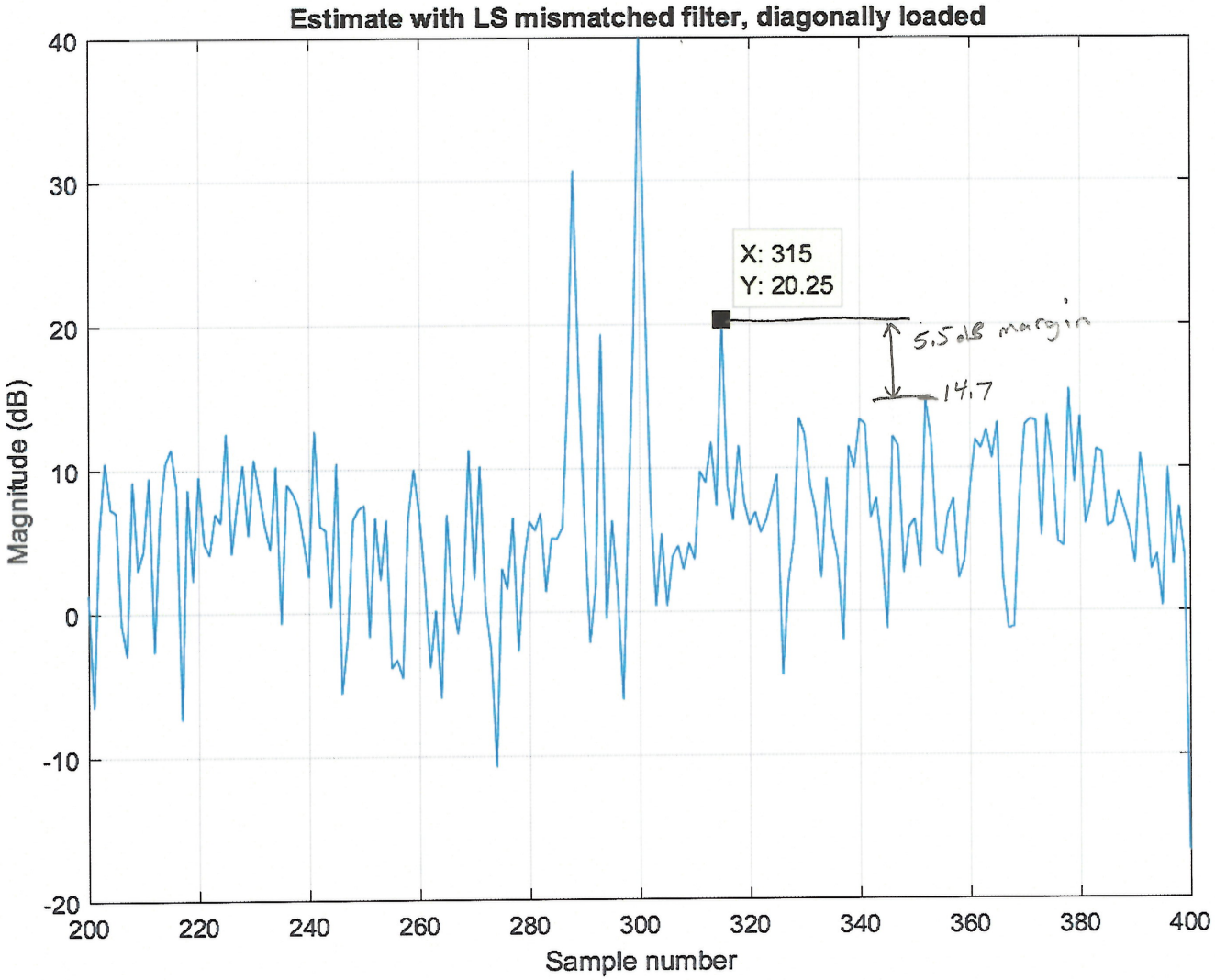
- The best performing (in terms of peak "uniqueness" relative to the noise floor) is the LS mismatched filter, as the lowest peak was 5dB above the noise floor.
- The LS mismatched filter with modified A performed similarly, albeit with slightly "fatter" correlation peaks; the noise floor was still below 10dB.
- When the filter matrix was diagonally-loaded, the noise floor increased slightly such that the highest peak was not as high as the previous two (the peaks were slightly fatter), but the lowest peak had the highest noise margin at 5.5dB (see plot). This was expected, since diagonal loading was shown to have the best SNR but worst decorrelator performance (as measured by the peak widths).
- Finally, the diagonally loaded modified A decorrelator had the lowest peak height for the tallest peak, but maintained a reasonable noise margin of 4.4dB for the lowest peak relative to the noise.

