Radiation from a Loop Antenna

End-Semester Examination

EE2703 - Applied Programming Lab

Abhigyan Chattopadhyay

EE19B146

25th May 2021

Contents

1 Introduction **2**

2 Setting up the Problem **3**

3 Steps **4**

4 Pseudocode **4**

5 Code **6**

5.1 Imports 6

5.2 Question 2 6

5.3 Question 3 and 4 6

5.4 Question 5 7

5.5 Question 6 8

5.6 Question 7 8

5.7 Question 8 8

5.8 Question 9 8

5.9 Question 10 9

5.10 Extra: Taking $I = |\cos(\phi)|$ 10

6 Question 11 - Inferences and Conclusion **13**

6.1 Sources of Error 13

7 Final Conclusions **13**

8 References **14**

1 Introduction

Before you start reading the rest of this document, I highly recommend that you kindly read the code first. I have spent a lot of time getting the right code and vectorizing every operation so that only a single summation is required and not a single for-loop anywhere.

I hope that you enjoy reading this report.

- Abhigyan C.

2 Setting up the Problem

A long wire carries the following current through a loop of wire:

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

ϕ is in (r, ϕ, z) coordinates here. The wire is on the x-y plane and centered at the origin. The radius of the loop is 10 cm and is also equal to 1/k = c/ω, meaning that the circumference is λ.

We need to compute and plot the magnetic field $\vec{B}$ from z = 1cm to 1000cm, plot it and fir the data to $|\vec{B}| = cz^b$.

This involves the computation of the vector potential:

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

Here, $\vec{R} = \vec{r} - \vec{r'}$ and k = $\omega/c = 0.1$. $\vec{r}$ is the point where we want the field, and $\vec{r'} = a\hat{r'}$ is the point on the loop.

As we are doing this discretely, we can write this as the following sum:

$$\vec{A}_{ijk} = \sum_{l=0}^{N-1} \frac{cos(\phi_l')\hat{\phi}\exp(-jkR_{ijkl})d\vec{l'}}{R_{ijkl}}$$

Here, $\vec{r}$ is at r_i, ϕ_j, z_k and $\vec{r'}$ is at $a\cos\phi_l'\hat{x} + a\sin\phi_l'\hat{y}$. This equation is valid at all points in space, and is to be summed over all the current elements in the loop.

Once we have A, we can obtain B as

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

And, when we take the discrete form of this, we get:

$$\frac{A_y(\Delta x, 0, z) - A(0, \Delta y, z) - A_y(-\Delta x, 0, z) + A_x(0, -\Delta y, z)}{4\Delta x \Delta y}$$

3 Steps

1. Create a mesh of values of x, y and z where we want to calculate the Magnetic field
2. Create and sample the current at the points on the loop carrying current
3. Plot the current elements on the wire
4. Calculate the value of the A field at each of the points of interest and sum over them.
5. Calculate the curl of A to find the magnetic field, B.
6. Plot B's variation with z
7. Fit B using the Least Squares algorithm and find the values of b and c in the approximation $|\vec{B}| = cz^b$.

Note: I have not taken -1,0,1 for x and y and instead taken an off-centered set of values so that the magnetic field values don't completely cancel out.

4 Pseudocode

Pseudocode written in simple English Instructions:

```
x = linearspace(-0.99,1.01,3)
y = linearspace(-1.01,0.99,3)
z = linearspace(1,1000,1000)

X,Y,Z = (x,y,z)

a = 10
N = 100
k = 0.1
phi = (linearspace(0,2 pi,N+1).remove(2 pi)
xLoop = a*cos(phi)
yLoop = a*sin(phi)
rPrime = matrix(xLoop,yLoop)

currMag = 10^7 * cos(phi)
dlPrime = 2pi/N * matrix(-yLoop,xLoop)

quiverPlot(rPrime, currMag * dlPrime * N/2pi)

r = array(X,Y,Z)
function calc(int l){
    rWithZeros = matrix(rPrime, zeros([N,1]))
    # adding the z coordinate, which is a row of zeros. r_prime_wz stands for "r
    ↪  prime with zeros"

    r1 = tile(r,(N,1,1,1)).reshape((N,3,3,3,1000))
    # r1 is a rearranged and repeated version of r. First, we tile r 100 times
    ↪ so that we have 100 different copies of it to handle each of the values of
    ↪  l. Then we reshape it into a 5D array with 100 copies of the X,Y,Z arrays
    ↪ , each of which are 3,3,1000 arrays. Hence, its shape is (100,3,3,3,1000)

