EE2703 End-semester Exam

Anvith Pabba [EE19B070]

22nd May 2021

1 Abstract

The main objectives of the Assignment are:

- Analysing various parameters of a loop of wire containing a non-uniform current and storing these parameters in the form of vectors.
- Plotting the position of current elements and visualising the direction and magnitude of respective currents using quiver plots.
- Computing the vector potentials ($\vec{\mathbf{A}}$) of points in space through vectorized code.
- Computing the magnetic flux density ($\vec{\mathbf{B}}$) using an approximation of the curl.
- Plotting the loglog plot of $|\vec{\mathbf{B}}(z)|$ vs z.
- Fitting the obtained values of $|\vec{\mathbf{B}}(z)|$ in the form of a polynomial and analysing its behaviour.

2 Problem Description

2.1 Given Information

In the given question, there is a loop antenna of length 'λ', which has a current

$$I = \frac{4\pi}{\mu_o} cos(\phi) exp(j\omega t) \tag{1}$$

flowing through it, where ϕ is the angle in the polar coordinates. i.e in (r,ϕ,z) coordinates.

- **Its important to note that we DO NOT consider the time dependence of the current throughout our analysis except in the value of the constant 'k' which is equal to w/c where w is the angular frequency.**

The wire is on the x - y plane and centered at the origin. The given radius of the loop is 10 cm. The problem is to compute and plot the magnetic field ($\vec{\mathbf{B}}$) along the z axis from 1cm to 1000 cm, plot it and then fit the data to a polynomial of the form

$$|\vec{B}| = cz^b \tag{2}$$

3 Theoretical Equations and Forumlas Required

3.1 The current

The current I is given by,

$$I = \frac{4\pi}{\mu_o} cos(\phi) exp(j\omega t) \tag{3}$$

where the x and y components are given by

$$Ix = -Isin(\phi)\hat{x} \tag{4}$$

$$Iy = Icos(\phi)\hat{y} \tag{5}$$

3.2 Calculation of the Vector Potential

$$\vec{A}(r, \phi, z) = \frac{\mu_o}{4\pi} \int \frac{I(\phi)\hat{\phi} e^{-jkR} a d\phi}{R} \tag{6}$$

where

- $\vec{R} = \vec{r} - \vec{r'}$
- a = radius = 10cm
- $k = 0.1cm^{-1}$
- $\vec{r}$ is the point where we want the field.
- and, $\vec{r'} = a\hat{r'}$ is the point on the loop.

This can be reduced to the sum:

$$\vec{A}_{ijk} = \sum_{N-1}^{l=0} \frac{cos(\phi_l')e^{-jkR_{ijk}}\vec{dl'}}{R_{ijk}} \tag{7}$$

where,

- $\vec{r} = x_i\hat{x} + y_j\hat{y} + z_k\hat{z}$
- $\vec{r'} = acos(\phi_l')\hat{x} + asin(\phi_l')\hat{y} + 0\hat{z}$
- R_{ijk} is a vector that contains all the magnitudes of the distances between all the points in the 3x3x1000 vector space with the 'l'th point on the loop that lies at $\vec{r'}$.
- dl is the vector that contains the incremental length vector of the 'l'th current element.

NOTE: Throughout our analysis, we split most of these parameters into their x and y components and evaluate each of them separately

3.3 The Magnetic Flux Density

The Magnetic flux density is given by:

$$\vec{B} = \nabla \times \vec{A} \tag{8}$$

Along the z-axis and using an approximation, this can be simplified to:

$$B_z(z) = \frac{A_y(\Delta x, 0, z) - A_x(0, \Delta y, z) - A_y(-\Delta x, 0, z) + A_x(0, -\Delta y, z)}{4\Delta x \Delta y} \tag{9}$$

Where

$$\Delta x = \Delta y = 1 \tag{10}$$

So now to equation is given by:

$$B_z(z) = \frac{A_y(1,0,z) - A_x(0,1,z) - A_y(-1,0,z) + A_x(0,-1,z)}{4} \tag{11}$$

NOTE: the formula given above uses the coordinates of the point in the 3d space, but when we write the same formula in a vectorized code manner, we need to use the EQUIVALENT array indices

For example, [-1,1,:] is given by [0,2,:] when written in the vectorized code form.

So now, the final code with equivalent array indices turns out to be:

$$B_z(:) = \frac{A_y(2,1,:) - A_x(1,2,:) - A_y(0,1,:) + A_x(1,0,:)}{4} \tag{12}$$

4 Visualising Higher Order Vectors

Throughout this analysis, we create many vectors that are 4 dimensional. These vectors can be quite complicated to visualise. Throughout the assignment, the general convention is that for any 4d vector of shape (N,3,3,1000) [N is either 1 or 3],

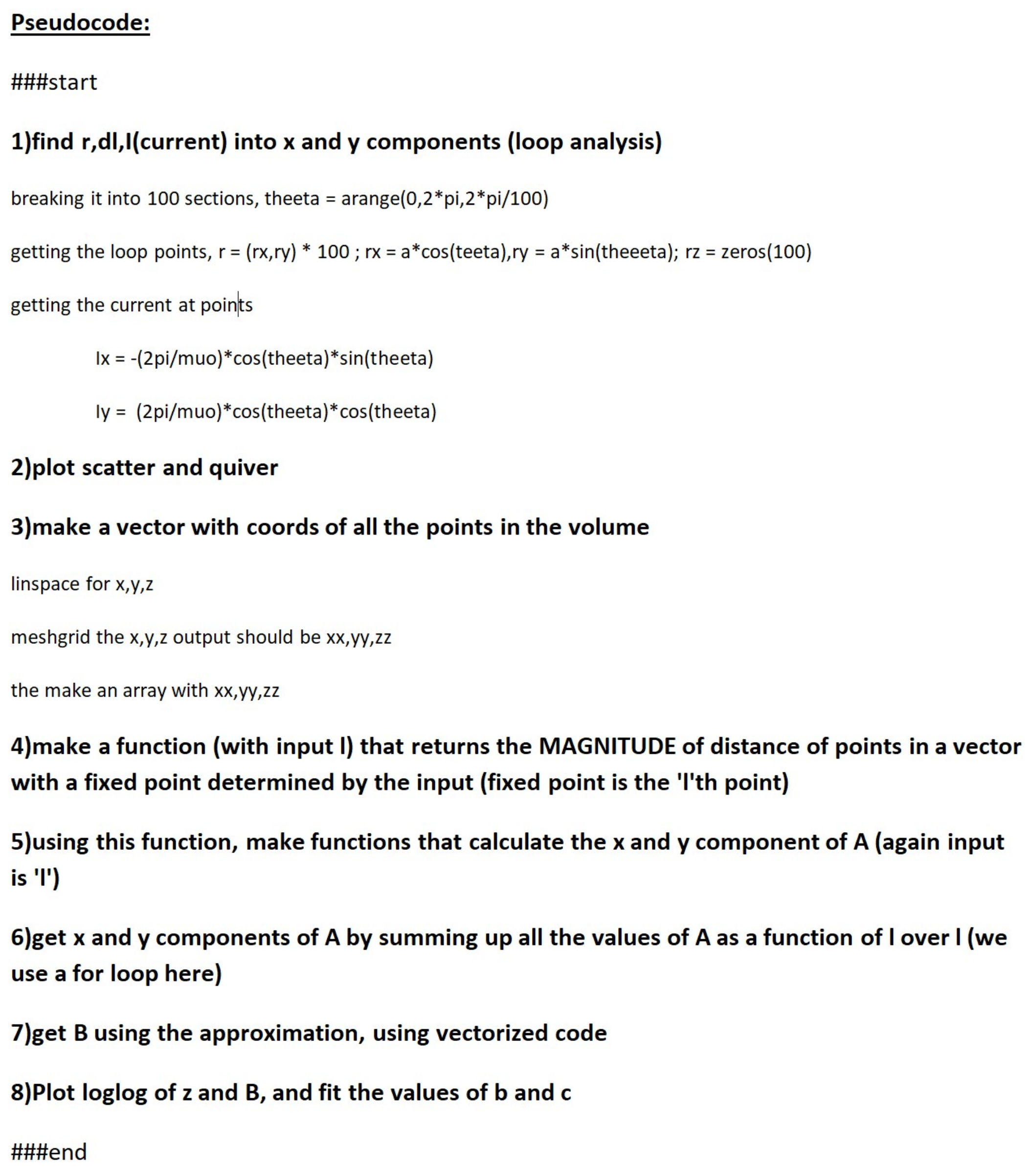
N[:,0,1,99] is equal to the parameters at the respective point [-1,0,100] in the vector space.