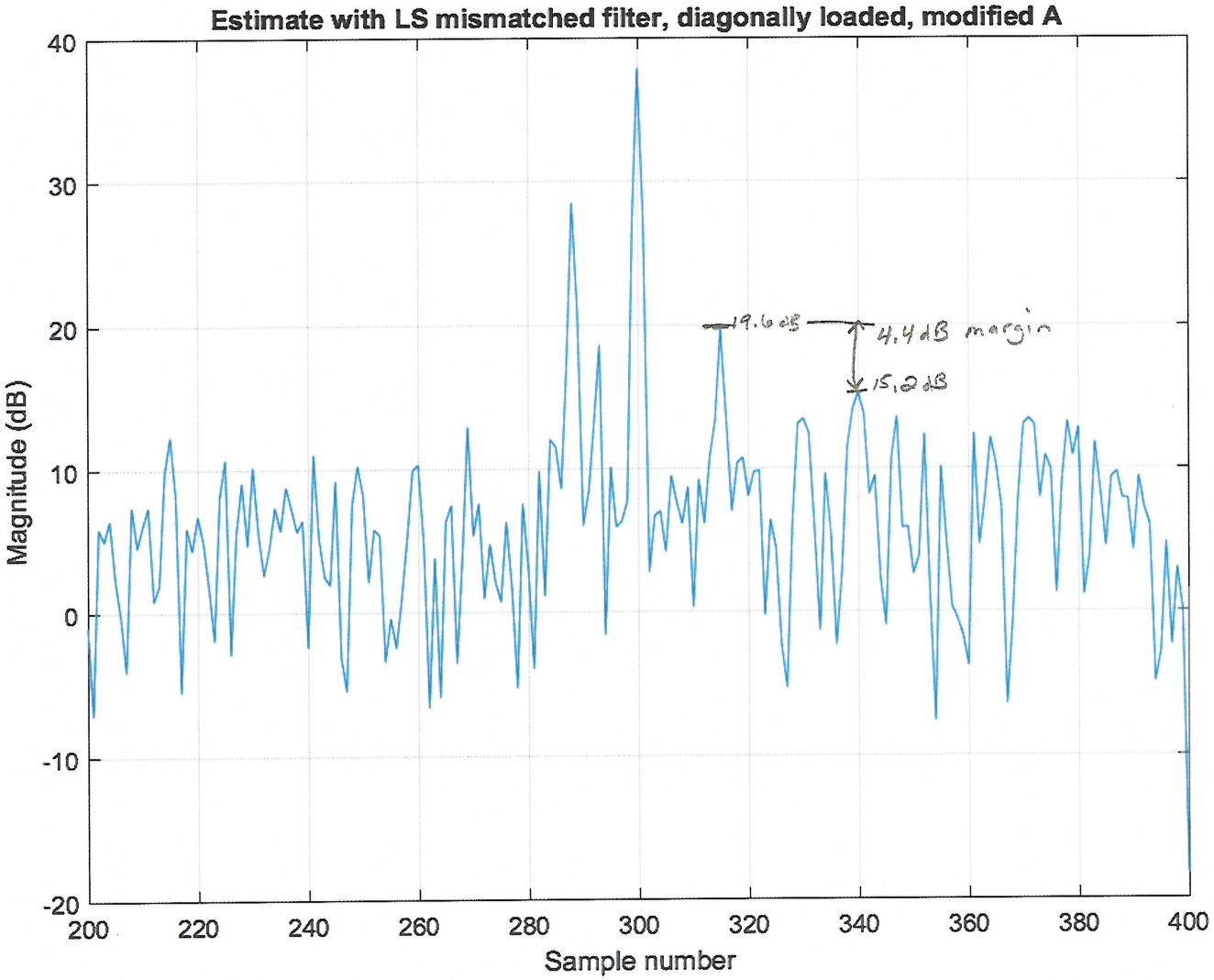
Problem 5


Estimate with Normalized matched filter
Magnitude (dB)
Sample number
X: 188
Y: 29.76
40
10 dB margin
50
40
30
20
10
0
-10
100
120
140
160
180
200
220
240
260
280
300


Estimate with LS mismatched filter
Magnitude (dB)
Sample number
40
30
20
10
0
-10
-20
200
220
240
260
280
300
320
340
360
380
400
14.6dB
5dB
margin
X: 352
Y: 9.645


Estimate with LS mismatched filter, modified A
Magnitude (dB)
Sample number
40
30
20
10
0
-10
-20
200
220
240
260
280
300
320
340
360
380
400
15.19 dB
4.8 dB
margin
X: 378
Y: 10.38


Estimate with LS mismatched filter, diagonally loaded
X: 315
Y: 20.25
5.5dB margin
14.7
Magnitude (dB)
Sample number
40
30
20
10
0
-10
-20
200
220
240
260
280
300
320
340
360
380
400


