Homework 6 submission

ECET 512 — Wireless Systems

Chris Uzokwe
ID: cnu25

February 23, 2021

1 Submitted files

For this assignment, besides this report, the following archives were created:

1.1 SRC Folder

+ *"main.py"*: This **script** simulates the response of a linear smart antenna array. Power and phase response of a signal uplink are plotted as a user moves through a single cell.

1.2 DOC Folder

+ *"user.gif"* Animation of a single user moving across one base cell.

+ *"bf-l2spacing"* Beamformer output power plotted against the user direction of arrival. Inter-element spacing is lambda/2.

+ *"bf-variedspacing"* Beamformer output power plotted against the user direction of arrival. Inter-element spacing varies between lambda and lambda/10 in steps of lambda/10..

2 Simulated Array Antenna Uplink

Below are simulations of the Bartlett Beamformer output. Each line represents a different output in the antenna array. You can see how the power is greatest when all of the antennas in the array have line of sight to the user. this power tapers off near the 0 and 180 degree axes. It's similar in the beamformer with the differentiated inter-element spacing, except the antennas interfere less at the 0 and 180 degree axes, and the power does not taper off as much. This might be solved with a circular array unit rather than a linear.

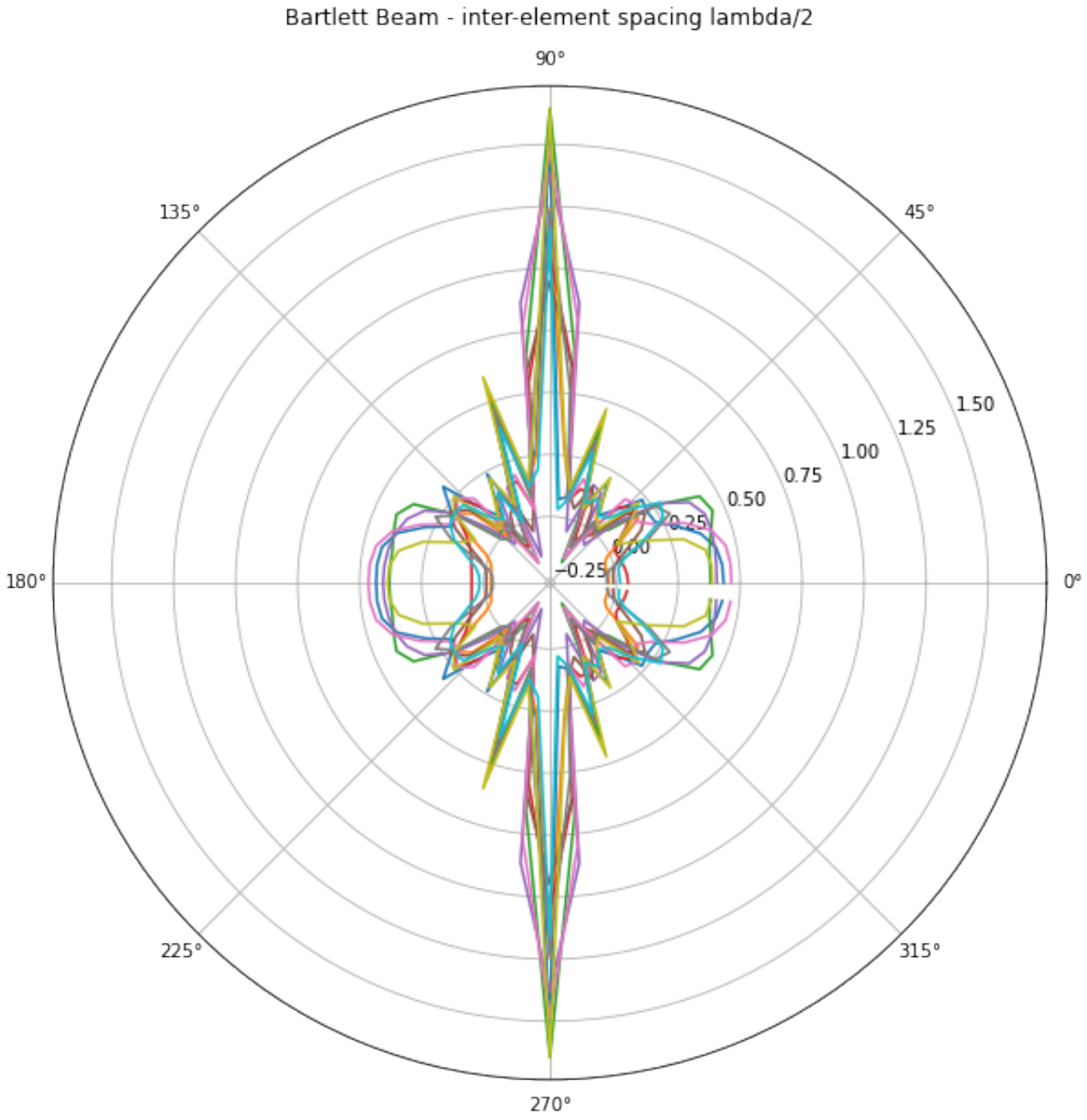

Figure 1: Conventional Bartlett Beamformer Output with inter-element array spacing of lambda/2.

