EE2703:
APPLIED PROGRAMMING LAB

FINAL EXAM

Half-Wave Dipole Antenna

Arjun R
EE20B016

May 13, 2022

Contents

1 Aim

In this assignment, we are tasked to do the following:

- Compute currents through each section of the dipole antenna
- Compare the computed current with the assumed sinusoidal current
- Plot both the currents against z and analyse the difference

2 Introduction

- A long wire carries a current I(z) in a dipole antenna with half length of 0.5m - so the antenna is a metre long, and has a wavelength of 2 metres. We have to determine the currents in the two wires of the antenna. The standard analysis assumes that the antenna current is given by

$$I = \begin{cases} Imsin(k(l-z)) & 0 \leq z \leq l \\ Imsin(k(l+z)) & -l \leq z < 0 \end{cases}$$

- Our task is to determine if this is a good assumption.

3 Importing necessary libraries and initializing variables

We declare the following variables to efficiently run the program.

Algorithm 1 Pseudo code to initialise values

```
    import required libraries/functions from pylab
  initialize N, l, Im, a, dz, mu0, k
```

The python code to execute the above tasks is as following:

```python
from pylab import *
N = 100 # number of sections in each half-section
l = 0.5 # quarter wavelength
Im = 1.0 # current injected at feed point
a = 0.01 # radius
dz = l/N # spacing b/w current samples
mu0 = 4e-7 * pi # permeability of free space
k = pi/(2*l) # wavenumber
```

4 Section 1: Initialising array to store section points

- We shall divide each half-section of the antenna into N sections each of length dz = l/N. For this purpose, we construct two vectors `z` and `u` such that

$$z = i \times dz, -N \leq i \leq N$$

$$u = i \times dz, i \in [-(N-1), -1] \cup [1, N-1]$$

- `z` and `u` vectors are initialized as points along the antenna but `u` vector doesn't have the terms corresponding to known currents, i.e., at i=0, N, and 2N.
- Initialise two vectors I and J, to store currents at points pointed to by z and u, respectively.
- Thus, I and z both have length (2N+1), and J and u both have length (2N-2).

Algorithm 2 Initialising vectors I, J, z, u

```
    initialize z vector as array from -l to l with step dz
  copy z values into u
  remove values at indices 0, N, 2N from u
```

The python code to execute the above tasks is as following:

```python
z = arange(-N, N+1)*dz # ranges from -N*dz to N*dz
u = delete(z, [0, N, 2*N]) # removes locations where current is known
# Since I and J aren't updated, they can be initialized later
```

Let's see what the z and u matrices contain (for N=4):

```
z:
[-0.5 -0.38 -0.25 -0.12 0.  0.12 0.25 0.38 0.5]
u:
[-0.38 -0.25 -0.12 0.12 0.25 0.38]
```

5 Section 2: Creating M matrix

Ampere's law gives us:

$$2\pi a H_\phi(z_i) = I_i$$

In matrix form, this can be written as:

$$\begin{pmatrix} H_\phi(z_1) \\ \dots \\ H_\phi(z_{N-1}) \\ H_\phi(z_{N+1}) \\ \dots \\ H_\phi(z_{2N-1}) \end{pmatrix} = \frac{1}{2\pi a} \begin{pmatrix} 1 & \dots & 0 & 0 & \dots & 0 \\ \dots & \dots & \dots & \dots & \dots & \dots \\ 0 & \dots & 1 & 0 & \dots & 0 \\ 0 & \dots & 0 & 1 & \dots & 0 \\ \dots & \dots & \dots & \dots & \dots & \dots \\ 0 & \dots & 0 & 0 & \dots & 1 \end{pmatrix} \begin{pmatrix} J_1 \\ \dots \\ J_{N-1} \\ J_{N+1} \\ \dots \\ J_{2N-1} \end{pmatrix} = M * J$$

We need to obtain this M matrix.

```
Algorithm 3 function to return M
    M = (1/(2*pi*a))*identity matrix of size 2N-2
```