Examples of 4-d vectors in the Assignment:

1. Vector $rijk\_$
   - The shape of this vector is (3,3,3,1000)
   - In this vector, the output of $rijk\_$[:,0,1,99] is a vector that has 3 coordinates of the respective point [-1,0,100].
   - That is, $rijk\_$[:,0,1,99] == [-1,0,100]
   - Therefore if the first parameter is 0, then $rijk\_$[0,0,1,99] == -1 i.e this gives us the x coordinate of the point, and similarly, first parameter = 1 gives us the y coordinate, =2 gives us the z coordinate
2. Vector B_x
   - The shape of this vector is (1,3,3,1000)
   - B_x[0,a,b,c] gives us the value of B_x at the respective [a,b,c] point.

5 Questions In The Assignment

5.1 Q1: Pseudo Code



Pseudocode:

```
###start

1)find r,dl,I(current) into x and y components (loop analysis)

breaking it into 100 sections, theeta = arange(0,2*pi,2*pi/100)

getting the loop points, r = (rx,ry) * 100 ; rx = a*cos(teeta),ry = a*sin(theeeta); rz = zeros(100)

getting the current at points

        Ix = -(2pi/muo)*cos(theeta)*sin(theeta)

        Iy =  (2pi/muo)*cos(theeta)*cos(theeta)

2)plot scatter and quiver

3)make a vector with coords of all the points in the volume

linspace for x,y,z

meshgrid the x,y,z output should be xx,yy,zz

the make an array with xx,yy,zz

4)make a function (with input l) that returns the MAGNITUDE of distance of points in a vector
with a fixed point determined by the input (fixed point is the 'l'th point)

5)using this function, make functions that calculate the x and y component of A (again input
is 'l')

6)get x and y components of A by summing up all the values of A as a function of l over l (we
use a for loop here)

7)get B using the approximation, using vectorized code

8)Plot loglog of z and B, and fit the values of b and c

###end
```

5.2 Q2: Assigning Coordinates to the Volume

To analyse the value of the magnetic flux density in the z direction, we first need to assign coordinates to all points in the test space.

The test space here refers to a 3x3x1000 cuboid with points on the x-y square grid centered at the z-axis going from -1 to 1 and the z axis going from z=1 to z=1000.

We perform this by first assigning the x,y and z vectors using linspace and then using meshgrid to obtain the xx,yy,and zz vectors.

We then create an array using the obtained xx,yy,zz which gives us a 4d vector **rijk** as the output.

5.2.1 Code:

```
#creating the mesh grid
x = np.linspace(0,2,3) # breaking the volume into 3x3x1000
x = x-1  #since we want to make a 3x3 grid in the x-y plane that is from [-1,0,
                                        1] NOT [0,1,2]
y = np.linspace(0,2,3)
y = y-1
z = np.linspace(1,1000,1000)
xx,yy,zz = np.meshgrid(x,y,z)
rijk = np.array((xx,yy,zz))

#this gives us a 4d vector rijk in which if we set the FIRST parameter to 0
                                        then it will give the y-coordinate
#of the point rijk, similarly if set it as 1 then it gives the x coordinate

#hence, we switch both of these for simpler use in a variable rijk_:
```

5.3 Q3 Part A: Loop Analysis

We divide the central 2π angle into 100 sections using the linspace function.

We then use this to assign the following parameters:

- angle theeta (which is equivalent to ϕ in the formulas)
- x and y component of the radial vector pointing to the current elements ($\vec{r}$)
- x and y current vectors of the current in each current element ($\vec{I}$)
- x and y components of the incremental length vector ($\vec{dl}$)

5.3.1 Code:

```
no_of_sections = 100
theeta = np.arange(0,2*np.pi,2*np.pi/no_of_sections) #diving the circle into
                                        100 sections
a = 10 #radius of the loop


#x and y components of the radial vector, r'l bar
rx = a*np.cos(theeta) #x component
ry = a*np.sin(theeta) #y component


#x and y components of the current carrying element at an angle of theeta
Ix = -1*(2*np.pi/(4*np.pi*1e-7))*np.sin(theeta)*np.cos(theeta)
Iy = (2*np.pi/(4*np.pi*1e-7))*np.cos(theeta)*np.cos(theeta)

#x and y components of dl'; the length vector of each incremental element
dlx = (2*np.pi*a/no_of_sections)*(-1*np.sin(theeta))
dly = (2*np.pi*a/no_of_sections)*(np.cos(theeta))
```

