Endsemester Assignment

Lakshya J ee19b035

May 30th, 2021

Introduction

In this assignment we are asked to estimate the magnetic field produced by a current carrying ring along the z axis, which passes through the center of the ring using vectorized array methods.

Pseudo code

We are asked to write a pseudo code of how we plan to achieve the same.

```
$\delta\theta \leftarrow \frac{\pi}{100}$
$\phi \leftarrow linspace(\delta\theta,\ 2\pi - \delta\theta, 100)$
$\Delta x \leftarrow 1$
$\Delta y \leftarrow 1$
$x_{mesh} \leftarrow linspace(-\Delta x,\ \Delta x,\ 3)$
$y_{mesh} \leftarrow linspace(-\Delta y,\ \Delta y,\ 3)$
$z_{mesh} \leftarrow linspace(1,\ 1000,\ 1000)$
$Y, X, Z \leftarrow meshgrid(x_{mesh},\ y_{mesh},\ z_{mesh})$
$x \leftarrow a\cos(\phi)$
$y \leftarrow a\sin(\phi)$
$r\_ \leftarrow column\_stack((x, y))$
$\delta l \leftarrow \frac{2\pi a}{100}$
$\delta lx \leftarrow \delta l(-\sin(\phi))$
$\delta ly \leftarrow \delta l(\cos(\phi))$
$\delta l\_ \leftarrow column\_stack((\delta lx, \delta ly))$
function CALC($l$)
    $R_{ijkl} \leftarrow ((X - r\_[l,0])^2 + (Y - r\_[l,1])^2 + Z^2)^{0.5}$
    $exp\_term \leftarrow jkR_{ijkl}$
    $A_{xijk} \leftarrow \frac{\cos(\phi[l])dl\_[l,0](\exp(-exp\_term))}{R_{ijkl}}$
    $A_{yijk} \leftarrow \frac{\cos(\phi[l])dl\_[l,1](\exp(-exp\_term))}{R_{ijkl}}$
    return $A_{xijk}, A_{yijk}$
end function
$Ax_f \leftarrow zeros(Y.shape)$
```

```
$Ay_f \leftarrow zeros(Y.shape)$
**for** $i \leftarrow 0$ **to** 100 **do**
    $Ax_{\_}, Ay_{\_} \leftarrow$ **calc(i)**
    $Ax_f \leftarrow Ax_f + Ax_{\_}$
    $Ay_f \leftarrow Ay_f + Ay_{\_}$
**end for**
$B \leftarrow zeros(1000)$
$B \leftarrow (Ay_f[2,1,:] + Ax_f[1,2,:] + Ay_f[0,1,:] + Ax_f[1,0,:])/4$
=0
```

Q2

Here we are asked to create a matrix of a specific dimension to help us calculate the curl of the vector potential to find the magnetic field. Here is how I achieved the same.

```python
x_mesh = np.linspace(-delta_x, delta_x, 3)
y_mesh = np.linspace(-delta_y, delta_y, 3)
z_mesh = np.linspace(1, 1000, 1000)
Y,X,Z = np.meshgrid(x_mesh, y_mesh, z_mesh)
```

Q3

Here we are asked to break the loop into 100 sections and plot the current elements. I used both a 3D approach and a surface plot to visualize the length and direction of flow of the currents.
Here is the code I used.

```python
d_theta = (np.pi)/100
mu0 = 4*np.pi*(10**(-7))
phi = np.linspace(d_theta,2*np.pi-d_theta, 100)
I = ((4*np.pi)/mu0)*np.cos(phi)
x = a*np.cos(phi)
y = a*np.sin(phi)

# 3D PLOT
fig = plt.figure()
ax = fig.add_subplot(111, projection='3d')
ax.title.set_text('Plot of current elements at t=0')
ax.plot(x, y, I, 'bo', markersize = 5)
ax.plot(x, y, np.amin(I), 'r', markersize = 1)
ax.legend(['Variation of current magnitude', 'Wire loop'])
ax.grid(True)
plt.show()

# Creating a quiver plot for the currents
YY, XX = np.meshgrid(x, y, sparse =False)
I = ((4*np.pi)/mu0)*(YY/a)
I[np.where(np.sqrt(XX**2+YY**2)!=a)] = 0
Ix = -I*(XX)/a
Iy = I*(YY)/a
```