The python code to execute the above task is as following:

```
def return_M(N, a):
    """Returns M matrix where H = M*J
    Inputs:
    N = number of sections
    a = radius at observing point"""
    M = identity(2*N-2)/(2*pi*a) # identity(N) returns identity matrix of size N
    return M
```

Let's see what M matrix looks like for N=4:
M:

```
[[15.92 0.  0.  0.  0.  0. ]
 [ 0.  15.92 0.  0.  0.  0. ]
 [ 0.  0.  15.92 0.  0.  0. ]
 [ 0.  0.  0.  15.92 0.  0. ]
 [ 0.  0.  0.  0.  15.92 0. ]
 [ 0.  0.  0.  0.  0.  15.92]]
```

6 Section 3: Finding P and P_B matrices

- We shall express vector potential as

$$A_{z,i} = \sum_j I_j \left(\frac{\mu 0}{4\pi} \frac{exp(-jkR_{ij})dz'_j}{R_{ij}} \right)$$

- This sum is reduced to summation of P matrix with J vector. With this, we also include the portion from already known currents, which would be PB matrix times In.

$$A_{z,j} = \sum_j P_{ij} I_j + P_B I_N$$

- P is a matrix of size (2N - 2)X(2N - 2) rows. Note that the vector potential is driven by all the currents, which is why we use the I vector.
- P_B is the contribution to the vector potential due to current I_N, and is given by

$$P_B = \frac{\mu 0}{4\pi}\frac{exp(-jkR_{iN}) \times dz'_j}{R_{iN}}$$

- We now require R_{ij} and R_{iN} matrices to compute P and P_B.
- R is the distance between observer(alternatively, the observation point) and source. R is given by

$$R_{ij} = \sqrt{a^2 + (z_i - z_j)^2}$$

- Finding this matrix using z array gives Rz and and u array gives Ru.
- R_{iN} is obtained by slicing the middle column from Rz and deleting the indices 0, N, and 2N.

Algorithm 4 computing P and PB matrices

```
create rectangular grid matrices zj and zi from z, z
create rectangular grid matrices uj and ui from u, u
create matrix Rz = (a**2+(zi-zj)**2)**0.5
create matrix Ru = (a**2+(ui-uj)**2)**0.5
RiN = array obtained by slicing the middle column of Rz, and then deleting elements at
0, N, and 2N indices
P = mu0*(1/Ru)*exp(-(1j)*k*Ru)*dz/(4*pi)
PB = mu0*(1/RiN)*exp(-(1j)*k*RiN)*dz/(4*pi)
```

The python code to execute the above tasks is as following:

```python
zj, zi = meshgrid(z, z) # creates a rectangular grid out of x and y arrays
uj, ui = meshgrid(u, u)
Rz = array(sqrt(a**2 + (zi-zj)**2)) # distance matrix
Ru = array(sqrt(a**2 + (ui-uj)**2))
# matrix containing distances to unknown currents
RiN = delete(Rz[:, N], [0, N, 2*N])
# size=2N-1; matrix with distance to center element Im

# Finding P matrices
P = mu0*(1/Ru)*exp(-(1j)*k*Ru)*dz/(4*pi)
PB = mu0*(1/RiN)*exp(-(1j)*k*RiN)*dz/(4*pi)
```

R matrices:

R_z:

```
[[0.01 0.13 0.25 0.38 0.5 0.63 0.75 0.88 1.  ]
 [0.13 0.01 0.13 0.25 0.38 0.5 0.63 0.75 0.88]
 [0.25 0.13 0.01 0.13 0.25 0.38 0.5 0.63 0.75]
 [0.38 0.25 0.13 0.01 0.13 0.25 0.38 0.5 0.63]
 [0.5 0.38 0.25 0.13 0.01 0.13 0.25 0.38 0.5 ]
 [0.63 0.5 0.38 0.25 0.13 0.01 0.13 0.25 0.38]
 [0.75 0.63 0.5 0.38 0.25 0.13 0.01 0.13 0.25]
 [0.88 0.75 0.63 0.5 0.38 0.25 0.13 0.01 0.13]
 [1.  0.88 0.75 0.63 0.5 0.38 0.25 0.13 0.01]]
```