5.4 Q3 Part B: Plotting the Graphs

5.4.1 Plot of the Current elements

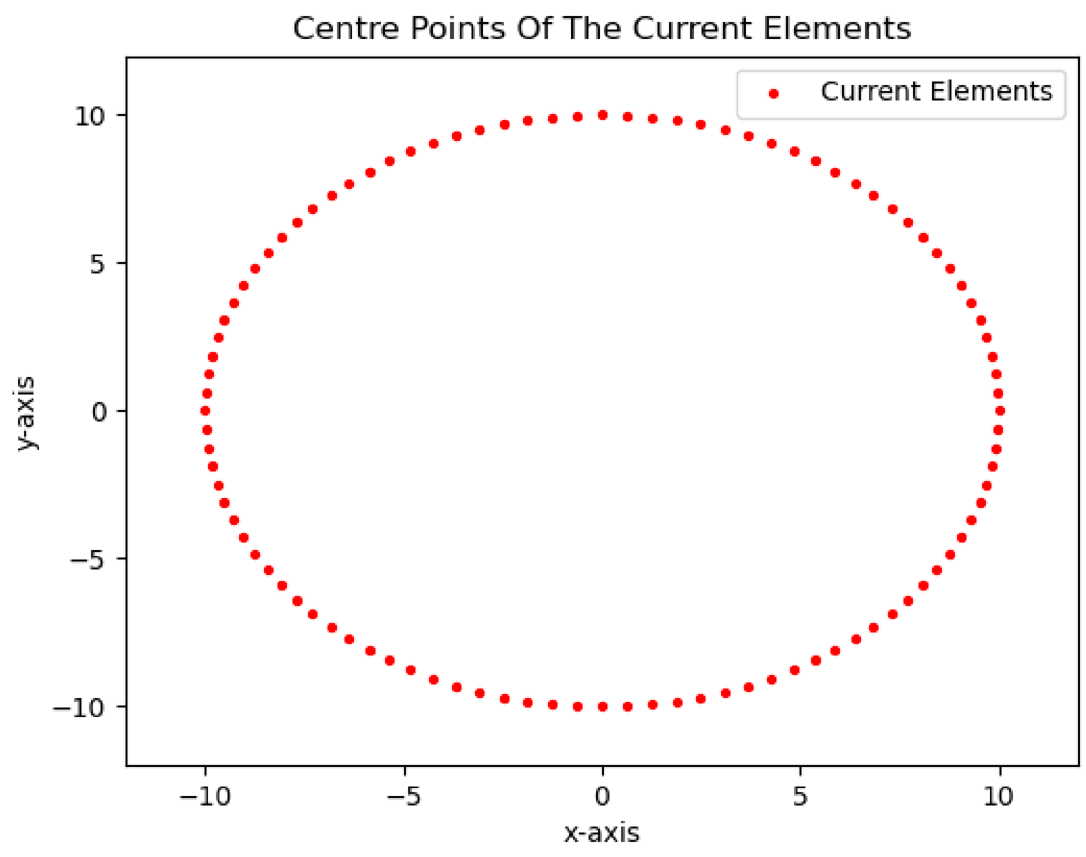


Figure 1: Current Elements

5.4.2 Code:

```
#plotting the scatter plot of all the current carrying elements in the loop
plt.scatter(rx,ry, c = 'red', s = 8, label = 'Current Elements')
plt.xlabel('x-axis')
plt.ylabel('y-axis')
plt.title('Centre Points Of The Current Elements')
plt.xlim((-12,12))
plt.ylim((-12,12))
plt.legend(loc = 'upper right')
plt.show()
```

5.4.3 Quiver Plot of the Currents

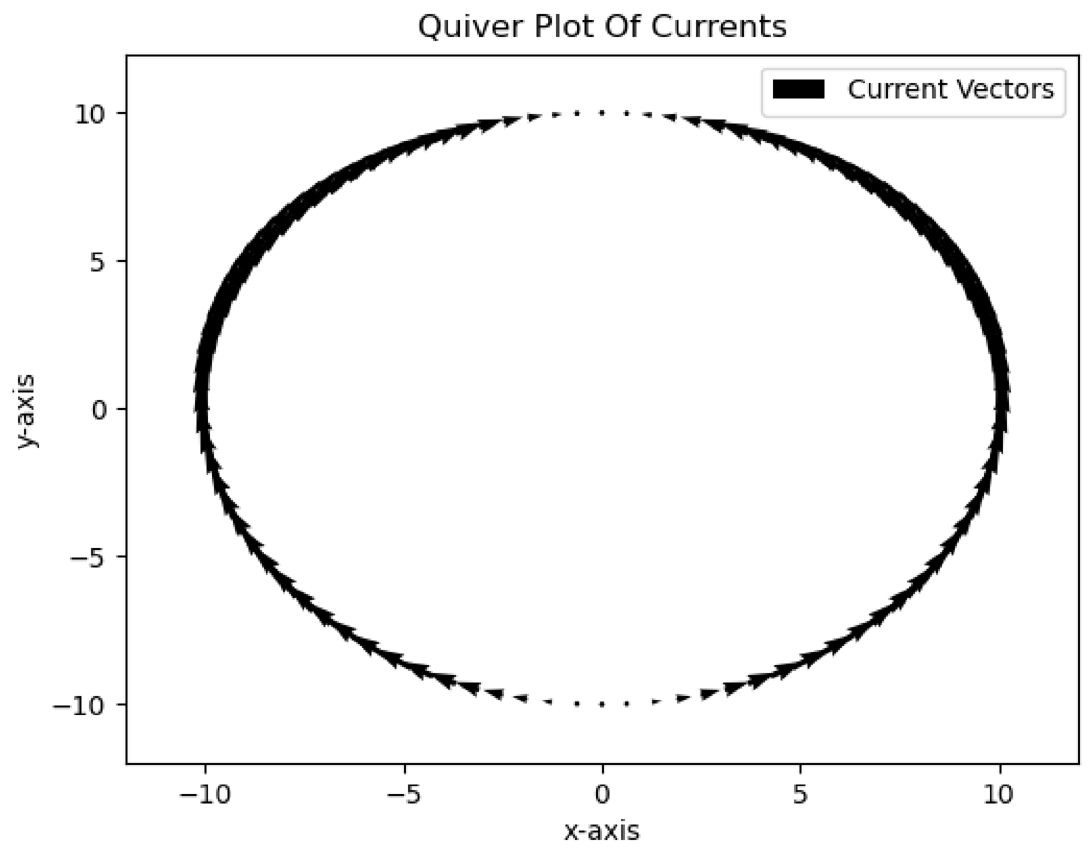


Figure 2: Quiver Plot of the Currents

5.4.4 Code2:

```
#plotting the quiver plot of currents in the current carrying elements
plt.quiver(rx,ry,Ix,Iy, color = 'black',label = 'Current Vectors')
plt.xlabel('x-axis')
plt.ylabel('y-axis')
plt.title('Quiver Plot Of Currents')
plt.xlim((-12,12))
plt.ylim((-12,12))
plt.legend(loc = 'upper right')
plt.show()
```

5.5 Q4: obtaining the vectorised loop parameters

These parameters have already been found in the **previous Q3 Part A**. Please refer to the **Q3 Part A: Loop Analysis** subsection, i.e subsection **5.3**.

