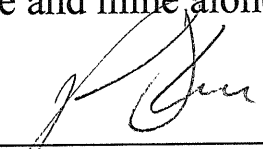


EECS 844 – Fall 2016
Exam 2 Cover page*

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

10/21/16

Date

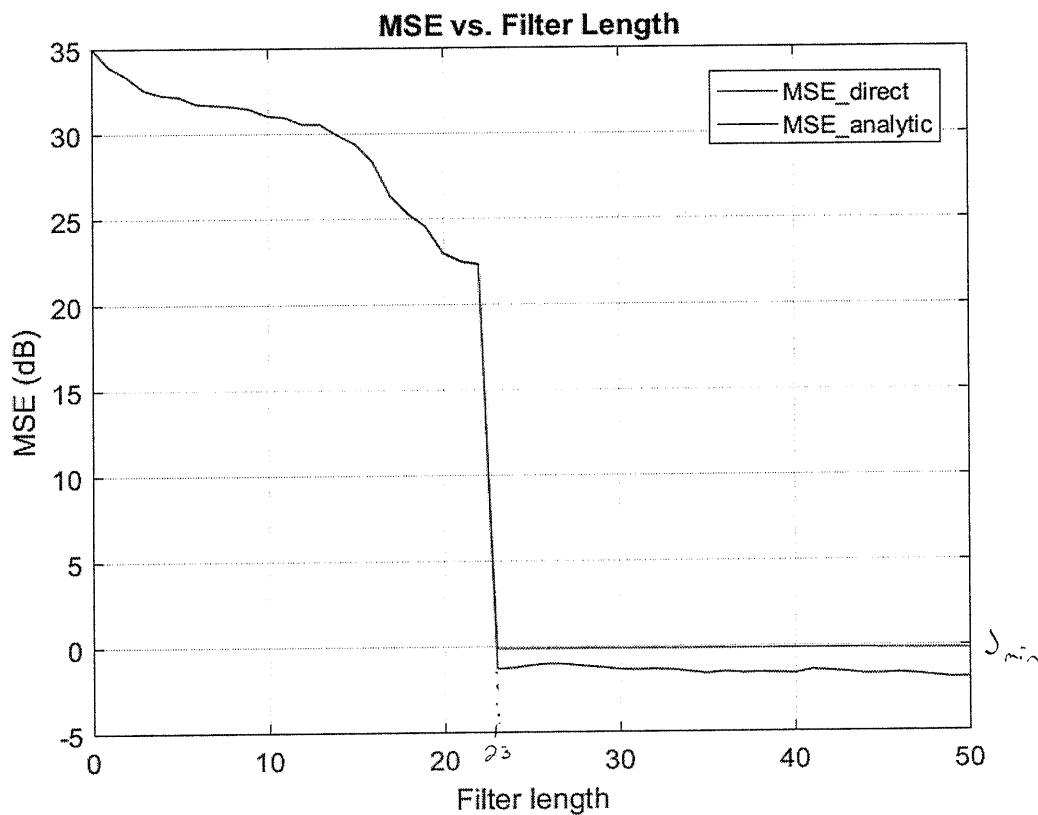
RICH SIMON

Name (printed)

Student ID #

* Attach as cover page to completed exam.

- 1/a) At $M=0$ (no filter), MSE is maximum at 35 dB
 MSE decreases as M increases (makes sense) as model is underfitted
 At $M=23$ taps, MSE drops sharply to 0 dB and levels off (overfitted for $M > 23$)
 \Rightarrow implies that the unknown system length is 23 samples long (critically fitted @ 23)
 Direct + analytic MSE match perfectly from $M=0$ to $M=22$
 Above 23 taps, differences between direct + analytic MSE are small (presumably due to round-off error)



Relevant equations:

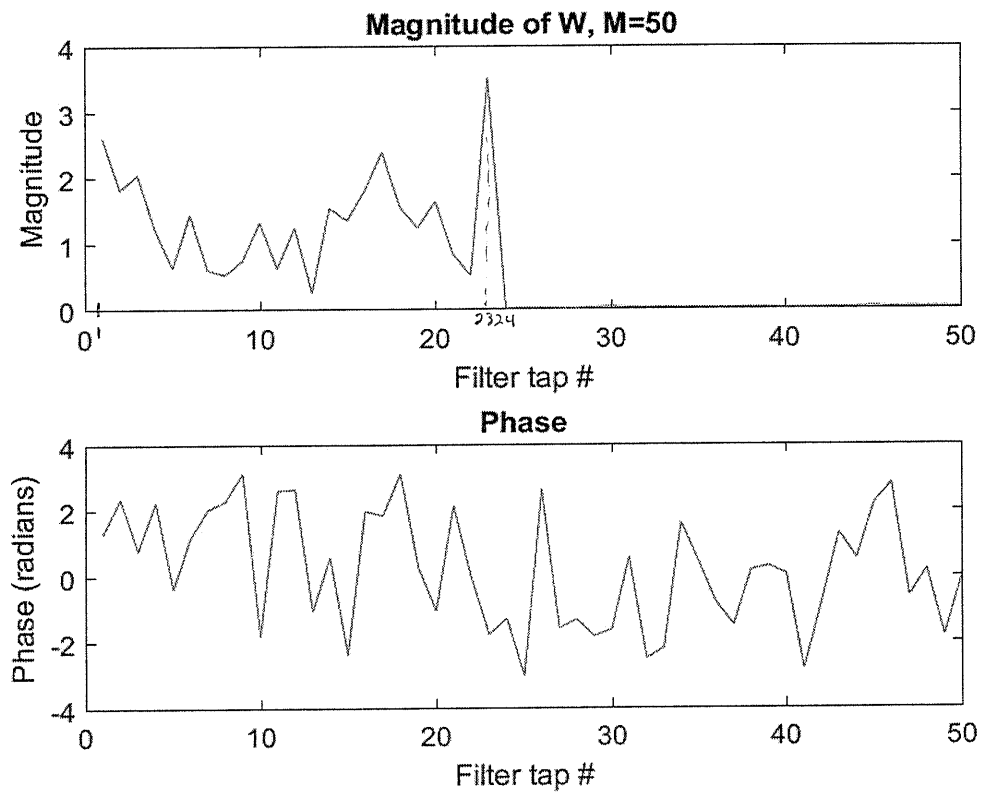
$$\vec{W} = R^{-1} \vec{P}, \quad R = E[\vec{x}\vec{x}^H]$$

$$P = E[\vec{x}\vec{d}^*]$$

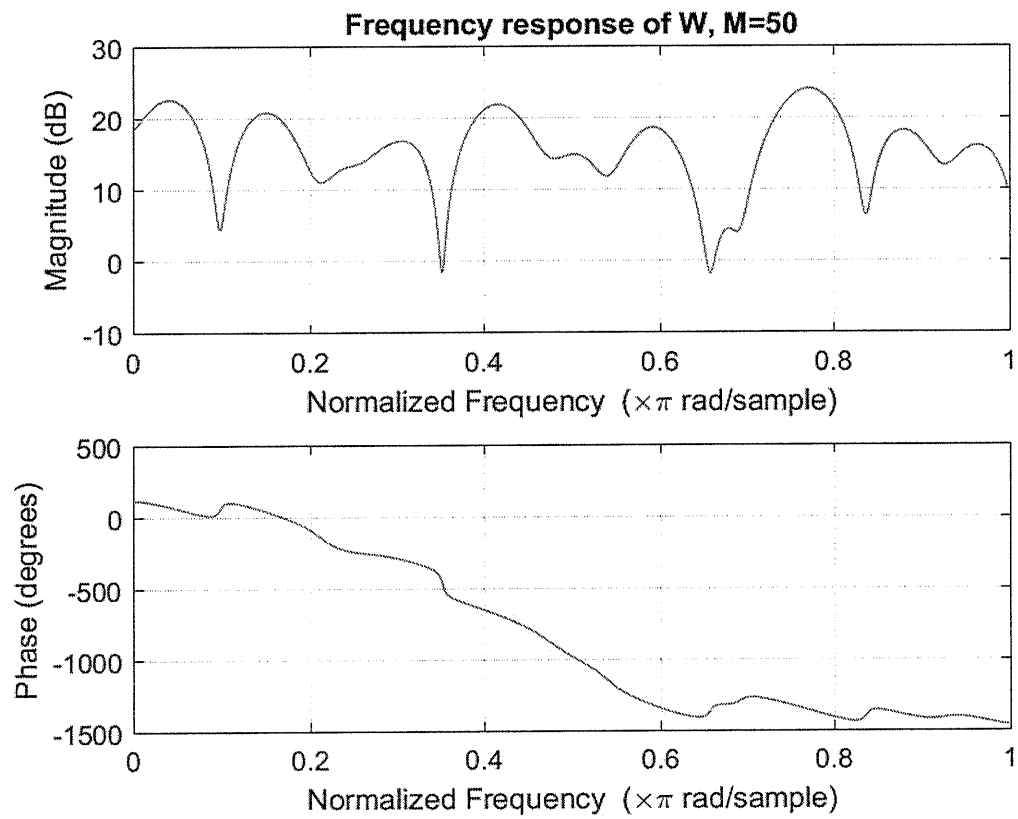
$$MSE_{analytic} = J_{min} = \sigma_d^2 - \hat{\sigma}_d^2, \quad \sigma_d^2 = \frac{d^H d}{length(d)}$$

$$\hat{\sigma}_d^2 = \vec{d}^H R^{-1} \vec{P}$$

1/
b)



d) At $m=50$, only 23 filter taps are needed to estimate the unknown system



- d) Referring to 1(b), why 23 filter taps are needed at $M=50$ to estimate the unknown system

```
% Exam 2 Problem 1

clear all;
close all;
hold off;

load p1.mat; % loads x and d

M_MAX = 50;
W = zeros(M_MAX,1);
mse_direct = zeros(M_MAX+1,1);
mse_analytic = zeros(M_MAX+1,1);
K = length(x);
e = zeros(K,1);

sigma_d_squared = (ctranspose(d)*d)/length(d); % for analytic MSE calc

% Cycle through all M filter lengths
for M=0:M_MAX
    % Calculate filter weights
    if( M == 0 )
        W(1) = 1.0; % no filter
    else
        % Calculate Wiener filter weights

        % Create X matrix
        X = zeros(M,K-M+1);
        for k=1:K-M+1
            X(:,k) = flipud(x(k:k+M-1));
        end

        % Calculate R
        R = (1/(K-M+1))*X*ctranspose(X);

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

        % Calculate W
        W = R\P;

        % Calculate MSE (analytic) = Jmin
        sigma_dhat_squared = real(ctranspose(P)*inv(R)*P);
        mse_analytic(M+1) = sigma_d_squared - sigma_dhat_squared;
    end

    % Calculate filter output
    y = filter(conj(W),1,x);
```

```
% Calculate estimation error
e = d - y(1:length(d));

% Calculate MSE (direct)
mse_direct(M+1) = (ctranspose(e)*e)/length(e);
if( M == 0 )
    mse_analytic(M+1) = mse_direct(M+1); % no R or P to calc analytic error
end
end

% Plots
figure(1)
x_axis_label = linspace(0,50,51);
mse_curve_direct = 20*log10(mse_direct);
mse_curve_analytic = 20*log10(mse_analytic);
plot(x_axis_label,mse_curve_direct,'color','blue');
hold on;
plot(x_axis_label,mse_curve_analytic,'color','red');
title('MSE vs. Filter Length');
legend({'MSE\_direct','MSE\_analytic'});
grid on;
xlabel('Filter length');
ylabel('MSE (dB) ');

figure(2)
h0(1)=subplot(2,1,1);
plot(abs(W)); % magnitude
title('Magnitude of W, M=50');
ylabel('Magnitude');
xlabel('Filter tap #');
h0(2)=subplot(2,1,2);
plot(angle(W)); % phase
title('Phase');
ylabel('Phase (radians) ');
xlabel('Filter tap #');
linkaxes(h0,'x');

figure(3)
freqz(W);
title('Frequency response of W, M=50');
```