```

```
    R = r1 - rWithZeros.reshape((N,3,1,1,1))
    # Before we start subtracting, we reshape r prime with zeros to a (100,3,1,1,1) array so that it can be subtracted from r1 using broadcasting

    modR = norm(R,axis=1)
    # Next, we take the norm of it along the first axis (i.e. the axis that has the individual X,Y,Z components). And therefore we get a 100,3,3,1000 array, and return its l'th element. This is the set of distances of the l'th section of the loop from all the 3x3x1000 points in space.

    tempval = currMag.reshape(N,1,1,1)/($10^7$)*exp(-1j*k*modR)/modR

    Aijkl = array(tempval*dlPrime.firstColumn.reshape(N,1,1,1), tempval*dlPrime.secondColumn.reshape(N,1,1,1))

    return Aijkl[l] if l is not None else return Aijkl
}

Aijk = sum(calc(),axis=1)

# Take the sum to get Aijk

Ax = Aijk[0]
Ay = Aijk[1]

# Separate the X and Y components

B = (Ay[2,1,:] - Ax[1,2,:] - Ay[0,1,:] + Ax[1,0,:])/4

# Find the magnetic field B
logLogPlot(z,abs(B))

b,c = leastSquaresFit(B,z)

BFit = c*(z^b)
logLogPlot(z,BFit)

print(b,c)
```

5 Code

The entire working code is given here in the order in which it appears in the file, with the output that comes out, if any.

5.1 Imports

Only 2 modules are required for this assignment, NumPy and MatPlotLib. NumPy is used for the Least Squares, which is the same as the one found in SciPy.

```
import numpy as np
import matplotlib.pyplot as plt
```

5.2 Question 2

Now, we will set up the volume which is a 3x3x1000 mesh, separated by 1cm each.

```
x = np.linspace(-0.99,1.01,3,dtype=np.longdouble)
y = np.linspace(-1.01,0.99,3,dtype=np.longdouble)
z = np.linspace(1,1000,1000,dtype=np.longdouble)

X,Y,Z = np.meshgrid(x,y,z,indexing='ij') # meshgrid returns a set of 3 3D arrays that helps to run
# the functions over the entire grid without having to index each of them individually
```