Estimate with LS mismatched filter, diagonally loaded, modified A
Magnitude (dB)
Sample number
40
30
20
10
0
-10
-20
200
220
240
260
280
300
320
340
360
380
400
19.6 dB
4.4 dB margin
15.2 dB

```
% Exam 3 Problem 4+5

clear all;
close all;
hold off;

load p4.mat; % loads x

M = length(x);
N = 2*M;
m = 2*M;

%% Compute normalized matched filter (h_nmf)
h_nmf = flipud(conj(x))/(ctranspose(x)*x);

%% Compute LS mismatched filter (h_mmf)

% Form A
a = zeros(3*M,1);
a(1:M) = x;
A = toeplitz(a,zeros(N,1));

% Form elementary vector e
e = zeros(3*M,1);
e(m) = 1;

h_mmf = inv(ctranspose(A)*A)*ctranspose(A)*e;
h_nmmf = h_mmf/(sqrtm(ctranspose(h_mmf)*h_mmf)*sqrtm(ctranspose(x)*x));
mismatch_loss = -20*log10( max(abs(conv(h_nmmf,x)))/max(abs(conv(h_nmf,x))) );
fprintf('Mismatch loss, LS mismatched filter: %.2f dB\n',mismatch_loss);

%% Compute LS mismatched filter with modified A
A_mod = A;
A_mod(m-1,:) = zeros(1,size(A,2));% zero out (m-1)th row
A_mod(m+1,:) = zeros(1,size(A,2));% zero out (m+1)th row

h_mmf_mod = inv(ctranspose(A_mod)*A_mod)*ctranspose(A_mod)*e;
h_nmmf_mod = h_mmf_mod/(sqrtm(ctranspose(h_mmf_mod)*h_mmf_mod)*sqrtm(ctranspose(x)*x));
mismatch_loss = -20*log10( max(abs(conv(h_nmmf_mod,x)))/max(abs(conv(h_nmf,x))) );
fprintf('Mismatch loss, LS mismatched filter with modified A: %.2f dB\n',mismatch_loss);

%% Compute LS mismatched filter with diagonal loading
h_mmf_diag = inv(ctranspose(A)*A + 0.02*max(eig(ctranspose(A)*A))*eye(N,N))*ctranspose(A)
*e;
h_nmmf_diag = h_mmf_mod/(sqrtm(ctranspose(h_mmf_diag)*h_mmf_diag)*sqrtm(ctranspose(x)
*x));
mismatch_loss = -20*log10( max(abs(conv(h_nmmf_diag,x)))/max(abs(conv(h_nmf,x))) );
fprintf('Mismatch loss, LS mismatched filter with diagonal loading: %.2f dB\n',
mismatch_loss);

%% Compute LS mismatched filter with modified A and diagonal loading
```

```
h_mmf_mod_diag = inv(ctranspose(A_mod)*A_mod + 0.02*max(eig(ctranspose(A_mod)*A_mod))*eye
(N,N))*ctranspose(A_mod)*e;
h_nmmf_mod_diag = h_mmf_mod_diag/(sqrtm(ctranspose(h_mmf_mod_diag)*h_mmf_mod_diag)*sqrtm
(ctranspose(x)*x));
mismatch_loss = -20*log10( max(abs(conv(h_nmmf_mod_diag,x)))/max(abs(conv(h_nmf,x))) );
fprintf('Mismatch loss, LS mismatched filter with modified A and diagonal loading: %.2f
dB\n',mismatch_loss);

