EE2703 : Applied Programming Lab
End-semester exam

Potta Muni Asheesh
EE19B048

May 30, 2021

1 Introduction

This problem is about a loop antenna whose length is equal to the wave length λ of the radiation.

The loop carries a current given by

$$I = \frac{4\pi}{\mu_0}\cos(\phi)\exp(j\omega t)$$

where, ϕ is the polar angle in cylindrical coordinates. The radius of the loop (a) is 10cm which is also equal to $\lambda/2\pi = 1/k = c/\omega$, this implies circumference $2\pi a = \lambda$. For a given current, the magnetic vector potential is given by the integral,

$$\boldsymbol{A}(\boldsymbol{r}, t) = \frac{\mu_0}{4\pi}\int_C \frac{I(\boldsymbol{r'}, t')d\boldsymbol{r'}}{|\boldsymbol{r} - \boldsymbol{r'}|}$$

where C is the curve of the loop, $\boldsymbol{r'}$ is the position of point on the loop and $t' = t - \frac{|\boldsymbol{r}-\boldsymbol{r'}|}{c}$. And from the magnetic vector potential $\boldsymbol{A}$, the magnetic field $\boldsymbol{B}$ is given as $\boldsymbol{B} = \nabla \times \boldsymbol{A}$

2 Pseudocode

In the pseudocode below, the procedure or algorithm to solve the problem is given. The declarations in the pseudocode have been written in english language.

```
X, Y, Z are the x, y and z components of meshgrid of x from -1 to 1, y
    ↪ from -1 to 1 and z from 1 to 1000 indexed i,j,k
x' is x-coordinate of 100 sections of the loop indexed by l.
y' is y-coordinate of 100 sections of the loop indexed by l.
phi' is the polar angle of mid point of the 100 sections of the loop
    ↪ indexed by l.
dx' is x-component of dl'
dy' is y-component of dl'
k is the wave constant

```

```
Ax and Ay is initially 0.

for l from 0 to 99
    R = sqrt((X-x'[l])^2 + (Y-y'[l])^2 + Z^2)
    Ax += cos(phi'[l])*exp(-j*k*R)*dx'[l]/R
    Ay += cos(phi'[l])*exp(-j*k*R)*dy'[l]/R

Bz = 0.5*(Ay[1, 0, 1 to 1000] - Ax[0, 1, 1 to 1000] - Ay[-1, 0, 1 to 1000] + Ax[0, -1, 1 to 1000])
```

3 Current elements in the loop

Since the loop of wire is in the x-y plane, has a radius of $a = 10$cm and centered at the origin. Each point on the loop ($\vec{r'}$) can be denoted in terms of polar angle ϕ as

$$\vec{r'} = a(\cos(\phi)\hat{x} + \sin(\phi)\hat{y}); 0 \leq \phi < 2\pi$$

And, the current in the loop at time $t = 0$ is given by,

$$I(\phi) = \frac{4\pi}{\mu_0} cos(\phi)$$

Consider an element of length $dl' = ad\phi$ on the loop, then in cartesian coordinates, the element length vector is given by,

$$\vec{dl'} = ad\phi(-\sin(\phi)\hat{x} + \cos(\phi)\hat{y})$$

For computation, the loop is broken into 100 sections and x and y components of $\vec{r'}$ and $\vec{dl'}$ for these 100 sections are obtained in an array as shown.

```
a = 10; k = 1/a

phi = np.linspace(0, 2*np.pi, 101); phi = phi[:-1]
delta_phi = phi[1]-phi[0]
phi += (phi[1]-phi[0])/2 # Setting the point to middle of the element.

x1, y1 = a*np.cos(phi), a*np.sin(phi) # r' vector components
dx1, dy1 = -a*delta_phi*np.sin(phi), a*delta_phi*np.cos(phi) # dl' vector components
```

The x and y components of vector $I\vec{dl'} = \cos(\phi)ad\phi\hat{\phi}$ for the 100 sections is obtained as

```
# x and y components of the current element vector (Idl'),
# alternating elements are considered for visual comfort.
I_x, I_y = (x1[::2]/a)*(dx1[::2]), (x1[::2]/a)*(dy1[::2])
```

Note that, the scaling factor $4\pi/\mu_0$ is not considered for the plot.

