CMSC715 Wireless and Mobile Systems for the IoT

Shramay Palta, 117678727 spalta@umd.edu

October 22, 2021

Results, Plots and Source Codes

1) Question 1:

```
[transmitted_signal, sample_rate_t] = ...
    audioread('Data/Task1_SignalDetection/Data/transmitSignal.wav');
[received_signal, sample_rate_r] = audioread('Data/Task1_SignalDetection/Data/7.wav');

dt_t = 1/sample_rate_t;
t_t = 0:dt_t:(length(transmitted_signal)*dt_t)-dt_t;

figure(1);
plot(t_t,transmitted_signal); xlabel('Time in Seconds'); ylabel('Amplitude');
title('Transmitted Signal');

dt_r = 1/sample_rate_r;
t_r = 0:dt_r:(length(received_signal)*dt_r)-dt_r;

figure(2);
plot(t_r,received_signal); xlabel('Time in Seconds'); ylabel('Amplitude');
title('Received Signal');

% I wrote a code in Python to find the sliding window correlation
% Between the transmitted and the received signal
% I cross checked with findsignal
% Which is an inbuilt function in Matlab and got the same answer

[start_index,stop_index,sim_dist] = findsignal(received_signal, transmitted_signal);
```

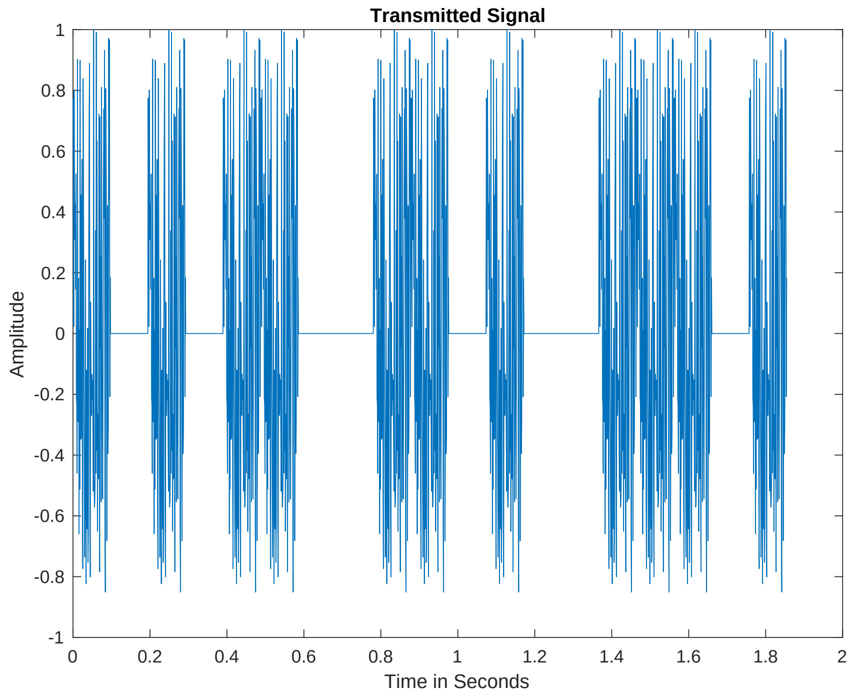


Figure 1: Plot of the Transmitted Signal

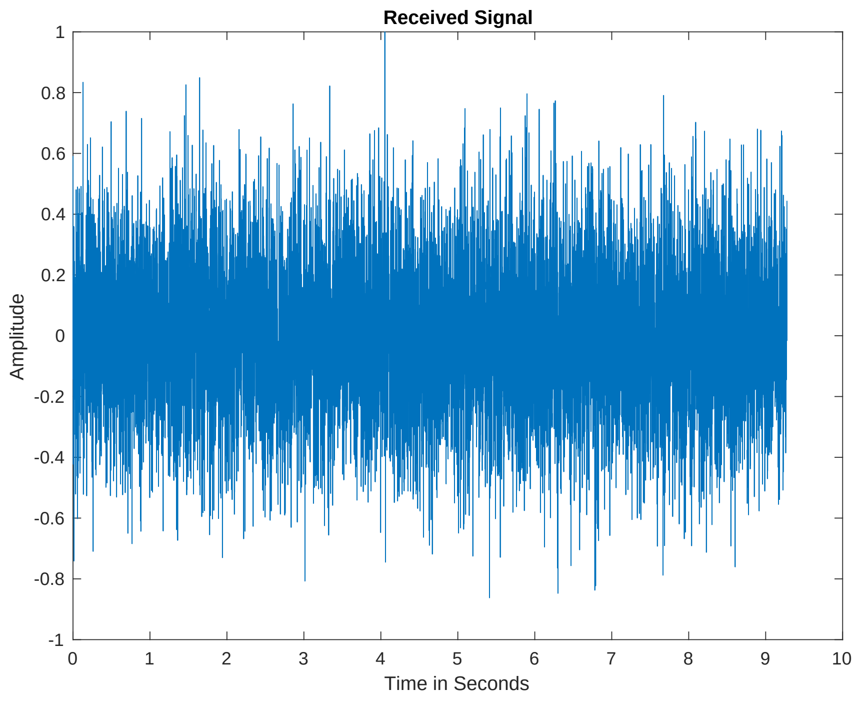
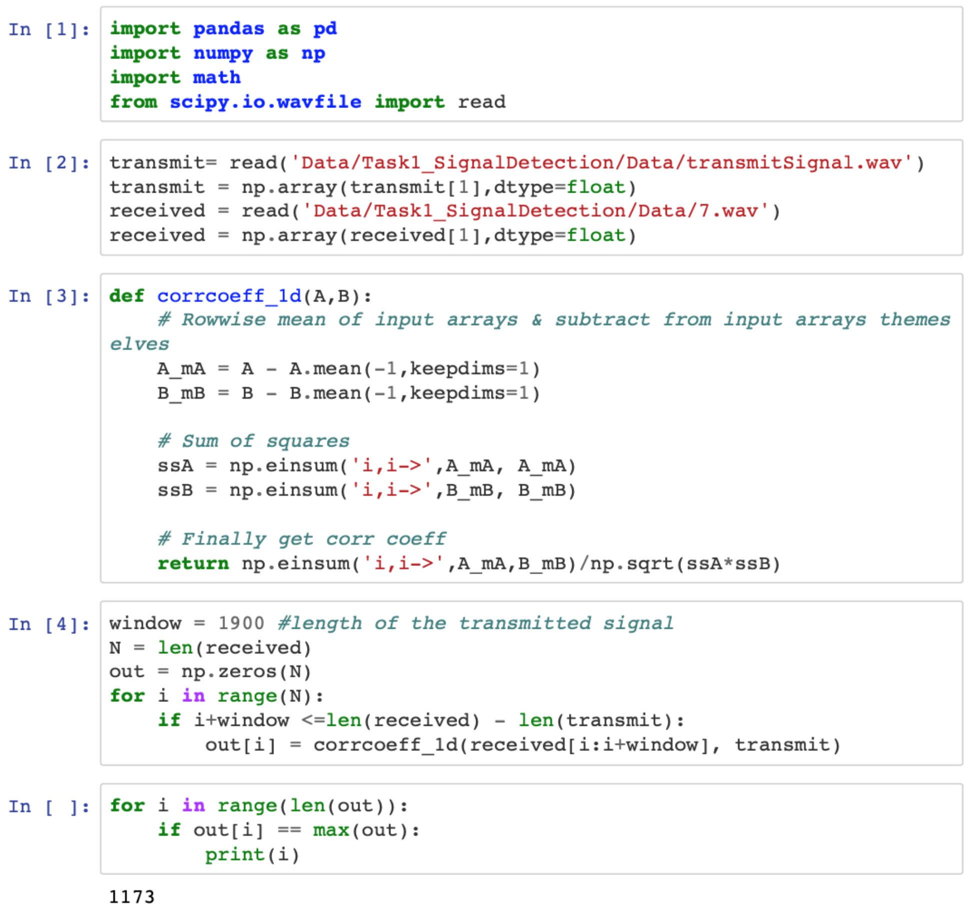
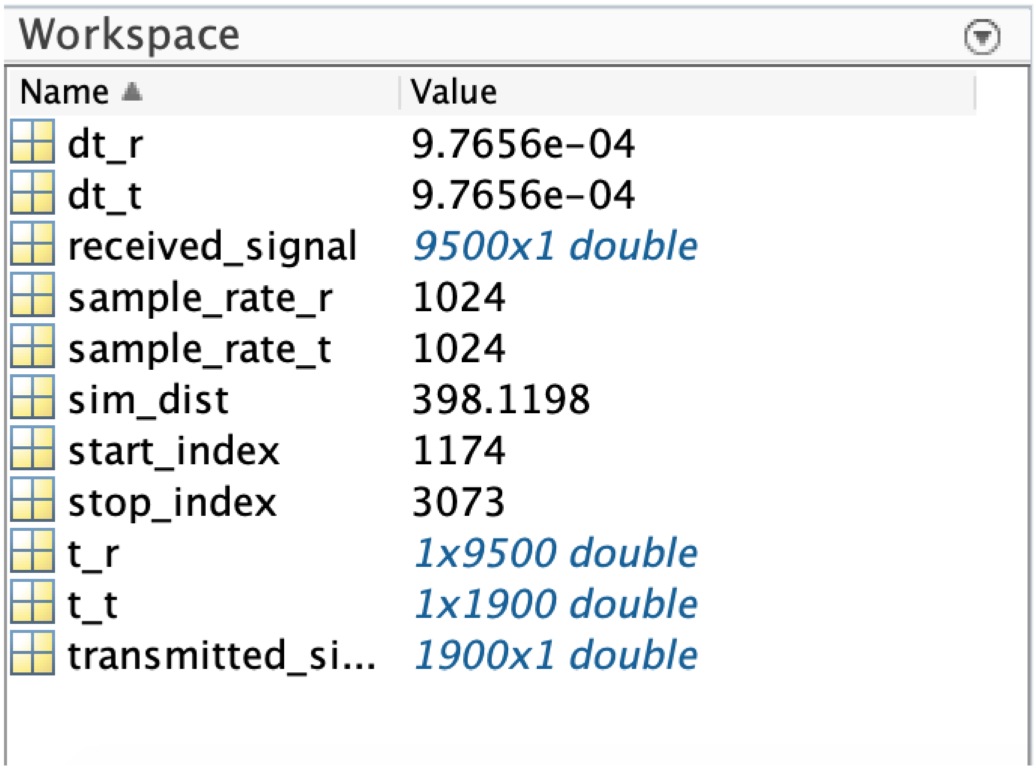