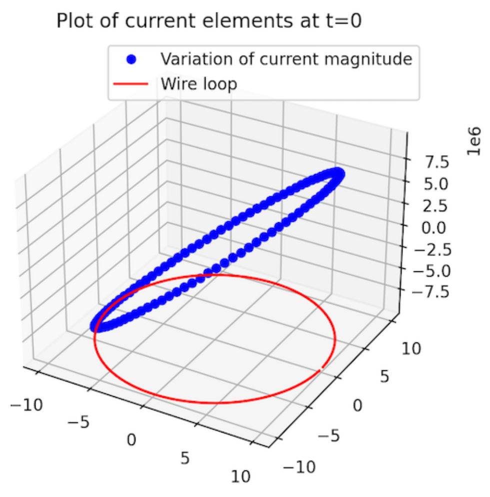

Figure 1: 3D PLOT of the current magnitude

```python
plt.figure()
plt.title("Surface plot of the currents")
plt.quiver(x, y, Ix, Iy)
plt.show()
```

Q4

Here I obtain the vectors $\vec{r'}$ and $\vec{dl}$ where l is the index for the section of the loop.

```python
x = a*np.cos(phi)
y = a*np.sin(phi)
r_ = np.column_stack((x,y))
dl = (a*2*np.pi)/100
dlx = dl*(-np.sin(phi))
dly = dl*(np.cos(phi))
dl_ = np.column_stack((dlx, dly))
```

Q5

Now we move on to defining **calc(l)** which calculates R_{ijkl} for a given loop segment l and returns the x and y components of the vector potential $\vec{A_{ijk}}$.

```python
def calc(l):
  # Finding Rijkl matrix for each segment l
```

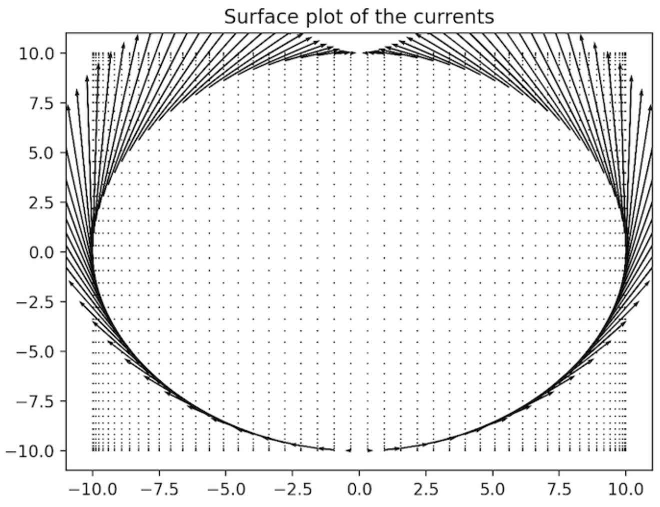

Figure 2: Surface plot of the currents

```
    Rijkl = np.sqrt(np.multiply(X-r_[l,0], X-r_[l,0]) + np.multiply(Y
      -r_[l,1], Y-r_[l,1]) + np.multiply(Z,Z))
    # Declaring a comlex variable jk
    jk = complex(0, 0.1)
    # Multiplying the complex term with Rijkl
    exp_term = jk*Rijkl
    # np.divide achieves element-wise division
    # Finding the x component of the potential
    Axijk = np.divide((np.cos(phi[l])*dl_[l,0]*(np.exp(-exp_term))),
      Rijkl)
    # Finding the y component of the potential
    Ayijk = np.divide((np.cos(phi[l])*dl_[l,1]*(np.exp(-exp_term))),
      Rijkl)
    # Returning the x and y components of the potential
    return Axijk, Ayijk
```