The plot of the current element vectors is shown below

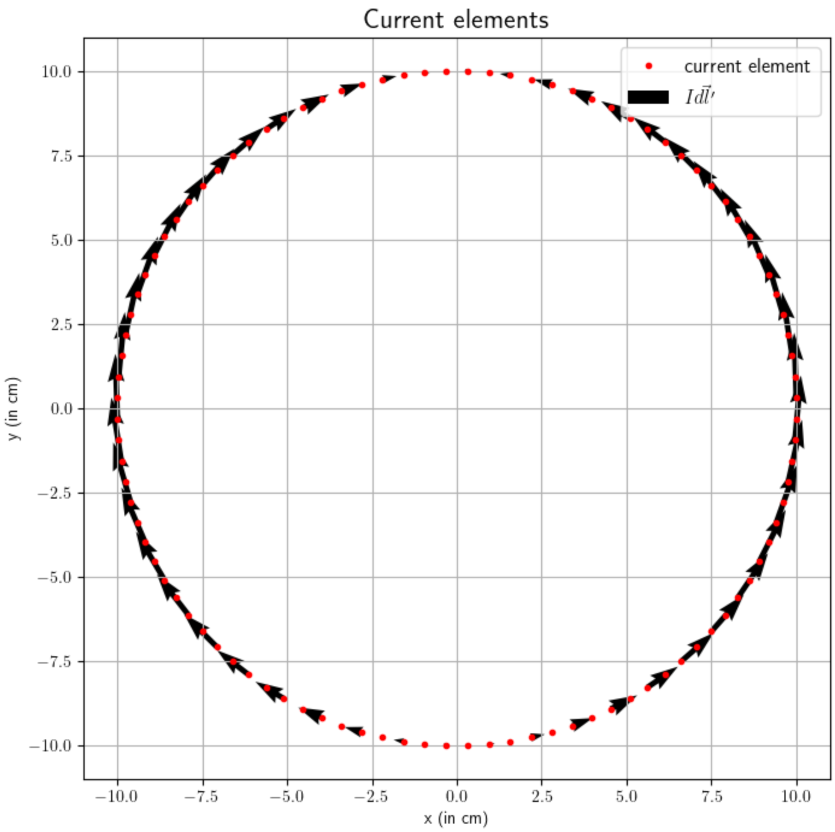

Figure 1: The vector $Id\vec{l}'$ at each section

Note that, alternate vector arrows are shown to avoid clumsiness in the plot.

It can be observed that the current is symmetric about the y-axis. From this, it can be expected that the magnetic field along z-axis (B_z) is 0.

4 Computing the vector potential $\vec{A}$ and magentic field $B_z(z)$

In order to compute the vector potential, consider a $3 \times 3 \times 1000$ 3D mesh in which x varies from -1cm to 1cm, y varies from -1cm to 1cm and z varies from 1cm to 1000cm. The seperation between each mesh point in a plane is 1cm. It is created as

```
x = np.linspace(-1.0,1.0,3)
y = np.linspace(-1.0,1.0,3)
z = np.linspace(1.0, 1000.0, 1000)

```

```
X, Y, Z = np.meshgrid(x, y, z, indexing='ij')
```

Since, the vector potential $\vec{A_{ijk}}$ is computed numerically as,

$$\left(\vec{A_{ijk}}\right)_x = \sum_{l=0}^{99} \frac{\cos(\phi_l)\exp(-jkR_{ijkl})(-\sin(\phi_l))dx'_l}{R_{ijkl}}$$
$$\left(\vec{A_{ijk}}\right)_y = \sum_{l=0}^{99} \frac{\cos(\phi_l)\exp(-jkR_{ijkl})(\cos(\phi_l))dy'_l}{R_{ijkl}}$$

where l is the index of the current element and i, j, k are the indices of the points in the mesh (x_i, y_j, z_k), $k = \omega/c$ is the wave propogation constant and $R_{ijkl} = |(x_i - a\cos(\phi_l))\hat{x} + (y_j - a\sin(\phi_l))\hat{y} + (z_k)\hat{z}|$