2

$$\begin{aligned} e(n) &= x(n) - z(n) \\ &= x(n) - \vec{y}^H(n) \vec{w} \end{aligned}$$

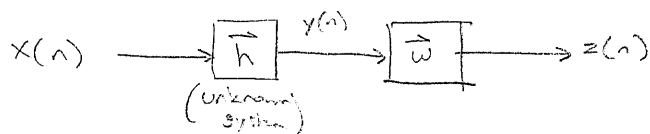
$$J(w) = E[e(n)e^*(n)] = E[x(n)x^*(n)] - \underbrace{2E[x(n)y^H(n)]}_{\vec{p}} \vec{w} - \underbrace{w^H E[y(n)y^H(n)] w}_{R_{yy}}$$

To find minimum,

$$\frac{\partial J(w)}{\partial \vec{w}} = -2\vec{p} + 2R_{yy}\vec{w} = 0$$

$$\vec{w} = R_{yy}^{-1} \vec{p}$$

By inspection, the system diagram is



When \vec{w} is the optimal value, it will act as a channel equalizer,

and $h(i) * w(i) = \delta(i)$ (the system of h and w will act like a pass-through filter)

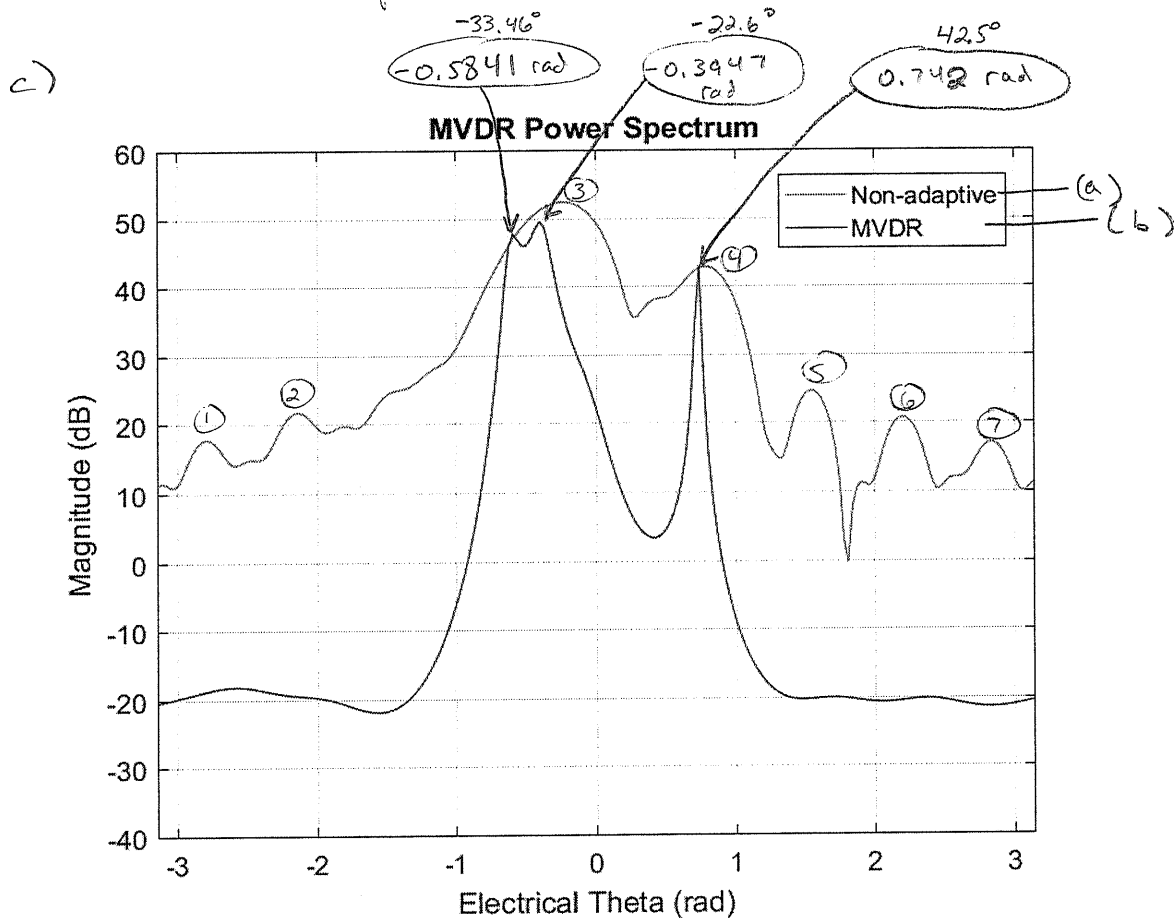
3/a) There appears to be about 7 signals (see below circled)

b) There appears to be about 3 signals (pointed to in part (c))

2 of the 3 correspond to spatial spectrum estimate (3)

1 of the 3 correspond to spatial spectrum estimate (4)

MVDR clearly is better at discerning closely spaced signals than nonadaptive estimate



Relevant equations:

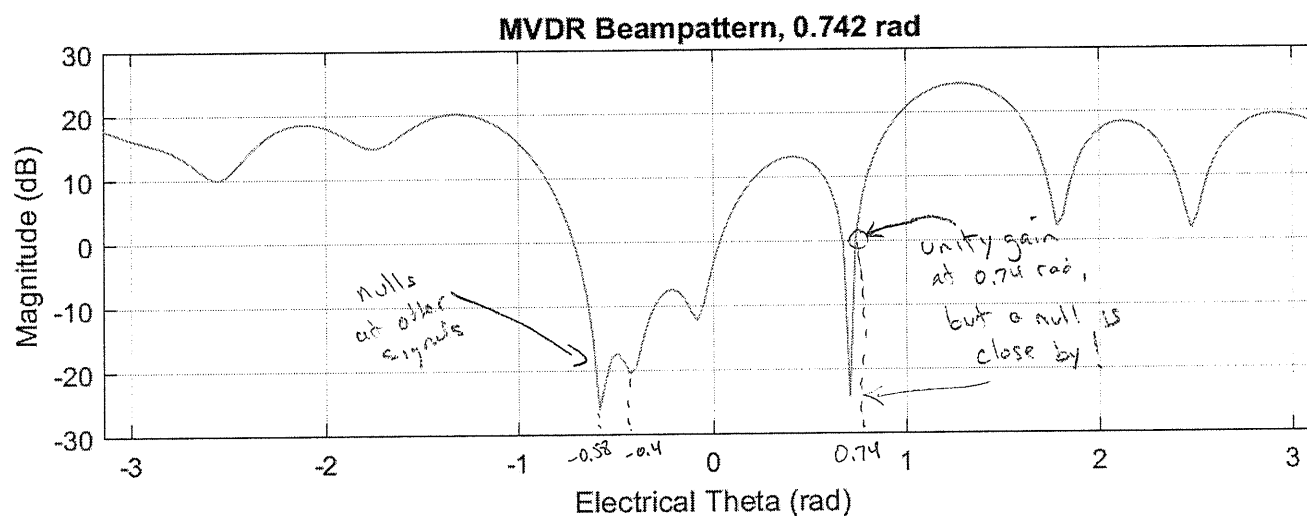
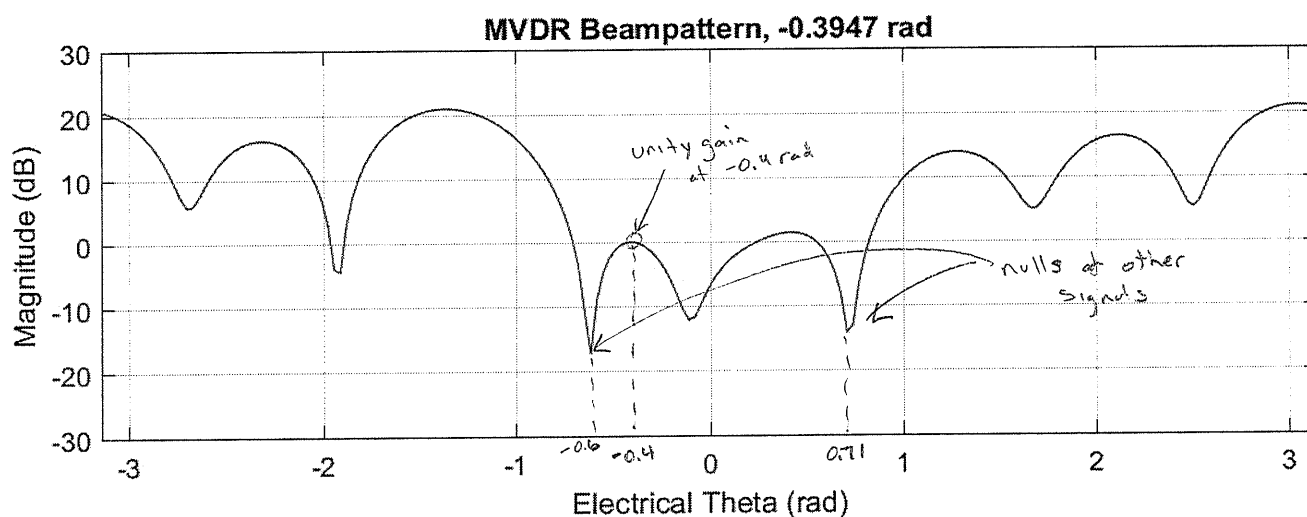
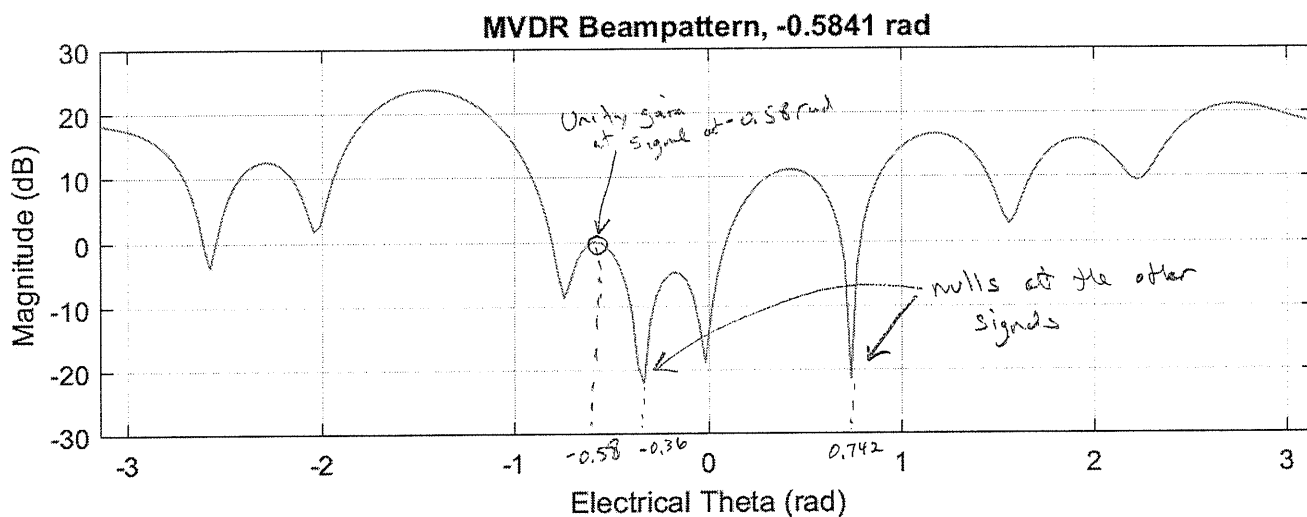
$$P(\theta) = \frac{1}{M} \sum_{j=1}^L |\vec{s}^H(\theta) \vec{x}_j|^2 \quad \text{Spatial spectrum estimate}$$

