Question 1

```
% Parameters
A1 = 1; A2 = exp(1i*pi/3); M = 64; N = 256; fc = 10/M
```

```
fc = 0.1562
```

```
delta_list = [5 2 1 0.8 0.5 0.3];
delta = delta_list(1);
f1 = fc - delta./(2*N)
```

```
f1 = 0.1465
```

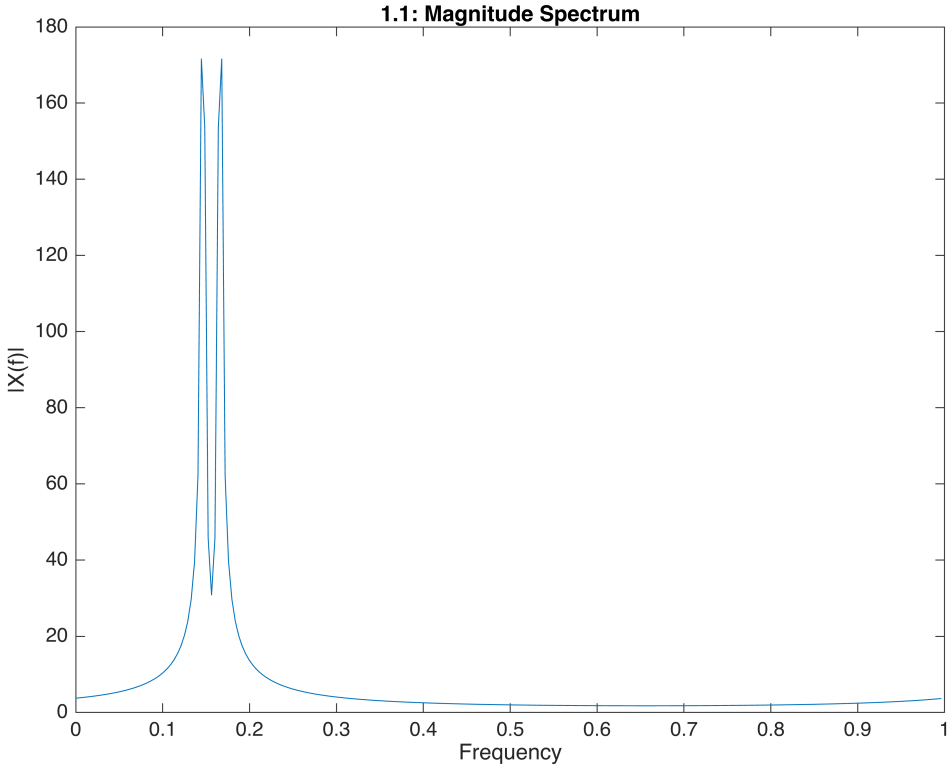
```
f2 = fc + delta./(2*N)
```

```
f2 = 0.1660
```

```
n = 1:1:N;

% Noise
sigma = 0;
mean = 0;

w = mean + sigma * (randn(1, N) + 1i*randn(1, N));
x = A1*exp(1i*2*pi*f1*(n-1))+A2*exp(1i*2*pi*f2*(n-1))+w;
```

1.

```
% DFT
X = fft(x, N);
f = (0:N-1)/N;

figure;
plot(f, abs(X));
xlabel('Frequency');
ylabel('|X(f)|');
title('1.1: Magnitude Spectrum');
```



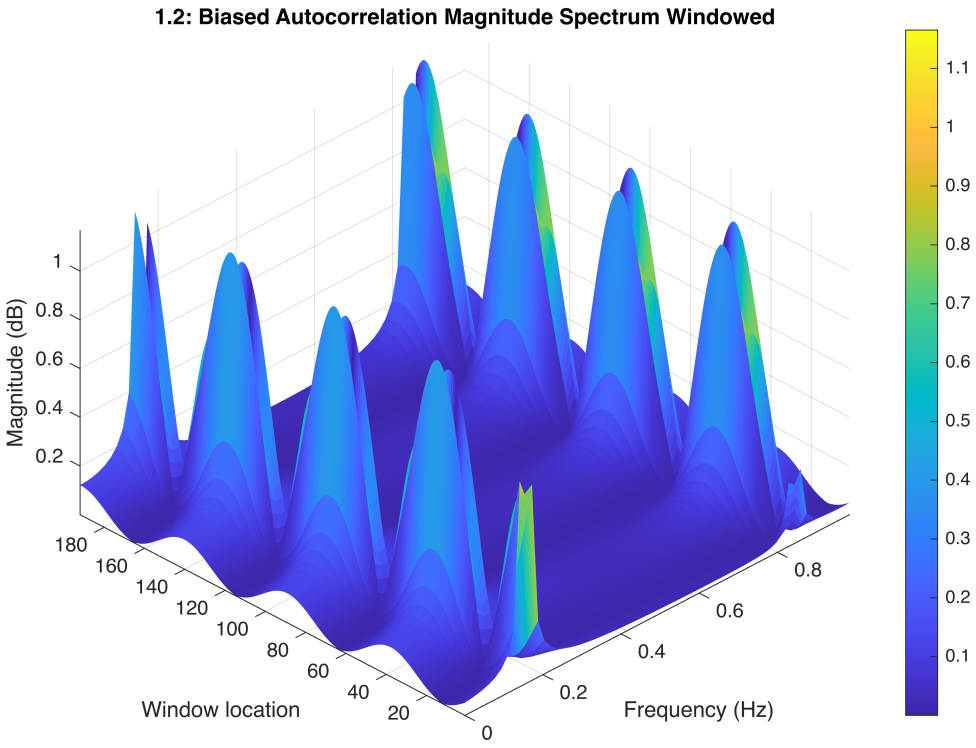
Changing σ changes the frequency spread of the noise. When σ is large, the magnitude spectrum is very cluttered with a lot of frequencies contributing to the spectrum (high noise). But when sigma is small, the magnitude spectrum peaks at f_1 and f_2 are visible (low noise).

When Δ is decreased, the difference between f_1 and f_2 becomes smaller and hence the two peaks gradually merge together into a single peak at $\Delta = 0.3$ centered at $\frac{f_1 + f_2}{2}$. However, we should ideally still see two peaks very close together but the reason we see one peak is due to the limit amount of samples that was taken.

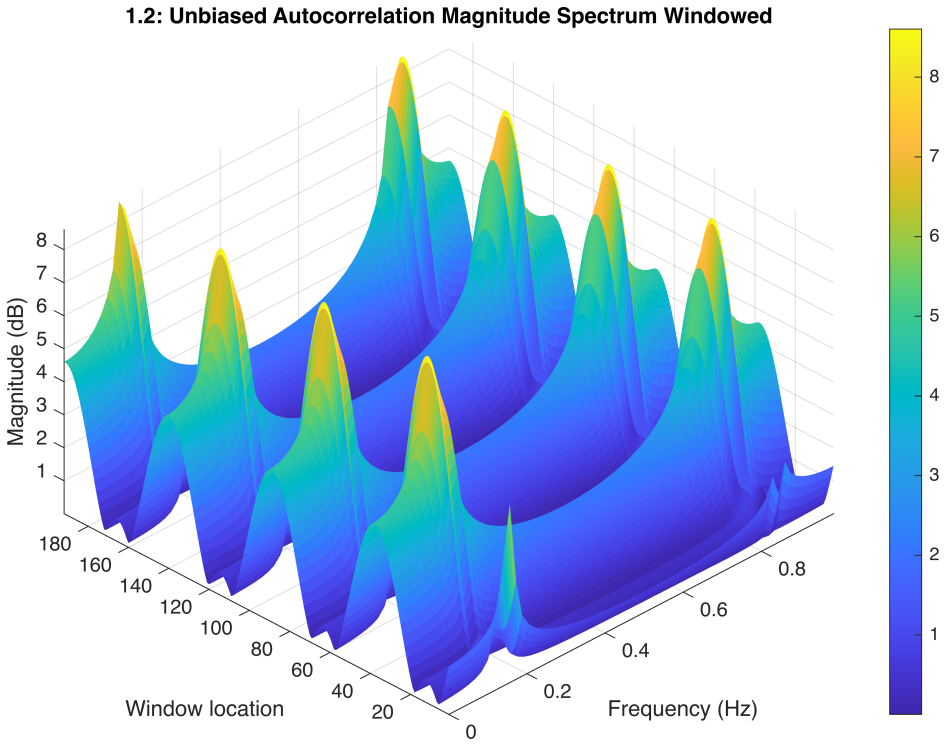
2.

```
N_wind = M;
r_biased = zeros(N-M, N_wind);
r_unbiased = zeros(N-M, N_wind);

% Biased
for n=1:N-M
    r_biased(n,:) = r_biased(n,:) + autocorr(n, 1,x);
end
f = (0:N_wind-1)*(1/N_wind);
n = 1:192;
surf(f, n, abs(r_biased), 'EdgeColor', 'none');
axis tight;
xlabel('Frequency (Hz)');
ylabel('Window location');
```