%% Plots (Problem 4)---------------

figure(1);
plot(20*log10(abs(conv(x,h_nmf))));
grid on;
title('Normalized Matched Filter');
ylabel('Magnitude (dB)');
xlabel('Sample number');

figure(2);
plot(20*log10(abs(conv(x,h_nmmf))),'color','black');
grid on;
title('LS Mismatched Filters');
hold on;
plot(20*log10(abs(conv(x,h_nmmf_mod))));
plot(20*log10(abs(conv(x,h_nmmf_diag))),'color','red');
plot(20*log10(abs(conv(x,h_nmmf_mod_diag))),'color','green');
legend({'MMF','MMF modified','MMF diag load','MMF modified diag load'});
ylabel('Magnitude (dB)');
xlabel('Sample number');

%% Plots (Problem 5)-------------
figure(3);
plot(20*log10(abs(conv(y,h_nmf))));
grid on;
title('Estimate with Normalized matched filter');
ylabel('Magnitude (dB)');
xlabel('Sample number');
axis([100 300 -10 50]);

figure(4);
plot(20*log10(abs(conv(y,h_nmmf))));
grid on;
title('Estimate with LS mismatched filter');
ylabel('Magnitude (dB)');
xlabel('Sample number');
axis([200 400 -20 40]);

figure(5);
plot(20*log10(abs(conv(y,h_nmmf_mod))));
grid on;
title('Estimate with LS mismatched filter, modified A);
ylabel('Magnitude (dB)');
```

```
xlabel('Sample number');
axis([200 400 -20 40]);

figure(6);
plot(20*log10(abs(conv(y,h_nmmf_diag))));
grid on;
title('Estimate with LS mismatched filter, diagonally loaded');
ylabel('Magnitude (dB)');
xlabel('Sample number');
axis([200 400 -20 40]);

figure(7);
plot(20*log10(abs(conv(y,h_nmmf_mod_diag))));
grid on;
title('Estimate with LS mismatched filter, diagonally loaded, modified A');
ylabel('Magnitude (dB)');
xlabel('Sample number');
axis([200 400 -20 40]);
```

Problem 6

From the non-adaptive power spectrum estimate, there appears to be 3 distinct signals of interest impinging on the array at {-61.5, -17.7, and 44.8} degrees. It can be expected to have the MVDR power spectrum estimate be more precise than the non-adaptive estimate; however, the MVDR power spectrum estimate shows 6 signals at {-61.5, -16.5, 40.5, 45.4, 95.3, and 112.7} degrees, but at very low power levels (almost 290dB below the non-adaptive estimate). This huge discrepancy is due to the autocorrelation matrix R being very ill-conditioned (condition number of 1.34e18). It was so low that it is shown on a separate plot.