$$P(\theta) = \frac{1}{\vec{s}^H R^{-1} \vec{s}} \quad \text{MVDR spectral estimate}$$

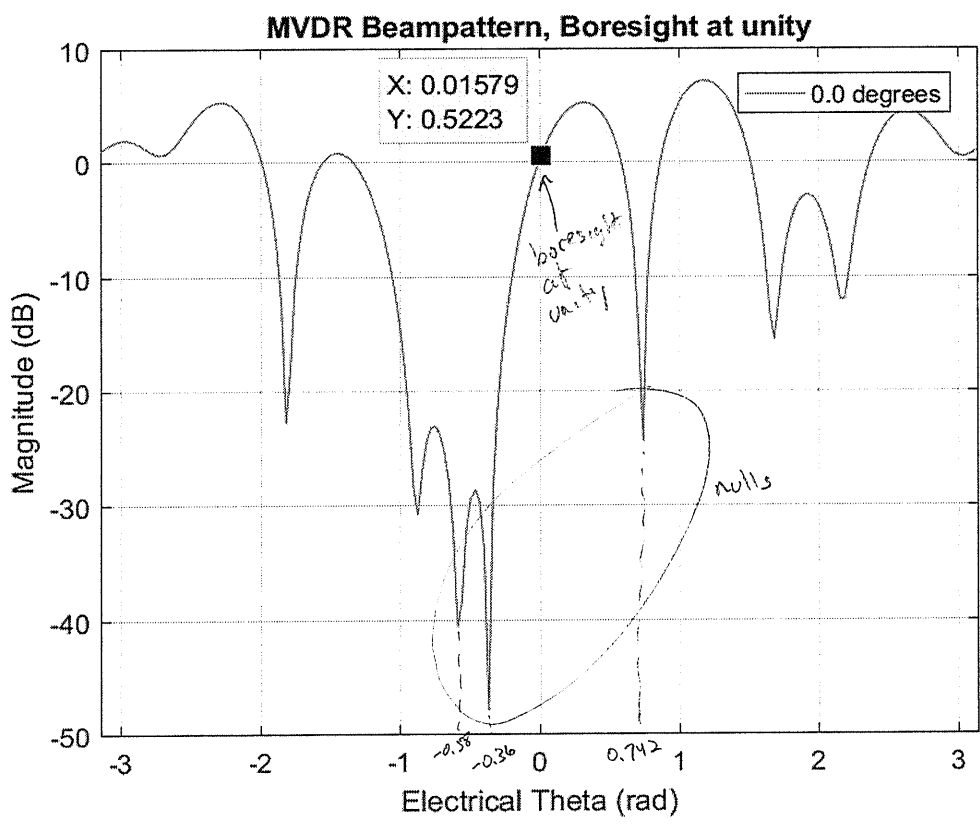
$$w = \frac{R^{-1} \vec{s}(\theta)}{\vec{s}^H R^{-1} \vec{s}(\theta)} \quad \text{MVDR filter for steering vector angle } \theta$$

3

d)



3/d)



```
% Exam 2 Problem 3

clear all;
close all;
hold off;

load p3.mat; % loads X

L = length(X); % number of snapshots
M = 10; % number of array elements
numsamps = 20*M; % oversampled angular increments

s = zeros(M,1); % steering vector matrix
MVDR = 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_inv = inv(R);

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

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

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

%% Compute MVDR beamformers
peak_angles = [-0.5841 -0.3947 0.742 0.0]; % peaks in radians
w = zeros(M,length(peak_angles)); % weights for each of the 3 observed peaks
MVDR_response = zeros(numsamps,length(peak_angles));

for k=1:length(peak_angles)
    s = transpose(exp(-j*peak_angles(k)*linspace(0,M-1,M)));
    w(:,k) = (R_inv*s) / (ctranspose(s)*R_inv*s);
    % Compute beampatterns
    for idx=1:numsamps
        s = transpose(exp(-j*theta(idx)*linspace(0,M-1,M)));
```

```
MVDR_response(idx,k) = ctranspose(s)*w(:,k);
end
end

%% Plots

% MVDR Power Spectrum (electrical angle)
figure(1);
plot(theta,20*log10(abs(p)));
title('MVDR Power Spectrum');
xlabel('Electrical Theta (rad)');
ylabel('Magnitude (dB)');
axis([-pi pi -40 60]);
hold on;
plot(theta,20*log10(abs(MVDR)), 'color', 'red');
legend({'Non-adaptive', 'MVDR'});
grid on;

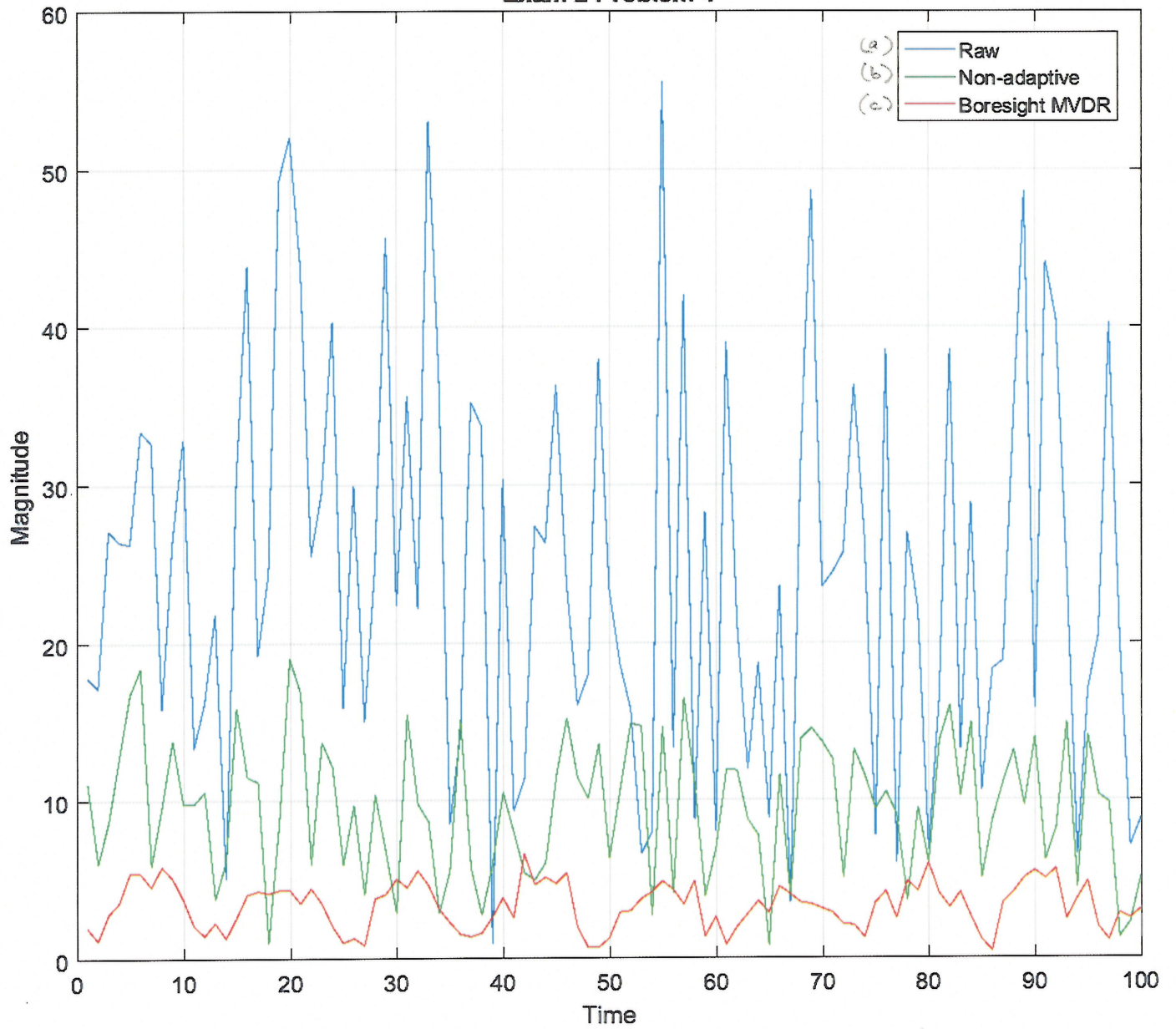
% MVDR beamformers
figure(2);
h0(1)=subplot(3,1,1);
plot(theta,20*log10(abs(MVDR_response(:,1))));
grid on;
title('MVDR Beampattern, -0.5841 rad');
xlabel('Electrical Theta (rad)');
ylabel('Magnitude (dB)');
axis([-pi pi -30 30]);
h0(2)=subplot(3,1,2);
plot(theta,20*log10(abs(MVDR_response(:,2))), 'color', 'red');
grid on;
title('MVDR Beampattern, -0.3947 rad');
xlabel('Electrical Theta (rad)');
ylabel('Magnitude (dB)');
axis([-pi pi -30 30]);
h0(2)=subplot(3,1,3);
plot(theta,20*log10(abs(MVDR_response(:,3))), 'color', 'green');
grid on;
title('MVDR Beampattern, 0.742 rad');
xlabel('Electrical Theta (rad)');
ylabel('Magnitude (dB)');
axis([-pi pi -30 30]);
linkaxes(h0,'x');

% Boresight constraint
figure(3);
plot(theta,20*log10(abs(MVDR_response(:,4))));
title('MVDR Beampattern, Boresight at unity');
xlabel('Electrical Theta (rad)');
```

```
ylabel('Magnitude (dB)');  
axis([-pi pi -50 10]);  
legend({'0.0 degrees'});  
grid on;
```

4

Exam 2 Problem 4



The raw response shows the composite signal of all signals coming from all directions.

