EE2703: Applied Programming Lab
End Semester Examination - 2022

Niyas Mon P — EE20B094

May 2022

1 Objectives

- Finding antenna currents in a half-wave dipole antenna.
- Solving the magnetic vector equations in the form of matrices.
- plotting the calculated currents and standard assumption.
- Determining how good is the standard assumption which is given by:

$$I = \begin{cases} I_m sin(k(l-z)) & 0 \le z \le l \\ I_m sin(k(l+z)) & -l \le z \le 0 \end{cases}$$

2 Calculation of Current and Position vectors

The position vector **z** includes all points while **u** vector includes only positions of unknown points. The current vector is **I** and **J** represents unknown currents.

$$z[i+N] = i \cdot dz, -N \le i \le N$$

u is obtained by removing endpoints and middle point from z

$$u = concat(z[1:N], z[N+1:-1])$$

The code is shown below

```
# creating z matrix ie, position of all currents
z = array([i*dz for i in range(-N,N+1,1)])
# creating current vector corresponding to the position of z
I = zeros((2*N+1,),dtype = float)
I[N] = Im

# u is unknown current points
u = concatenate((z[1:N],z[N+1:-1]))
# J is unknown current vector
J = zeros((2*N-2,1),dtype = float)
```

For N =4

$$z = \begin{pmatrix} -0.5 \\ -0.375 \\ -0.25 \\ -0.125 \\ 0 \\ 0.125 \\ 0.25 \\ 0.375 \\ 0.5 \end{pmatrix}, u = \begin{pmatrix} -0.375 \\ -0.25 \\ -0.125 \\ 0.125 \\ 0.25 \\ 0.375 \end{pmatrix} \tag{1}$$

I is initialised vector of length 2N+1 with middle values I_m and endpoint values as zeros. J is initialised as a zero vector of length 2N-2

3 Creation of M matrix

From Ampere's Law, we have

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

$$\begin{pmatrix} H_\phi[z_1] \\ \dots \\ H_\phi[z_{N-1}] \\ H_\phi[z_{N+1}] \\ \dots \\ H_\phi[z_{2N-1}] \end{pmatrix} = \frac{1}{2\pi a} \begin{pmatrix} 1 & \dots & 0 & 0 & \dots & 0 \\ \dots & \dots & \dots & \dots & \dots & \dots \\ 0 & \dots & 1 & 0 & \dots & 0 \\ 0 & \dots & 0 & 1 & \dots & 0 \\ \dots & \dots & \dots & \dots & \dots & \dots \\ 0 & \dots & 0 & 0 & \dots & 1 \end{pmatrix} \begin{pmatrix} J_1 \\ \dots \\ J_{N-1} \\ J_{N+1} \\ \dots \\ J_{2N-1} \end{pmatrix}$$

$$H = M * J \tag{3}$$

Here we create a function which takes the value of a and dimension of Matrix and return M.
Here is the code for it:

```
# defining function for computing matrix M
def compute_M(radius,no_unknown_currents):
  M = zeros((no_unknown_currents,no_unknown_currents),dtype = float)
  fill_diagonal(M,1)
  M = M / (2*pi*radius)
  return M
```

4 Calculation of R_z, R_u, P and P_B

Calculation of vector potential $A(r,z)$,can be further simplified into a equation consisting of P which is a square matrix of order $2N-2$ and P_B is a column vector. P_B is the contribution to the vector potential due to current I_N.

$$\vec{A}(r,z) = \frac{\mu_0}{4\pi} \int \frac{I(z')\hat{z}e^{-jkR}dz'}{R}$$

where vector R is the distance between observation point and source current:

$$\vec{R_{ij}} = \vec{r}\hat{r} + (z_i - z_j)\hat{z} \tag{4}$$

$$\vec{R_{iN}} = \vec{r}\hat{r} + z_i\hat{z}$$

For N = 4 , Rz and Ru we obtain as follows:

```
Rz :
-----------
[[0.01       0.12539936 0.25019992 0.37513331 0.50009999 0.62507999
  0.75006666 0.87505714 1.00005   ]
 [0.12539936 0.01       0.12539936 0.25019992 0.37513331 0.50009999
  0.62507999 0.75006666 0.87505714]
 [0.25019992 0.12539936 0.01       0.12539936 0.25019992 0.37513331
  0.50009999 0.62507999 0.75006666]
 [0.37513331 0.25019992 0.12539936 0.01       0.12539936 0.25019992
  0.37513331 0.50009999 0.62507999]
 [0.50009999 0.37513331 0.25019992 0.12539936 0.01       0.12539936
  0.25019992 0.37513331 0.50009999]
 [0.62507999 0.50009999 0.37513331 0.25019992 0.12539936 0.01
  0.12539936 0.25019992 0.37513331]
 [0.75006666 0.62507999 0.50009999 0.37513331 0.25019992 0.12539936
  0.01       0.12539936 0.25019992]
 [0.87505714 0.75006666 0.62507999 0.50009999 0.37513331 0.25019992
  0.12539936 0.01       0.12539936]
 [1.00005    0.87505714 0.75006666 0.62507999 0.50009999 0.37513331
  0.25019992 0.12539936 0.01      ]]
Ru :
-----------
[[0.01       0.12539936 0.25019992 0.50009999 0.62507999 0.75006666]
 [0.12539936 0.01       0.12539936 0.37513331 0.50009999 0.62507999]
 [0.25019992 0.12539936 0.01       0.25019992 0.37513331 0.50009999]
 [0.50009999 0.37513331 0.25019992 0.01       0.12539936 0.25019992]
 [0.62507999 0.50009999 0.37513331 0.12539936 0.01       0.12539936]
 [0.75006666 0.62507999 0.50009999 0.25019992 0.12539936 0.01      ]]
```