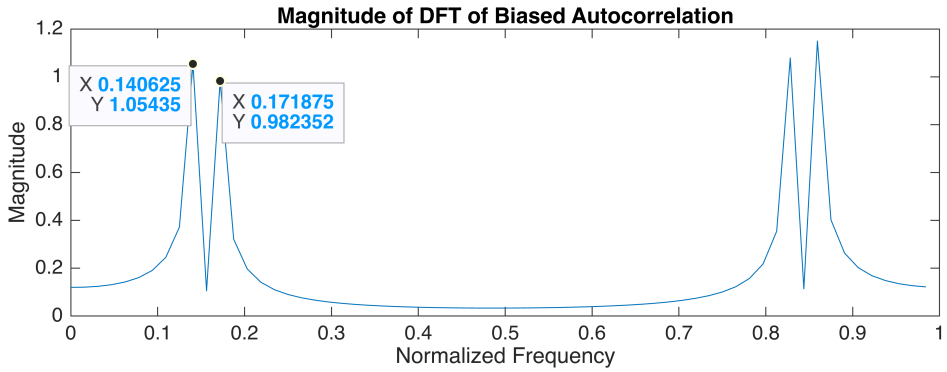
```
zlabel('Magnitude (dB)');
title('1.2: Biased Autocorrelation Magnitude Spectrum Windowed');
colorbar;
view(-45, 45);
```

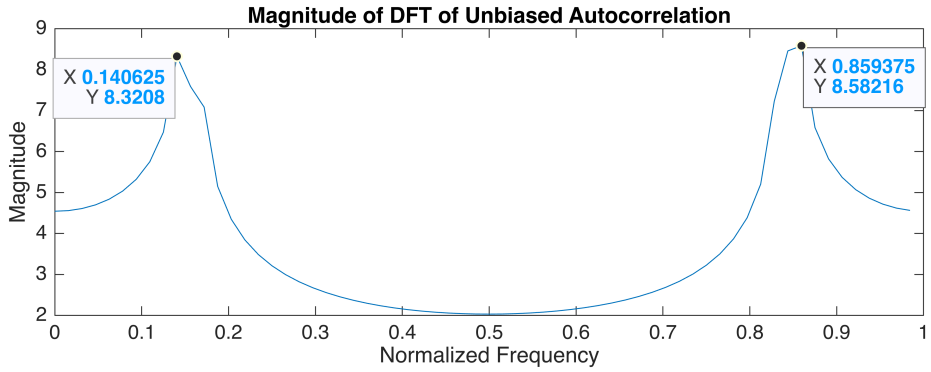



```
% Unbiased
for n=1:N-M
    r_unbiased(n,:) = r_unbiased(n,:) + autocorr(n, 0,x);
end
f = (0:N_wind-1)*(1/N_wind);
n = 1:192;
surf(f, n, abs(r_unbiased), 'EdgeColor', 'none');
axis tight;
xlabel('Frequency (Hz)');
ylabel('Window location');
zlabel('Magnitude (dB)');
title('1.2: Unbiased Autocorrelation Magnitude Spectrum Windowed');
colorbar;
view(-45, 45);
```




```
realisation = 36;
figure;
subplot(2,1,1);
plot(f, abs(r_biased(realisation,:)));
title('Magnitude of DFT of Biased Autocorrelation');
xlabel('Normalized Frequency');
ylabel('Magnitude');

subplot(2,1,2);
plot(f, abs(r_unbiased(realisation,:)));
title('Magnitude of DFT of Unbiased Autocorrelation');
xlabel('Normalized Frequency');
ylabel('Magnitude');
```





We see peaks in the expected f1 and f2 locations. But there's also repetitions of those peaks around 0.8Hz. I probably messed up my code somewhere. The magnitude oscillates with changes to the window location. This suggests that the autocorrelation magnitude is dependent on the alignment of the window with the data's inherent periodicities. I.e. how the beginning and end points of the window coincide with the phase of periodic components in the data. The biased method should provide lower variance than the unbiased. This can be seen in the plots where the unbiased two peaks cannot be distinguished. This effect will become worse for smaller window sizes. The unbiased magnitude is larger since the normalisation constant in that case is $\frac{1}{N-k}$ which is smaller than the biased case. It's required to take the magnitude of the DFT of the autocorrelation function since the outputs are complex.

Question 2

DFT uses discrete samples so it's considered to have low resolution

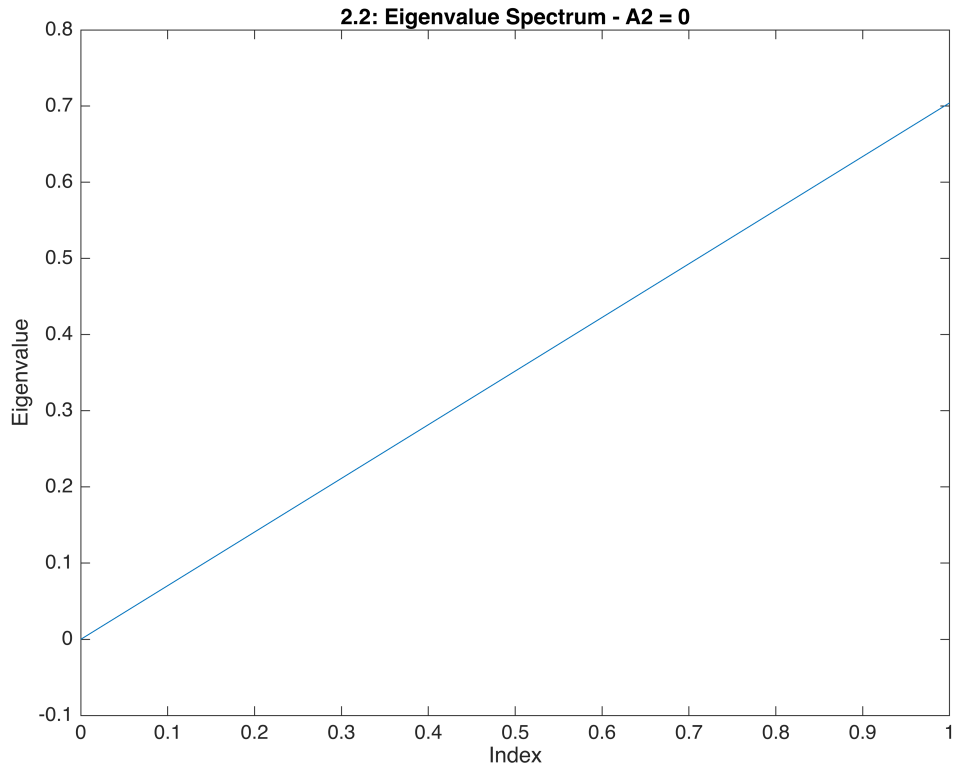
1.

```
sigma = 0.1;
n = 1:1:N;
w = mean + sigma * (randn(1, N) + 1i*randn(1, N));
x = A1*exp(1i*2*pi*f1*(n-1))+A2*exp(1i*2*pi*f2*(n-1))+w;

% Correlation matrix
R = zeros(M, M);
start = 1;
x_windowed = x(start:start+M-1);
```