To compute each term in the summation, a function named `calc` is defined, which computes and returns both x and y components of the vector potential. It is defined as shown below.

```
def calc(l, x1, y1, dx1, dy1, X, Y, Z, phi, k, dynamic=True):
    """
    This function finds the vector potential due to the current
    element of index l. Here, the current I = (4*pi/mu0)*cos(phi),
    and depending on whether the current is time-dependent or not
    (determined by the boolean parameter 'dynamic'), the current
    is multiplied by exp(j*w*t). Here, first, R = |r - r'| is
    found and then vector potential A is computed, which has
    only x and y components.
    """
    Rl = np.sqrt((X-x1[l])**2 + (Y-y1[l])**2 + Z**2)
    if dynamic:
        A_xl = (np.cos(phi[l])*np.exp(-1j*k*Rl)*dx1[l])/Rl
        A_yl = (np.cos(phi[l])*np.exp(-1j*k*Rl)*dy1[l])/Rl
    else:
        A_xl = (np.cos(phi[l])*dx1[l])/Rl
        A_yl = (np.cos(phi[l])*dy1[l])/Rl
    return np.array([A_xl, A_yl])
```

Note, the function can be used to compute vector potential for both dynamics and statics, based on the boolean value of the parameter `dynamic`.

Finally, the summation is done using for loop to iterate over the index l. *Vectorized code* could not be used here because a 4 dimensional array cannot be created in any way from a 3 dimensional and a 1 dimensional arrays using math operations.

The summation is done is done as follows,

```
A = calc(0, x1, y1, dx1, dy1, X, Y, Z, phi, k, dynamic=True)
for l in range(1, 100):
    A += calc(l, x1, y1, dx1, dy1, X, Y, Z, phi, k, dynamic=True)
```

The magentic field is given by the curl of the vector potential,

$$\vec{\nabla} \times \vec{A} = \vec{B}$$

The z component of the magnetic field aloong z-axis $B_z(z)$ is given by,

$$B_z(z) = \frac{\partial A_y}{\partial x}(0, 0, z) - \frac{\partial A_x}{\partial y}(0, 0, z)$$

Numerically, it computed using the central difference method as

$$B_z(z) = \frac{A_y(\Delta x, 0, z) - A_y(-\Delta x, 0, z)}{2\Delta x} - \frac{A_x(0, \Delta y, z) - A_x(0, -\Delta y, z)}{2\Delta y}$$

where, $\Delta x = \Delta y = 1\text{cm}$
This is done as shown below.

```
# Finding the curl of the vector potential along the z-axis to find the
    magentic field.
B_z = np.abs(0.5*(A[1,2,1,:]-A[0,1,2,:]-A[1,0,1,:]+A[0,1,0,:]))
```

The logarithmic plot of the z-component of the magnetic field along the z-axis is shown below.