5.6 Q5: Defining function calc(l)

now we define a function **calc** which has one input 'l' and the output is a vector with all the magnitude of distances between each point in the volume and the 'l'th point on the loop.

The below code defines the function.

5.6.1 Code:

```
'''
now we define a function calc
where, the input is 'l'
and the output is a vector with all the magnitude of distances between each
                                        point in the volume and the lth point
                                        on the loop
'''
def calc(l):
    Rijk = np.zeros((1,3,3,1000))
    Rijk[0,:,:,:] = ((rijk_[0,:,:,:]-rx[l])**2 + (rijk_[1,:,:,:]-ry[l])**2 + (
                                            rijk_[2,:,:,:])**2 )**(1/2)
    return Rijk
```

5.6.2 Testing the function

```
example = calc(0)
print(example[0,1,1,999])
```

output is **1000.049**.. which is the magnitude of the distance between [10,0,0] ('0'th element of the loop) and [0,0,1000] ((1,1,999)th element in the vector space)

```
TEST WAS SUCCESSFUL!
```

5.7 Q6: Extending calc to calculate $\vec{A}$

Now, using calc, we have obtained R_{ijk} for every index of 'l', we now use this to find using the previously stated formula.

$$\vec{A}_{ijkl} = \frac{cos(\phi_l')e^{-jkR_{ijk}}\vec{dl'}}{R_{ijk}} \tag{13}$$

Now, we can us calc(l) to calculate R_{ijk}. The final $\vec{A}_{ijk}$ is obtained through the summation of $\vec{A}_{ijkl}$ over all indices of l.

Again, as stated above, This function is analysed in the x and y direction separately, so we have to two functions A_x and A_y which we calculate using A_{xl} and A_{yl}

5.7.1 Code:

```
def A_x_l(l): #now we define a function that gives us the x component of A at
                                        all points from a the lth point on the
                                        loop
    A_x_ = np.zeros((1,3,3,1000))
    A_x_ = ( np.cos(theeta[l])* np.exp(-1*1j*0.1*calc(l))* dlx[l] )/ (calc(l))
    return A_x_

def A_y_l(l): #now we define a function that gives us the y component of A at
                                        all points from a the lth point on the
                                        loop
    A_y_ = np.zeros((1,3,3,1000))
    A_y_ = ( np.cos(theeta[l])* np.exp(-1*1j*0.1*calc(l))* dly[l] )/ (calc(l))
    return A_y_
```

5.8 Q7: Computing A_{ijk}

Now, we can finally find A_{ijk} by summing all the individual terms over the index 'l'.

5.8.1 Justification for using the for loop

The reason we use a for loop instead of vectorised code is because to make this operation possible in vectorised code, we would have to extend the A_x and A_y vector with a 5th parameter that has 100 terms (to match the 'l' parameter). Only then could this operation be carried out using vectorised code.

This would use up alot of space even though it is faster, hence using a for loop is justified here.

5.8.2 Code:

```
#Finally, getting the x and y components of A through a summation
A_x = np.zeros((1,3,3,1000))
for l in range(100): #we us a for loop as we need to find a summantion through
                                      iterating 'l'
    A_x = A_x + A_x_l(l)

#The reason we use a for loop instead of vectorised code is because to make
                                      this operation possible in vectorised
                                      code,
#we would have to extend the A_x and A_y vector with a 5th parameter that has
                                      100 terms (to match the 'l' parameter).
                                       then we could use vectorisation.
#this would us up alot of space even though it is faster, hence using a for
                                      loop is justified here.

A_y = np.zeros((1,3,3,1000))
for l in range(100):
    A_y = A_y + A_y_l(l) #see reason above
```

5.9 Q8: Computing B along the z-axis

Finally, using the approximation that:

$$B_z(z) = \frac{A_y(\Delta x,0,z) - A_x(0,\Delta y,z) - A_y(-\Delta x,0,z) + A_x(0,-\Delta y,z)}{4\Delta x\Delta y} \tag{14}$$

Where

$$\Delta x = \Delta y = 1 \tag{15}$$

So now to equation is given by:

$$B_z(z) = \frac{A_y(1,0,z) - A_x(0,1,z) - A_y(-1,0,z) + A_x(0,-1,z)}{4} \tag{16}$$

NOTE: the formula given above uses the coordinates of the point in the 3d space, but when we write the same formula in a vectorized code manner, we need to use the EQUIVALENT array indices

For example, [-1,1,:] is given by [0,2,:] when written in the vectorized code form.

So now, the final code with equivalent array indices turns out to be:

$$B_z(:) = \frac{A_y(2,1,:) - A_x(1,2,:) - A_y(0,1,:) + A_x(1,0,:)}{4} \tag{17}$$

We can get B through vectorised code.

5.9.1 Code:

```
#finding the final B output
B = np.zeros(1000)
B = (A_y[0,2,1,:] - A_x[0,1,2,:] - A_y[0,0,1,:] + A_x[0,1,0,:])/(4) #delta x =
                                                    delta y = 1
#but point [-1,0,z] in space corresponds to [0,1,z-1] in the array
```

The remaining Questions 9,10,11 are answered below.

6 ALL POSSIBLE CASES

We mainly analyse **four** possible cases regarding the current in the loop. These four cases are give below:

1. AC current (i.e time dependence) with spatial variation [This is the case when we follow everything in the assignment pdf with NO extra assumptions].
2. AC current (i.e time dependence) with spatial variation **BUT!, we ONLY take the magnitude of the current** i.e the current is always along the direction of dl].
3. DC current (i.e **NO** time dependence) with spatial variation [i.e 'w' is '0'].
4. and, DC current (i.e **NO** time dependence) with **NO** spatial variation.

There are many more cases that we can analyse, but for now these are the most important ones.