Figure 2: Plot of the Received Signal



```
In [1]: import pandas as pd
        import numpy as np
        import math
        from scipy.io.wavfile import read

In [2]: transmit= read('Data/Task1_SignalDetection/Data/transmitSignal.wav')
        transmit = np.array(transmit[1],dtype=float)
        received = read('Data/Task1_SignalDetection/Data/7.wav')
        received = np.array(received[1],dtype=float)

In [3]: def corrcoeff_1d(A,B):
            # Rowwise mean of input arrays & subtract from input arrays themes
        elves
            A_mA = A - A.mean(-1,keepdims=1)
            B_mB = B - B.mean(-1,keepdims=1)

            # Sum of squares
            ssA = np.einsum('i,i->',A_mA, A_mA)
            ssB = np.einsum('i,i->',B_mB, B_mB)

            # Finally get corr coeff
            return np.einsum('i,i->',A_mA,B_mB)/np.sqrt(ssA*ssB)

In [4]: window = 1900 #length of the transmitted signal
        N = len(received)
        out = np.zeros(N)
        for i in range(N):
            if i+window <=len(received) - len(transmit):
                out[i] = corrcoeff_1d(received[i:i+window], transmit)

In [ ]: for i in range(len(out)):
            if out[i] == max(out):
                print(i)

        1173
```

Figure 3: Python Code to find the sliding window correlation between the transmitted and the received signal.