R_u:

```
[[0.01 0.13 0.25 0.5 0.63 0.75]
 [0.13 0.01 0.13 0.38 0.5 0.63]
 [0.25 0.13 0.01 0.25 0.38 0.5 ]
 [0.5 0.38 0.25 0.01 0.13 0.25]
 [0.63 0.5 0.38 0.13 0.01 0.13]
 [0.75 0.63 0.5 0.25 0.13 0.01]]
```

R_{iN}:

```
[0.38 0.25 0.13 0.13 0.25 0.38]
```

Let's look at P and P_B matrices (multiplied by 10^8) for N=4:

$P \times 10^8$:

$$\begin{pmatrix}
124.94-3.93j & 9.2-3.83j & 3.53-3.53j & -0.-2.5j & -0.77-1.85j & -1.18-1.18j \\
9.2-3.83j & 124.94-3.93j & 9.2-3.83j & 1.27-3.08j & -0.-2.5j & -0.77-1.85j \\
3.53-3.53j & 9.2-3.83j & 124.94-3.93j & 3.53-3.53j & 1.27-3.08j & -0.-2.5j \\
-0.-2.5j & 1.27-3.08j & 3.53-3.53j & 124.94-3.93j & 9.2-3.83j & 3.53-3.53j \\
-0.77-1.85j & -0.-2.5j & 1.27-3.08j & 9.2-3.83j & 124.94-3.93j & 9.2-3.83j \\
-1.18-1.18j & -0.77-1.85j & -0.-2.5j & 3.53-3.53j & 9.2-3.83j & 124.94-3.93j
\end{pmatrix}$$

$P_B \times 10^8$:

```
[1.27-3.08j 3.53-3.53j 9.2 -3.83j 9.2 -3.83j 3.53-3.53j 1.27-3.08j]
```

7 Section 4: Computing Q and Q_B matrices from Relation connecting H and A

- We shall compute Q and Q_B matrices from magnetic field intensity expression along ϕ direction.

$$H_\phi(r,z) = -\frac{1}{\mu}\frac{\partial A_z}{\partial r}$$

$$H_\phi(r,Z_i) = \sum P_{ij}\frac{a}{\mu 0}(\frac{jk}{R_{ij}} + \frac{1}{R_{ij}^2})I_j + P_B\frac{a}{\mu 0}(\frac{jk}{R_{iN}} + \frac{1}{R_{iN}^2})I_m$$

$$H_\phi(r,z_i) = \sum Q_{ij}I_j + Q_{Bi}I_m$$

- Now comparing the both expressions, we shall get the expression for Q and Q_B

$$Q = P_{ij}\frac{a}{\mu 0}(\frac{jk}{R_{ij}} + \frac{1}{R_{ij}^2})$$

$$Q_B = P_B\frac{a}{\mu 0}(\frac{jk}{R_{iN}} + \frac{1}{R_{iN}^2})$$

- The Q'_{ij} in the equation is over all the currents. However this needs to be split into the unknown currents, J_j and the boundary currents. Only one of the boundary currents is non-zero, namely the feed current at i = N. The matrix corresponding to J_j we call Q_{ij}, and the boundary current we call $Q_B = Q_{iN}$.

Algorithm 5 Computing Q and QB matrices

```
Q = P*a*(1/(Ru**2), j*k*(1/Ru))/mu0
QB = PB*a*(1/(RiN**2), j*k*(1/RiN))/mu0
```

The python code to execute the above tasks is as following:

```python
def vec_complex(a, b):
    return complex(a, b)

vec_complex = vectorize(vec_complex)
Q = P*a*vec_complex(1/(Ru**2), k*(1/Ru))/mu0
QB = PB*a*vec_complex(1/(RiN**2), k*(1/RiN))/mu0
```

Let's see the values stored inside Q and Q_B matrices:

