Communications Project

Angle of Arrival
HW 7

Homework Results

Authors: Saqib Faraz 458317
Ankita Kannan Iyer 457620
Harsh Godhani 461986
Richard Grünert 289427
Kanaiyalal Thummar 435534

8.6.2023

Homework Tasks

HW7)
due June 8th 23:59 LT

Simulate the Angle-of-Arrival(AOA) coverage of for antenna array arrangements.

1) define an equilateral antenna array (triangle with equal distances/length) for 3 antennas with

a) 0.5 lambda wavelength (WL)

b) 1.0 lambda wavelength (WL)

c) 1.5 lambda wavelength (WL)

2) Create synthetic RANDOM complex data to calculate AOA of suitable size

plot/show example I/Q-diagram of the data and histograms of I/Q data.

3) derive the AOA for the synthetic data, for a,b,c configurations

plot the AOA in dcosx,dcosy

finally you should the the coverage area of the chosen

Proposal:

- use the script 25b_read_rawdata_AOA as template

- remove the read "real raw data" and other section you don't need

- generate a synthetic data (random complex numbers) - matrix with size [2000,100,3]

1 Introduction

Antenna arrays can be constructed from multiple similar antennas to increase the gain in comparison to a single of these antennas. [1] Under the assumption that every antenna has the same radiation pattern and that there is no coupling between the antennas, one can calculate an *Array Factor* (*AF*) which is simply multiplied with the base field pattern $E(\phi, \theta)$ to yield the pattern of the total array $Y(\phi, \theta)$.

$$Y(\phi, \theta) = E(\phi, \theta) \cdot AF$$

The array factor is commonly written in terms of the directional cosines u, v

$$AF(\theta, \phi) = \sum_{n=0}^{N-1} w_n \cdot e^{-jk\cdot(x_n u + y_n v)}$$

$$u = \sin\theta \cos\phi$$

$$v = \sin\theta \sin\phi$$

Angle-of-Arrival (AOA) coverage refers to the range of angles from which an antenna array can accurately discern the direction of incoming signals. By calculating the time delay between different antenna elements direction of the of an incoming signal can be determined. Relative time delay between the array elements can be calculated from the phase difference between received signals. Received signal can be IQ demodulated and stored as a complex number, whose phase can be used for AOA calculation. AOA is a critical parameter in many applications, including wireless communication systems, radar systems, and navigation systems. AOA can be measured using a variety of techniques, such as antenna arrays or interferometry. These methods employ several antennas or sensors to measure the phase and amplitude of the signal at various points, which can then be utilized to compute the angle of arrival. The AOA coverage of an antenna array arrangement is determined by a number of parameters, including the array's geometry and layout, the number and spacing of individual antenna elements, and the signal processing algorithms used.

2 Antenna Array Simulation

There are various antenna array layouts that can provide differing AOA coverage characteristics. Some examples of typical array configurations are: Uniform Linear Array (ULA), Uniform Circular Array (UCA), Planar array (Two-dimensional array with antenna elements arranged in a rectangle or square grid), and Equilateral array, where antenna elements are arranged in a grid of triangles and distance between adjacent elements is equal.

Three Equilateral Array configurations with different spacing, $d = 0.5\,\lambda, 1.0\,\lambda$, $1.5\,\lambda$, are simulated. The coordinates of the antenna positions are (0,0), (d, 0),