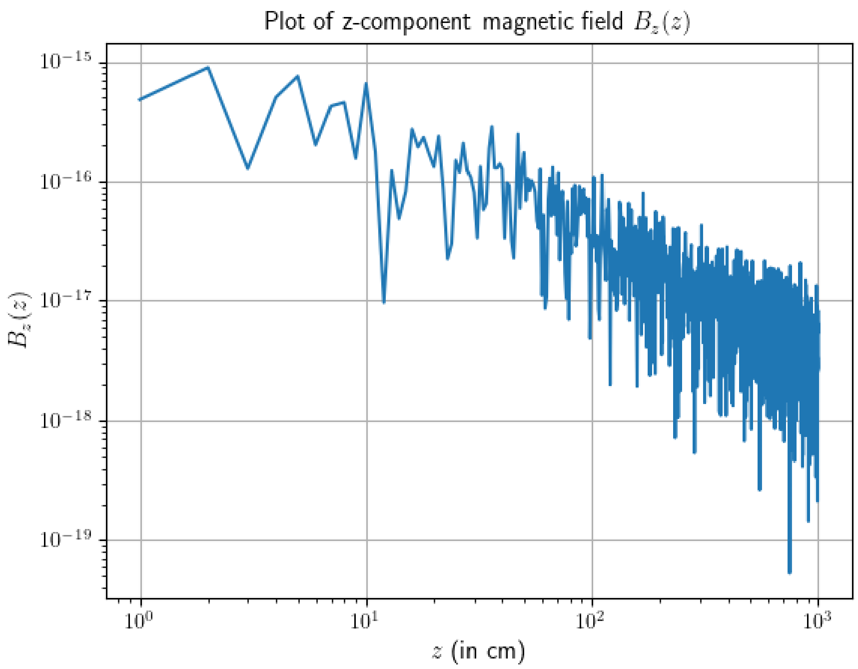

Figure 2: The magentic field along the z-axis

It can be seen that the magentic field is of the order $10^{-16} - 10^{-17}$ which can practical be considered as 0, which is as expected.

5 Least squares fit

Consider the model for the magentic field along z-axis to be $B_z = cz^b$, to find a least squares fit for this model, the exponential has to be converted to something linear by taking logarithm, so

$$\log(B_z) = b\log(z) + \log(c)$$

The corresponding matrix equation is

$$\begin{pmatrix} \log(z_0) & 1 \\ \log(z_1) & 1 \\ . & . \\ . & . \\ . & . \\ \log(z_{999}) & 1 \end{pmatrix} \begin{pmatrix} b \\ \log(c) \end{pmatrix} = \begin{pmatrix} \log(B_z(z_0)) \\ \log(B_z(z_1)) \\ . \\ . \\ . \\ \log(B_z(z_{999})) \end{pmatrix}$$

The parameters b and c are found as

```python
# Finding the least squares fit (b, c) for the model, Bz = c*(z^b)
M = np.c_[np.log(z),np.ones(z.size)]
fit = lstsq(M, np.log(B_z))[0]
b = fit[0]
c = np.exp(fit[1])
print("The approximated values of b and c are {}, {}".format(b, c))
```

The plot of the magnetic field given by least squares fit is shown below

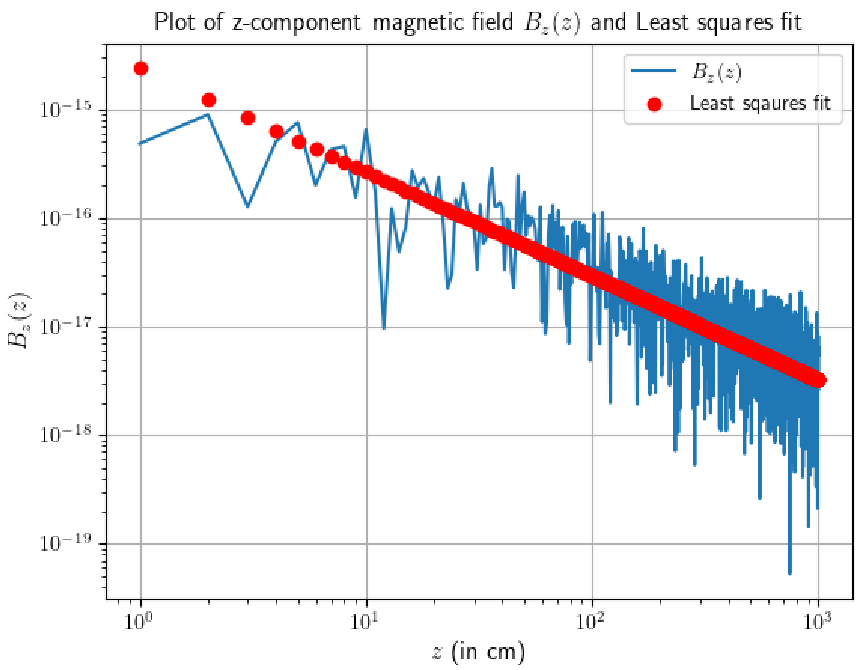

Figure 3: The magentic field along the z-axis from least squares fit

The approximated values of b and c are -0.9558345894653539, 2.3929363959585138e-15

Since the magnetic field is actually 0, the order of the decay of magnetic field can be considered the computational error.