7 CASE 1: AC current (i.e time dependence) with spatial variation:

7.1 Q9: Plotting the Magnetic Flux Density B

7.1.1 loglog Plot of Bz vs z

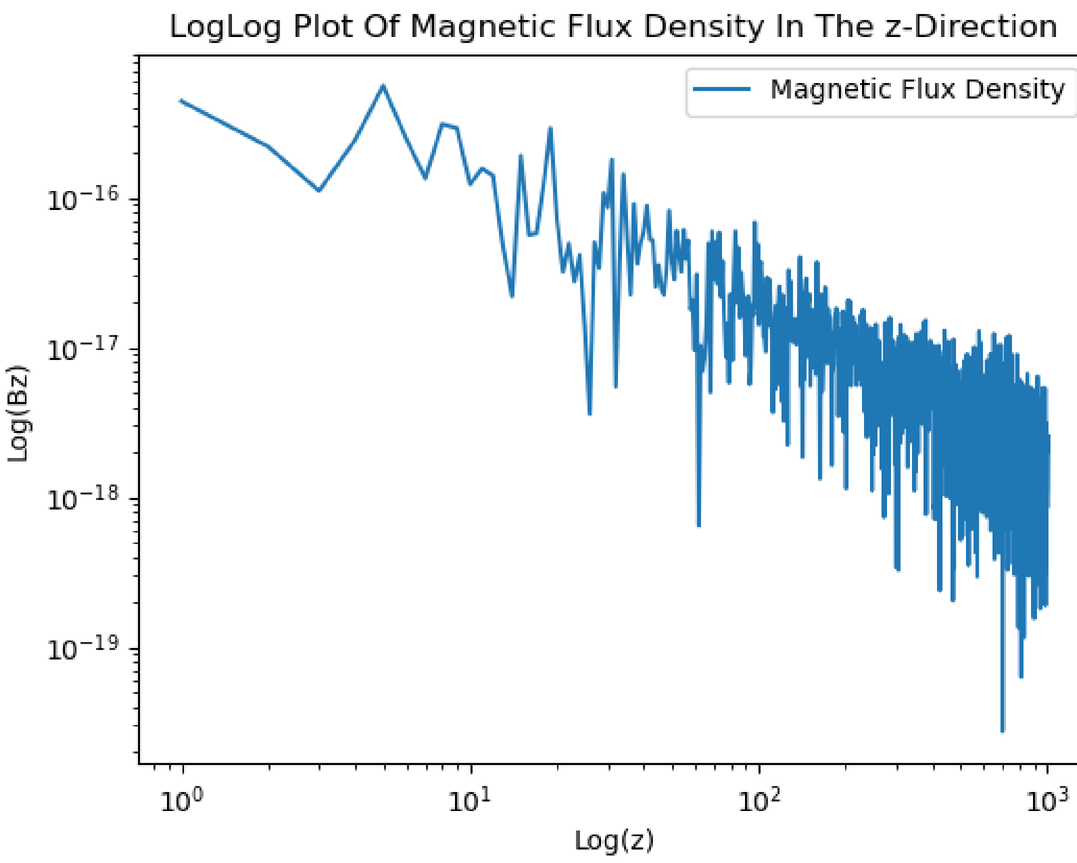


Figure 3: loglog plot of Bz vs z

7.1.2 Code:

```
#loglog plot of B vs z
z = np.arange(1,1001,1)
plt.loglog(z,np.abs(B),label = 'Magnetic Flux Density')
plt.xlabel('Log(z)')
plt.ylabel('Log(Bz)')
plt.title('LogLog Plot Of Magnetic Flux Density In The z-Direction')
plt.legend(loc = 'upper right')
plt.show()
```

7.2 Q10: Fitting the field in a polynomial

We now fit the field into a polynomial of the form

$$B(z) = cx^b \tag{18}$$

To do this, we use the np.linalg.lstsq function to find the value of log(c) and b where,

$$log(B(z)) = blog(z) + log(c) \tag{19}$$

this can be solved by the linear equation

$$Ax = B \tag{20}$$

Where A = vector containing log(z), and ones of suitable size and B = vector containing log(B(z))

Looking at the graph, its clear that it is not initially linear, but it turns slightly linear after a certain value of 'z'. Hence we find two fitting values

- First is only for the samples after the loglog plot is linear (or slightly resembles it.)
- Second is for all the samples

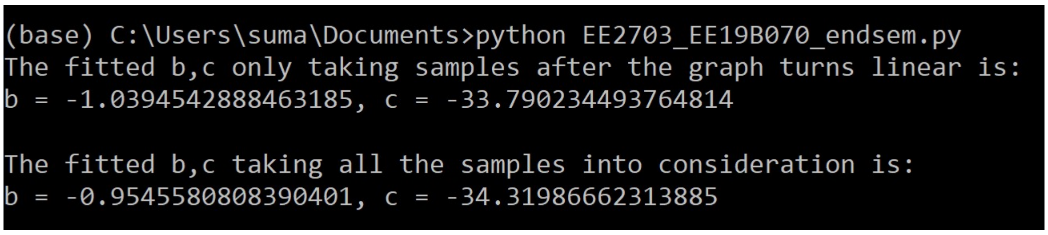
7.2.1 Code:

```
#in this, we only fit the values for the samples after z=100, this is because
                                        the loglog plot is linear here
A_fit = np.c_[np.log(z[100:]),np.ones(len(B)-100)]
B_fit = np.log(abs(B)[100:])
a_real,b_real = np.linalg.lstsq(A_fit, B_fit, rcond=None)[0]
print ("The fitted b,c only taking samples after the graph turns loglog is:\nb
                                        = {}, c = {}\n" .format(a_real,b_real))


#in this, we find a fit for all samples
A_fit = np.c_[np.log(z),np.ones(len(B))]
B_fit = np.log(abs(B))
a_no_approx,b_no_approx = np.linalg.lstsq(A_fit, B_fit, rcond=None)[0]
print ("The fitted b,c taking all the samples into consideration is:\nb = {}, c
                                         = {}" .format(a_no_approx,b_no_approx)
                                        )
```

7.3 The output of the fitting data

The output that one obtains is:



```
(base) C:\Users\suma\Documents>python EE2703_EE19B070_endsem.py
The fitted b,c only taking samples after the graph turns linear is:
b = -1.0394542888463185, c = -33.790234493764814

The fitted b,c taking all the samples into consideration is:
b = -0.9545580808390401, c = -34.31986662313885
```

Figure 4: Output of the terminal

$$|\vec{B}(z)| = cz^b \tag{21}$$

Here, we can see that the fitted values of b and c taking all the samples is given by:

$$b = -0.9545580808390401 \tag{22}$$

$$c = -34.31986662313885 \tag{23}$$

Where **b** is the decay rate.