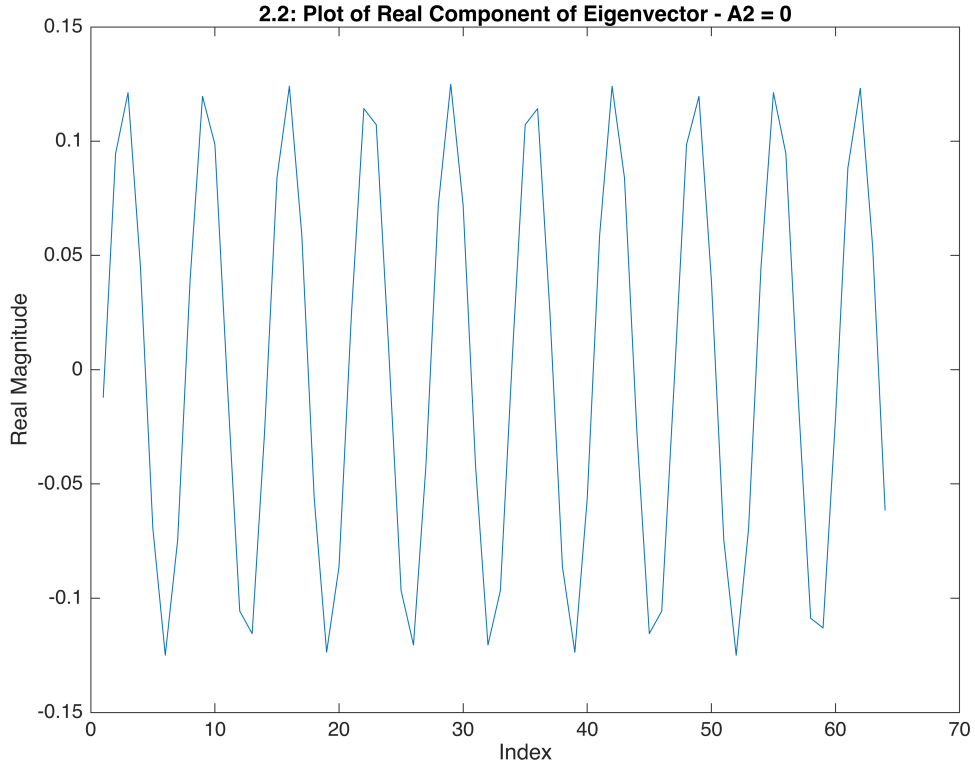
```
for j = 1:M
    for k = 1:M
        R(j, k) = R(j, k) + x_windowed(j) * x_windowed(k);
    end
end
%%%% Do we need to take the expectation value for R? %%%%%
```

2.

```
% A1 = 0
R = genR(0,64,2);
[eigVectors, eigValues] = eig(R);
figure;
plot(diag(eigValues)); %%%% Check this is correct %%%%
title('2.2: Eigenvalue Spectrum - A2 = 0');
xlabel('Index');
ylabel('Eigenvalue');
```



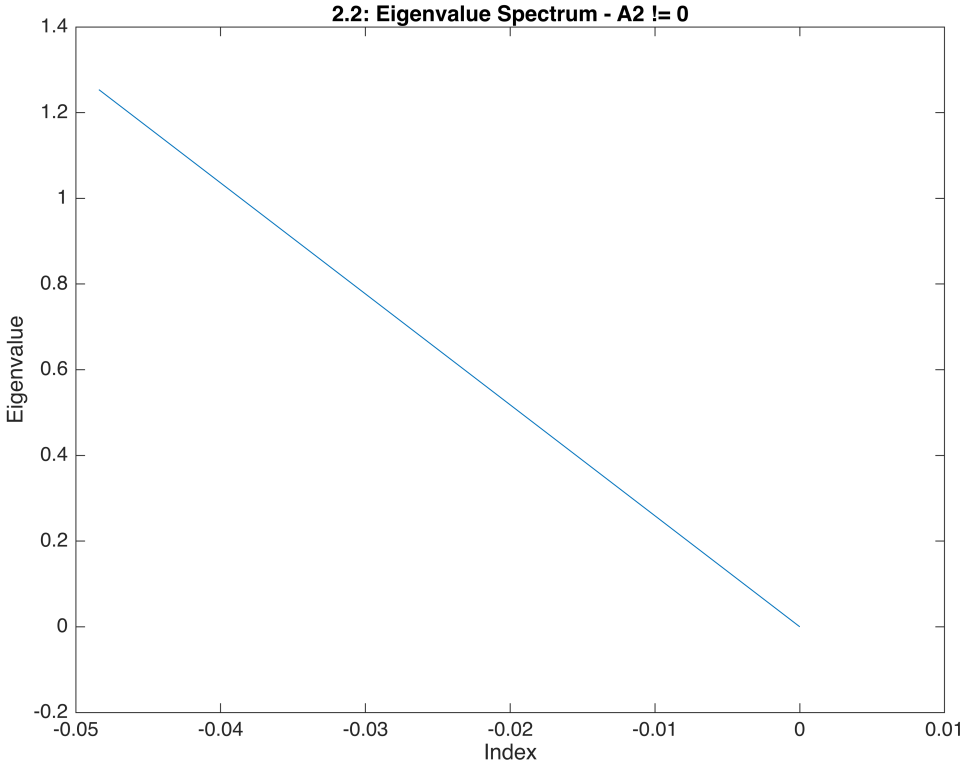

```
% Max eigenvalue
[max_eig, index] = max(diag(eigValues));
max_vec = eigVectors(:, index);
plot(real(max_vec))
title('2.2: Plot of Real Component of Eigenvector - A2 = 0');
xlabel('Index');
ylabel('Real Magnitude');
```