I have vectorized the function by using R_{ijkl} as a matrix and then use the values requires for the calculation later on in the code.
I have also used broadcasting techniques for the matrix addition.
For the case of taking the absolute value of the current, I use *abs* to make the value returned by $np.\cos(\phi)$ absolute.

```
Axijk = np.divide((abs(np.cos(phi[l]))*dl_[l,0]*(np.exp(-exp_term))
    ), Rijkl)
Ayijk = np.divide((abs(np.cos(phi[l]))*dl_[l,1]*(np.exp(-exp_term))
    ), Rijkl)
```

Q6, Q7

I call the function **calc(l)** by looping 100 times, which corresponds to the number of sections made in the ring.
I am using a loop of 100 iterations here because the value of the magnetic field at each of these points should be calculated separately depending on the part of the wire which contributes to it. As we are considering the loop to me made up of smaller 100 segments, the magnetic field is a resultant of the contribution of each of the individual segments.

```
Ax_f = np.zeros(Y.shape)
Ay_f = np.zeros(Y.shape)
# for loop for each small length of the ring
for i in range(100):
  # Assigning Ax and Ay the values retured by calc
  Ax_, Ay_  = calc(i)
  # Adding the returned values to the final matrices
  Ax_f = np.add(Ax_f, Ax_)
  Ay_f = np.add(Ay_f, Ay_)
```

Q8

Now we move on to computing the value of the magnetic field $\vec{B}$. This will be along the z axis.

```
# Using matrix manipulation to find the final B value
B = (Ay_f[2,1,:]-Ax_f[1,2,:]-Ay_f[0,1,:]+Ax_f[1,0,:])/4
```

There is another method to achieve the same using the command *where* from the *numpy* library.

```
x1 = np.where((Y==0) & (X==1))
y1 = np.where((X==0) & (Y==1))
x1_ = np.where((Y==0) & (X==-1))
y1_ = np.where((X==0) & (Y==-1))

dAy = Ay_f[x1]
dAx = Ax_f[y1]
dAy_ = Ay_f[x1_]
dAx_ = Ax_f[y1_]

B = (dAy - dAx - dAy_ + dAx_)/4.0
```

Both the methods give the same results.

Q9

Now we obtain the plots of the same.
Since the current is symmetric around the center of the ring, the net magnetic field will be zero. However I have plotted this case as well, where the numbers are due to computer errors caused due to finite precision. Hence when we use

the absolute values of the current, we get a viable graph. I have plotted both the cases here.
First two imaged are the results I obtained without taking absolute values of current. The next two figures are obtained when absolute values of the current is considered. It can be noticed that in the case where we take absolute values, we get a more readable graph.

```python
# Declaring the figure where the plot is to appear
plt.figure(1)
# Declaring the title of the plot
plt.title("Loglog Plot of B")
# Declaring the type of plot, in this case it is loglog
plt.loglog(np.linspace(1,1000,1000), np.abs(B), 'o')
# Labelling the x axis
plt.xlabel(r"log(z) $\longrightarrow$")
# Labelling the y axis
plt.ylabel(r"log(|B|) $\longrightarrow$")
# Option for making grid appear for readability
plt.grid(True)
# Finally plot is displayed until exited
plt.show()

# Declaring the figure where the plot is to appear
plt.figure(2)
# Declaring the title of the plot
plt.title("Plot of B")
# Declaring the type of plot
plt.plot(np.linspace(1,1000,1000), B, 'o')
# Labelling the x axis
plt.xlabel(r"z $\longrightarrow$")
# Labelling the y axis
plt.ylabel(r"B $\longrightarrow$")
# Option for making grid appear for readability
plt.grid(True)
# Finally plot is displayed until exited
plt.show()
```