The non-adaptive filter, which simply peaks at the boresight angle, will filter out signals from outside the main lobe, but as seen in problem 3, there are signals of interest close to the boresight and presumably within the main lobe or first sidelobe, so the signal magnitude, which smaller than the raw signal, is still fairly high.

The MVDR filter, developed in problem 3d, is designed to admit signals at the boresight with unity gain, and reject (null) stronger signals outside the boresight. As expected, the time domain magnitude response is much lower, and is very close to unity.

```
% Exam 2 Problem 4

clear all;
close all;
hold off;

load p3.mat; % loads X

L = length(X); % number of snapshots
M = 10; % number of array elements
s = zeros(M,1); % steering vector matrix

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

%% Compute MVDR beamformers
s = transpose(exp(-j*0.0*linspace(0,M-1,M))); % boresight direction (0.0 deg)
w = (R_inv*s) / (ctranspose(s)*R_inv*s);

y_mvdr = ctranspose(w)*X; % MVDR
y_na = (1/M)*(ctranspose(s)*X); % non-adaptive

P_mvdr = sqrt((1/length(y_mvdr))*y_mvdr*ctranspose(y_mvdr))

%% Plots

% MVDR Power Spectrum (electrical angle)
figure(1);
plot(abs(X(1,:)));
title('Exam 2 Problem 4');
xlabel('Time');
ylabel('Magnitude');
hold on;
plot(abs(y_na(1,:)),'color','green');
plot(abs(y_mvdr(1,:)),'color','red');
legend({'Raw', 'Non-adaptive', 'Boresight MVDR'});
grid on;
```


5

For each signal direction found in problem 3 (-0.5851 radians, -0.3947 radians, and 0.742 radians), the MVDR boresight filter beampattern is overlaid.

In general, the GSC works as well or better than the MVDR filter. By using a degree of freedom to concentrate a null at a chosen location, the null is much deeper than that created by the MVDR filter.

Since there are only 3 (known) signals, and 10 array elements, the GSC has enough degrees of freedom to effectively create the null while still meeting the unity gain constraint and the single null constraint.

In all three cases where a null was placed at the known signals, the unity gain constraint is met, and the desired null is very sharp and deep. In the -0.5851 and -0.3947 cases, the beampattern response is very close to the MVDR, signifying that the cost of giving up one degree of freedom for null placement was not that great (it also appears to be because the MVDR filter had a null there anyway). For the null at 0.742 radians, the beampattern response differs more than the other two null placements. Overall rejection is about the same at all three signal locations, and even exhibits better rejection at other directions.

Placing a null away from detected signals (at -1.437 radians) shows that the combined adaptive and non-adaptive filters can reject the three rejected signals, while placing a null at the desired direction without penalty.

Relevant equations:

$$C = \begin{bmatrix} \vec{s}_0 & \vec{s}_1 \end{bmatrix} \xrightarrow{\text{SVD}} U \Sigma V$$

$$\hookrightarrow U = \begin{bmatrix} \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \end{bmatrix}$$

$$C_a$$

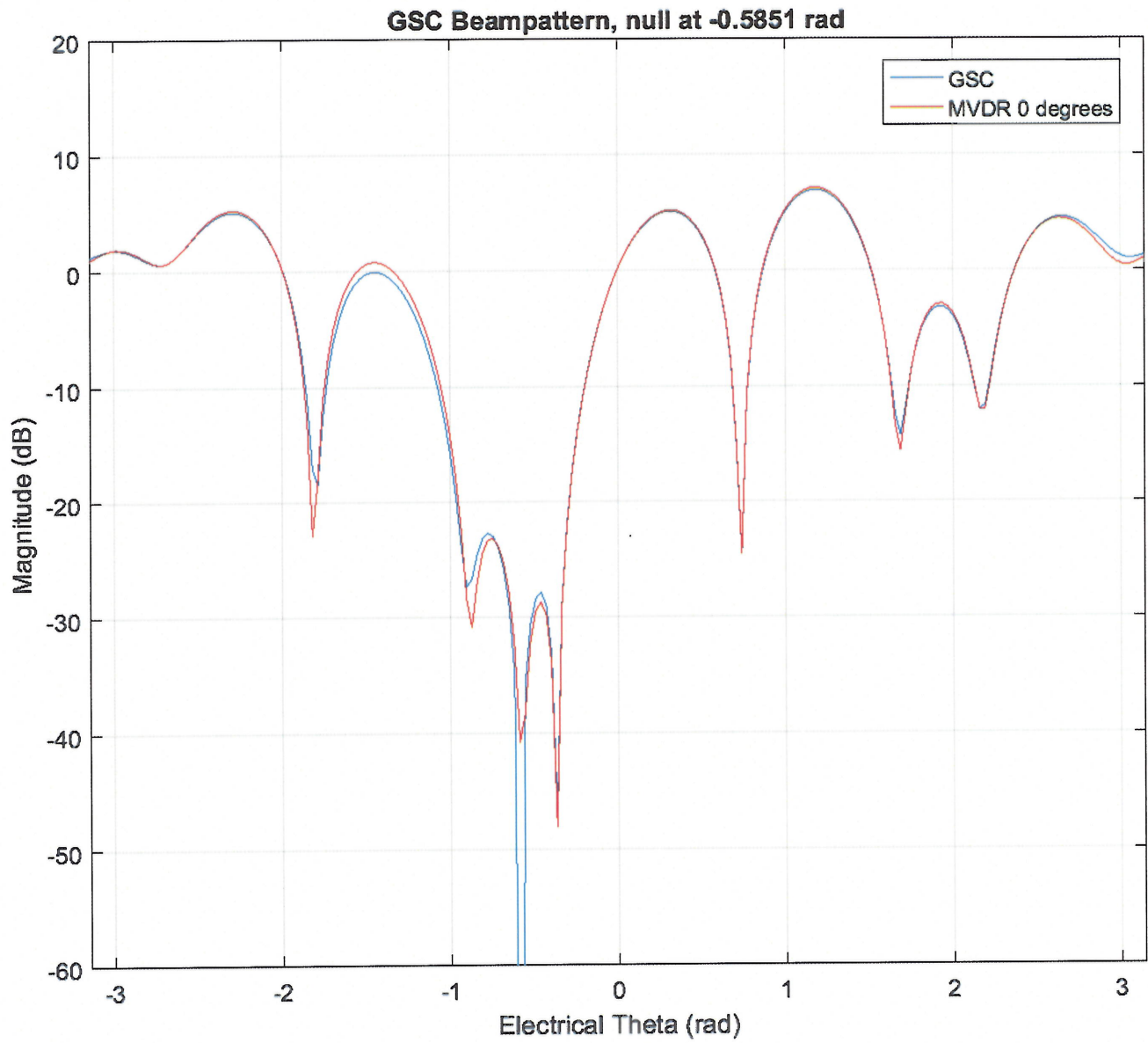
$$w_q = C \vec{v}, \quad \vec{v} = (C^H C)^{-1} \vec{g} \quad (\text{Fixed weights based on } g \text{ and } C)$$

$$w_a = (C_a^H R C_a)^{-1} C_a^H R w_q \quad (\text{Adaptive weights based on } R)$$