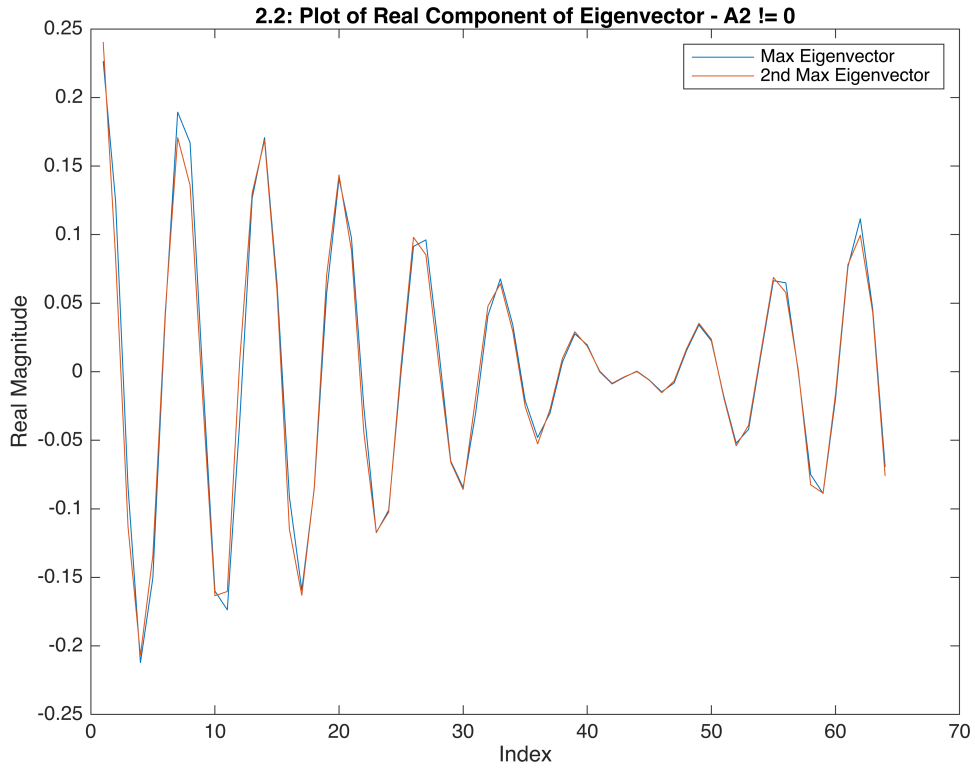

```
% calculate frequency of signal
N = length(real(max_vec));
Y = fft(real(max_vec));
P2 = abs(Y/N);
P1 = P2(1:N/2+1);
P1(2:end-1) = 2*P1(2:end-1);
f = (0:(N/2))/N;
[~, peakIndex] = max(P1);
% plot(f,P1)
frequency = f(peakIndex)
```

```
frequency = 0.1562
```

```
% A1 != 0
R = genR(1,64,2);
[eigVectors, eigValues] = eig(R);
figure;
plot(diag(eigValues)); %%%% Check this is correct %%%%
title('2.2: Eigenvalue Spectrum - A2 != 0');
xlabel('Index');
ylabel('Eigenvalue');
```




```
% Find max and 2nd max eigenvalue
[sortedEigValues, sortOrder] = sort(diag(eigValues), 'descend');
eigVectorsSorted = eigVectors(:, sortOrder);
max_vec = eigVectorsSorted(:, 1:2);  % change this for max and 2nd max

% Plot real part of each graph with legend
plot(real(max_vec(:,1)), 'DisplayName', 'Max Eigenvector');
hold on;
plot(real(max_vec(:,2)), 'DisplayName', '2nd Max Eigenvector');
hold off;
title('2.2: Plot of Real Component of Eigenvector - A2 != 0');
xlabel('Index');
ylabel('Real Magnitude');
legend;
```




```
% calculate frequency of signal
max_vec = max_vec(:,2);
N = length(real(max_vec));
Y = fft(real(max_vec));
P2 = abs(Y/N);
P1 = P2(1:N/2+1);
P1(2:end-1) = 2*P1(2:end-1);
f = (0:(N/2))/N;

[~, peakIndex] = max(P1);
frequencyA2 = f(peakIndex)
```

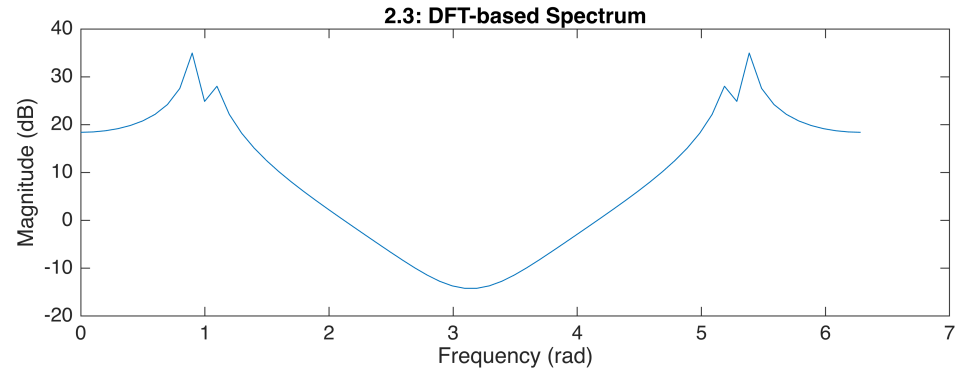
```
frequencyA2 = 0.1562
```

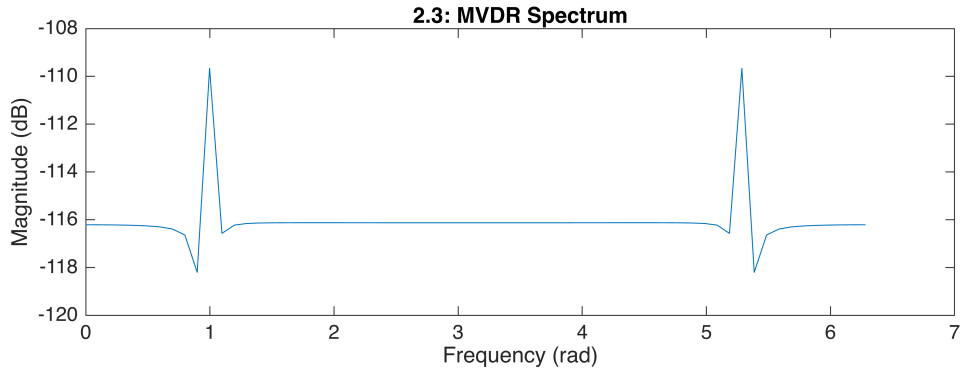
A2 = 0: The frequency of oscillation is f = 0.154Hz which is due to the frequency f1, though it is slightly larger.

A2 !=0 (two component signal): Frequency stays the same, but oscillations look nearly the same for the max and 2nd max eigenvalues. When $\Delta = 2$, max/2nd max waves are in phase, but when $\Delta = 0.8$, these waves are out of phase by 180◦.

```
N = size(R,1);
w = linspace(0, 2*pi, N);
```

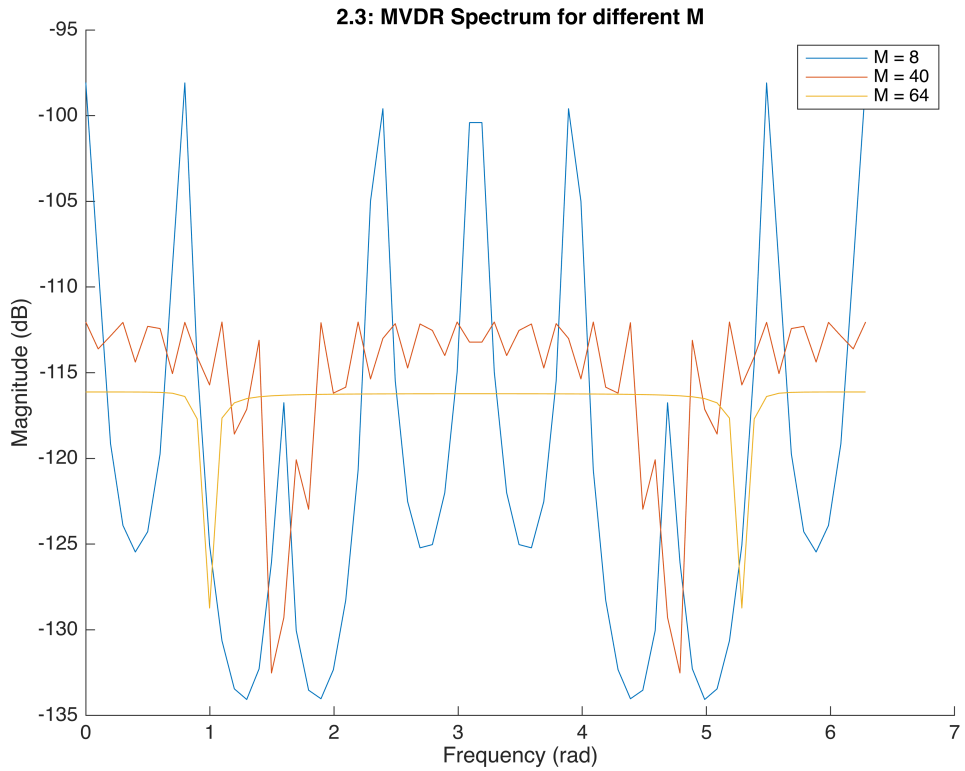
```
% Regularising the matrix R
epsilon = 1e-4;
R_reg = R + epsilon * eye(N);

% DFT spectrum
dft_spectrum = zeros(1, N);
for k = 1:N
    a = exp(-1j * w(k) * (0:N-1)');
    dft_spectrum(k) = real(a' * R_reg * a);
end

% MVDR spectrum
mvdr_spectrum = zeros(1, N);
for k = 1:N
    a = exp(-1j * w(k) * (0:N-1)');
    mvdr_spectrum(k) = 1 / (a' * (R_reg \ a));
end

figure;
subplot(2,1,1);
plot(w, 20*log10(abs(dft_spectrum)));
title('2.3: DFT-based Spectrum');
xlabel('Frequency (rad)');
ylabel('Magnitude (dB)');

subplot(2,1,2);
plot(w, 20*log10(abs(mvdr_spectrum)));
title('2.3: MVDR Spectrum');
xlabel('Frequency (rad)');
ylabel('Magnitude (dB)');
```






```
% Generate MVDR spectra for various M
figure;
hold on;

M_loop = [8, 40, 64];

for M = M_loop
    R = genR(1, M, 0.2); % Generate R with M changing
    R_reg = R + 1e-4 * eye(M); % Regularisation

    mvdr_spectrum = zeros(1, N);
    for k = 1:N
        a = exp(-1j * w(k) * (0:M-1)');
        mvdr_spectrum(k) = 1 / (a' * (R_reg \ a));
    end

    plot(w, 20*log10(abs(mvdr_spectrum)), 'DisplayName', sprintf('M = %d',
M));
end

title('2.3: MVDR Spectrum for different M');
xlabel('Frequency (rad)');
ylabel('Magnitude (dB)');
legend show;
```

```
hold off;
```