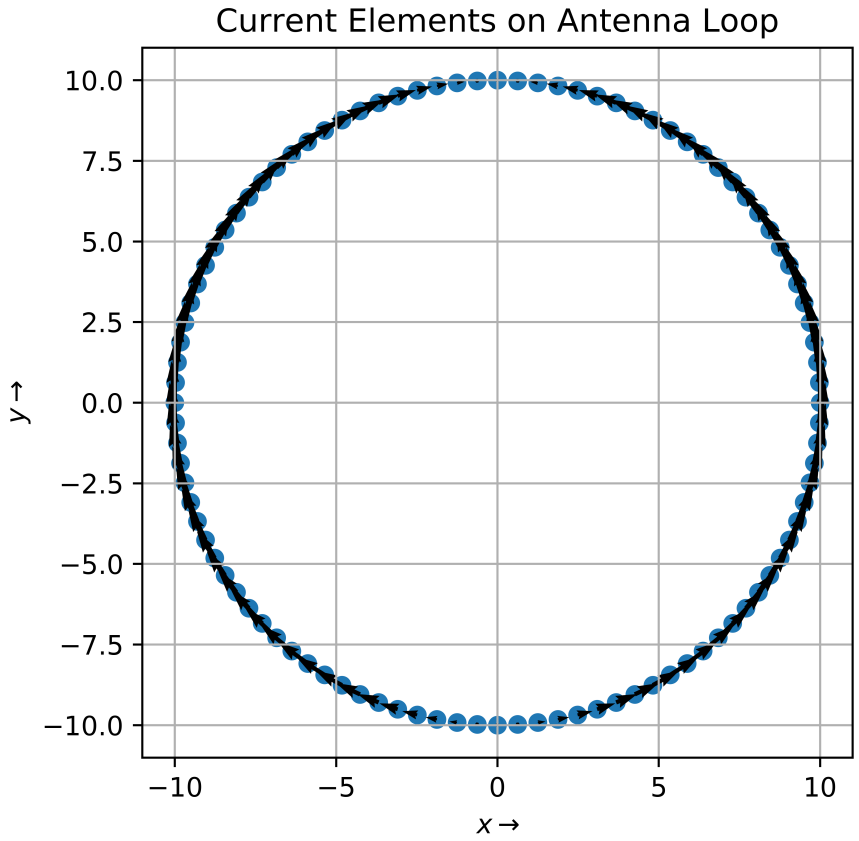
5.3 Question 3 and 4

First, we will be generating the points on the loop antenna:

```
a = 10 # radius of the loop
N = 100  # number of loop sections
k = 0.1  # k = w/c = 1/10 = 0.1 rad/cm
phi = np.linspace(0,2*np.pi,N+1,dtype=np.longdouble)[:-1]
exs = a*np.cos(phi) # x values
wys = a*np.sin(phi) # y values

# Question 4: The r' vector

r_prime = np.c_[exs,wys] # converting the 2 individual arrays to a 100x2 array

# Question 3 contd.
# Part 2: Finding the current elements on the loop
# Now finding the current magnitudes and directions at these points

curr_mag = 1e7*np.cos(phi)

# Question 4 contd.
# Obtaining dl' = [-r*d(phi)sin(phi)x-hat, r*d(phi)cos(phi)y-hat]:
# Hence dl' = d(phi)*[-y,x] for each [x,y], and d(phi)=2pi/100

dl_prime = 2*np.pi/N * np.asarray([-wys,exs],dtype=np.longdouble).T

# Current will be curr_mag * dl

```

```
plt.figure(1)
plt.scatter(*(r_prime.T))
plt.quiver(*(r_prime.T),*(curr_mag*dl_prime.T*N/(2*np.pi)))
plt.gca().set_aspect('equal') # plots distances equally on both axes
plt.grid(True)
plt.title('Current Elements on Antenna Loop')
plt.xlabel(r'$x\rightarrow$')
plt.ylabel(r'$y\rightarrow$')
plt.savefig('images/fig1.png',dpi=1000)
plt.show()
```

The plot obtained is below:



5.4 Question 5

We begin by adding the z coordinate to r_prime, which is a row of zeros to get "r prime with zeros".

r1 is a rearranged and repeated version of r

First, we tile r 100 times so that we have 100 different copies of it to handle each of the values of l.

Then we reshape it into a 5D array with 100 copies of the X,Y,Z arrays, each of which are 3,3,1000 arrays. Hence, its shape is (100,3,3,3,1000)

Before we start subtracting, we reshape "r prime with zeros" to a (100,3,1,1,1) array so that it can be subtracted from r1 using NumPy broadcasting.

Finally, we take the norm of it along the first axis (i.e. the axis that has the individual X,Y,Z components). And therefore we get a 100,3,3,1000 array, and return its l'th element.

This is the set of distances of the lth section of the loop from all the 3x3x1000 points in space.

```
r = np.array((X,Y,Z))

def calc(l):
    r_prime_wz = np.c_[r_prime,np.zeros([N,1])]
    r1 = (np.tile(r,(N,1,1,1)).reshape((N,3,3,3,1000)))
    R = r1 - r_prime_wz.reshape((N,3,1,1,1))
    return np.linalg.norm(R,axis=1)[l]
```

5.5 Question 6

Here we begin in the same way as for Question 5, but add a few steps and changes to the function.

Notice that the function now takes a default value of l as None, so that we can leave it empty so as to get all the values at once.