Workspace

Name	Value
dt_r	9.7656e-04
dt_t	9.7656e-04
received_signal	*9500x1 double*
sample_rate_r	1024
sample_rate_t	1024
sim_dist	398.1198
start_index	1174
stop_index	3073
t_r	*1x9500 double*
t_t	*1x1900 double*
transmitted_si...	*1900x1 double*

Figure 4: MATLAB workspace to show results of ***findsignal()*** which align with the output from the Python code.

Calculations to find the distance from the base station:

```
% Calculations:
% Maximum correlation between the transmitted and received signal occurs
% between index 1174-3074, which was obtained using a sliding window correlation.
% Therefore, assuming t = 0 are synchronized, there was delay of 1174 samples.
% Sample rate of both the transmitted and received signal is 1024.
% Time Delay in seconds = 1174/1024
%                       = 1.146484375 seconds
% Speed of sound = 340 meters per second
% Therefore Distance from Base Station = Speed * Time Delay in seconds
% Distance = 340 * 1.146484375
%          = 389.8046875 meters
```

Therefore, using this data, we find that the distance from the base station is **389.8046875 meters.**

Rationale:

The goal here was to locate the transmitted signal within the received signal. The index from where the transmitted signal was located would give us the delay.

I used the Python code to find the sliding window correlation between the transmitted and the received signal. The output was the starting index as **1173**, which meant that the transmitted signal was located between the indices **1173-3073** in the received signal.

I cross checked using the ***findsignal()*** command in MATLAB which gave me the starting index as **1174**, which accurately matches my answer. *Note that indices in Python start from 0 and in MATLAB, they start from 1.*

Using the starting index, as shown in the calculations, I computed the time delay in seconds, because we already knew the sample rate for both the transmitted and the received signals, which was 1024 samples per second. We knew the speed of sound to be 340 m/s.From here the distance from the base station was found using the formula :

$$Distance = Speed * Time \tag{1}$$

2) Question 2:

```
target_frequency = 2e3;
sound_speed = 340;
lamda = sound_speed/target_frequency;
element_distance = lamda/2;
elements = 10;

angle = 0:pi/180:2*pi;
alpha = zeros(1, length(angle));

for n=1:elements
    beta(n,:) = (n-1)*element_distance*cos(angle);
    alpha = alpha+exp(-1i*2*pi*(beta(n,:)/lamda));
end

figure(1);
polarplot(angle, abs(alpha), 'b');
title('Directional Gain of a 10-element ULA with D = 0.5*wavelength');
```