6 Difference between statics and dynamics

In the above case, the current changes with time, this is the case of magnetodynamics.

$$I = I(\phi)\exp(j\omega t)$$

In this case, the vector potential for a given current is given by

$$\vec{A}(\vec{r},t) = \frac{\mu_0}{4\pi}\int \frac{I(\phi)\exp(j(\omega t - kR))ad\phi}{R}$$

where, $R = |\vec{r} - \vec{r'}|$, $\vec{r}$ is the point where the vector potential is being caluculated and $\vec{r'}$ is the point on the loop.

Now, consider a current independent of time, say

$$I = I(\phi)$$

This is the case of magnetostatics. Here, the vector potential is given by

$$\vec{A}(\vec{r}) = \frac{\mu_0}{4\pi}\int \frac{I(\phi)ad\phi}{R}$$

where, R is same as mentioned above.

Numerically, this becomes,

$$\left(\vec{A_{ijk}}\right)_x = \frac{\mu_0}{4\pi}\sum_{l=0}^{99}\frac{I(\phi_l)(-\sin(\phi_l))dx'_l}{R_{ijkl}}$$
$$\left(\vec{A_{ijk}}\right)_y = \frac{\mu_0}{4\pi}\sum_{l=0}^{99}\frac{I(\phi_l)(\cos(\phi_l))dy'_l}{R_{ijkl}}$$

For $I(\phi) = \frac{4\pi}{\mu_0}\cos(\phi)$, There is not much difference between dynamics and statics case, both have zero magnetic field along z-axis.

In order to observe the difference between these two cases, consider $I(\phi) = \frac{4\pi}{\mu_0}$. For this current, in case of magnetostatics, we can analytically find out that magnetic field along z-axis, is given by

$$B_z(z) = 2\pi\frac{a^2}{(\sqrt{z^2+a^2})^3}$$

So, the expected decay of the magnetic field is z^{-3}. The magnetic field is computed numerically, a function `calc1` is defined for this (see complete python code given below), and a least squares fit of type $B_z(z) \approx cz^b$ is found, it looks like

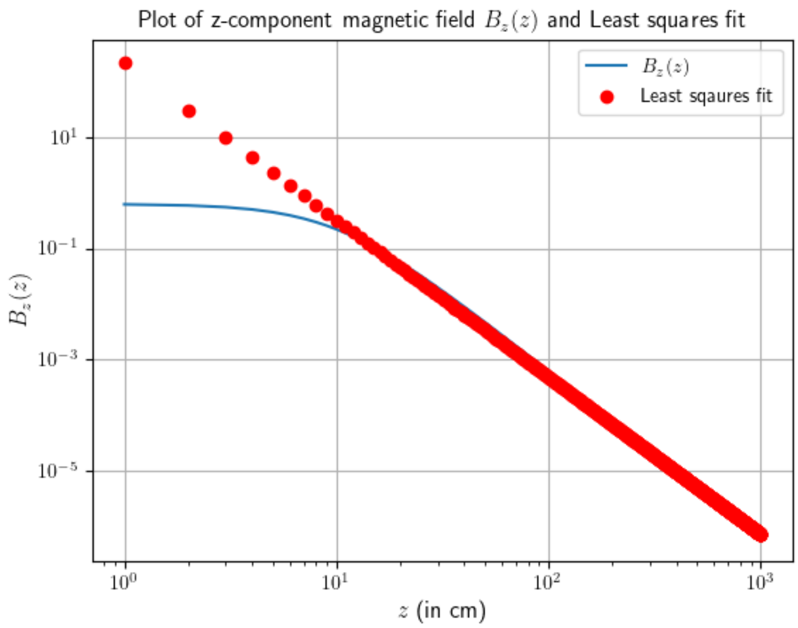

Figure 4: The magentic field along the z-axis from least squares fit

Here, the approximated values of b and c are -2.8261920569266086, 215.85790244337815, which is very near to the analytical value of b = -3.