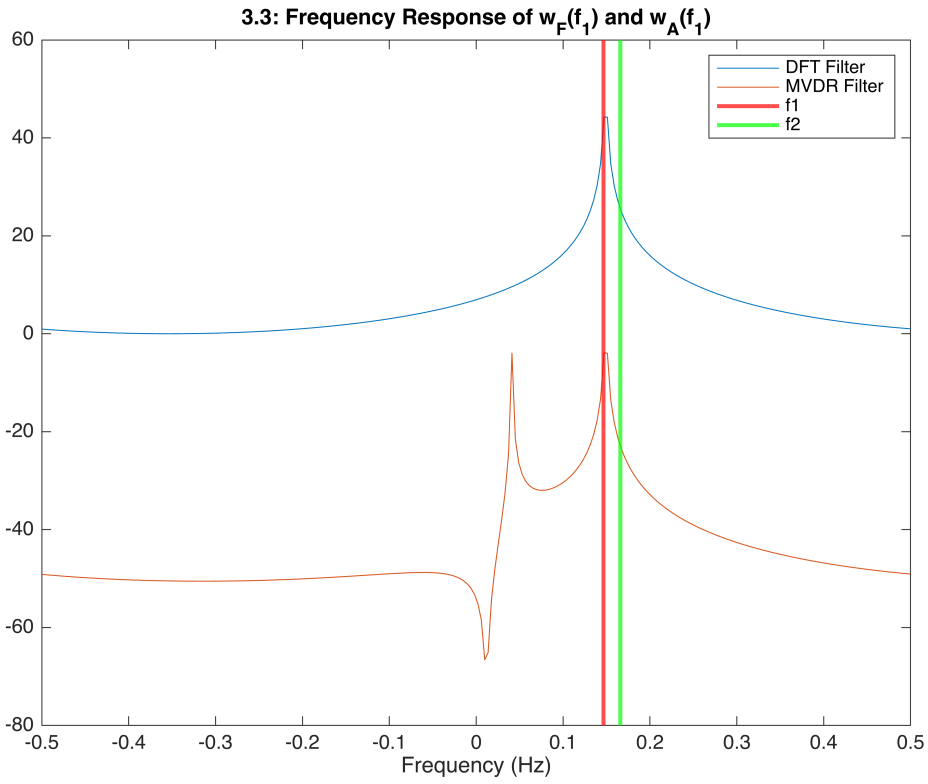
The larger the value of M, the more defined the spectrum is, which makes sense because we essentially have more samples to work with. However, my magnitude seems to be flipped for some reason when plotting multiple M. The MVDR spectrum is more defined and less variance than the DFT since it is an adaptive to the inputs. Further discussion will occur about this in question 3.6.

Question 3

1. $w_F(f)[n] = e^{-i2\pi fn/N}$
2. $w_A(f) = \frac{R^{-1}a(f)}{a(f)^H R^{-1} a(f)}$
3.

```
N_wind = 256;
% Parameters
n = 0:N_wind-1;
% DFT Filter Weight Vector
wF = exp(1j * 2 * pi * f1 * n);

% MVDR Filter Weight Vector
R = genR(1,N_wind,0.2);
R_reg = R + 1e-4 * eye(N_wind); % Regularisation

a_f1 = exp(1j * 2 * pi * f1 * n).';
```

```
wA = (R_reg \ a_f1) / (a_f1' * (R_reg \ a_f1));
% Frequency responses
fs = 1;
f = linspace(-fs/2,fs/2,N_wind);

figure;
plot(f, 20 * log10(abs(fftshift(fft(wF)))));
hold on;
plot(f, 20 * log10(abs(fftshift(fft(wA)))));
title('3.3: Frequency Response of w_F(f_1) and w_A(f_1)');
xlabel('Frequency (Hz)');
xline(f1, 'r', 'LineWidth', 2);
xline(f2, 'g', 'LineWidth', 2);
legend('DFT Filter', 'MVDR Filter', 'f1', 'f2');
```

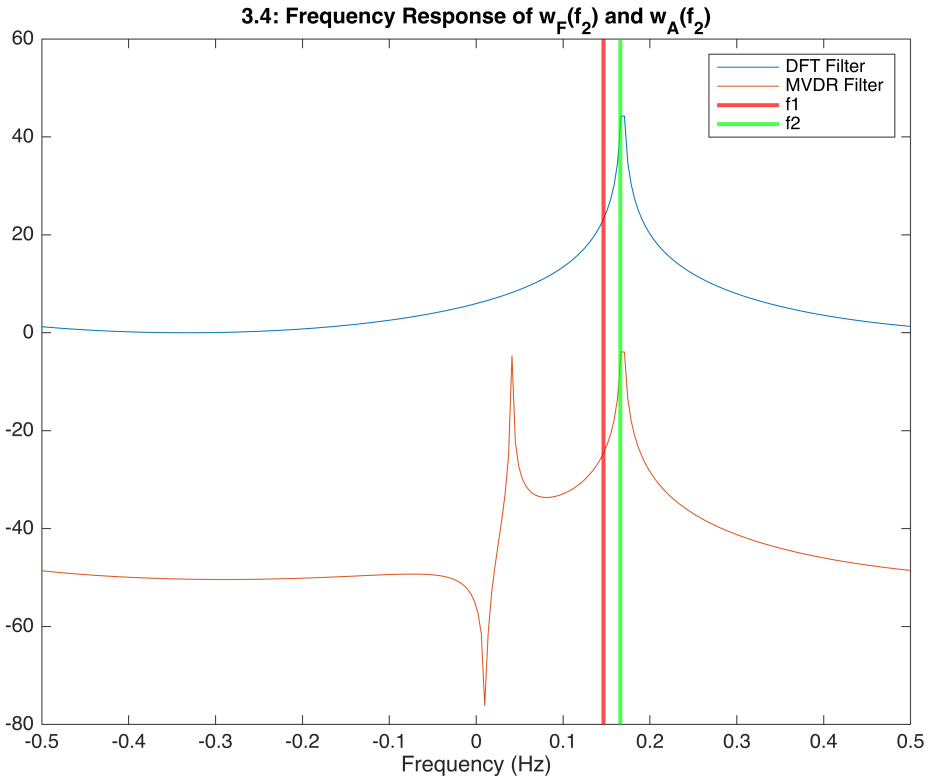


The MVDR and Fourier weights both peak at (approximately) f1, this just shows how the weights were designed for high selectivity at f1. The MVDR peak and Fourier peak both have the same sharpness and there is a peak in the MVDR spectrum around 0.04Hz which is weird, perhaps just a coding error. The magnitude of MVDR is lower since the weight vector is less dependent on the f1 frequency.