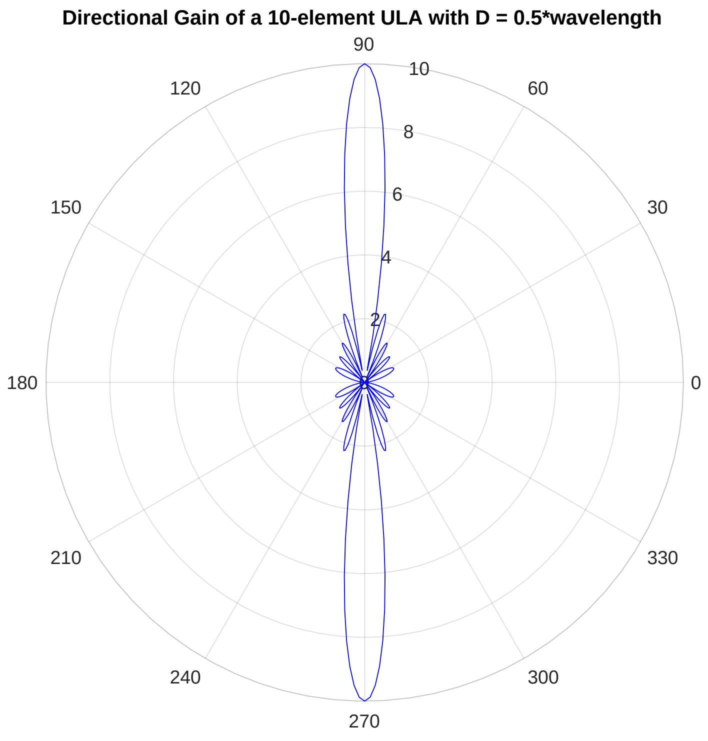


Figure 5: Directional Gain of a 10-element ULA with D = 0.5*λ

```
target_frequency = 2e3;
sound_speed = 340;
lamda = sound_speed/target_frequency;
element_distance = lamda*2;
elements = 10;

angle = 0:pi/180:2*pi;
alpha = zeros(1, length(angle));

for n=1:elements
    beta(n,:) = (n-1)*element_distance*cos(angle);
    alpha = alpha+exp(-1i*2*pi*(beta(n,:)/lamda));
end

figure(1);
polarplot(angle, abs(alpha), 'b');
title('Directional Gain of a 10-element ULA with D = 2*wavelength');
```

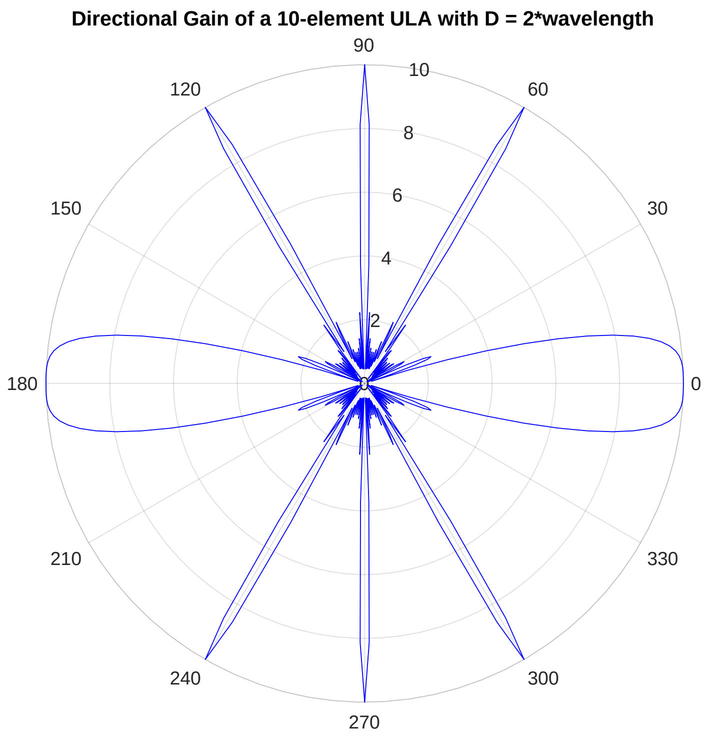


Figure 6: Directional Gain of a 10-element ULA with D = 2*λ.

```
target_frequency = 2e3;
sound_speed = 340;
lamda = sound_speed/target_frequency;
element_distance = lamda/2;
elements = 10;

angle = 0:pi/180:2*pi;
alpha = ones(1, length(angle));

for n=5:elements
    beta(n,:) = (n-1)*element_distance*cos(angle);
    alpha = alpha+exp(-1i*2*pi*(beta(n,:)/lamda));
end

figure(2);
polarplot(angle, abs(alpha), 'b');
title('Directional Gain of a 10-element ULA with Faulty Receivers');
```