$Q \times 10^4$:

$$\begin{pmatrix}
9.95e{+}05{-}10.28j & 5.42e{+}02{-}10.12j & 8.02e{+}01{-}9.66j & 1.24e{+}01{-}7.96j & 5.83e{+}00{-}6.82j & 2.26e{+}00{-}5.59j \\
5.420e{+}02{-}10.12j & 9.95e{+}05{-}10.28j & 5.42e{+}02{-}10.12j & 2.77e{+}01{-}8.92j & 1.24e{+}01{-}7.96j & 5.83e{+}00{-}6.82j \\
8.02e{+}01{-}9.66j & 5.42e{+}02{-}10.12j & 9.95e{+}05{-}10.28j & 8.02e{+}01{-}9.66j & 2.77e{+}01{-}8.92j & 1.24e{+}01{-}7.96j \\
1.24e{+}01{-}7.96j & 2.77e{+}01{-}8.92j & 8.02e{+}01{-}9.66j & 9.95e{+}05{-}10.28j & 5.42e{+}02{-}10.12j & 8.02e{+}01{-}9.66j \\
5.83e{+}00{-}6.82j & 1.24e{+}01{-}7.96j & 2.77e{+}01{-}8.92j & 5.42e{+}02{-}10.12j & 9.95e{+}05{-}10.28j & 5.42e{+}02{-}10.12j \\
2.26e{+}00{-}5.59j & 5.83e{+}00{-}6.82j & 1.24e{+}01{-}7.96j & 8.02e{+}01{-}9.66j & 5.42e{+}02{-}10.12j & 9.95e{+}05{-}10.28j
\end{pmatrix}$$

$Q_B \times 10^4$:

```
[ 27.72 -8.92j, 80.2 -9.66j, 542.08-10.12j, 542.08-10.12j, 80.2 -9.66j, 27.72 -8.92j]
```

8 Section 5: Obtaining J vector from M and Q matrices

- The matrices M, Q, Q_B and J are connected by the equation

$$MJ = QJ + Q_B I_m$$

- Thus, J vector can be obtained by matrix multiplication of inv(M-Q) and Q_B I_m

$$J = (M - Q)^{-1} * Q_B \times I_m$$

- We should add the known currents at i = 0, N, and 2N to J vector to get I vector.
- I vector can be compared to the assumed sinusoidal expression for current distribution in a half-wave dipole given at the top of the assignment.

Algorithm 6 Computing J from Q, QB, M

```
    Initialize M vector
J = inv(M-Q)*QB X Im
copy contents of J into vector I insert center(Im) and boundary(0) values
Compute I vector
```

The python code to execute the above tasks is as following:

```python
M = return_M(N, a)
J = Im * dot(inv(M-Q), QB) # Calculates J array
I = J.copy()
I = insert(I, 0, 0)
I = insert(I, N, Im)
I = insert(I, 2*N, 0)
# calculates final I array after inserting center and boundary values
```

Let's see what are the values of I, J and I_assumed vectors:

abs($I \times 10^5$):

```
[0.00e+00 3.47e+00 9.62e+00 6.48e+01 1.00e+05 6.48e+01 9.62e+00 3.47e+00 0.00e+00]
```

abs($J \times 10^5$):

```
[ 3.47 9.62 64.84 64.84 9.62 3.47]
```

I_assumed:

```
[0.  0.38 0.71 0.92 1.  0.92 0.71 0.38 0.  ]
```

We notice that the value of I vector is very small comparing to I_assumed vector other than at the middle index, i.e., value = 1.
This large difference is reduced when we increase the value of N.

9 Analysing difference between current plots

- We shall now plot these two current vectors vs z vector for N=4 and N=100.