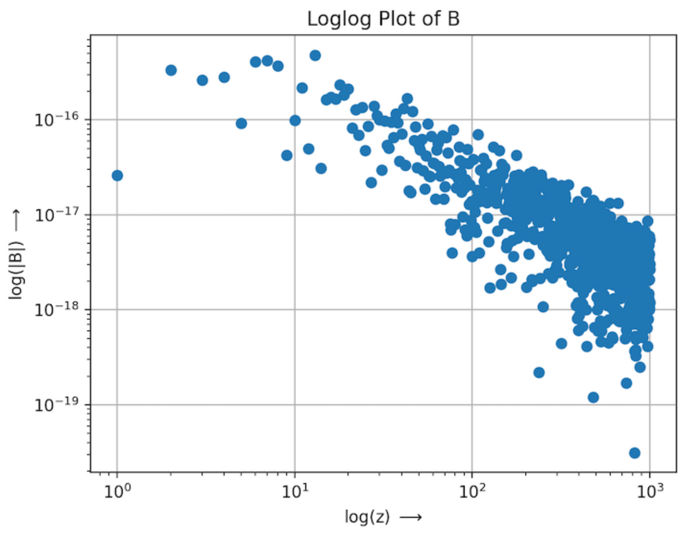

Figure 3: Loglog plot of the Magnetic field

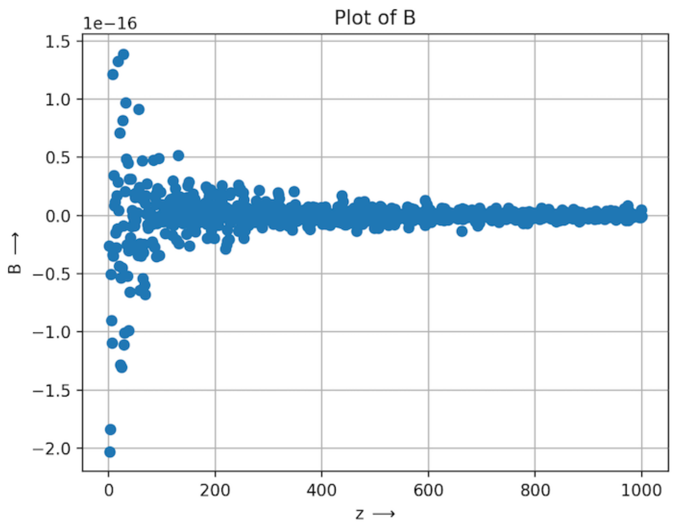

Figure 4: Normal plot of the magnetic field

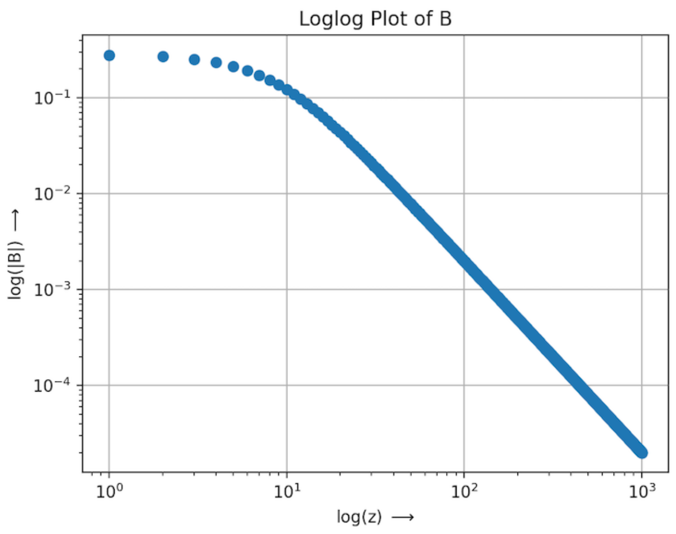

Figure 5: Loglog plot of the Magnetic field, taking absolute values of the current