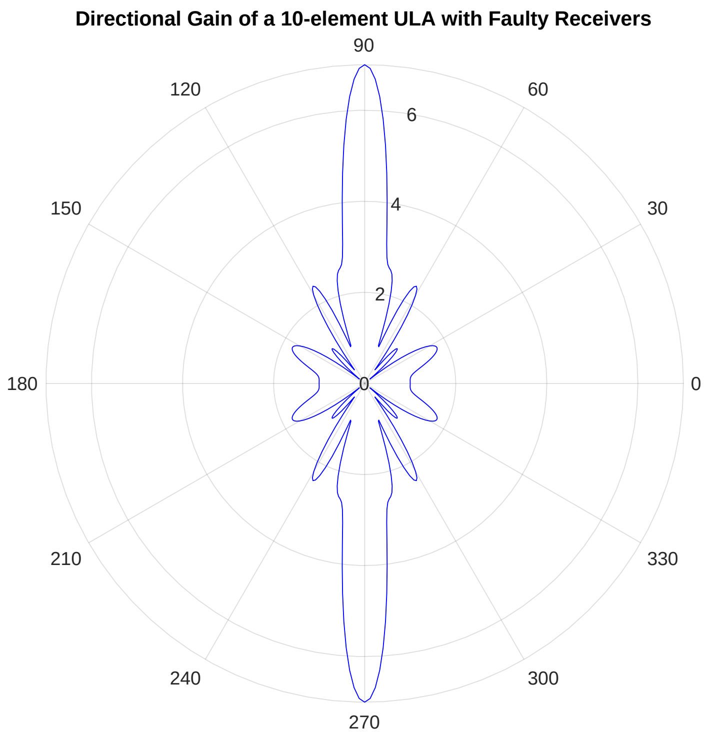


Figure 7: Directional Gain of a 10-element ULA with Faulty Receivers and with D = $0.5*\lambda$.

Differences Observed:

For the case where the inter element distance varies as $0.5^*\lambda$ and $2^*\lambda$, we can observe that as the distance between the antenna elements increases, grating lobe formation starts and this leads to a reduction in gain.

When the inter element distance is less than or equal to half the wavelength of the target signal, only the main lobe exists in the visible space, with no other grating lobes. There are no grating lobes when the inter element distance less than or equal to half the wavelength of the target signal. Grating lobes start to appear in the visible space when the inter element distance is greater than half the wavelength of the target signal.

The grating lobes that appear have an amplitude equal to the main beam. There is also a reduction in the main lobe beam-width. It also ultimately reduces the usefulness of the ULA.

As we increase it beyond the wavelength of the target signal, the number of side lobes also increase and the width of the main lobe decreases.

For the case where the inter element distance is $0.5^*\lambda$, but there are faulty receivers present at the 2nd, 3rd and the 4th position, we observe a change in the gain pattern, in particular there is a distorted directivity pattern. Because of the faulty elements, the symmetry of the array also gets distorted, and the side lobe levels also increase.

The phase difference is directly proportional to the inter element distance, and since that distance also gets altered, we observe a change in the phase difference, as compared to the case where all receivers were working perfectly.

Faulty elements also leads to errors in the measurement of DoA or the Direction of Arrival and also causes performance degradation.

4) Question 3:

```matlab
data = csvread('Data/Task3_DoA_Estimation/Data/7.csv');

figure(1);
plot(real(data));
xlabel('Number of Elements (Index)');

figure(2);
plot(imag(data));
xlabel('Number of Elements (Index)');

target_frequency = 2e3;
sound_speed = 340;
lamda = sound_speed/target_frequency;
element_distance = lamda/2;
elements = 100;
doa = zeros(1, length(-pi/2:pi/180:pi/2));
k = 0;

for angle = -pi/2:pi/180:pi/2
    alpha = [];
    for n = 1:elements
        alpha(n,:) =exp(-1i*2*pi*(n-1)*element_distance*sin(angle)/lamda);
    end
    k = k+1;
    doa(k,:) = data*alpha;
end

%[max_val max_ind]=max(doa)

phi = -90:1:90;

figure(3);
plot(phi, abs(doa), 'b');
xlabel('Possible direction of arrival in degrees');
ylabel('Absolute Value of Magnitude');
```