For the magnetodynamics case, find the analytical answer is difficult. So, directly the numerical answer is computed at time $t = 0$ and it is obtained as,

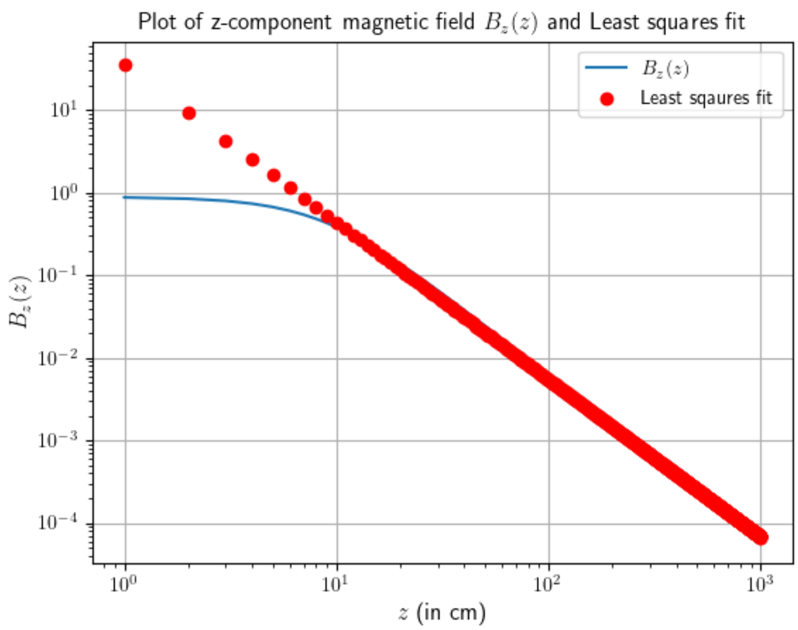

Figure 5: The magentic field along the z-axis from least squares fit

Here, the approximated values of b and c are -1.905750316850618, 35.243198919063666.

It is observed that b is near -2, which is a change of one degree from the case of statics. This change is because of the exponential term in the integral for vector potential.