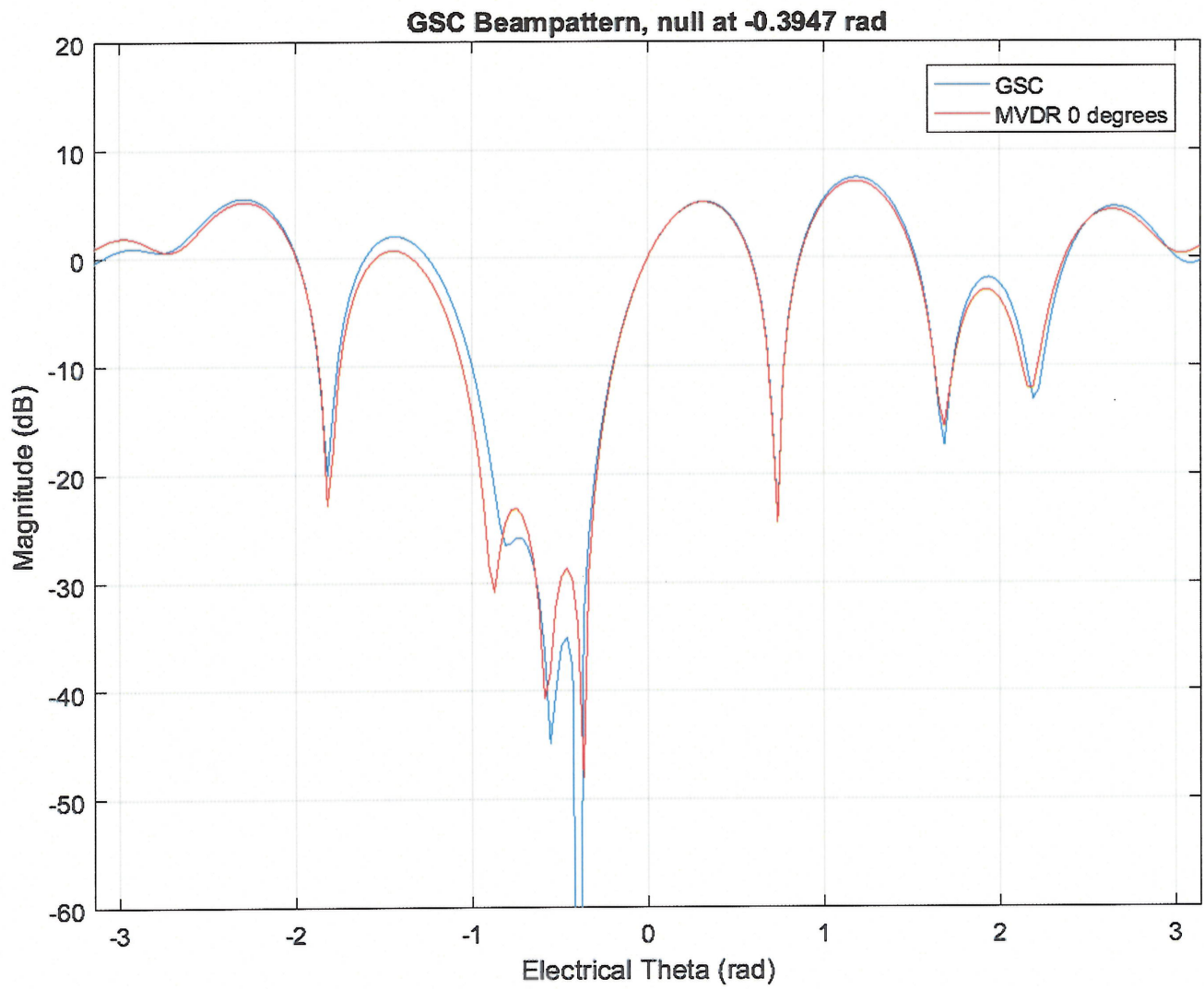
$$w = w_q - C_a w_a$$

5

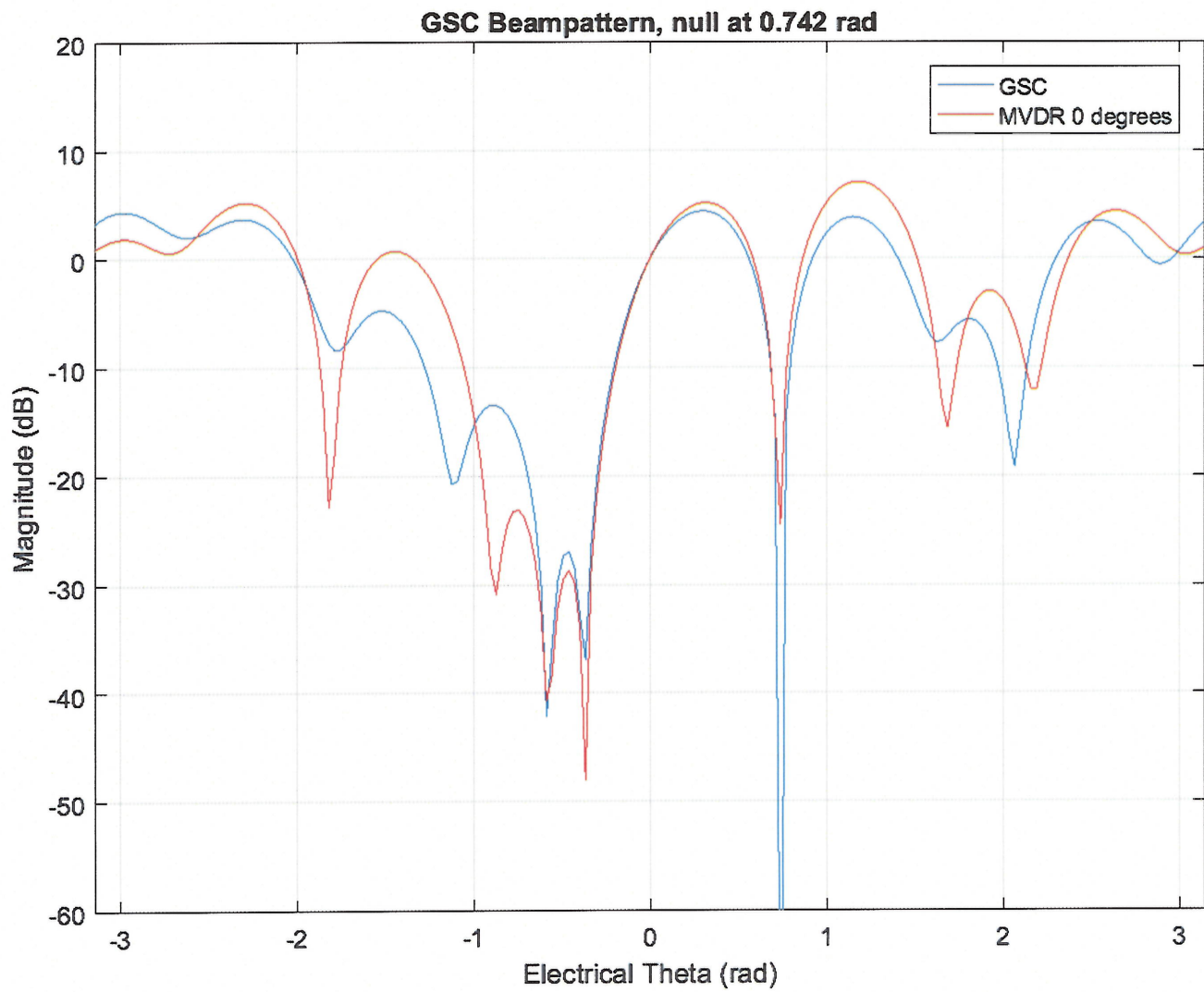
Null @ -0.5851 radians



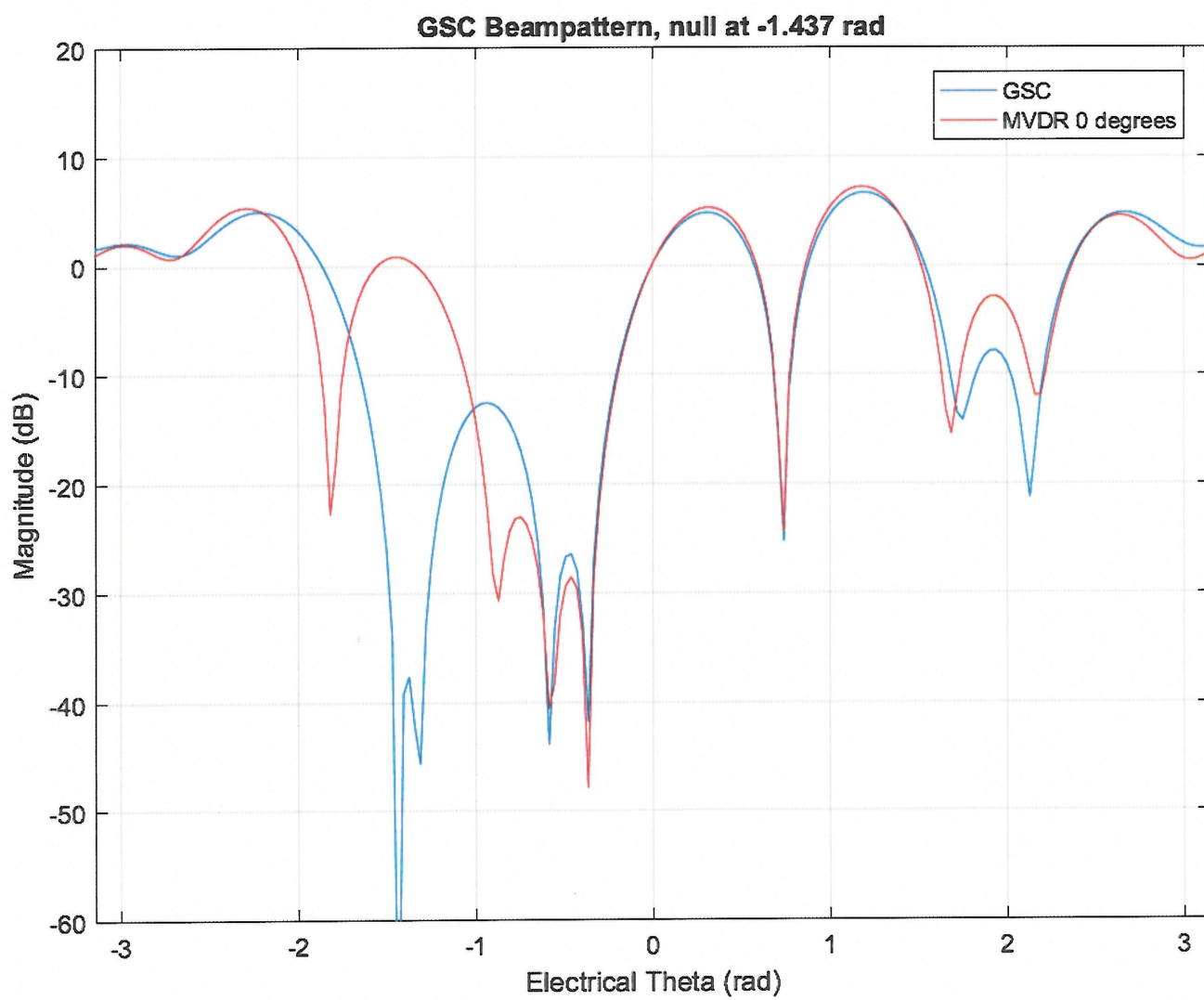
5
Null @ -0.3947 radians



5
Null @ 0.742 radians



S/ Null @ -1.437 rad (no known signal here, just pretend there's an intermittent source?)



```
% Exam 2 Problem 5
% Note: Only one null angle is simulated per run
% Change the null angle by setting null_angle (in radians)

clear all;
close all;
hold off;

load p3.mat; % loads X

Lx = length(X); % number of snapshots
L = 2; % number of constraints
M = 10; % number of array elements
numsamps = 20*M; % oversampled angular increments

null_angle = -0.3947; % null angle, radians

p = zeros(numsamps,1); % GSC power spectrum
pp = zeros(numsamps,1); % non-adaptive power spectrum

% Compute correlation matrix based on L snapshots
R = (1/Lx)*X*ctranspose(X);
R_inv = inv(R);

% Form steering vectors
s0 = transpose(exp(-j*0.0*linspace(0,M-1,M))); % boresight
s1 = transpose(exp(-j*null_angle*linspace(0,M-1,M))); % null
g = [1.0 0.0]';

% Form C, Ca
C = [s0 s1];
[U,S,V] = svd(C);
Ca = U(:,L+1:M);

% Compute wq
v = inv(ctranspose(C)*C) * g;
wq = C*v;

% Compute wa(optimal)
wa = inv(ctranspose(Ca)*R*Ca)*ctranspose(Ca)*R*wq;
w = wq - Ca*wa;

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

% Compute non-adaptive beamformer response
for idx=1:numsamps
    % Compute steering matrix (electrical angle)
    s = transpose(exp(-j*theta(idx)*linspace(0,M-1,M)));
    pp(idx) = ctranspose(s)*wq; % adaptive power spectrum
    p(idx) = ctranspose(s)*w; % GSC power spectrum
end
```

```
end

%% Compute MVDR beamformers
MVDR_response = zeros(numsamps,1);
s = transpose(exp(-j*0.0*linspace(0,M-1,M)));
w_mvdr = (R_inv*s) / (ctranspose(s)*R_inv*s);
% Compute beampatterns
for idx=1:numsamps
    s = transpose(exp(-j*theta(idx)*linspace(0,M-1,M)));
    MVDR_response(idx) = ctranspose(s)*w_mvdr;
end

%% Plots

% MVDR Power Spectrum (electrical angle)
figure(1);
plot(theta,20*log10(abs(p)));
title('GSC Beampattern, null at -0.3947 rad');
xlabel('Electrical Theta (rad)');
ylabel('Magnitude (dB)');
axis([-pi pi -60 20]);
hold on;
%plot(theta,20*log10(abs(pp)),'color','green');
plot(theta,20*log10(abs(MVDR_response)),'color','red');
%legend({'GSC','Non-adaptive','MVDR 0 degrees'});
legend({'GSC','MVDR 0 degrees'});
grid on;
```