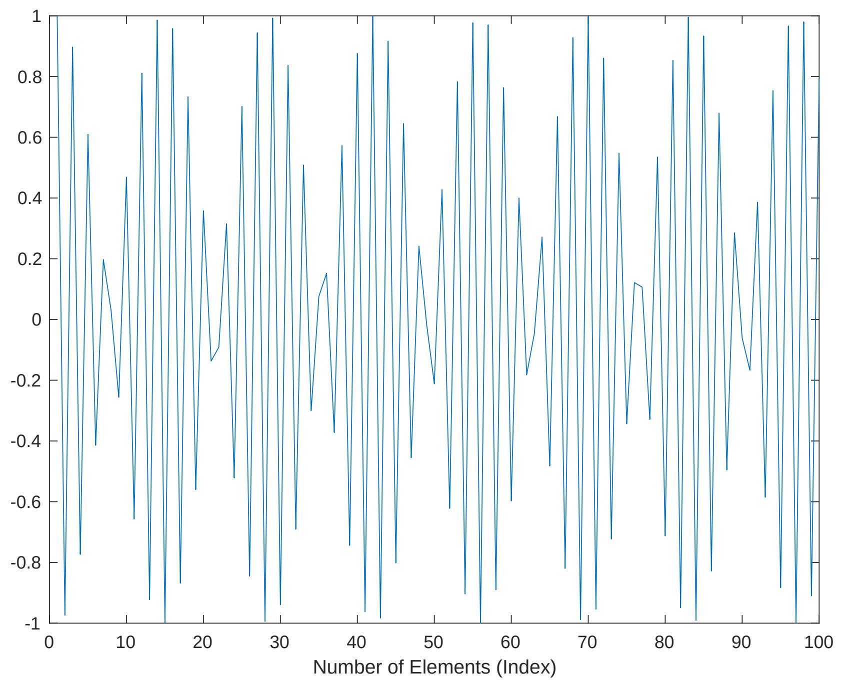


Figure 8: Real Part of the Array Snapshot.