$(dcos(60°), dsin(60°))$. The antenna configurations are plotted below,

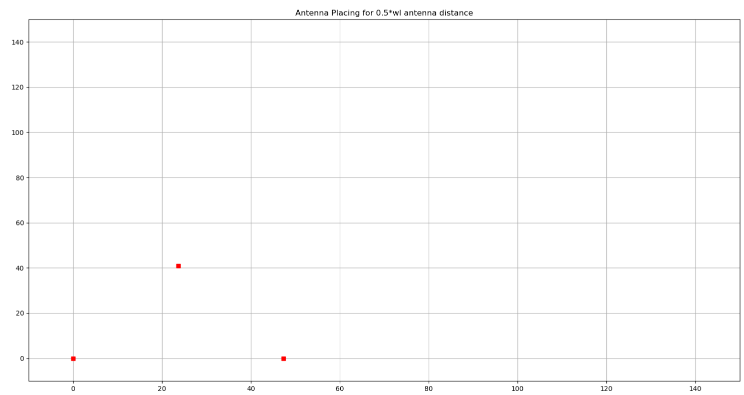

Figure 1: Equilateral grid array structure for $d = 0.5\,\lambda$

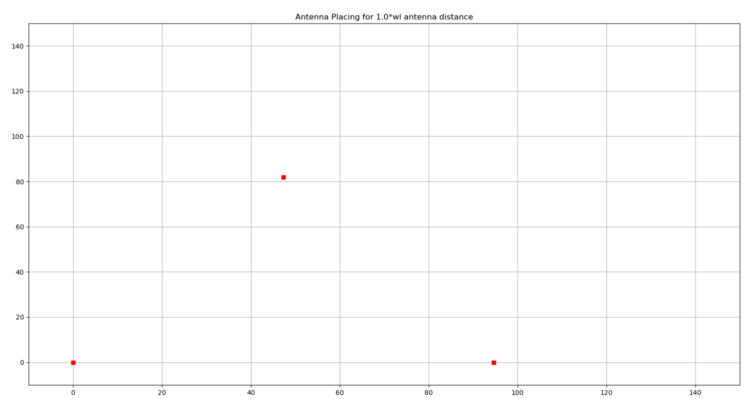

Figure 2: Equilateral grid array structure for $d = 1.0\,\lambda$

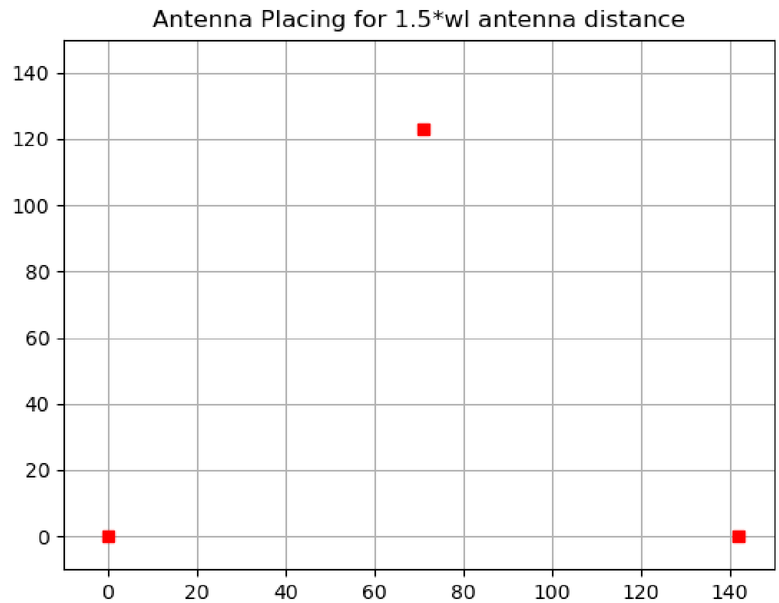

Figure 3: Equilateral grid array structure for $d = 1.5\,\lambda$

3 Random IQ data generation

Random complex data is generated with dimensions as (2000, 100, 3). Gaussian distribution is used for random number generation. Since, phases of the complex numbers determine the angle of arrival, generating large set of random complex numbers is equivalent to simulating signal sources or targets in all possible angles. The magnitude of the complex numbers is varied as it would not impact angle of arrival calculations. IQ data is coherently integrated over 8 pulses.