8 CASE 2: AC current with spatial variation, BUT ONLY taking the magnitude of the current! :

In this case, while finding the currents, we take the $cos(\phi)$ term in $\vec{I}$ as the absolute value, i.e $|cos(\phi)|$.

$$I = \frac{4\pi}{\mu_o}|cos(\phi)|exp(j\omega t) \tag{24}$$

And the subsequent is given by:

$$\vec{A}_{ijk} = \sum_{N-1}^{l=0} \frac{|cos(\phi_l')|e^{-jkR_{ijk}}\vec{dl'}}{R_{ijk}} \tag{25}$$

8.1 Current Quiver plot

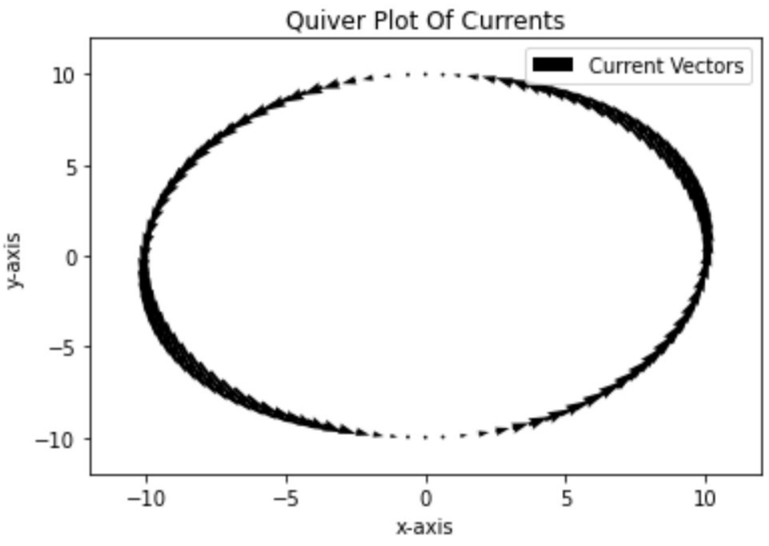


Figure 5: Current quiver plot in this case

8.1.1 loglog Plot of Bz vs z

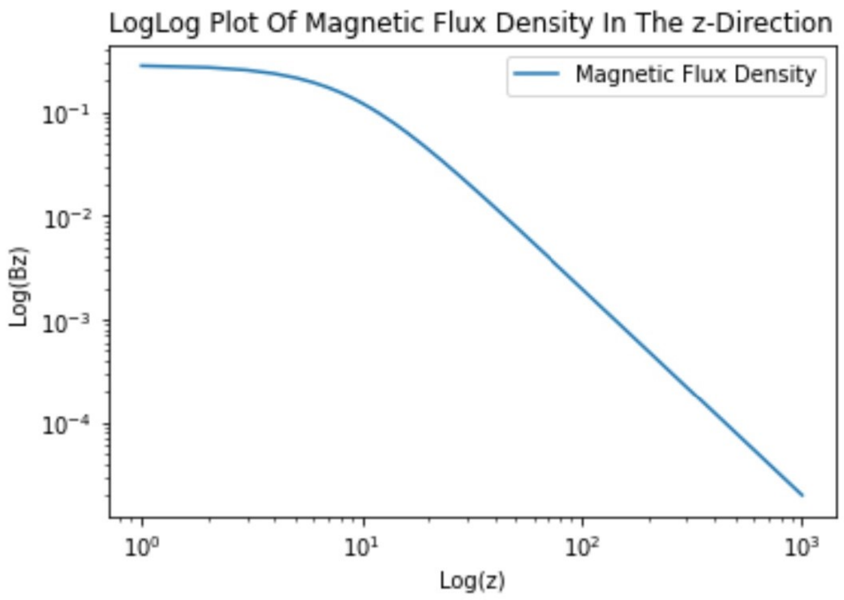
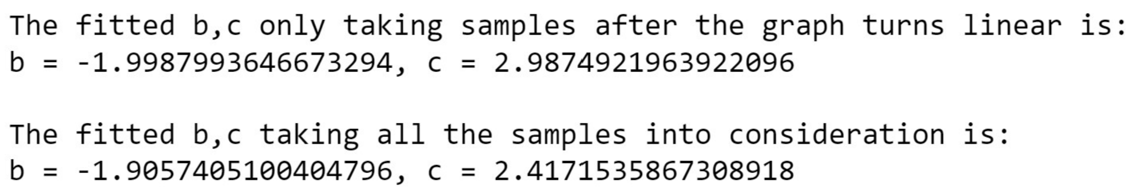


Figure 6: loglog plot of Bz vs z

8.2 Output



```
The fitted b,c only taking samples after the graph turns linear is:
b = -1.9987993646673294, c = 2.9874921963922096

The fitted b,c taking all the samples into consideration is:
b = -1.9057405100404796, c = 2.4171535867308918
```

Figure 7: Output

8.3 Q10: Fitting the field in a polynomial

We now fit the field into a polynomial of the form

$$B(z) = cx^b \tag{26}$$

The fitted value are given by:

- b = -1.9057405100404796 ≈ -2
- c = 2.4171535867308918

9 CASE 3: STATIC FIELD: DC current with spatial variation:

In the case of a DC current, the current has no time dependence, this means that in the formula for the current i.e:

$$I = \frac{4\pi}{\mu_o} cos(\phi) exp(j\omega t) \tag{27}$$

the value of ω is equal to **'0'**

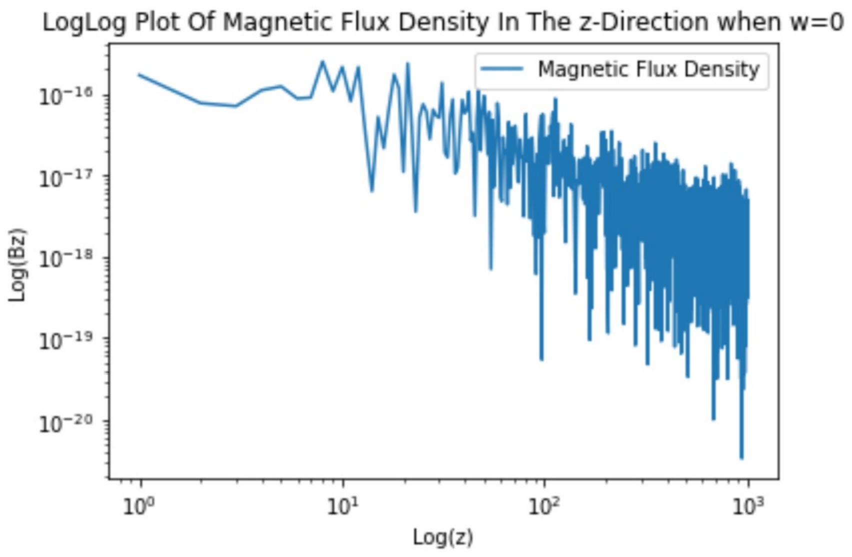
Therefore the value of the constant **k** which is equal to $\frac{w}{c}$ will be equal to **0**. i.e $k = 0$.