We find the value of Aijkl as per the equation and return all the values if l is unspecified, else only the value for the specified l value.

```
def calc(l=None):
    r_prime_wz = np.c_[r_prime,np.zeros([N,1])]
    r1 = (np.tile(r,(N,1,1,1)).reshape((N,3,3,3,1000)))
    R = r1 - r_prime_wz.reshape((N,3,1,1,1))
    modR = np.linalg.norm(R,axis=1)
    # Same as earlier version till here
    tempval = curr_mag.reshape(N,1,1,1)/(1e7)*np.exp(-1j*k*modR)/modR
    Aijkl = np.asarray([tempval*dl_prime[:,0].reshape(N,1,1,1),tempval*dl_prime
    ↪ [:,1].reshape(N,1,1,1)])
    return Aijkl[l] if l else Aijkl
```

5.6 Question 7

Here, we are finding Aijk from the individual terms calculated in the above formula. It is a one liner, but we need to be careful to only add along the first axis, i.e. the axis where we have the variation in l.

```
Aijk = np.sum(calc(),axis=1)
```

5.7 Question 8

First, we separate the x and y parts of Aijk, and the we calculate it as per the curl formula we found earlier.

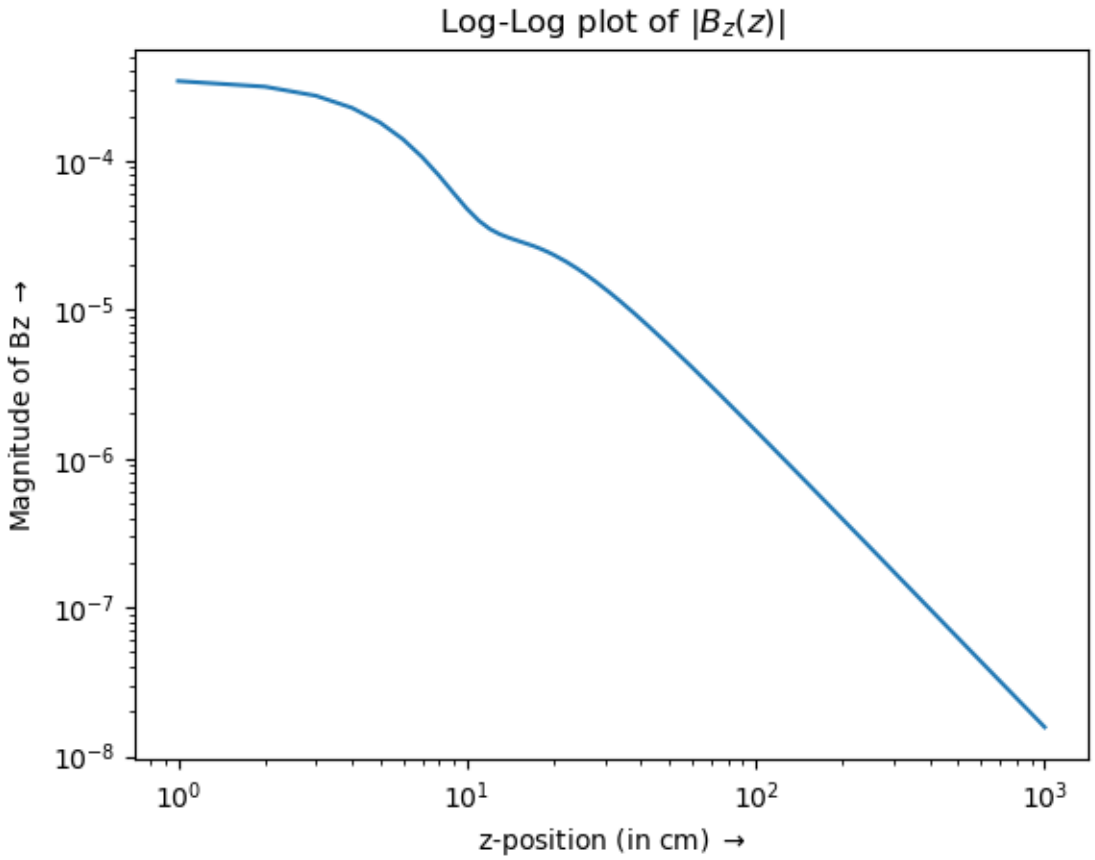
```
Ax = Aijk[0]
Ay = Aijk[1]

B = (Ay[2,1,:] - Ax[1,2,:] - Ay[0,1,:] + Ax[1,0,:])/4
```

5.8 Question 9

Here, we will plot B_z vs z in a log-log plot

```
plt.figure(2)
plt.loglog(z,abs(B))
plt.xlabel(r"z-position (in cm) $\rightarrow$")
plt.ylabel(r"Magnitude of Bz $\rightarrow$")
plt.title(r"Log-Log plot of $|B_z(z)|$")
plt.savefig('images/fig2.png',dpi=100)
plt.show()
```