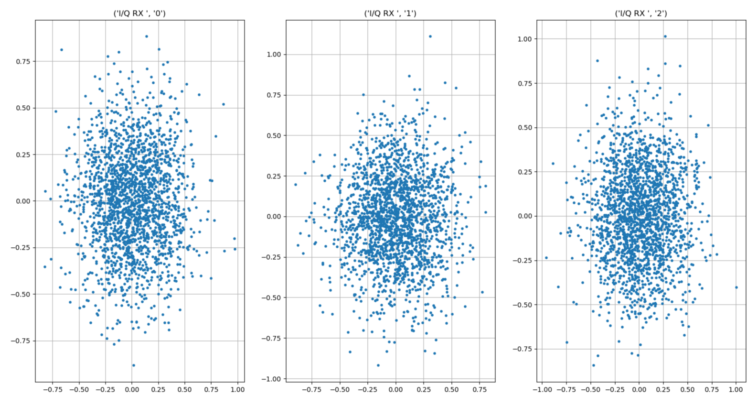

Figure 4: I/Q data plot for 3 receiving antennas

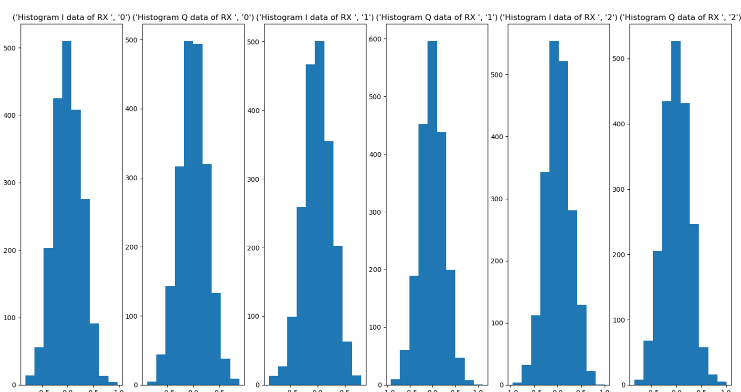

Figure 5: I/Q Histogram plots for 3 receiving antennas

All the histograms resemble Gaussian or Normal distribution probability density function (PDF).

4 Angle of Arrival

Angle of arrival for the three configurations was calculated with the same IQ data. The increase in distance between the elements decreases the main lobe beam width and AOA coverage, as effective aperture is increasing, and also cause grating lobes to form, creating ambiguous angle measurements. The plots of AOA for different antenna array configurations, in dcosx & dcosy axes.

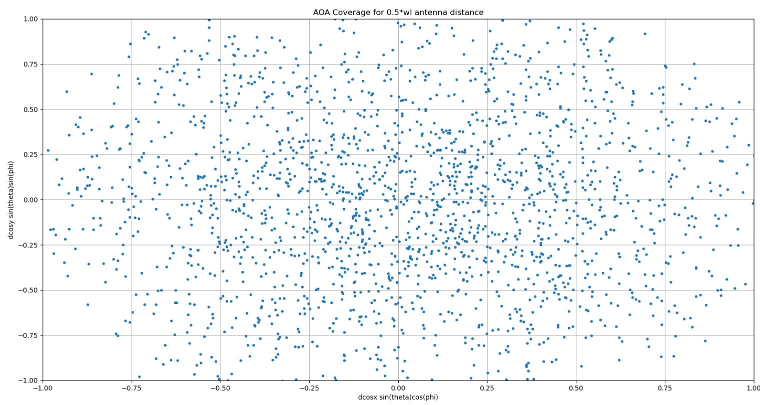

Figure 6: AOA coverage for Equilateral grid array structure with $d = 0.5\,\lambda$