7 Complete python code

```
import numpy as np
import matplotlib.pyplot as plt
from scipy.linalg import lstsq

# Using Latex in plots.
plt.rcParams.update({'text.usetex':True})

a = 10; k = 1/a

def calc(l, x1, y1, dx1, dy1, X, Y, Z, phi, k, dynamic=True):
    """
    This function finds the vector potential due to the current
    element of index l. Here, the current I = (4*pi/mu0)*cos(phi),
    and depending on whether the current is time-dependent or not
    (determined by the boolean parameter 'dynamic'), the current
    is multiplied by exp(j*w*t). Here, first, R = |r - r'| is
    found and then vector potential A is computed, which has
    only x and y components.
    """
    Rl = np.sqrt((X-x1[l])**2 + (Y-y1[l])**2 + Z**2)
    if dynamic:
        A_xl = (np.cos(phi[l])*np.exp(-1j*k*Rl)*dx1[l])/Rl
        A_yl = (np.cos(phi[l])*np.exp(-1j*k*Rl)*dy1[l])/Rl
    else:
        A_xl = (np.cos(phi[l])*dx1[l])/Rl
        A_yl = (np.cos(phi[l])*dy1[l])/Rl
    return np.array([A_xl, A_yl])

def calc1(l, x1, y1, dx1, dy1, X, Y, Z, phi, k, dynamic=True):
    """
    This function finds the vector potential due to the current
    element of index l. Here, the current I = (4*pi/mu0)*1,
    and depending on whether the current is time-dependent or not
    (determined by the boolean parameter 'dynamic'), the current
    is multiplied by exp(j*w*t). Here, first, R = |r - r'| is
    found and then vector potential A is computed, which has
    only x and y components.
    """
    Rl = np.sqrt((X-x1[l])**2 + (Y-y1[l])**2 + Z**2)
    if dynamic:
```

```python
        A_xl = (1*np.exp(-1j*k*Rl)*dx1[l])/Rl
        A_yl = (1*np.exp(-1j*k*Rl)*dy1[l])/Rl
    else:
        A_xl = (1*dx1[l])/Rl
        A_yl = (1*dy1[l])/Rl
    return np.array([A_xl, A_yl])

x = np.linspace(-1.0,1.0,3)
y = np.linspace(-1.0,1.0,3)
z = np.linspace(1.0, 1000.0, 1000)

X, Y, Z = np.meshgrid(x, y, z, indexing='ij')
phi = np.linspace(0, 2*np.pi, 101); phi = phi[:-1]
delta_phi = phi[1]-phi[0]
phi += (phi[1]-phi[0])/2 # Setting the point to middle of the element.

x1, y1 = a*np.cos(phi), a*np.sin(phi) # r' vector components
dx1, dy1 = -a*delta_phi*np.sin(phi), a*delta_phi*np.cos(phi) # dl' vector components
# x and y components of the current element vector (Idl'),
# alternating elements are considered for visual comfort.
I_x, I_y = (x1[::2]/a)*(dx1[::2]), (x1[::2]/a)*(dy1[::2])

# plotting the vector arrows of current elements in the loop.
fig1 = plt.figure(1, figsize=(8,8))
ax = fig1.add_subplot(111)
ax.plot(x1, y1, 'r.', label='current element')
ax.quiver(x1[::2], y1[::2], I_x, I_y, label=r"$I\vec{dl'}$")
plt.grid(True)
plt.legend(loc=1, fontsize='large')
plt.title('Current elements', size=16)
plt.xlabel('x (in cm)')
plt.ylabel('y (in cm)')

# Since summation over a single dimension is not possible with vectorized code,
# for loop needs to be used.
A = calc(0, x1, y1, dx1, dy1, X, Y, Z, phi, k, dynamic=True)
for l in range(1, 100):
    A += calc(l, x1, y1, dx1, dy1, X, Y, Z, phi, k, dynamic=True)

# Un-commment this section and comment the above part, to find magentic field
# Bz for constant current w.r.t phi, i.e; I = 4pi/mu0.
# A = calc1(0, x1, y1, dx1, dy1, X, Y, Z, phi, k, dynamic=True)
# for l in range(1, 100):
```

```python
#     A += calc1(l, x1, y1, dx1, dy1, X, Y, Z, phi, k, dynamic=True)

# Finding the curl of the vector potential along the z-axis to find the magentic field.
B_z = np.abs(0.5*(A[1,2,1,:]-A[0,1,2,:]-A[1,0,1,:]+A[0,1,0,:]))

plt.figure(2)
plt.loglog(z, B_z, label=r'$B_z(z)$')
plt.title(r'Plot of z-component magnetic field $B_z(z)$')
plt.xlabel(r'$z$ (in cm)', size=12)
plt.ylabel(r'$B_z(z)$', size=12)
plt.grid(True)

# Finding the least squares fit (b, c) for the model, Bz = c*(z^b)
M = np.c_[np.log(z),np.ones(z.size)]
fit = lstsq(M, np.log(B_z))[0]
b = fit[0]
c = np.exp(fit[1])
print("The approximated values of b and c are {}, {}".format(b, c))
plt.figure(3)
plt.loglog(z, B_z, label=r'$B_z(z)$')
plt.title(r'Plot of z-component magnetic field $B_z(z)$ and Least squares fit')
plt.xlabel(r'$z$ (in cm)', size=12)
plt.ylabel(r'$B_z(z)$', size=12)
plt.grid(True)
plt.loglog(z, np.exp(np.dot(M, fit)), 'ro', label=r'Least sqaures fit')
plt.legend()

plt.show()

# print(X, Y, Z, sep='\n')
```