The difference between Rz and Ru is that the former computes distances including distances to known currents, while Ru is a vector of distances to unknown currents. R_{iN} is the distance from the middle current I_N.

$$
\begin{aligned}
A_{z,i} &= \frac{\mu_0}{4\pi}\sum_j \frac{I_j \exp\left(-jkR_{ij}\right) dz'_j}{R_{ij}} \\
&= \sum_j I_j \left(\frac{\mu_0}{4\pi}\frac{\exp(-jkR_{ij})}{R_{ij}} dz'_j\right) \\
&= \sum_j P_{ij} I_j + P_B I_N
\end{aligned}
$$

P and PB are given by :

$$P_{ij} = \frac{\mu_0}{4\pi}\frac{exp(-jkR_{ij})}{R_{ij}}dz$$

$$PB = \frac{\mu_0}{4\pi}\frac{exp(-jkR_{iN})}{R_{iN}}dz$$

P and PB for N = 4 are obtained as

```
====================
The P matrix is :  (after multiplying by 10^8)
[[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]]
The PB matrix is :  (after multiplying by 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]
====================
```

Code for them is given below

```python
# defining function for computing Rz and Ru
# as specified in question
def compute_Rz_Ru(N,r):   # N : no of sections in each half length , r: radius
  Rz = zeros((2*N+1,2*N+1),dtype = float)
  Ru = zeros((2*N-2,2*N-2),dtype = float)
  for i in range(0,2*N+1):
    for j in range(0,2*N+1):
      Rz[i][j] = sqrt(r**2 + (z[i]-z[j])**2)
  for i in range(0,2*N-2):
    for j in range(0,2*N-2):
      Ru[i][j] = sqrt(r**2 + (u[i]-u[j])**2)
  return Rz,Ru

# computing Rz and Ru
Rz ,Ru = compute_Rz_Ru(N,a)
# computing the matrix P
# P is the matrix of vector potential contributed by unknown currents
P = exp(-1j*k*Ru)/Ru *1e-7 * dz
# computing the matrix PB
# PB is contribution to vector potential due to current I[N]
R_iN = array([sqrt(a**2 + u[i]**2) for i in range(0,2*N-2)])
PB = exp(-1j*k*R_iN)/R_iN *1e-7 * dz
```

5 Calculation of Q and Q_B

$H_\phi(r, z_i)$ can be calculated as :

$$\begin{aligned} H_\phi(r,z_i) &= -\sum_j \frac{dz'_j}{4\pi}\left(\frac{-jk}{R_{ij}} - \frac{1}{R_{ij}^2}\right)\exp\left(-jkR_{ij}\right)\frac{rI_j}{R_{ij}} \\ &= -\sum_j P_{ij}\frac{r}{\mu_0}\left(\frac{-jk}{R_{ij}} - \frac{1}{R_{ij}^2}\right)I_j + P_B\frac{r}{\mu_0}\left(\frac{-jk}{R_{iN}} - \frac{1}{R_{iN}^2}\right)I_m \\ &= \sum_j Q'_{ij}I_j \end{aligned}$$

Q and QB are given by

$$Q = -\frac{r}{\mu_0}\left(\frac{-jk}{R_{ij}} - \frac{1}{R_{ij}^2}\right) \tag{5}$$

$$QB = \frac{r}{\mu_0}\left(\frac{-jk}{R_{iN}} - \frac{1}{R_{iN}^2}\right) \tag{6}$$

The Q and QB for N=4 are obtained as:

```
====================
Q matrix:
[[9.952e+01-0.j 5.000e-02-0.j 1.000e-02-0.j 0.000e+00-0.j 0.000e+00-0.j
  0.000e+00-0.j]
 [5.000e-02-0.j 9.952e+01-0.j 5.000e-02-0.j 0.000e+00-0.j 0.000e+00-0.j
  0.000e+00-0.j]
 [1.000e-02-0.j 5.000e-02-0.j 9.952e+01-0.j 1.000e-02-0.j 0.000e+00-0.j
  0.000e+00-0.j]
 [0.000e+00-0.j 0.000e+00-0.j 1.000e-02-0.j 9.952e+01-0.j 5.000e-02-0.j
  1.000e-02-0.j]
 [0.000e+00-0.j 0.000e+00-0.j 0.000e+00-0.j 5.000e-02-0.j 9.952e+01-0.j
  5.000e-02-0.j]
 [0.000e+00-0.j 0.000e+00-0.j 0.000e+00-0.j 1.000e-02-0.j 5.000e-02-0.j
  9.952e+01-0.j]]
QB matrix:
[-0.  +0.j -0.01+0.j -0.05+0.j -0.05+0.j -0.01+0.j -0.  +0.j]
====================
```