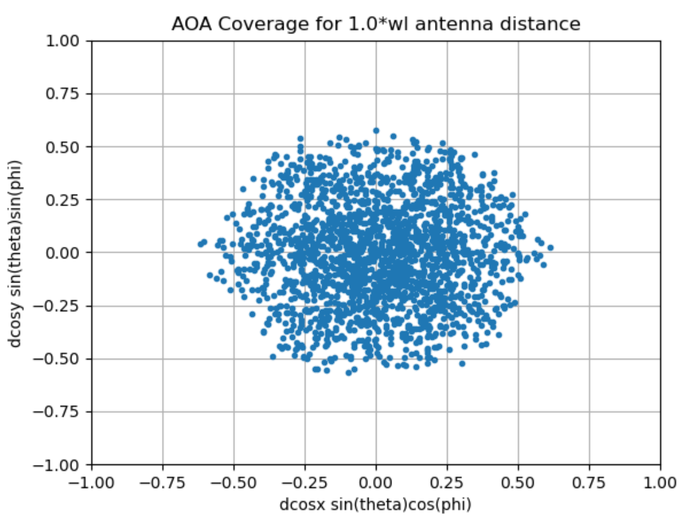

Figure 7: AOA coverage for Equilateral grid array structure with $d = 1.0\,\lambda$

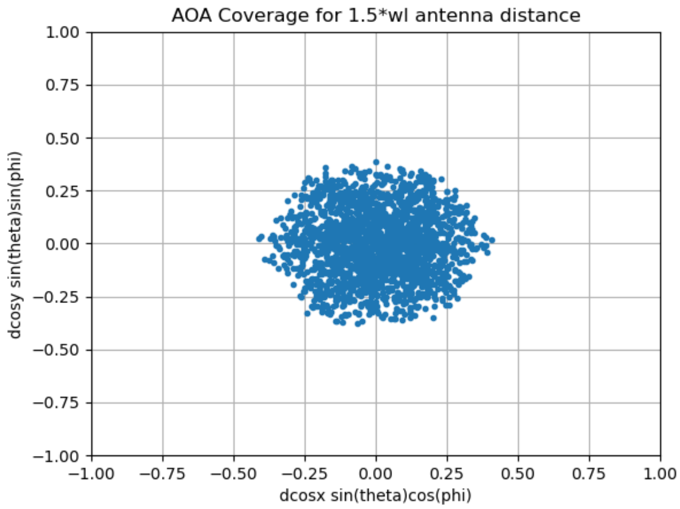

Figure 8: AOA coverage for Equilateral grid array structure with $d = 1.5\,\lambda$

From the above plots, it is observed that AOA coverage decreases with increase in antenna element distance beyond $d = 0.5\,\lambda$. AOA can be calculated from the plots by taking the maximum angle along both axes. As dcosx axis can be taken as Phi=0°cut and dcosy axis can be taken as Phi=90°cut, giving the AOA coverage in the two perpendicular cuts of radiation pattern.

Table 1: A Table

Array Configuration	Angle coverage in Phi=0°cut	Angle coverage in Phi=90°cut
$d = 0.5\,\lambda$	90°	90°
$d = 1.0\,\lambda$	37.7°	34.7°
$d = 1.5\,\lambda$	24.0°	22.3°

References

[1] Hysell, David,"Antennas and Radar for Environmental Scientists and Engineers", Cambridge University Press, url = https://doi.org/10.1017/9781108164122, 2018.

[2] Bevelacqua, Peter Joseph,"The Array Factor", url = https://www.antenna-theory.com/arrays/arrayfactor.php, 2023-05-28.

[3] Badawy, A., Khattab, T., Trinchero, D., ElFouly, T., and Mohamed, A., “A Simple Angle of Arrival Estimation Scheme,” https://doi.org/10.48550/arXiv.1409.5744, pp. 473-480, June 1996.

Software Used

[1] Charles R. Harris et al. “Array programming with NumPy”. In: Nature 585.7825 (Sept. 2020), pp. 357– 362. doi: 10.1038/s41586-020-2649-2. url: https://doi.org/10.1038/s41586-020-2649-2.

[2] J. D. Hunter. “Matplotlib: A 2D graphics environment”. In: Computing in Science & Engineering 9.3 (2007), pp. 90–95. doi: 10.1109/MCSE.2007.55.