3 Python code

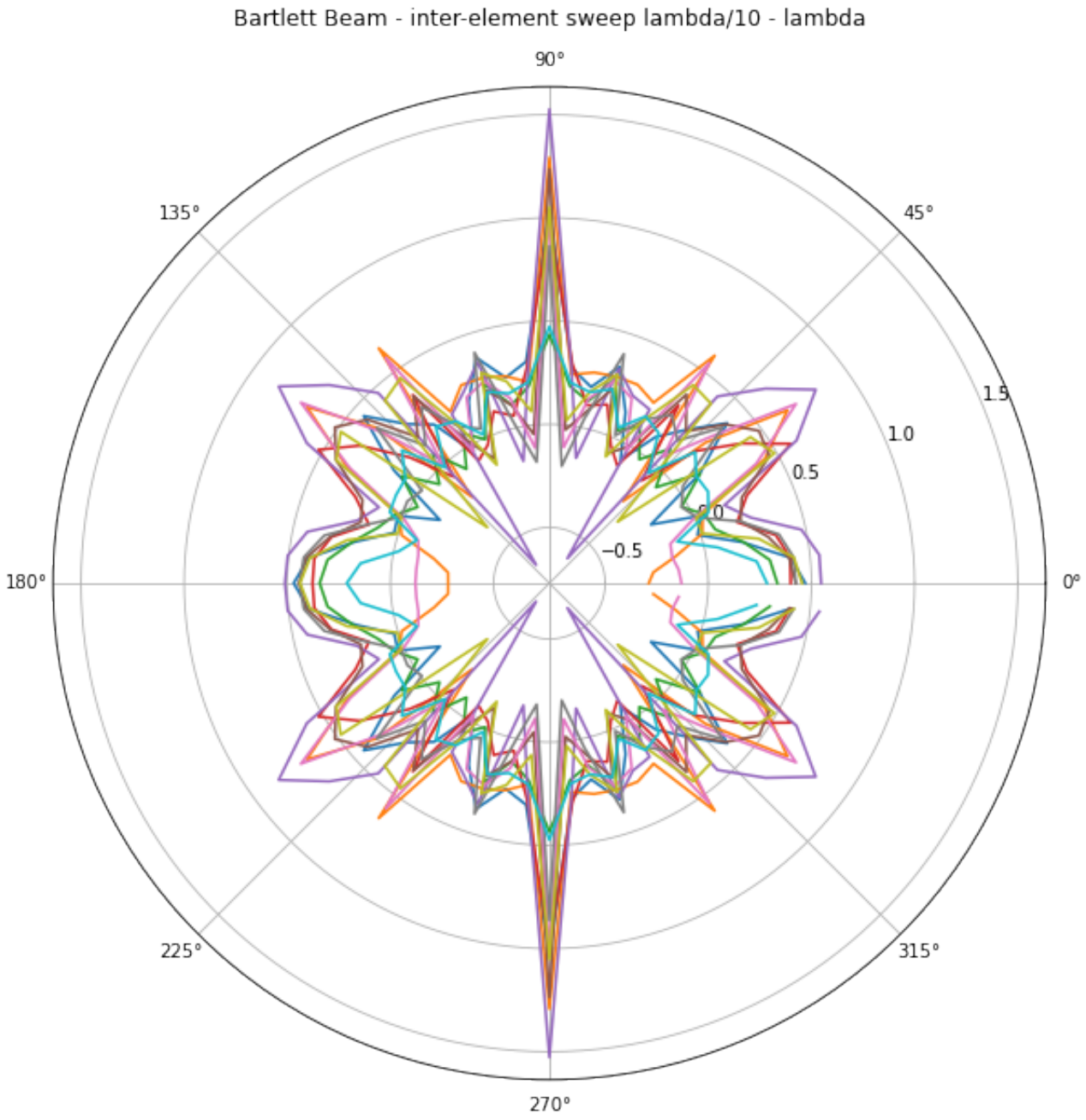

Figure 2: Conventional Bartlett Beamformer Output with inter-element array spacing varied between lambda and lambda/10.

```
# steering vector
def st_vec_lin(theta, M, d, k):
  alu = np.array([], dtype=np.float64)
  inc = d
  for i in range(M)[::-1]:
    r = exp(-1j*k*d*(M-i+1)*cos(theta))
    d= d+inc
    alu = np.append(alu, r)

  return alu

alu_vecs = []
l = 1/1.8E9
M = 10
d = l/10
k = 2*np.pi/l

def bs_power(d):
  return 0 - 10*2.9*np.log10(d)


for i in range(numFrames):
  alu = st_vec_lin(mobileAngle[i], M, d, k)
  alu_vecs.append(alu)


x = []
# array output
for  i in range(numFrames):
  out = alu_vecs[i] + np.random.normal(0, 1, 1)
  out = out.reshape(len(out),1)
  x.append(out)

x = np.asarray(x)

# spatial covariance matrix

def R_est(N, x, Ts):
  R = []
  #xH = x.conjugate().T
  sum = 0
  for i in range(N):
    #print(x[i*Ts].conjugate().shape)
    sum += np.dot(x[i*Ts],x[i*Ts].conjugate().T)
   # print(sum)
```

Figure 3: Snippet of main.py.