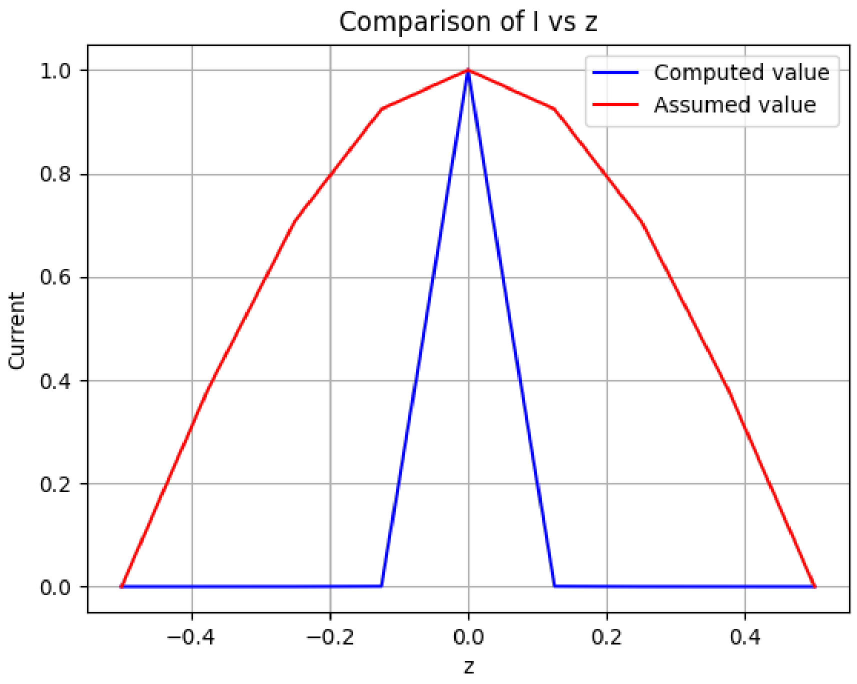

Figure 1: Current vs Distance along the dipole for N=4

- We notice a large deviation in N=4 plot. But, we see that the curve moves closer to the assumed curve as we increase N to 100.
- The reason for this discrepancy is because we aren't considering the section to be infinitesimally small. For N=4, we get large errors as expected, because the approximation of short dipoles is no longer valid and simple summation won't give accurate results.

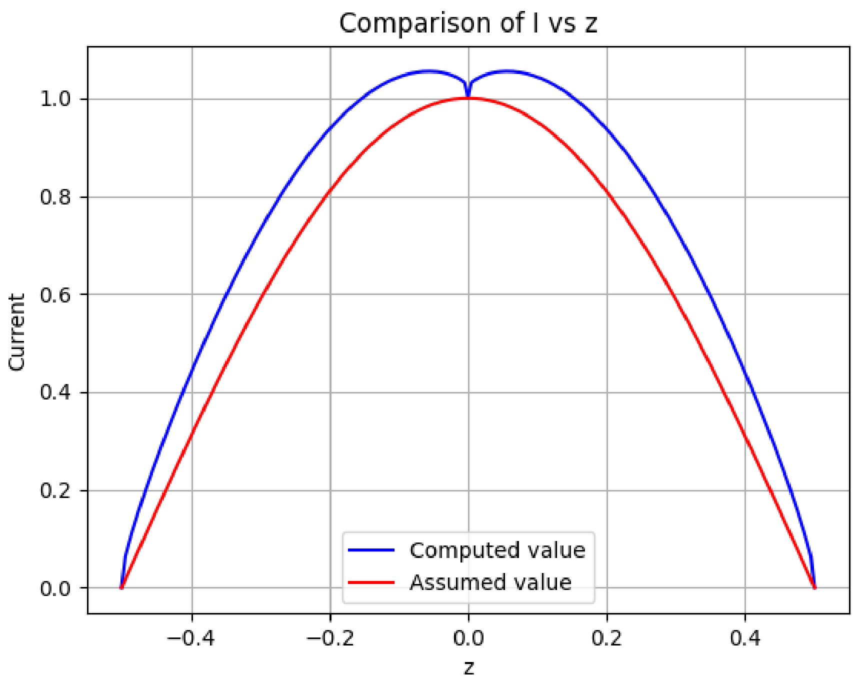

Figure 2: Current vs Distance along the dipole for N=100

- At larger N (N=100), the curve moves closer to the sinusoidal curve but we still see a deviation. This, in fact, points to the fact that assumed sinusoidal current distribution is just **a very good approximation** and is not exact.
- The sudden matching value at distance 0 is because we explicitly give its value to be Im. Otherwise we would have seen a smooth curve.

10 Conclusion

Thus, we have analysed and observed how the current of half-wave dipole antennas varies across the length of the dipole, and how the value differs from the standard analysis current expression (the sinusoidal one). We have also inferred from the matrices we used how vectorization is quick and easier than using for loops.