4.

```
% Parameters
n = 0:N_wind-1;
% DFT Filter Weight Vector
wF = exp(1j * 2 * pi * f2 * n);
```

```
% MVDR Filter Weight Vector
R = genR(1,N_wind,0.2);
R_reg = R + 1e-4 * eye(N_wind); % Regularisation

a_f2 = exp(1j * 2 * pi * f2 * n).';
wA = (R_reg \ a_f2) / (a_f2' * (R_reg \ a_f2));
% Frequency responses
fs = 1;
f = linspace(-fs/2,fs/2,N_wind);

figure;
plot(f, 20 * log10(abs(fftshift(fft(wF)))));
hold on;
plot(f, 20 * log10(abs(fftshift(fft(wA)))));
title('3.4: Frequency Response of w_F(f_2) and w_A(f_2)');
xlabel('Frequency (Hz)');
xline(f1, 'r', 'LineWidth', 2);
xline(f2, 'g', 'LineWidth', 2);
legend('DFT Filter', 'MVDR Filter', 'f1', 'f2');
```

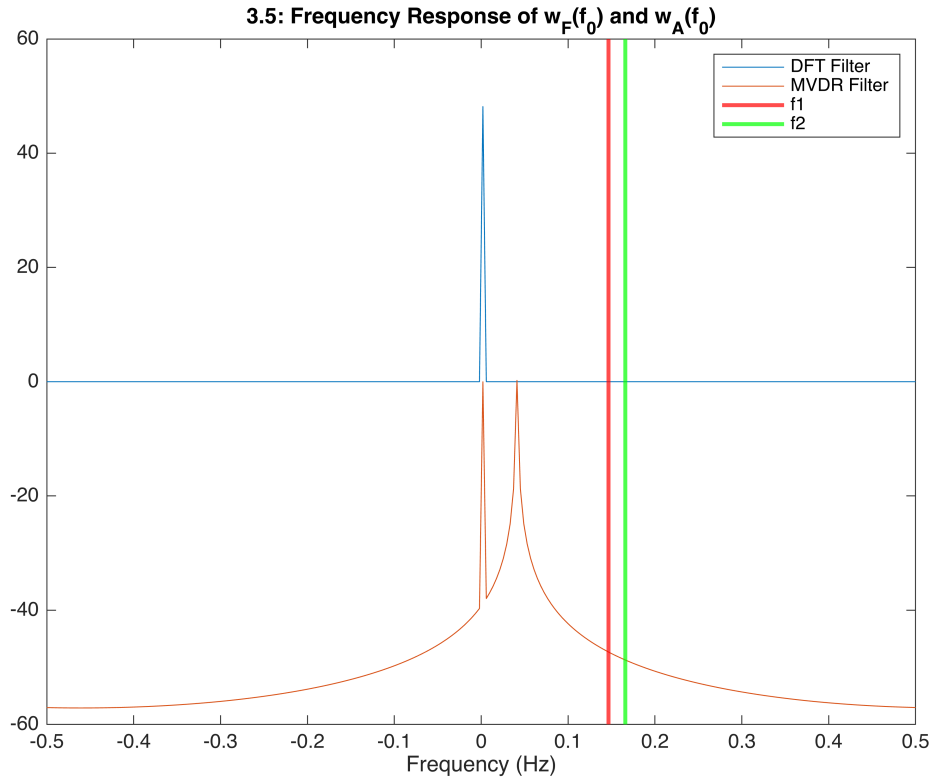


The two peaks in the DFT and MVDR plots now peak around f2 which is what we expect. The features of this graph is similar to the last graph.

5.

```
% Parameters
n = 0:N_wind-1;
```

```
f0 = 0;
% DFT Filter Weight Vector
wF = exp(1j * 2 * pi * f0 * n);

% MVDR Filter Weight Vector
R = genR(1,N_wind,0.2);
R_reg = R + 1e-4 * eye(N_wind); % Regularisation

a_f0 = exp(1j * 2 * pi * f0 * n).';
wA = (R_reg \ a_f0) / (a_f0' * (R_reg \ a_f0));
% Frequency responses
fs = 1;
f = linspace(-fs/2,fs/2,N_wind);

H = (abs(fftshift(fft(wF))));
H(129) = mag2db(H(129));
figure;
plot(f, H);
hold on;
plot(f, mag2db(abs(fftshift(fft(wA)))));
title('3.5: Frequency Response of w_F(f_0) and w_A(f_0)');
xlabel('Frequency (Hz)');
xline(f1, 'r', 'LineWidth', 2);
xline(f2, 'g', 'LineWidth', 2);
legend('DFT Filter', 'MVDR Filter', 'f1', 'f2');
```



Both MVDR and Fourier plots has a peak at 0Hz as expected and the side peak is still there in the MVDR plot. I have plotted taken the log of only the non-zero entry in the fft(wF) because otherwise I would get -Inf in my array which is unplottable.

6.

The MVDR allows for spatial filtering. In the previous tasks, it involves specifying a window of size M to select what signals to use. In real applications, this allows the focus on signals from a specific direction. It does this by using the the covariance matrix of the signal. The covariance matrix holds the strength and spatial correlation of signals and noise. The weights use this matrix to minimize the variance and interference of noise from other directions. In contrast, the Fourier method splits the signal into its constituent frequencies. It does not provide spatial filtering. Therefore, the MVDR method allows for more effective at improving signal clarity and interference rejection compared to the Fourier method. This was seen in exerciese 1 and 2. However, in exercise 3, the MVDR method gave additional peaks which was unexapected. It might have been an error when calculating the covariance matrix.

```
% Function that computes autocorrelation matrix
function output = autocorr(start, isbiased, x)
    M = 64;
    x_windowed = x(start:1:start+(M-1));

    N = length(x_windowed);
    r_biased = zeros(1, N);
    r_unbiased = zeros(1, N);

    for m = 1:N
        for n = 1:N-m+1
            r_biased(m) = r_biased(m) + x_windowed(n)*x_windowed(n+m-1);
            r_unbiased(m) = r_unbiased(m) + x_windowed(n)*x_windowed(n+m-1);
        end
        r_biased(m) = r_biased(m) / N;
        r_unbiased(m) = r_unbiased(m) / (N-m+1);
    end

    r_biased = fft(r_biased, N);
    r_unbiased = fft(r_unbiased, N);

    if isbiased == 1
        output = r_biased;
    else
        output = r_unbiased;
    end
```

```
end

% Generates the correlation matrix. Coeff controls if A2 is 0, and inputs
% are window size M and delta.
function R = genR(coeff, M, delta)
% Parameters
A1 = 1; A2 = coeff * exp(1i*pi/3); N = 256; fc = 10/M;
f1 = fc - delta./(2*N);
f2 = fc + delta./(2*N);
n = 1:1:N;
% Noise
sigma = 0;
mean = 0;

w = mean + sigma * (randn(1, N) + 1i*randn(1, N));
x = A1*exp(1i*2*pi*f1*(n-1))+A2*exp(1i*2*pi*f2*(n-1))+w;

% Correlation matrix
R = zeros(M, M);
start = 1;
x_windowed = x(start:start+M-1);
for j = 1:M
    for k = 1:M
        R(j, k) = R(j, k) + x_windowed(j) * x_windowed(k);
    end
end
%%%% Do we need to take the expectation value for R? %%%%%
end
```

Question 4

1.

```
[X,Targs] = radar_sig_gen(4);
size_X = size(X);
size_Targs = size(Targs);
Targs
```

Targs = 1×4 struct

Fields	Range	DOA	SINR
1	105	-0.4276	15.4614
2	50	-0.6840	24.2372
3	202	0.5239	21.4505
4	101	-0.1414	20.7532

```
Na = size_X(1); % Number of antenna elements
Np = size_X(2); % Number of pulses
Nr = size_X(3); % Angle grid
```