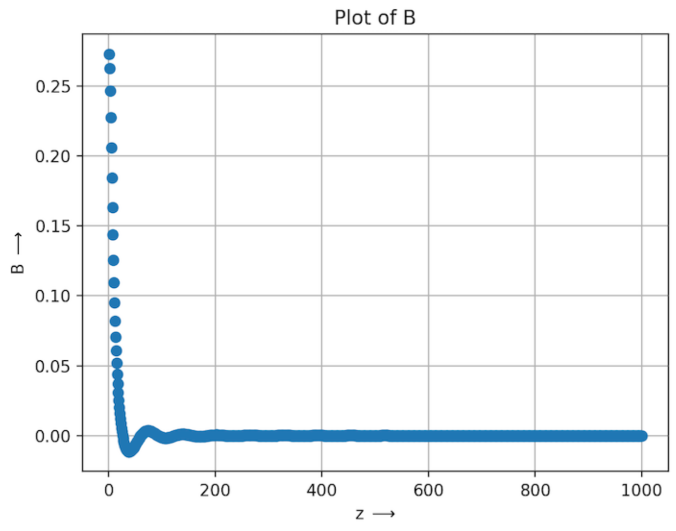

Figure 6: Normal plot of the Magnetic field, taking absolute values of the current

Q10

Now we try to fit the obtained magnetic field. I try to fit the magnetic fields obtained in both cases and fit them separately.
I use the least squares method as for fitting the curve from the library *numpy.linalg*. I take the first argument returned as the answer to the fitting question. I later display the final result in each case in the output window.

```
# Declaring z array
z = np.linspace(1,1000,1000)
# Fing the log of z for least squares manipulation
ln_z = np.log(z)
# Finding the log of B for least squares manipulation
ln_B = np.log(np.abs(B))
# Declaring a column of ones which represents the constant log c
# multiplied to it
col1 = np.ones(1000)
# Stacking the columns side by side to create the least squares
    matrix
A = np.column_stack((col1, ln_z))
# Applying the lstsq method from linalg and taking the first
    argument
lsq = np.linalg.lstsq(A, ln_B, rcond=None)[0]
# Taking the exponent value for finding c
c = np.exp(lsq[0])
# b is used as is as it was a part of z^b
b = lsq[1]
# Finding the estimate using the constants found
B_est = c*(np.power(z, b))
```