6

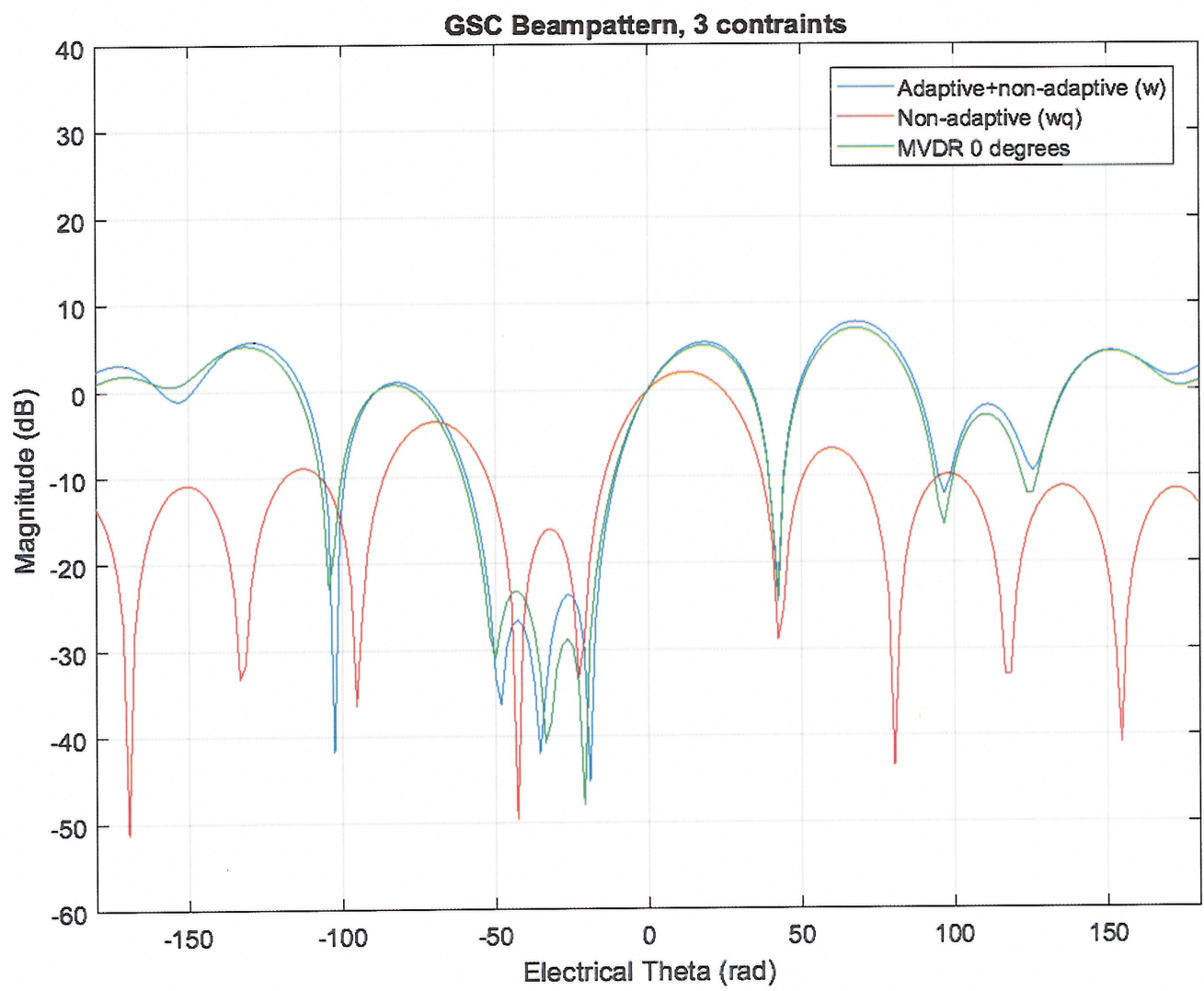
A 10x10 constraint matrix of nulls from -50 degrees to -15 degrees was created, with a unity gain constraint at the boresight. Using eigendecomposition of the constraint matrix, it was found that there were 2 dominant eigenvalues (75.8, 22.7, 1.41, 0.03, 0.0002, and zeros for the rest). Thus it would be expected that about 3 eigenvalues would be optimal in producing a broad null across the desired range of directions.

At 3 constraints (including the unity gain constraint), there are three nulls across the range, and the GSC beampattern matches closely with the MVDR, which is expected given that the MVDR likely sees the two signals around those angles (and subsequently places three nulls there, the same number of nulls that the GSC used).

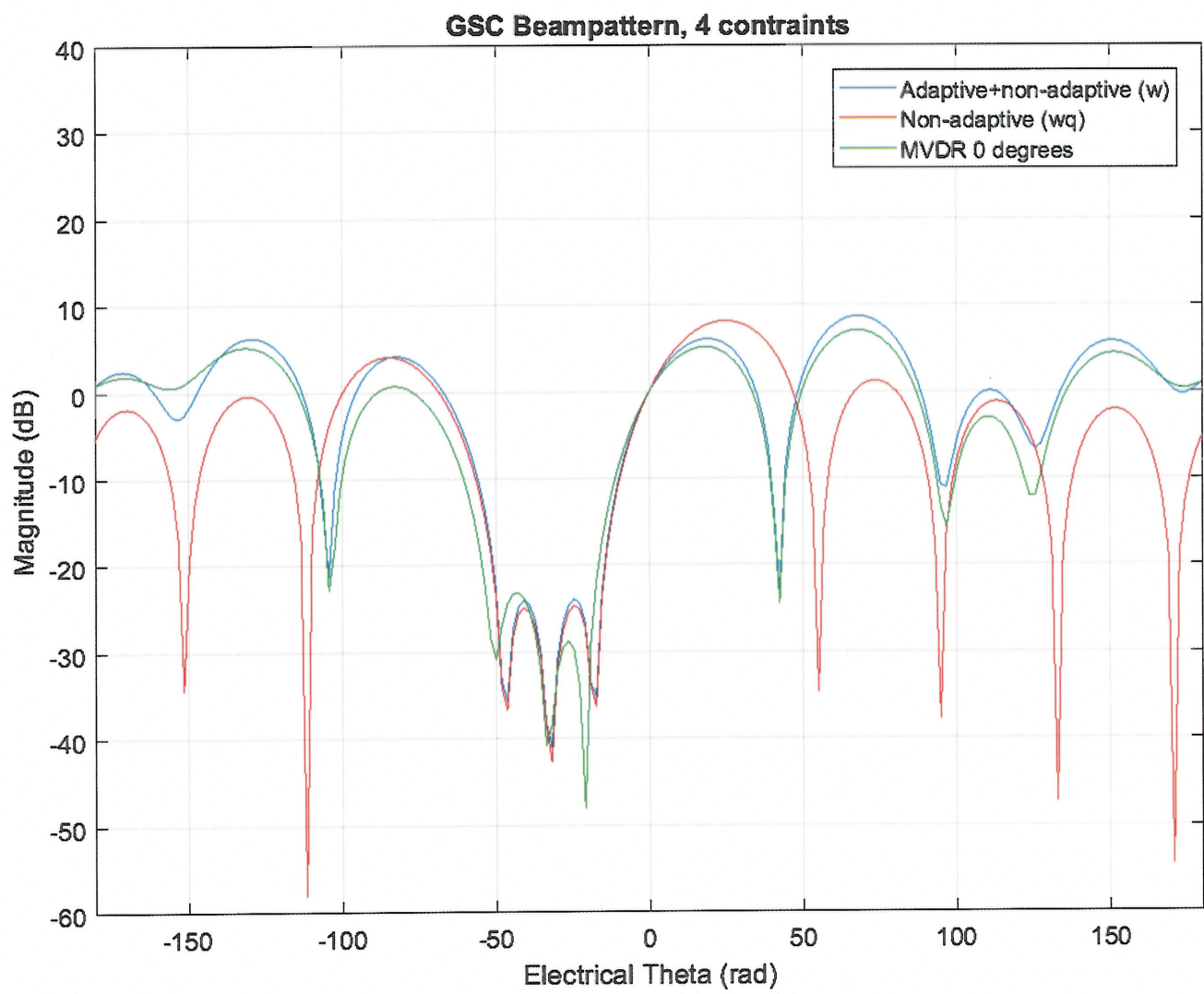
At 4 constraints, three nulls are still placed across the range, but are collectively deeper than the MVDR. You can see the response at other directions is worse than the MVDR (higher gains).

At 5 and 6 constraints, more nulls are used in the range, at the cost of significantly higher gain outside of the null range. The available degrees of freedom needed for overall rejection outside the null range is lessened.