The plot obtained is below:



5.9 Question 10

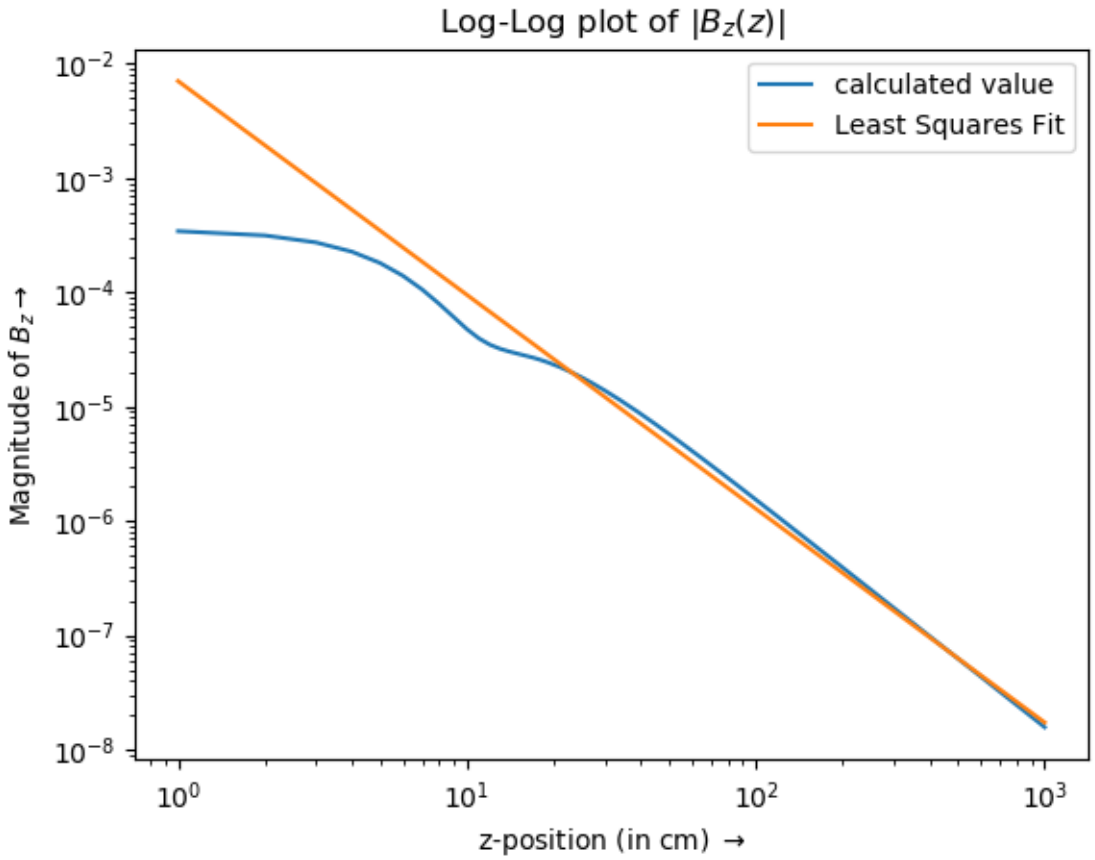
Now, we will be Fitting the obtained values to $|B| = cz^b$ and plotting it too.

We will also print the values of b and c thus obtained.

```
p = np.c_[np.ones(len(z)),np.log(z)]
log_c,b = np.linalg.lstsq(p,np.log(np.abs(B)),rcond=None)[0]
c = np.exp(log_c)
B_fit = c*(z**b)

plt.figure(3)
plt.loglog(z,np.abs(B),label="calculated value")
plt.loglog(z,np.abs(B_fit),label="Least Squares Fit")
plt.xlabel(r"z-position (in cm) $\rightarrow$")
plt.ylabel(r"Magnitude of $B_z \rightarrow$")
plt.title(r"Log-Log plot of $|B_z(z)|$")
plt.legend()
```

```
plt.savefig('images/fig3.png',dpi=100)
plt.show()

print(b,c)

log_c1,b1 = np.linalg.lstsq(p[100:],np.log(np.abs(B))[100:],rcond=None)[0]
c1 = np.exp(log_c1)
print(c1,b1)

log_c2,b2 = np.linalg.lstsq(p[250:],np.log(np.abs(B))[250:],rcond=None)[0]
c2 = np.exp(log_c2)
print(c2,b2)
```