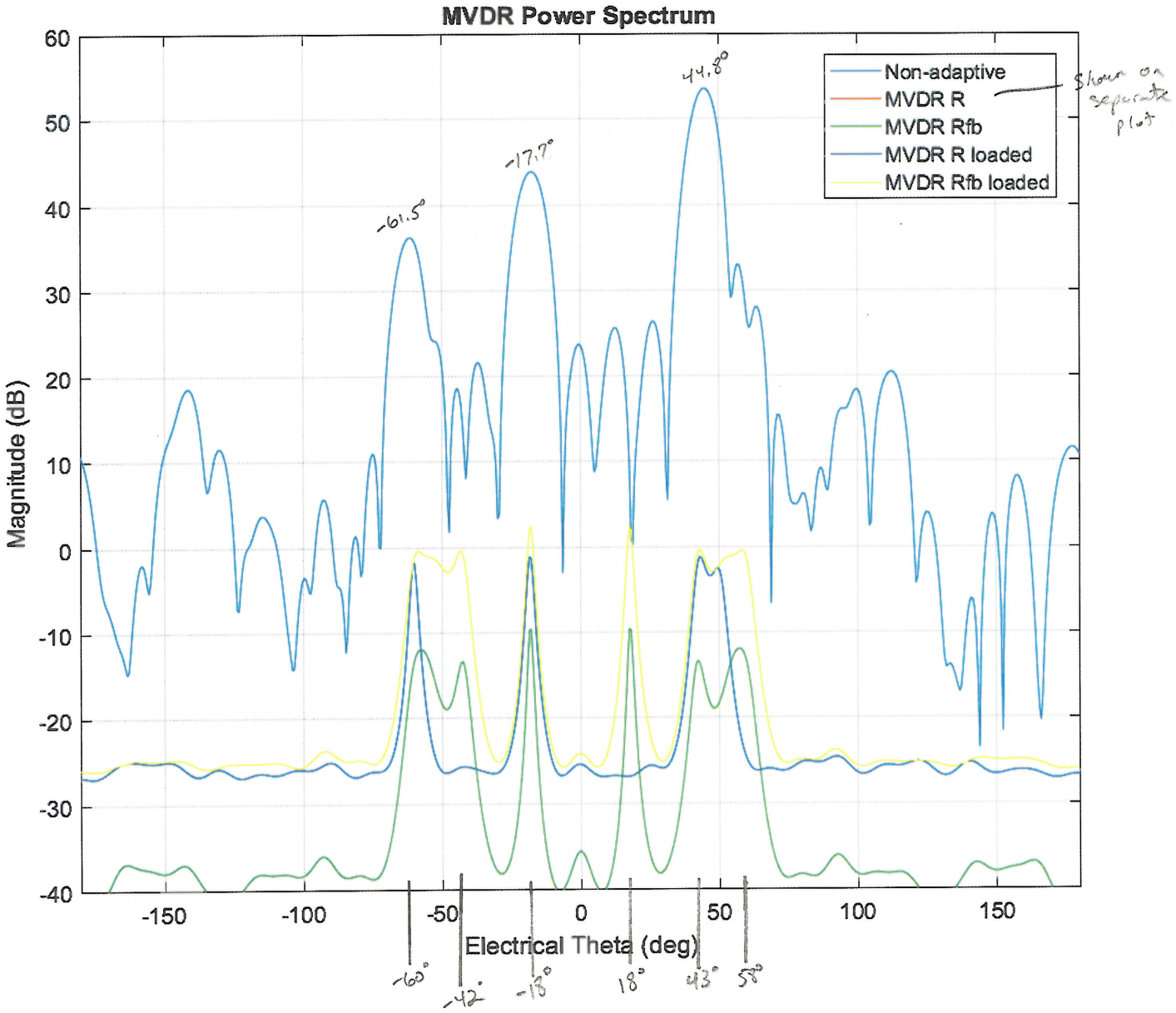
By using forward/backward averaging, the condition number of the R matrix (denoted Rfb) is decreased by orders of magnitude, to 9823. The corresponding MVDR (denoted by the green Rfb line in the plot) has well defined peaks at {-57.4, -42.4, -17.7, 18.3, 43.0, and 58.0} degrees.