[3] Python Core Team. Python: A dynamic, open source programming language. Python version 3.7. Python Software Foundation. 2019. url: https://www.python.org/.

[4] Pauli Virtanen et al. “SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python”. In: Nature Methods 17 (2020), pp. 261–272. doi: 10.1038/s41592-019-0686-2.

A Python Code

```
import scipy.io as spio
import os
import matplotlib.pyplot as plt
import numpy as np

####
f=3.17e6
c=3e8
wl=c/f

## wl=94.5718 # in m

# Question 1
d = 0.5 * wl    # for q1a
#d = 1.0 * wl    # for q1b
#d = 1.5 * wl    # for q1c

def antennae_position(d):
  # positions of antennae in Equilateral 3 antenna array (triangle with equal sides)
  antpos=np.empty([3]).astype(complex)
  antpos[0] = 0+0j
  antpos[1] = d+0j
  antpos[2] = d*np.cos(60*np.pi/180) + d*np.sin(60*np.pi/180)*1j
  return antpos

antpos = antennae_position(d)

# Question 2
# Generate complex random data with matrix size (2000,100,3)
# Gaussian distribution
data = np.random.normal(loc=0, scale=1/np.sqrt(2), size=(2000,100,3)) + \
           1j*np.random.normal(loc=0, scale=1/np.sqrt(2), size=(2000,100,3))

# number of range gates , data points, receivers
noRG=np.size(data,0)
noDP=np.size(data,1)
noRx=np.size(data,2)

# perform coherent integrations
#def make_ci(t, y, ci):
def make_ci(y, ci):
    nptsn=int(np.floor(len(y)/ci))
    yn=np.empty(nptsn)+1j*np.empty(nptsn)
```

```
    #tn=np.empty(nptsn)
    for i in range(0,nptsn):
        yn[i]=np.mean(y[i*ci:i*ci+ci-1])
        #tn[i]=np.mean(t[i*ci:(i+1)*ci])
    #return tn,yn
    return yn

ci=8;

noDPn=int(np.floor(noDP/ci))
# predefine matrix for integrated raw data
datan=np.zeros([noRG,noDPn,noRx])+1j*np.zeros([noRG,noDPn,noRx])

for rx in range(noRx):
    for rg in range(noRG):
        #tn,datan[rg,:,rx]=make_ci(t,data[rg,:,rx],ci)
        datan[rg,:,rx]=make_ci(data[rg,:,rx],ci)

data=datan[:]
#t=tn[:]
noDP=np.size(data,1)

data.shape

plt.figure()
for rx in range(3):
    plt.subplot(1,3,rx+1)
    plt.plot(np.real(data[:,0,rx]),np.imag(data[:,0,rx]),'.')
    #plt.axis([-10,10,-10,10])
    plt.grid(1)
    plt.title(('I/Q RX ', str(rx)))
#

plt.figure()
for rx in range(3):
    plt.subplot(1,6,2*rx+1)
    plt.hist(np.real(data[:,0,rx]))
    plt.title(('Histogram I data of RX ', str(rx)))

    plt.subplot(1,6,2*rx+2)
    plt.hist(np.imag(data[:,0,rx]))
    plt.title(('Histogram Q data of RX ', str(rx)))
plt.show()
#

def AOA(antpos, data, wl):
```

```
noRG=np.size(data,0)
noDP=np.size(data,1)
noRx=np.size(data,2)

pairs=[[0,1],[0,2],[1,2]]

# pairs=np.zeros((2,3))
# pairs=np.ndarray([0,1],[0,2],[1,2])

#pairs
nopairs=np.size(pairs,0)

dx=np.zeros([nopairs])
dy=np.zeros([nopairs])
#phases=np.zeros([noRG,noDP,nopairs])
phases=np.zeros([noRG,nopairs])