Code for computing Q and QB is below

```
# creation of matrix Q and QB as in question 4
Q = -P * (a/mu0) * ((-1j*k/Ru)-1/(Ru*Ru))
QB = PB * (a/mu0) * ((-1j*k/R_iN)-1/(R_iN*R_iN))
```

6 Solving the equation

The final equation is given by :

$$MJ = QJ + Q_B I_M$$

ie,

$$(M - Q)J = Q_B I_M$$
$$J = (M - Q)^{-1} Q_B I_M$$

The value of J can thus be obtained as all other quantities have been obtained in the earlier section. J is the current vector corresponding to unknown currents. From this , I is obtained by adding the boundary conditions ie zero at i=0, i=2N, and I_m at i=N).

The J and I vectors obtained for N = 4 is given below:

```
The J vector is :
------------------
[0.    0.    0.001 0.001 0.    0.   ]

The I vector is :
------------------
[0.    0.    0.    0.001 1.    0.001 0.    0.    0.   ]
```

The code is given below

```
# finding the final solution
M = compute_M(a,2*N-2)
# finding the J vector
J = matmul(inv(M-Q),QB) * Im
J = abs(real(J))

# filling I vector with the values in the J vector
I[1:N] = J[0:N-1]
I[N+1:-1] = J[N-1:]
```

7 Plots and Errors

7.1 Plots

For N = 4

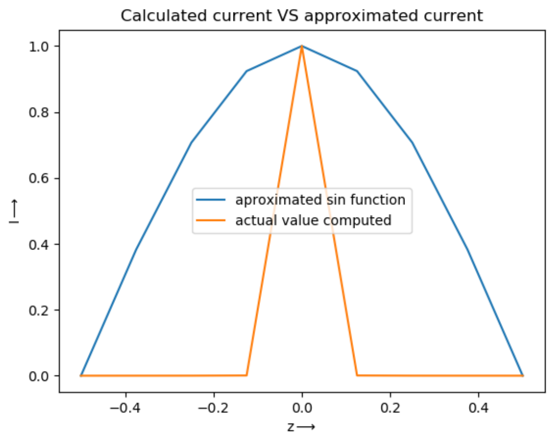

Figure 1: Calculated current Vs Standard assumption for N= 4

For N = 100

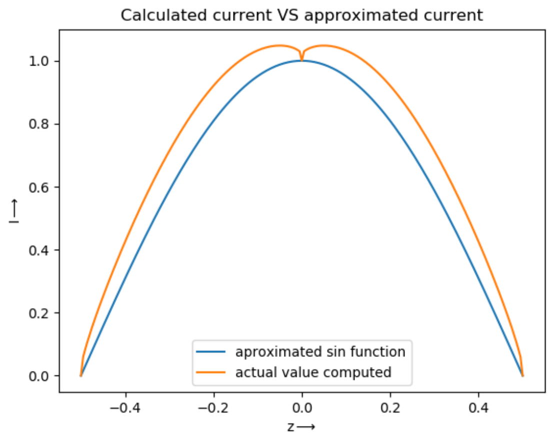

Figure 2: Calculated current Vs Standard assumption for N = 100

For N = 900

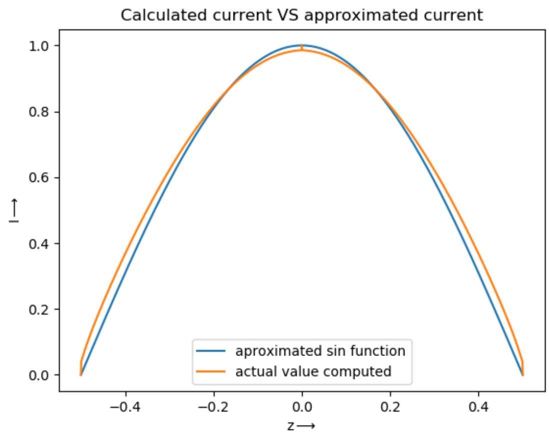

Figure 3: Calculated current Vs Standard assumption for N = 900

From above graphs it is clear that the standard assumption is a good estimation of current in the half-wave dipole antenna.

7.2 Errors

Now, let's check the mean squared errors in the estimation

For N = 4, mse = 0.33303
For N = 100, mse = 0.01025
For N = 900, mse = 0.00127

Here we can see that the errors are small for high N .Hence it is clear that this assumption is best one

8 Conclusion

- The standard assumption for dipole antenna current is a very good assumption.
- For very small value of N(say N =4) the errors are big but, for higher N , it is proven that the error are small and the plot of both almost coincide
- We solved magnetic equations using matrices with the help of python `numpy` library
- We have learnt to use python to solve different type of problems in the domain of physics and electronics