Hence the Summation to find the A is given by:

$$\vec{A}_{ijk} = \sum_{N-1}^{l=0} \frac{cos(\phi_l')\vec{dl'}}{R_{ijk}} \tag{28}$$

9.1 Q9: Plotting the Magnetic Flux Density B

9.1.1 loglog Plot of Bz vs z



```
The fitted b,c only taking samples after the graph turns linear is:
b = -0.9892626977324922, c = -34.32642732275246

The fitted b,c taking all the samples into consideration is:
b = -0.9088595273502664, c = -34.82814176186121
```

Figure 8: loglog plot of Bz vs z

9.2 Q10: Fitting the field in a polynomial

We now fit the field into a polynomial of the form

$$B(z) = cx^b \tag{29}$$

The fitted value are given by:

- b = -0.9088595273502664
- c = -34.82814176186121

10 CASE 4: DC current (i.e NO time variance) with NO spatial variation:

This is the case where we can use simple mathematics to find out the final formula to find the Magnetic field.

10.1 Current Quiver plot

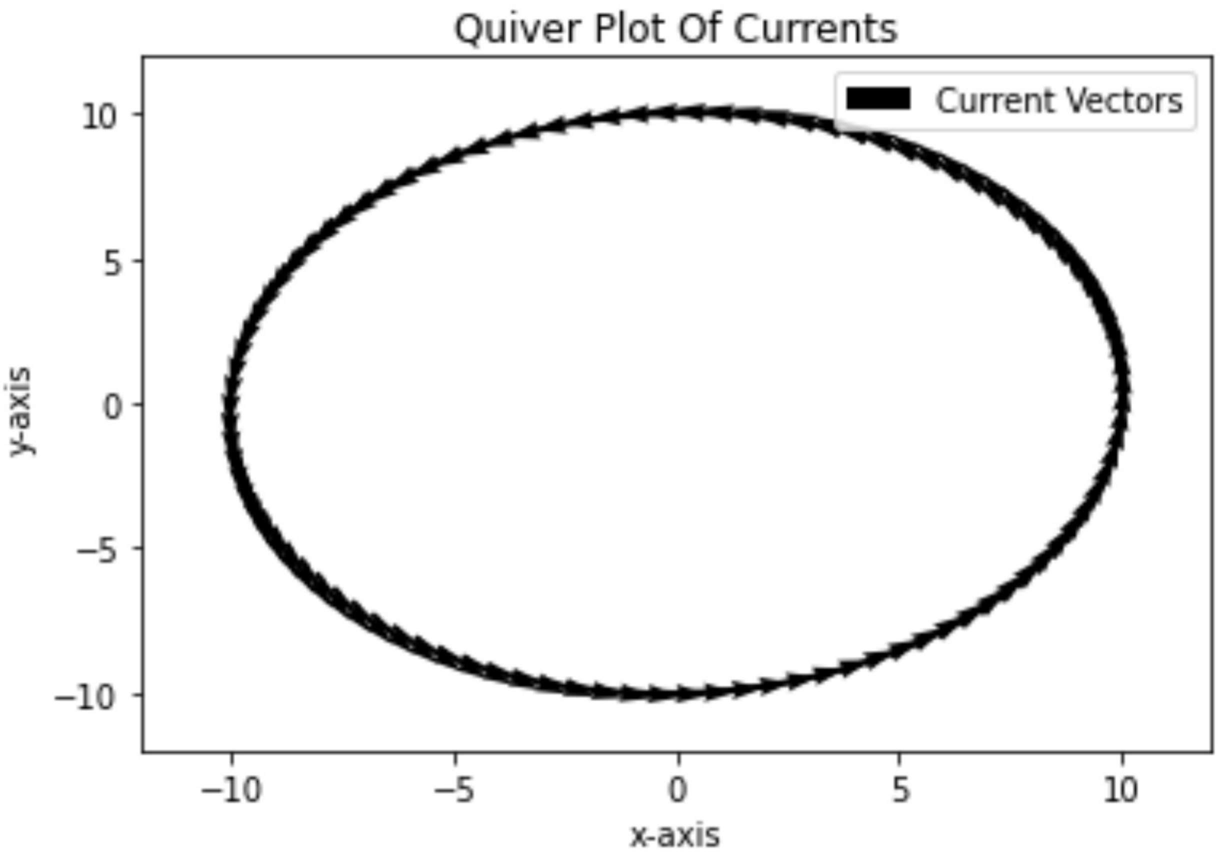
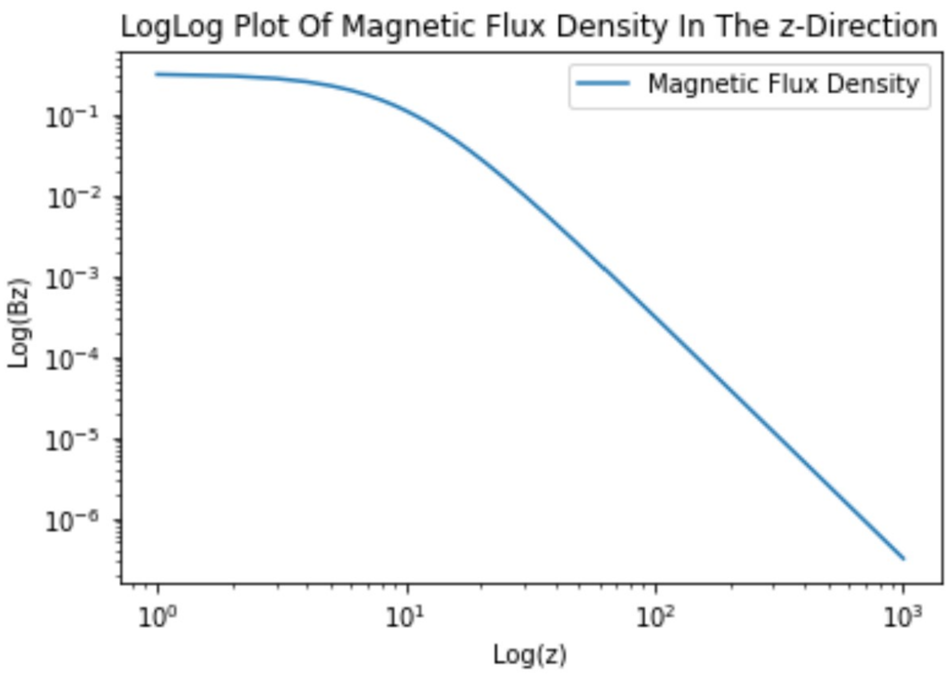


Figure 9: Current quiver plot in this case

10.2 Q9: Plotting the Magnetic Flux Density B

10.2.1 loglog Plot of Bz vs z



```
The fitted b,c only taking samples after the graph turns linear is:
b = -2.9964607171464435, c = 5.726576979789056

The fitted b,c taking all the samples into consideration is:
b = -2.826192056926662, c = 4.681473151581831
```

Figure 10: loglog plot of Bz vs z for Case 4

10.3 Q10: Fitting the field in a polynomial

We now fit the field into a polynomial of the form

$$B(z) = cx^b \tag{30}$$

The fitted value are given by:

- b = -2.826192056926662 ≈ -3
- c = 4.681473151581831

11 ANALYSIS OF ALL THE OUTPUTS

In both Case 1 and 3, we can clearly see that the output is of the order of **10^{-17}**$whichis\approx$ equal to '0'.