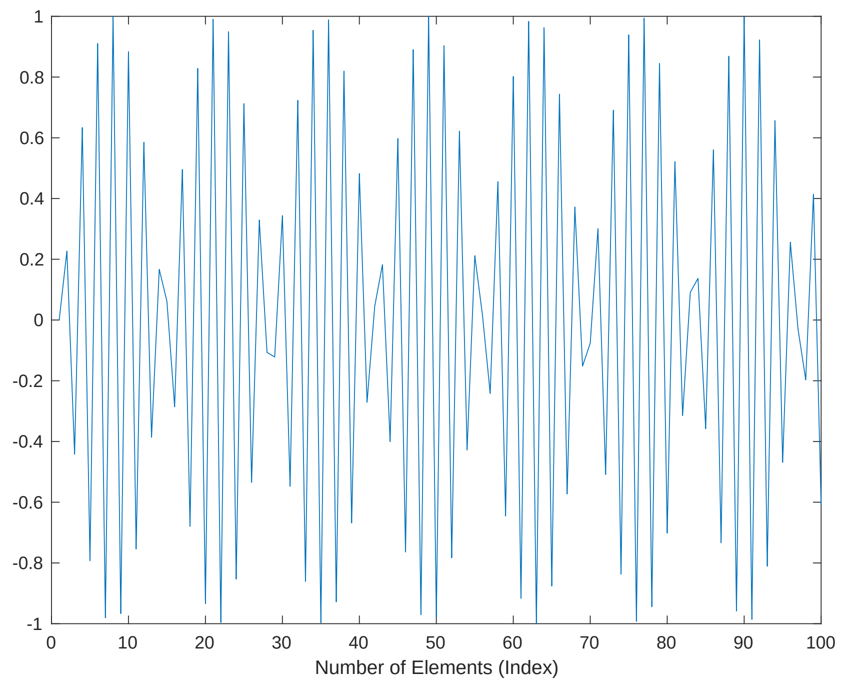


Figure 9: Imaginary Part of the Array Snapshot

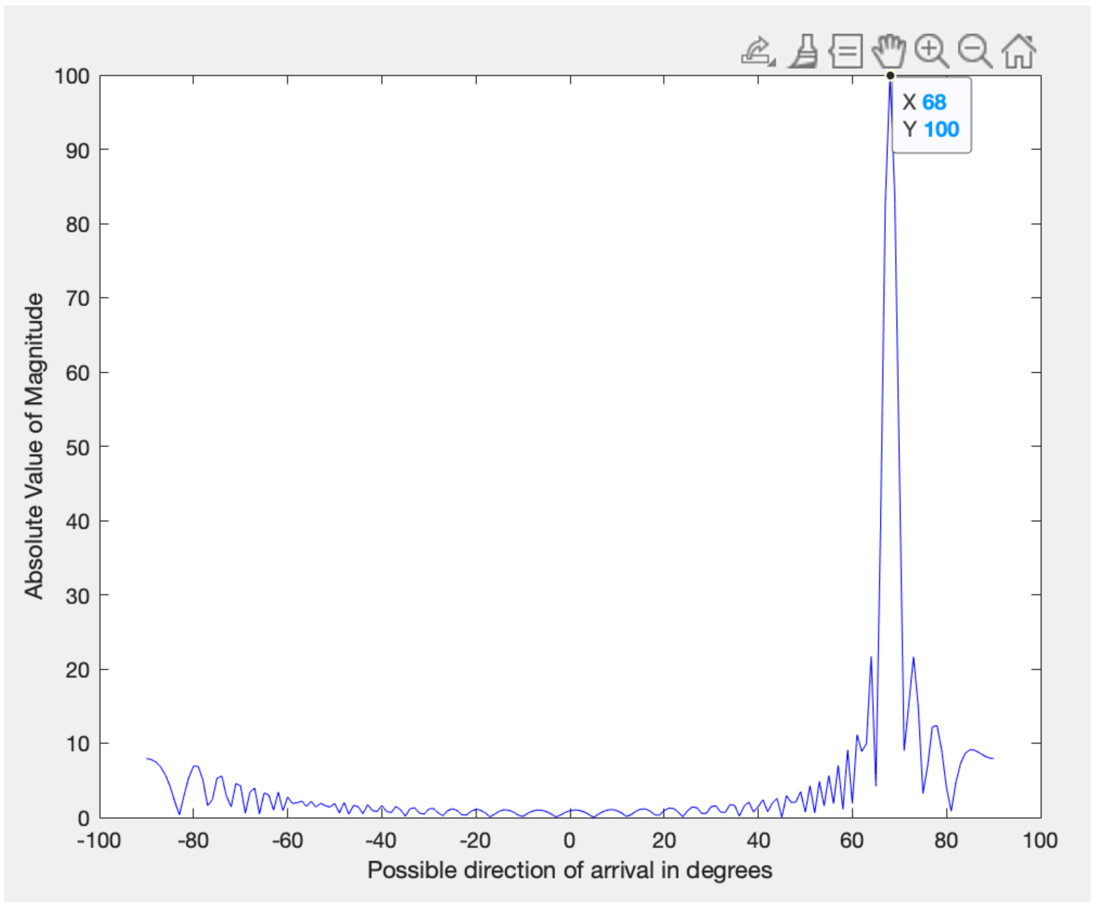


Figure 10: Possible Values of the DoA with a peak at the estimated value of 68 degrees.

```
% Calculations:
% The maximum value of the DoA vector occurs at index 159
% (the value of k).
% Relating it back to the corresponding angle, we get
% angle = -pi/2 + (159-1)*pi/180
% angle = 1.1868 radians
% angle = rad2deg(1.1868)
% angle = 67.9986 degrees.
```

From the graph we can approximate the estimated Direction of Arrival (DoA) to be **approximately 68 degrees**. While if we calculate it using the max value of the DoA vector and finding the corresponding angle, it comes out to be **1.1868 radians or 67.9986 degrees.**