The graph obtained is below:



The values of b and c printed are:

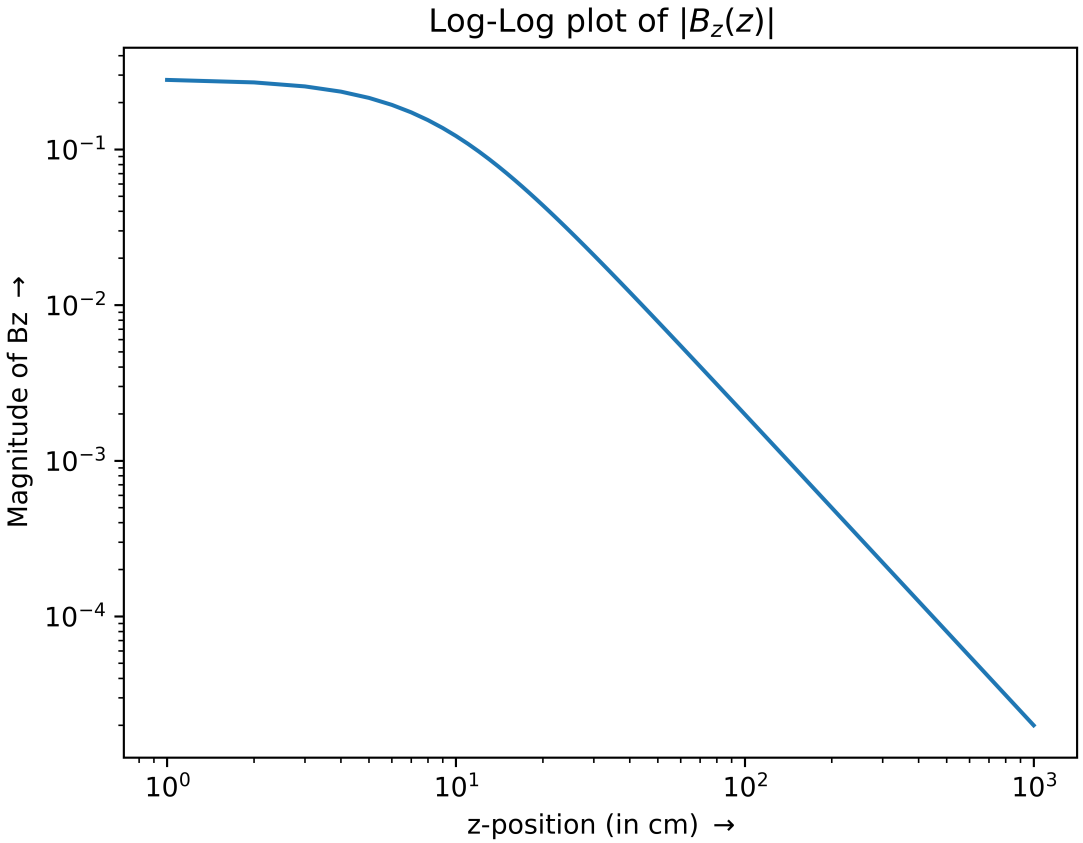
```
-1.8704814017152471 0.007053315946927306
-1.9952701214583952 0.01522597089568533
-1.9982888926250073 0.015525175941160424
```