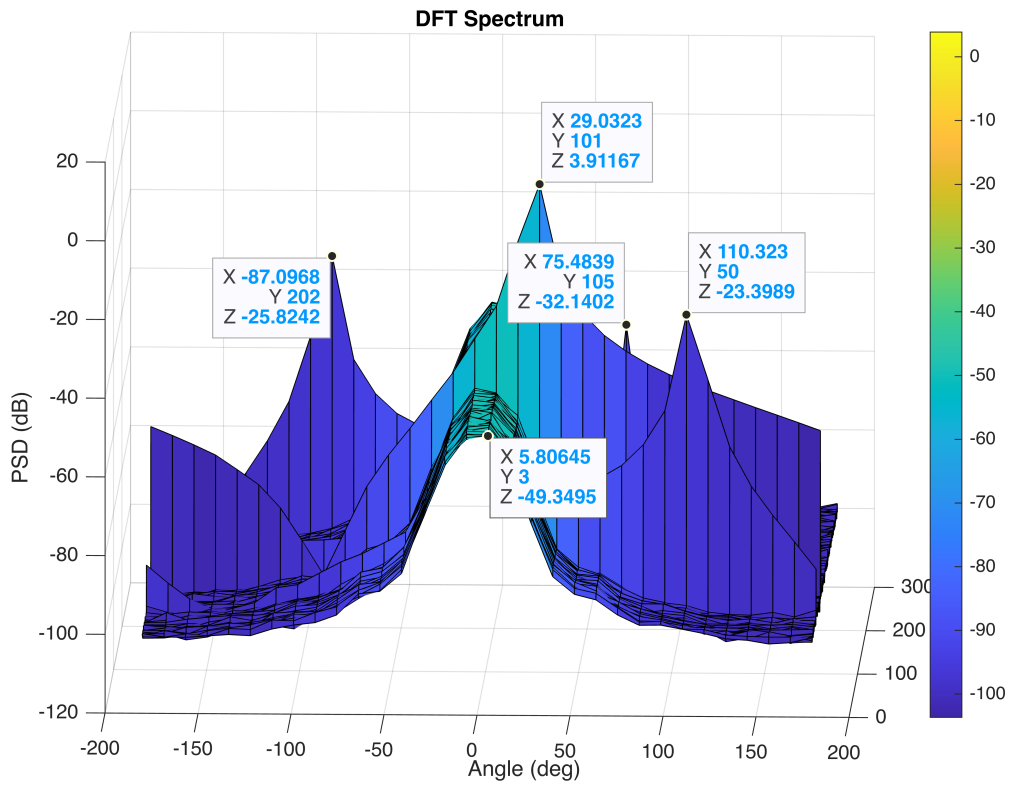
2.

```
R = zeros(Na);
for r = 1:Nr
    data_r = X(:,:,r);
    R(:,:,r) = (data_r * data_r') / Np;
end
```

3.

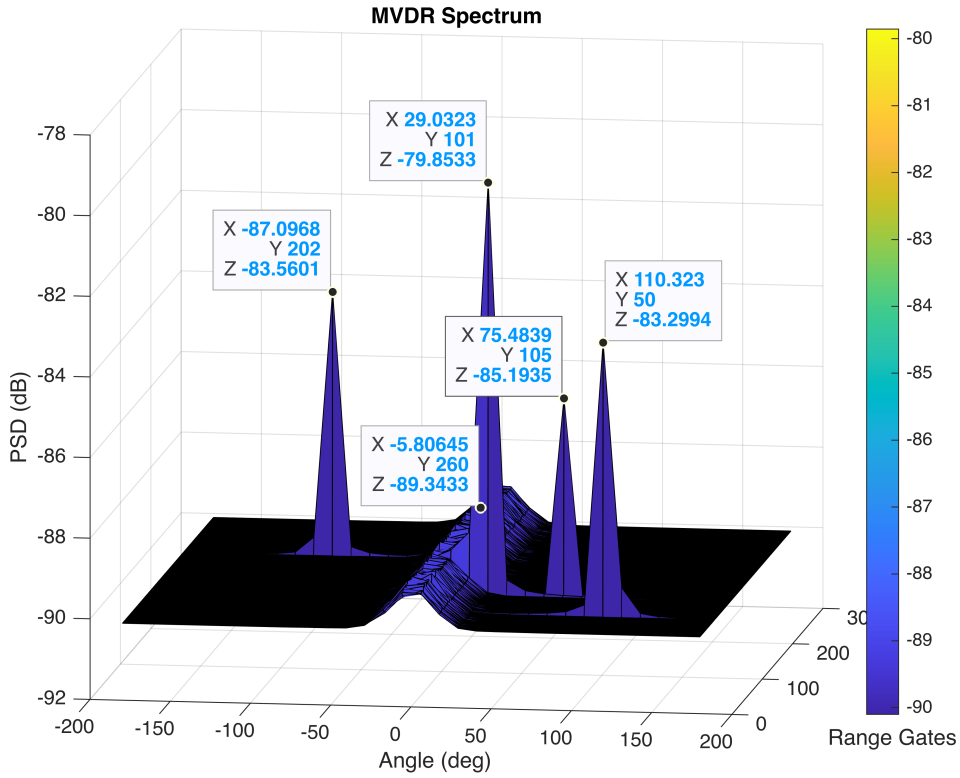
```
Na = 32;
Np = 128;
Nr = 300;
w = linspace(-pi, pi, Na);

dft_spectrum = zeros(Nr, Na);
mvdr_spectrum = zeros(Nr, Na);

for r = 1:Nr
    % Extract range gate data from R
    R_r = squeeze(R(:,:,r));

    % Regularization
    epsilon = 1e-3;
    R_reg = R_r + epsilon * eye(Na);

    for k = 1:Na
        a = exp(-1j * w(k) * (0:Na - 1).');
```

```
        % DFT
        dft_spectrum(r, k) = real(a' * R_r * a);

        % MVDR
        mvdr_spectrum(r, k) = 1 / (a' * (R_reg \ a)); % Backslash does
inverse matrix thing
    end
end

% Convert to dB
dft_spectrum_db = mag2db(abs(dft_spectrum) + eps); % Adding eps to avoid
log of zero
mvdr_spectrum_db = mag2db(abs(mvdr_spectrum) + eps);

% Plottingg
w = rad2deg(w);
figure;
surf(w, 1:Nr, dft_spectrum_db);
xlabel('Angle (deg)');
ylabel('Range Gates');
zlabel('PSD (dB)');
title('DFT Spectrum');
colorbar;
```




```
figure;
surf(w, 1:Nr, mvdr_spectrum_db);
```

```
xlabel('Angle (deg)');
ylabel('Range Gates');
zlabel('PSD (dB)');
title('MVDR Spectrum');
colorbar;
```




```
Targs
```

Targs = 1×4 struct

Fields	Range	DOA	SINR
1	105	-0.4276	15.4614
2	50	-0.6840	24.2372
3	202	0.5239	21.4505
4	101	-0.1414	20.7532

The targets can be seen in both MVDR and DFT spectrum. Evidently, the MVDR peaks for the targets are much more defined and sharp than the corresponding DFT spectrum. In both spectrums, there is a bump at angle = 0. This is due to the main lobe of the antenna design. It's the direction in which the antenna is designed to radiate/receive energy most effectively. It's usually directed to the region where we expect to detect targets which can be seen from the plots since the peaks tend to be around angle=0.

4. Both the DFT and MVDR methods predict the exact values for the range of the targes. The DFT method was harder to make out the targets because of the wide peaks covering up the plot. The DOA were not accurate for

both methods, it seems to be off by a certain proportion for each peak. Perhaps it's an error with the bounds of the "w = linspace(-pi, pi, Na); " line.