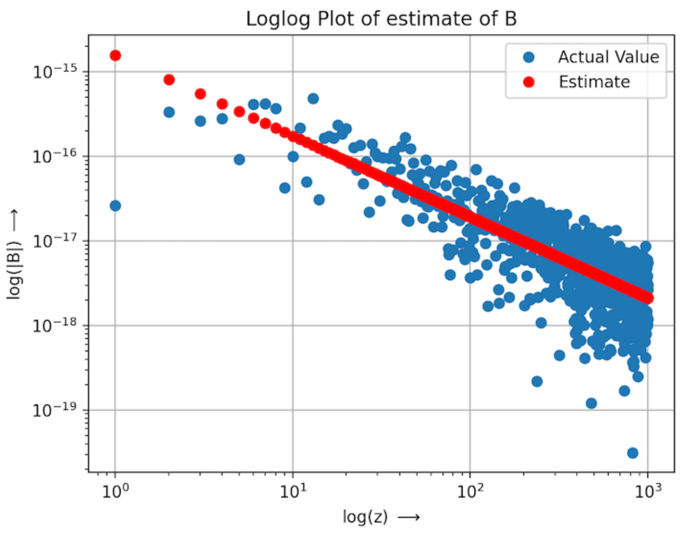

Figure 7: Fitting loglog plot of the Magnetic field

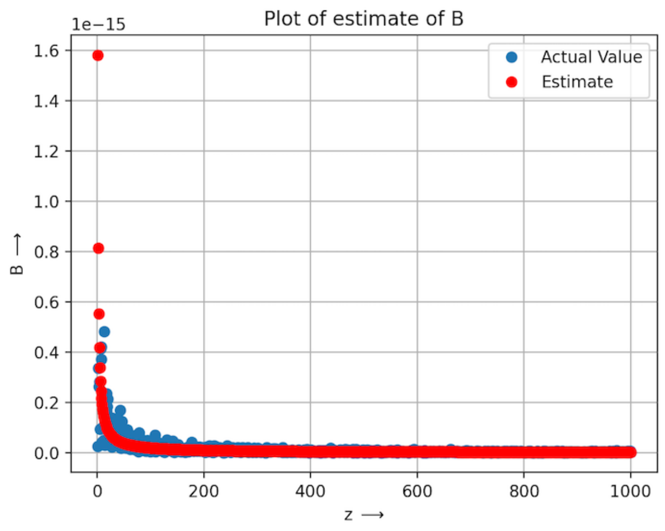

Figure 8: Fitting Normal plot of the magnetic field

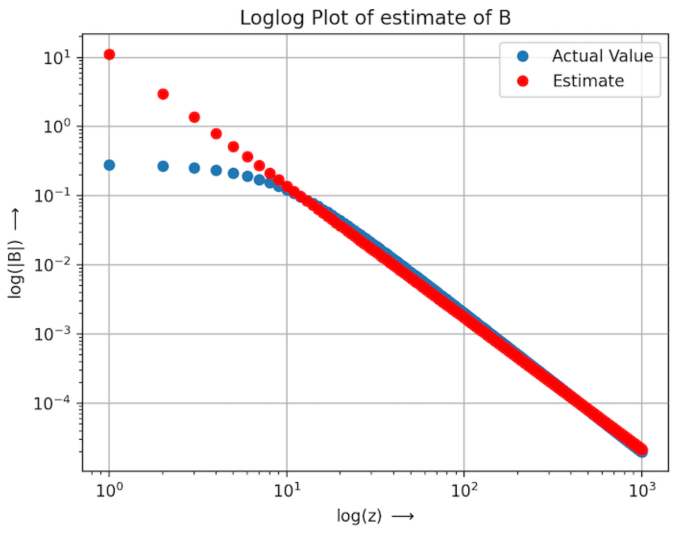

Figure 9: Fitting loglog plot of the Magnetic field, taking absolute values of the current

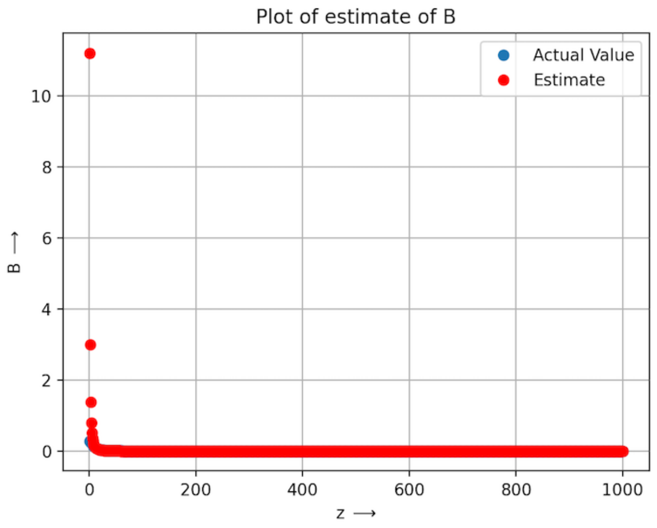

Figure 10: Fitting normal plot of the magnetic field, taking absolute values of the current

Q11

This is the result I obtained:
Case I: This is where I don't take the absolute values of the current into consideration.
The rate at which B falls off is -0.957351
Case II: This is where I take the absolute values of the current into consideration.
The rate at which B falls off is -1.905741
In case of a circular wire carrying constant current the magnetic field falls of z^{-3} for values of z larger than the radius of the ring.
In the first case we get this decay mainly due to finite precision of computation of a computer system.
In the second case this difference is coming in due to the fact that the magnetic field $\vec{B}$ depends on $exp(jkR_{ijkl})$. Since the magnetic field is also dependent on $cos(\phi)$ due to the current being space variant, such a decay is possible.

Conclusion

From this assignment came to appreciate how Python can be used to simulate real world applications by reducing computational time to a great extent by using vectorized methods.
I enjoyed tackling these problems through the various assignments throughout the whole course. I can say I have become more confident in using matplotlib, Python in general and the introduction to Overleaf was definitely an added bonus.
Thank You, Prof. HSR for such a well structured course!!