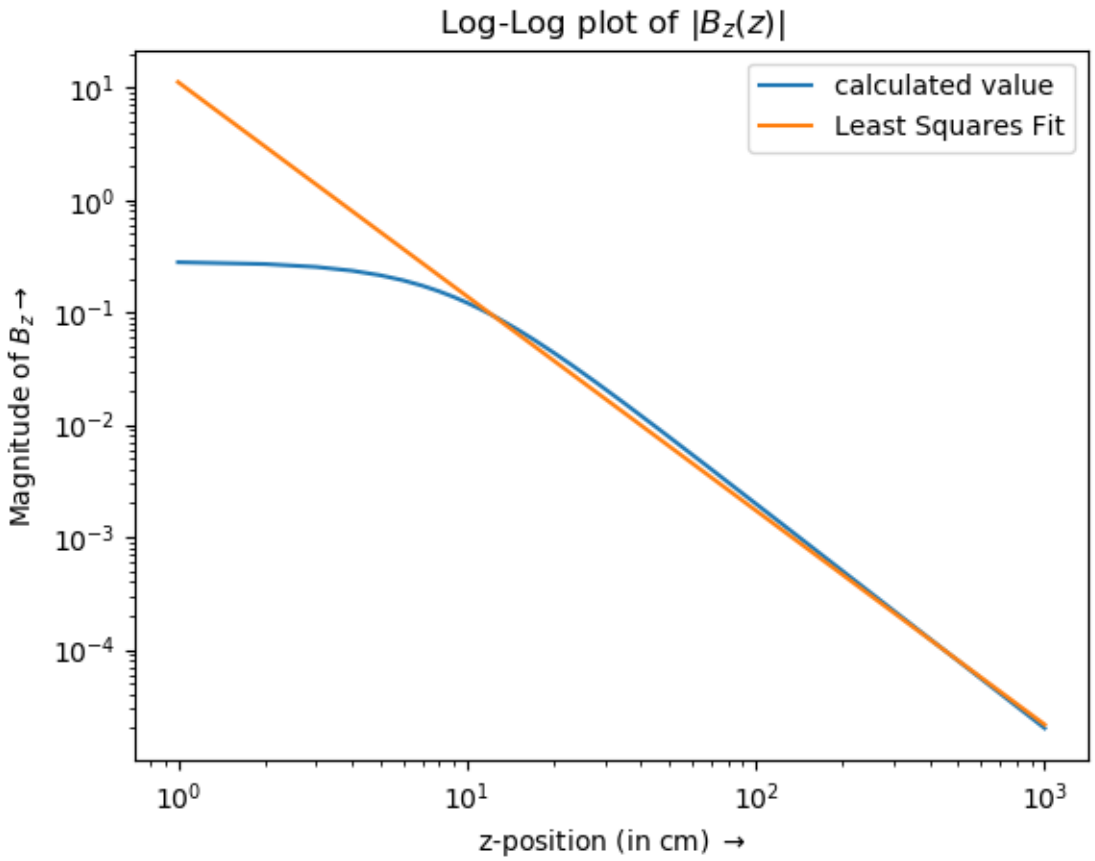
5.10 Extra: Taking $I = |\cos(\phi)|$

Here, `curr_mag` is replaced as `abs(np.cos(phi))`.

```
...
curr_mag = 1e7*abs(np.cos(phi))
...
```

This results in the following graphs:





This concludes the coding part.

6 Question 11 - Inferences and Conclusion

Magnetic field of a static current carrying circular loop along z-axis falls of as z^{-3} for large values of z. This is the expression of the magnetic field produced by a current carrying loop:

$$\frac{\mu_0 I a^2}{2(a^2+z^2)^{3/2}}\hat{k}$$

Because of precision and finite number of points and due to the time varying nature of the magnetic field, we have a plot that decays as approximately $r^{-1.87}$.

In fact, if we take individual parts of z and find the values of b and c obtained, we get b approaching -2 as we move away from the origin.

This is taken off the z axis, since we have taken x and y sampled at [-0.99,1.01] and [-1.01,0.99] respectively.

6.1 Sources of Error

There are various sources of error including:

1. the finite number of points
2. the precision of the bits (although np.longdouble was chosen to reduce that error)
3. the curl formula is not exact but approximate

7 Final Conclusions

1. Due to the symmetric nature of the current in the loop, if we try to find the Magnetic field at the center of the loop, it turns out to be completely zero, and we only get some residual noise as our plot.
2. If we evaluate the magnetic field off the axis, we get a magnetic field that decays as $z^{-1.87}$ on average, and reaches -2 as we move away from the origin.
3. If we use $|\cos(\phi)|$ as our current, then it is almost constant for lower values of z, and falls of as z^{-2} for larger values of z.

8 References

Some of these links were used extensively while coding this assignment

1. NumPy Reshape
2. NumPy Broadcasting
3. NumPy Summation
4. NumPy Meshgrid
5. NumPy and SciPy Least Squares