# phases=np.angle(data[30,:,0]*np.conjugate(data[30,:,1]))

# pp=0
for pp in range(nopairs):
    dx[pp]=(antpos[pairs[pp][0]]-antpos[pairs[pp][1]]).real
    dy[pp]=(antpos[pairs[pp][0]]-antpos[pairs[pp][1]]).imag
    # phases[:,pp]=np.angle(datamean[:,pairs[pp][0]]*np.conjugate(datamean[:,pairs[pp][1]
    phases[:,pp]=np.angle(np.nanmean(data[:,:,pairs[pp][0]]*
                            np.conjugate(data[:,:,pairs[pp][1]]),1))

R=2*np.pi/wl * np.array([dx, dy])
#R=np.transpose(R)
R=R.T


# least squares matrix solving
# numpy linalg
# https://numpy.org/doc/stable/reference/generated/numpy.linalg.solve.html

#B=np.matmul(np.transpose(np.mat(R)),np.mat(R))
B=np.matmul(np.mat(R).T,np.mat(R))


pos_data=np.zeros([noRG,2])
phi=np.zeros([noRG,1])
theta=np.zeros([noRG,1])

rg=0
for rg in range(noRG):
```

```
  #b=np.matmul(np.transpose(np.mat(R)),np.transpose(np.mat(phases[0,:])))
      b=np.matmul(np.mat(R).T,np.mat(phases[rg,:]).T)

      r=np.linalg.solve(B.T.dot(B), B.T.dot(b))
      #r=np.linalg.lstsq(B,b,rcond=None)[0]
      pos_data[rg,:]=r.T
      phi[rg]=np.arctan2(r[1],r[0])/np.pi*180
      theta[rg]=np.arcsin(np.sqrt(r[0]**2+r[1]**2))/np.pi*180

  return pos_data

# Question3
#3a
d = 0.5 * wl    # for q3a
antpos = antennae_position(d)
pos_data = AOA(antpos, data, wl)
plt.figure()
plt.plot(np.real(antpos), np.imag(antpos),'rs')
plt.xlim([-10,150])
plt.ylim([-10,150])
plt.grid(1)
plt.title('Antenna Placing for 0.5*wl antenna distance')

plt.figure()
plt.plot(pos_data[15:,0],pos_data[15:,1],'.')
plt.xlim([-1,1])
plt.ylim([-1,1])
plt.grid(1)
plt.xlabel('dcosx sin(theta)cos(phi)')
plt.ylabel('dcosy sin(theta)sin(phi)')
plt.title('AOA Coverage for 0.5*wl antenna distance')


#3b
d = 1.0 * wl    # for q3a
antpos = antennae_position(d)
pos_data = AOA(antpos, data, wl)
plt.figure()
plt.plot(np.real(antpos), np.imag(antpos),'rs')
plt.xlim([-10,150])
plt.ylim([-10,150])
plt.grid(1)
plt.title('Antenna Placing for 1.0*wl antenna distance')

plt.figure()
plt.plot(pos_data[15:,0],pos_data[15:,1],'.')
```

```python
plt.xlim([-1,1])
plt.ylim([-1,1])
plt.grid(1)
plt.xlabel('dcosx sin(theta)cos(phi)')
plt.ylabel('dcosy sin(theta)sin(phi)')
plt.title('AOA Coverage for 1.0*wl antenna distance')

#3c
d = 1.5 * wl    # for q3a
antpos = antennae_position(d)
pos_data = AOA(antpos, data, wl)
plt.figure()
plt.plot(np.real(antpos), np.imag(antpos),'rs')
plt.xlim([-10,150])
plt.ylim([-10,150])
plt.grid(1)
plt.title('Antenna Placing for 1.5*wl antenna distance')

plt.figure()
plt.plot(pos_data[15:,0],pos_data[15:,1],'.')
plt.xlim([-1,1])
plt.ylim([-1,1])
plt.grid(1)
plt.xlabel('dcosx sin(theta)cos(phi)')
plt.ylabel('dcosy sin(theta)sin(phi)')
plt.title('AOA Coverage for 1.5*wl antenna distance')
```