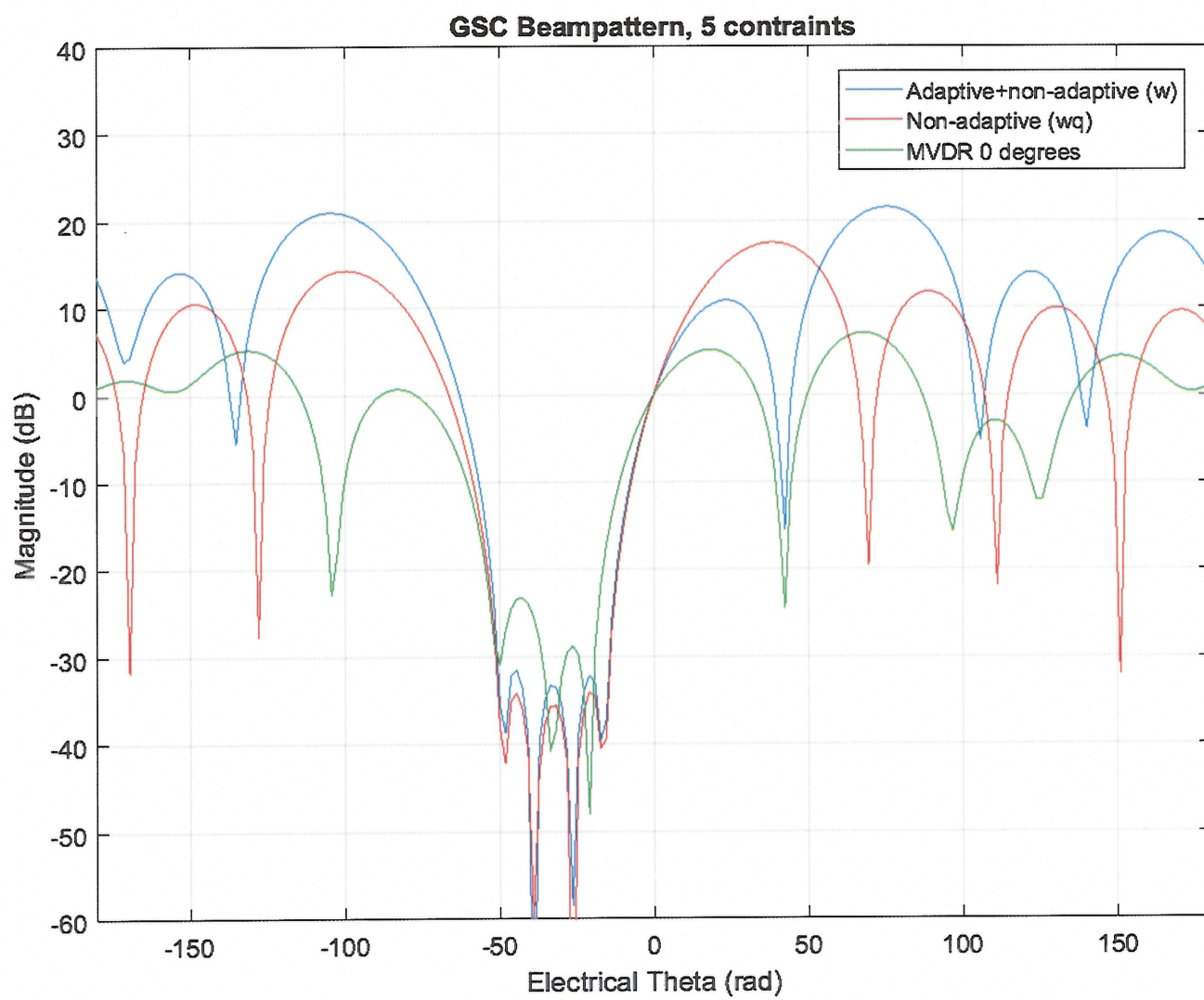
6



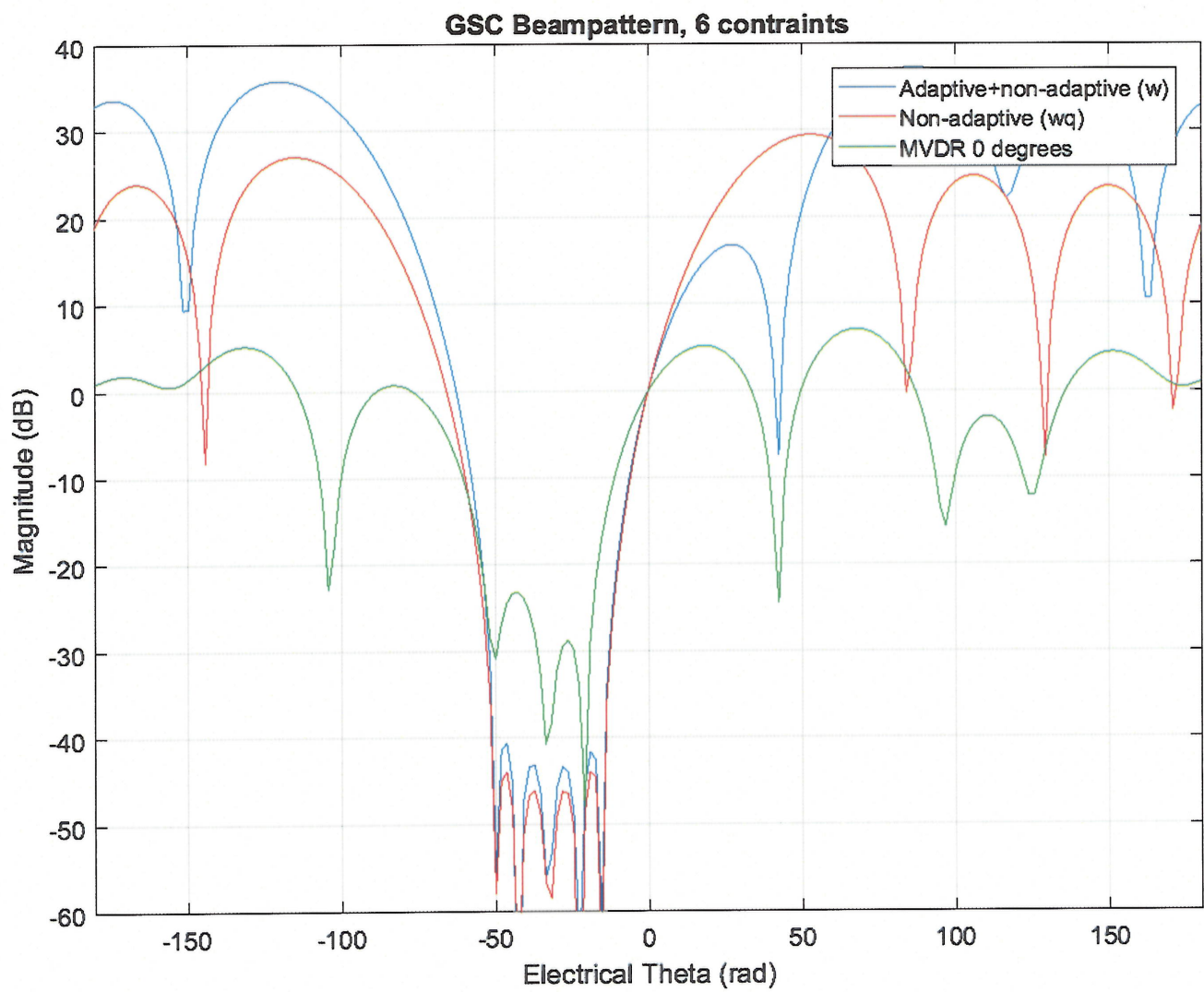
6



6



6



```
% Exam 2 Problem 6

clear all;
close all;
hold off;

load p3.mat; % loads X

Lx = length(X); % number of snapshots
L = 6;          % number of constraints
M = 10;         % number of array elements
numsamps = 20*M; % oversampled angular increments

MVDR = zeros(numsamps,1); % MVDR power spectrum
p = zeros(numsamps,1);    % GSC power spectrum
pp = zeros(numsamps,1); % non-adaptive power spectrum

% Compute correlation matrix based on Lx snapshots
R = (1/Lx)*X*ctranspose(X);
R_inv = inv(R);

% Form steering vectors
s0 = transpose(exp(-j*0.0*linspace(0,M-1,M))); % boresight
null_angles = (pi/180)*linspace(-50,-15,M); % radians
Rc = zeros(M,M);
for k=1:M
    s = transpose(exp(-j*null_angles(k)*linspace(0,M-1,M)));
    Rc = Rc + s*ctranspose(s);
end
[Q,lambda] = eig(Rc);

% by inspection, there are two principle eigenvalues, but take up to L
% eigenvalues (L constraints)
C = Q(:,M-L+1:M);
C(:,1) = s0; % boresight constraint

g = zeros(L,1);
g(1) = 1.0; % unity gain constraint

% Form Ca
[U,S,V] = svd(C);
Ca = U(:,L+1:M);

% Compute wq
v = inv(ctranspose(C)*C) * g;
wq = C*v;

% Compute wa(optimal)
wa = inv(ctranspose(Ca)*R*Ca)*ctranspose(Ca)*R*wq;
w = wq - Ca*wa;
```

```
%% Compute Power spectrums
theta = linspace(-pi,pi,numsamps);
theta_deg = theta*180/pi;

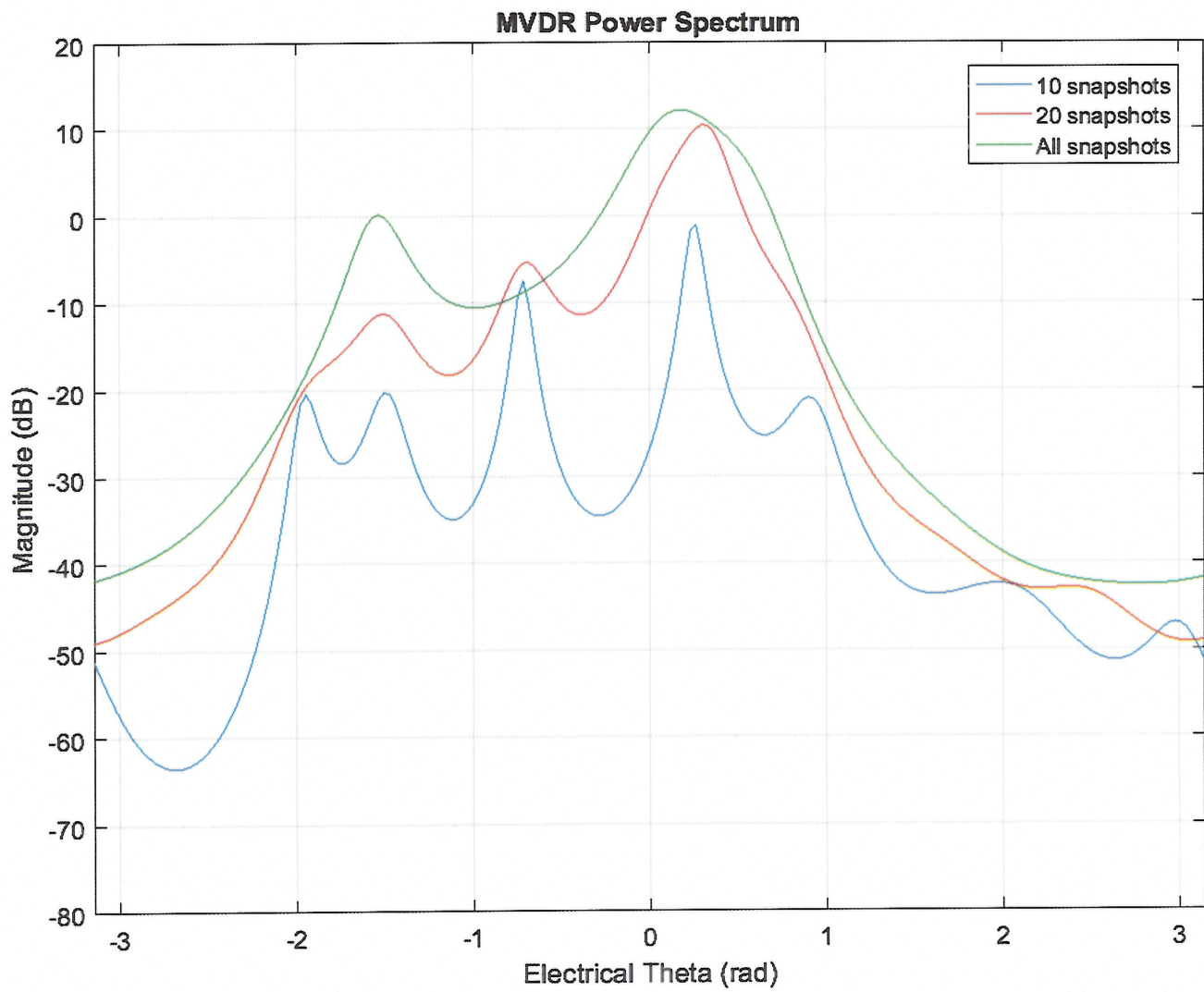
% Compute non-adaptive beamformer response
for idx=1:numsamps
    % Compute steering matrix (electrical angle)
    s = transpose(exp(-j*theta(idx)*linspace(0,M-1,M)));
    pp(idx) = ctranspose(s)*wq; % adaptive power spectrum
    p(idx) = ctranspose(s)*w; % GSC power spectrum
end

%% Compute MVDR beamformers
MVDR_response = zeros(numsamps,1);
s = transpose(exp(-j*0.0*linspace(0,M-1,M)));
w_mvdr = (R_inv*s) / (ctranspose(s)*R_inv*s);
% Compute beampatterns
for idx=1:numsamps
    s = transpose(exp(-j*theta(idx)*linspace(0,M-1,M)));
    MVDR_response(idx) = ctranspose(s)*w_mvdr;
end

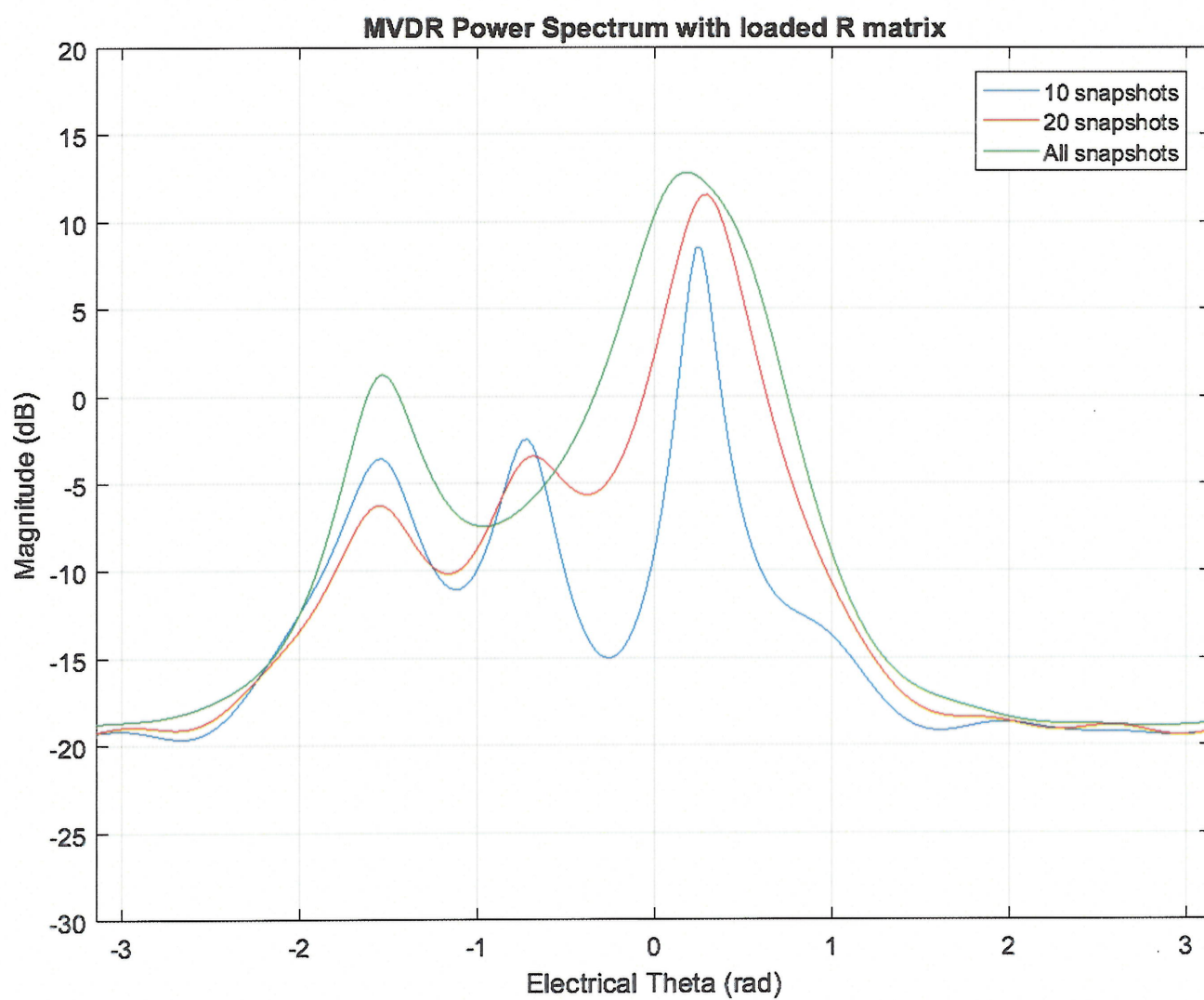
%% Plots

% MVDR Power Spectrum (electrical angle)
figure(1);
plot(theta_deg,20*log10(abs(p)));
title('GSC Beampattern, 6 constraints');
xlabel('Electrical Theta (rad)');
ylabel('Magnitude (dB)');
axis([-180 180 -60 40]);
hold on;
plot(theta_deg,20*log10(abs(pp)),'color','red');
plot(theta_deg,20*log10(abs(MVDR_response)),'color','green');
legend({'Adaptive+non-adaptive (w)','Non-adaptive (wq)','MVDR 0 degrees'});
grid on;
```

7
a)



✓
b)



3

Taking fewer snapshots to form the correlation matrix R that is used in the MVDR power spectrum leads to more peaks in the spectral output, while taking the maximum number of snapshots to form R shows only two peaks. It can be surmised that fewer snapshots leads to less "smoothing" of the snapshots (higher variance in the estimates of R) and thus the higher number of local peaks. As the number of snapshots increases to 20, the spectral estimate is smoother and displays less peaks.