Question 4

1.

```
[X,Targs] = radar_sig_gen(4);
size_X = size(X);
size_Targs = size(Targs);
Targs
```

Targs = 1×4 struct

Fields	Range	DOA	SINR
1	105	-0.4276	15.4614
2	50	-0.6840	24.2372
3	202	0.5239	21.4505
4	101	-0.1414	20.7532

```
Na = size_X(1); % Number of antenna elements
Np = size_X(2); % Number of pulses
Nr = size_X(3); % Angle grid
```

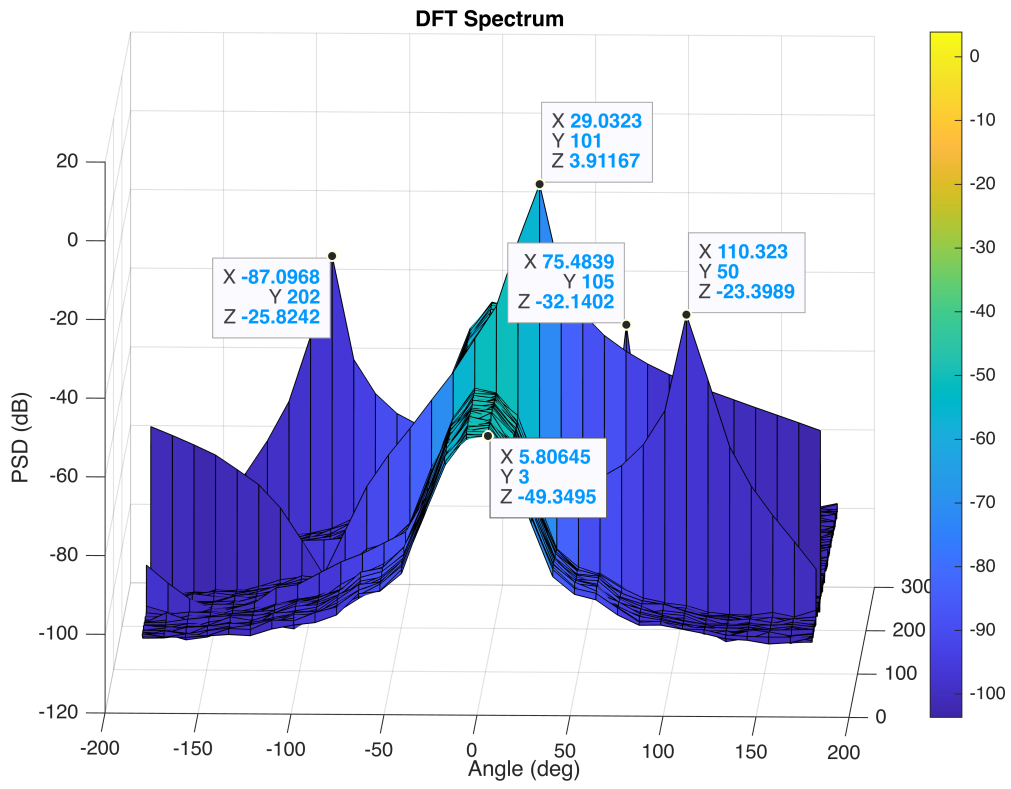
2.

```
R = zeros(Na);
for r = 1:Nr
    data_r = X(:,:,r);
    R(:,:,r) = (data_r * data_r') / Np;
end
```

3.

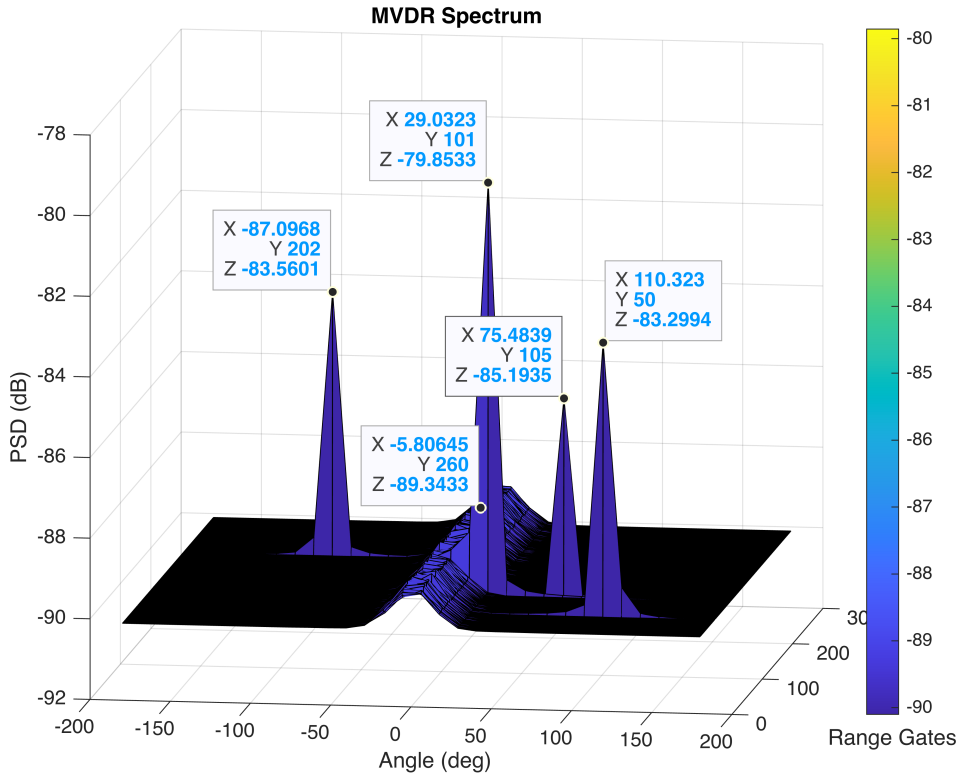
```
Na = 32;
Np = 128;
Nr = 300;
w = linspace(-pi, pi, Na);

dft_spectrum = zeros(Nr, Na);
mvdr_spectrum = zeros(Nr, Na);

for r = 1:Nr
    % Extract range gate data from R
    R_r = squeeze(R(:,:,r));

    % Regularization
    epsilon = 1e-3;
    R_reg = R_r + epsilon * eye(Na);

    for k = 1:Na
        a = exp(-1j * w(k) * (0:Na - 1).');
```

```
        % DFT
        dft_spectrum(r, k) = real(a' * R_r * a);

        % MVDR
        mvdr_spectrum(r, k) = 1 / (a' * (R_reg \ a)); % Backslash does
inverse matrix thing
    end
end

% Convert to dB
dft_spectrum_db = mag2db(abs(dft_spectrum) + eps); % Adding eps to avoid
log of zero
mvdr_spectrum_db = mag2db(abs(mvdr_spectrum) + eps);

% Plottingg
w = rad2deg(w);
figure;
surf(w, 1:Nr, dft_spectrum_db);
xlabel('Angle (deg)');
ylabel('Range Gates');
zlabel('PSD (dB)');
title('DFT Spectrum');
colorbar;
```




```
figure;
surf(w, 1:Nr, mvdr_spectrum_db);
```

```
xlabel('Angle (deg)');
ylabel('Range Gates');
zlabel('PSD (dB)');
title('MVDR Spectrum');
colorbar;
```




```
Targs
```

Targs = 1×4 struct

Fields	Range	DOA	SINR
1	105	-0.4276	15.4614
2	50	-0.6840	24.2372
3	202	0.5239	21.4505
4	101	-0.1414	20.7532

The targets can be seen in both MVDR and DFT spectrum. Evidently, the MVDR peaks for the targets are much more defined and sharp than the corresponding DFT spectrum. In both spectrums, there is a bump at angle = 0. This is due to the main lobe of the antenna design. It's the direction in which the antenna is designed to radiate/receive energy most effectively. It's usually directed to the region where we expect to detect targets which can be seen from the plots since the peaks tend to be around angle=0.

4. Both the DFT and MVDR methods predict the exact values for the range of the targes. The DFT method was harder to make out the targets because of the wide peaks covering up the plot. The DOA were not accurate for

both methods, it seems to be off by a certain proportion for each peak. Perhaps it's an error with the bounds of the "w = linspace(-pi, pi, Na); " line.