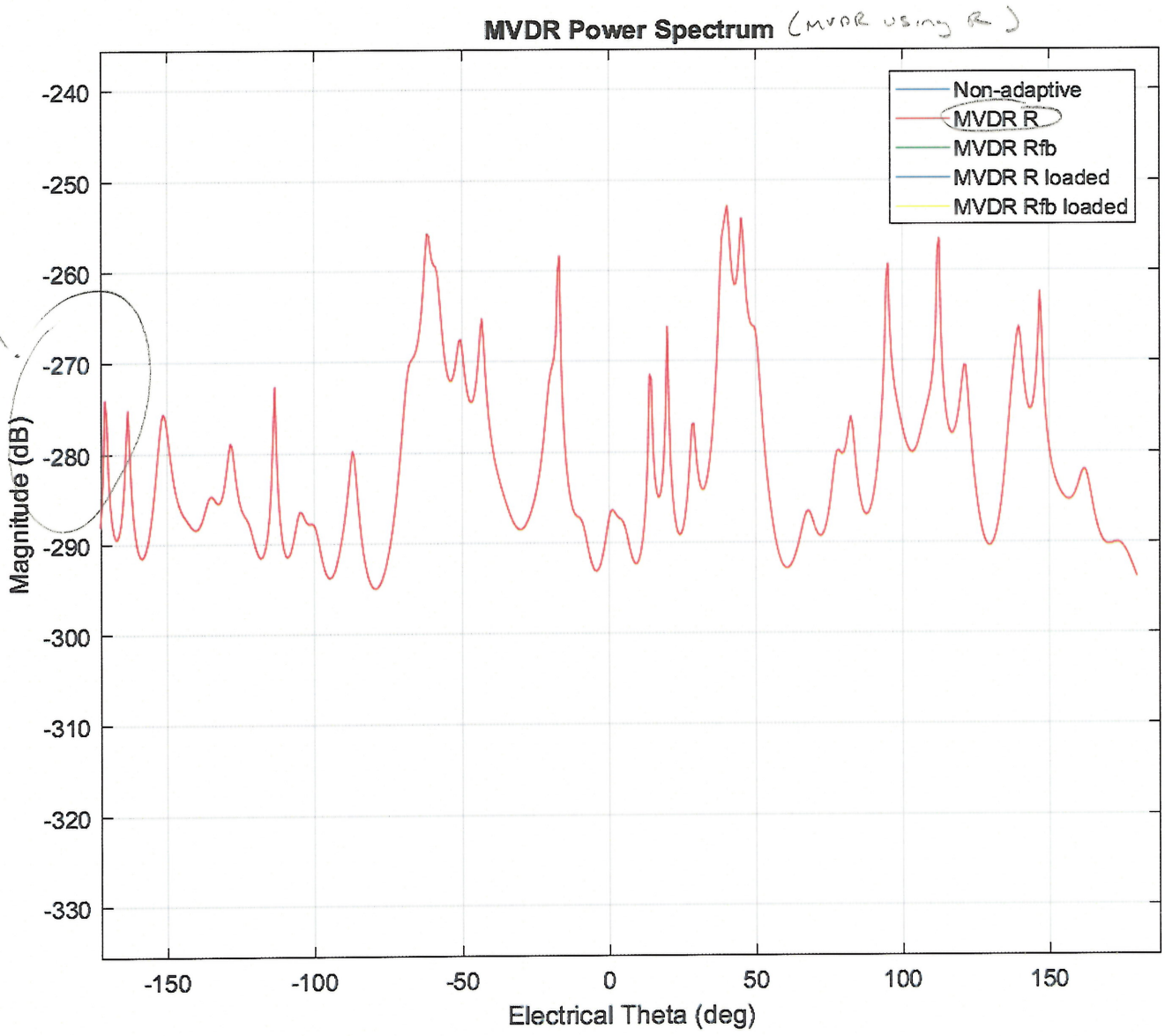
Diagonal loading of the original R matrix (to make the matrix more well-conditioned, in particular condition number = 1312) shows a smoother response (shown in purple) than the unloaded (original R) matrix, but the effect of the dominant diagonal is to smooth out the response; thus, only 4 peaks are seen at {-59.8, -18.3, 43.0, and 49.6} degrees.

Diagonal loading of the Rfb (forward-backward averaged) autocorrelation matrix again "smoothed out" the spectral response of the original Rfb MVDR; the peaks were higher (denoted by the yellow line) and had identical angular locations, but almost hid the unique peaks at -57/-42 degrees and 43/58 degrees. A lower diagonal loading value would likely prevent the blending of the peaks more.