The reason for this is quite simple. In both cases, the current is **Circularly Symmetric** about the z-axis.

So when we analyse the Magnetic field at any point on the z-axis, the z components of the magnetic field due to circularly opposite points on the current carrying loop cancel out each other.

The reason the output is not exactly '0' is due to 2 reasons:

- We are using an approximate summation of an integral. We divide the loop into 100 parts and find the summation but to get even more accurate results we need to divide the loop into smaller parts.
- The second reason is that there may be errors in the calculation performed, or precision of the computer. This results in the output not being a perfect '0'.

In both cases, **We cannot really estimate the reason for why B_z falls at a rate close to -1**.

Where as in Case 2 and 4, the graph looks very smooth similar to the response of a low pass filter and the decay constants we obtain are ≈ -2 and -3.

In the case 4, we can see that the decay rate is -3, i.e,

$$|\vec{B}(z)| = cz^{-3} \tag{31}$$

11.1 Explanation of Case 4:

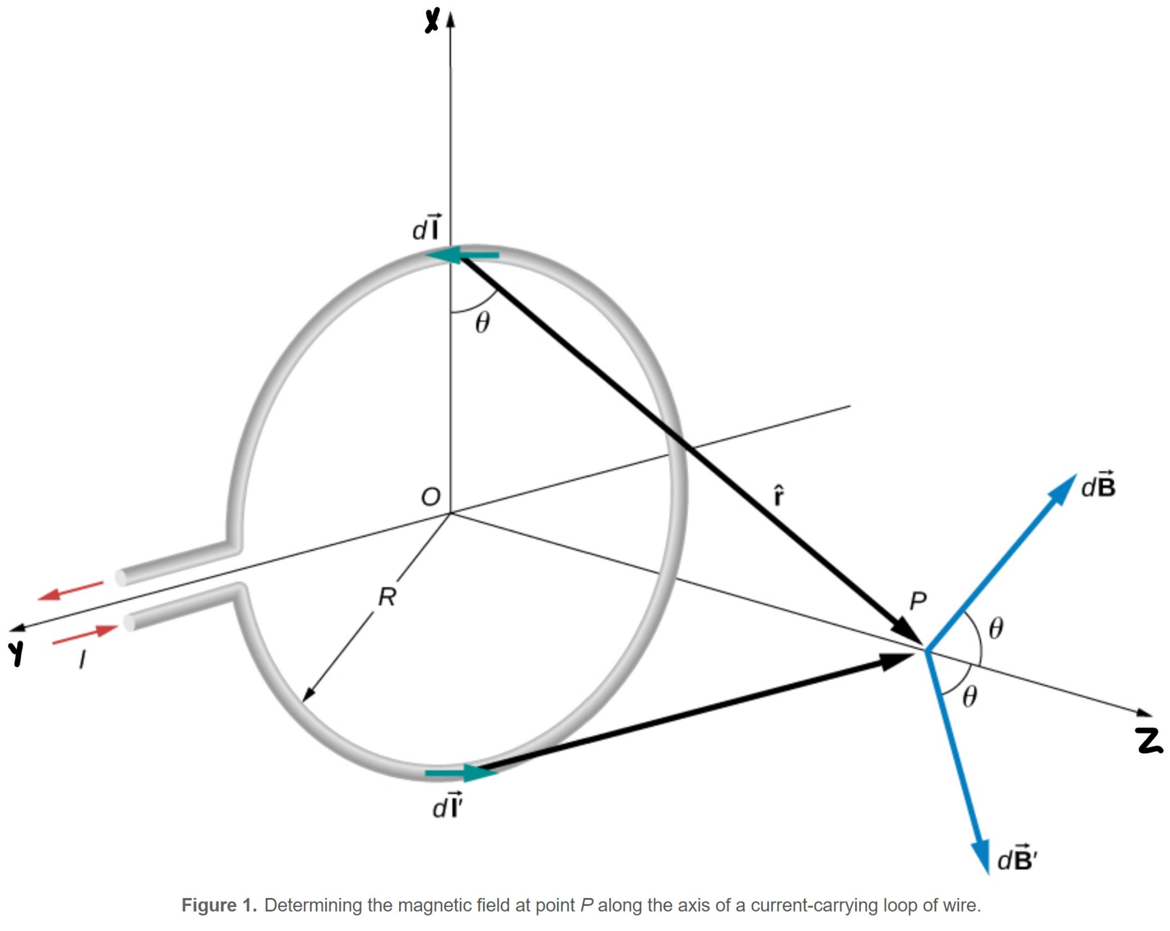


Figure 1. Determining the magnetic field at point *P* along the axis of a current-carrying loop of wire.

Figure 11: Case 4

We can use Biot-Savarts law to find the magnetic field due to a small conductor in the loop and then integrate it to find the final value.

$$d\vec{B} = \frac{\mu_o I d\vec{L} \times \hat{r}}{4\pi r^2} \tag{32}$$

Here, r is given by:

$$r = (z^2 + a^2)^{1/2}, \tag{33}$$

where a is the radius = 10cm.

After integrating, the final result we get is:

$$|\vec{B}(z)| = \frac{\mu_o I a^2}{2(z^2 + a^2)^{3/2}} \tag{34}$$

For z >> a,

$$|\vec{B}(z)| \approx \frac{\mu_o I a^2}{2(z)^3} \tag{35}$$

i.e,

$$|\vec{B}(z)| \approx c z^{-3} \tag{36}$$

11.2 Q11: Answering the questions

1. The $|\vec{B}(z)|$ we obtain in Case 1 does NOT fall off at the rate we expected.
2. The decay rate we would expect for a magneto static field is **-3** i.e Case 4 (or, it would be similar if u take Case 3 as in both Case 1 and 3, the currents are circularly symmetric.).
3. The difference comes from the fact that the currents in Case 1 are circularly symmetric whereas they are not in Case 4.

12 Conclusion

Through the use of vectorised code, we have obtained a numerical solution to find the magnetic field value of a point on the z-axis caused by a current carrying loop. Though it is not exact, it gives us an easy way to obtain all the necessary data efficiently and without us having to perform any of the complex calculations . It also allows us to plot and visualise their behaviour.