Taking all snapshots to form R really smooths out the spectral estimate, but is the most accurate if the signal statistics are stationary relative to the snapshot window.

The effect of adding noise to the diagonal of R causes the spectral estimates with few snapshots to be more accurate than the spectral estimate without noise (for the same number of snapshots). R is more well-conditioned, making the inversion of R less susceptible to noise enhancement, which is likely the cause of the higher number of peaks when the number of snapshots is small and R is not diagonally loaded.

```
% Exam 2 Problem 7

clear all;
close all;
hold off;

load p7.mat; % loads x

K = length(x); % number of samples
M = 10; % filter length
numsamps = 20*M; % oversampled angular increments
L = [10 20 K-M+1]; % snapshot lengths

s = zeros(M,1); % steering vector matrix
MVDR = zeros(numsamps,length(L)); % MVDR power spectrums
MVDR_loaded = zeros(numsamps,length(L)); % loaded MVDR power spectrum

% Create X matrix (snapshots)
X = zeros(M,K-M+1);
for k=1:K-M+1
    X(:,k) = flipud(x(k:k+M-1));
end

% Cycle through correlation lengths of 10,20,and all
theta = linspace(-pi,pi,numsamps);
for L_idx=1:3
    % Compute correlation matrix based on L snapshots
    R = (1/L(L_idx))*X(:,1:L(L_idx))*ctranspose(X(:,1:L(L_idx)));
    R_loaded = R + eye(M);
    R_inv = inv(R);
    R_loaded_inv = inv(R_loaded);

    % Compute steering matrix
    for idx=1:numsamps
        % Compute steering matrix (electrical angle)
        s = transpose(exp(-j*theta(idx)*linspace(0,M-1,M)));
        MVDR(idx,L_idx) = 1.0 / ( ctranspose(s)*R_inv*s );
        MVDR_loaded(idx,L_idx) = 1.0 / ( ctranspose(s)*R_loaded_inv*s );
    end
end

% MVDR Power Spectrum (electrical angle)
figure(1);
plot(theta,20*log10(abs(MVDR(:,1))));
title('MVDR Power Spectrum');
xlabel('Electrical Theta (rad)');
ylabel('Magnitude (dB)');
axis([-pi pi -80 20]);
hold on;
plot(theta,20*log10(abs(MVDR(:,2))), 'color', 'red');
plot(theta,20*log10(abs(MVDR(:,3))), 'color', 'green');
```

```
legend({'10 snapshots','20 snapshots','All snapshots'});
grid on;

% MVDR Power Spectrum, loaded correlation matrix (electrical angle)
figure(2);
plot(theta,20*log10(abs(MVDR_loaded(:,1))));
title('MVDR Power Spectrum with loaded R matrix);
xlabel('Electrical Theta (rad)');
ylabel('Magnitude (dB)');
axis([-pi pi -30 20]);
hold on;
plot(theta,20*log10(abs(MVDR_loaded(:,2))),color','red');
plot(theta,20*log10(abs(MVDR_loaded(:,3))),color','green');
legend({'10 snapshots','20 snapshots','All snapshots'});
grid on;
```