Overall, the signals at -60, -18, 43 and 50 degrees (corresponding to the associated eigenvectors of the 4 dominant eigenvalues) showed up in all the power spectrums, with the exception of the non-adaptive spectrum, which didn't have enough resolution to identify the two peaks at 43 and 50 degrees.

Problem 6


MVDR Power Spectrum
Non-adaptive
MVDR R
MVDR Rfb
MVDR R loaded
MVDR Rfb loaded
shown on separate plot
Magnitude (dB)
Electrical Theta (deg)
60
50
40
30
20
10
0
-10
-20
-30
-40
-150
-100
-50
0
50
100
150
-61.5°
-17.7°
44.8°
-60°
-42°
-18°
18°
43°
58°


MVDR Power Spectrum (MVDR using R)
Non-adaptive
MVDR R
MVDR Rfb
MVDR R loaded
MVDR Rfb loaded
Very low!
-240
-250
-260
-270
-280
-290
-300
-310
-320
-330
Magnitude (dB)
-150
-100
-50
0
50
100
150
Electrical Theta (deg)

```
% Exam 3 Problem 6

clear all;
close all;
hold off;

load p6.mat; % loads X

L = length(X);   % number of snapshots
M = 30;          % number of array elements
numsamps = 20*M; % oversampled angular increments

s = zeros(M,1); % steering vector matrix
MVDR = zeros(numsamps,1); % MVDR power spectrum
MVDR_fb = zeros(numsamps,1); % MVDR power spectrum
p = zeros(numsamps,1); % non-adaptive power spectrum

% Compute correlation matrix based on L snapshots
R = (1/L)*X*ctranspose(X);
R_I = R + eye(length(R));
R_inv = inv(R);
R_I_inv = inv(R_I);

%% Compute Power spectrums
theta = linspace(-180,180,numsamps);

% Compute non-adaptive power spectrum
for idx=1:numsamps
    % Compute steering matrix (electrical angle)
    s = transpose(exp(-j*(theta(idx)*pi/180)*linspace(0,M-1,M)));
    for k=1:L-5
        p(idx) = p(idx) + (ctranspose(s)*X(:,k))^2;
    end
    p(idx) = p(idx) / (M*L);
end

% Form a reflection matrix and forward-backward estimate of R
J = zeros(M,M);
for idx=1:M
    J(M-idx+1,M-idx+1) = 1;
end
Rfb = (1/(2*L))*(X*ctranspose(X) + J*conj(X)*transpose(X)*J);
Rfb_I = Rfb + eye(length(Rfb));
Rfb_inv = inv(Rfb);
Rfb_I_inv = inv(Rfb_I);

% Compute MVDR for both R matrices
for idx=1:numsamps
    % Compute steering matrix (electrical angle)
    s = transpose(exp(-j*(theta(idx)*pi/180)*linspace(0,M-1,M)));
    MVDR(idx) = 1.0 / ( ctranspose(s)*R_inv*s );
```

```
    MVDR_fb(idx) = 1.0 / (ctranspose(s)*Rfb_inv*s );
    MVDR_I(idx) = 1.0 / ( ctranspose(s)*R_I_inv*s );
    MVDR_fb_I(idx) = 1.0 / (ctranspose(s)*Rfb_I_inv*s );
end


%% Plots

% MVDR Power Spectrum (electrical angle)
figure(1);
plot(theta,20*log10(abs(p)));
title('MVDR Power Spectrum');
xlabel('Electrical Theta (deg)');
ylabel('Magnitude (dB)');
axis([-180 180 -40 60]);
hold on;
plot(theta,20*log10(abs(MVDR)),'color','red');
plot(theta,20*log10(abs(MVDR_fb)),'color','green');
plot(theta,20*log10(abs(MVDR_I)),'color','blue');
plot(theta,20*log10(abs(MVDR_fb_I)),'color','yellow');
legend({'Non-adaptive','MVDR R','MVDR Rfb','MVDR R loaded','MVDR Rfb loaded'});
grid on;
```