Endsem, Magnetic Field

Pranav Phatak, EE19B105

May 30, 2021

Objective

In this assignment, we analyse Magnetic Field on the Z-axis for current carrying wire loop of radius 10 centred at origin in the X-Y plane. For this purpose, we will use the fact that B is given by Curl of A where A is the Magnetic Potential Vector.
Note: I have used I proportional to absolute(cos(phi)) since due to symmetry of the system in I proportional to cos(phi), Bz could turn out to be 0 in static case (time independant current case), so for analysis purposes a slight modified version of the question has been solved

Physics and Maths to solve for B

The general equation for Magnetic Potential Vector for a time varying current is as follows using the Green's Theorem term $e^{-jkR_{ijkl}}$:

$$\vec{A}(r,\phi,z) = \frac{\mu_0}{4\pi}\int \frac{I(\phi)\hat{\phi}e^{-jkR}ad\phi}{R} \tag{1}$$

In our case, since the current is given by : $I = \frac{4\pi}{\mu_0}\cos(\phi)\exp(j\omega t)$ so the the integral can be reduced to the following summation:

$$A_{ijkl} = \sum_{l=0}^{N-1} \frac{\mid cos(\acute{\phi_l}) \mid \exp^{-jkR_{ijkl}} \vec{d\acute{l}}}{R_{ijkl}} \tag{2}$$

Using this we calculate $\vec{B}$ as

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

Note: In my code, I have solved for a time dependant current (dynamic) as well a static one, so in the calculations for the dynamic current case, we will have a green's theorem term e^{-jkR} in numerator and not in the one for the static current case.
Also, instead of taking grid as -1,0,1 will take one as (-10^{-5},0,10^{-5}) for better calculation of derivative near Z-axis.

Pseudo-Code

The logic used in the code goes as follows:

Step (i) Making phi,r,I,dl matrices

Make phi linspace by dividing (0,2π) into N parts
Make the rl vector using [r,phi,z] corresponds to [rcos(phi),rsin(phi),z]
Make the I vector, dl vector similarly
Since the magnitude of I is high because of $1/\mu$ factor, scale it up by same for easier quiver plot implementation, plot the graphs for current elements and quiver

Step (ii) Make a meshgrid

Make a meshgrid corresponding to 3,3,1000 points. X and Y go from 10^{-5} to 10^{-5}, Z goes from 1 to 1000
i)Make r of size (3,3,1000) to solve for A using calc(l) and for loop approach
ii)Make r' by copying r 100 times to solve for A using calc(all) and vectorization approach

Step (iii) Define calc(l), calc(all)

Calc(l) gives the norm of the vector r-rl at any specified value of l
Calc(all) gives the norm of the vector r'-rl for all values (It's shape is (3,3,1000,100))

Step (iv) Solving for A

I have solved the same using 2 approaches to get the value of A i.e the Magnetic Potential Vector:
i)For Loop approach to get A: Make empty A(loop) lists for x,y; append the value of dA at each value of l as given by the formula above, summate over the values finally to get final value of Ax, Ay for each of the (3,3,1000) points.
ii)Vectorization approach to get A: Since we are doing vectorization, define A as sum of the dA over l found by using r' matrix, since size of dA is (3,3,1000,100); the size of A is (3,3,1000) as expected.

Step (v) Taking Curl of A to get B

Since we have that:

$$\begin{aligned}\vec{B} &= \nabla \times \vec{A} \\ B_z &= \tfrac{dA_y}{dx} - \tfrac{dA_x}{dy}\end{aligned} \tag{3}$$

WE will do $\frac{Ay[2,1,:]-Ay[0,1,:])}{dx} - \frac{Ax[1,2,:]-Ax[1,0,:])}{dy}$ to get value of Bz.
This shall be plotted vs z on loglog scale now.

Step (vi) Finding Best Fit

Our next aim is to fit the plot we have of B(z) to cz^b, to do this is equivalent to fitting $log(B(z))$ to $blog(z) + log(c)$.
We will do this using the lstsq function of python.
Next thing we will do is show the results for the code I have executed:

Current Elements and plots

The code script used to make the current elements is as follows:

```
#Making the vectors of r and current in x,y,z form
phi = linspace(0,2*pi,N)
rl = c_[radius*cos(phi),radius*sin(phi),zeros(N)]          #rl[l] gives expected rl
dphi = (2*np.pi/N)*c_[-sin(phi),cos(phi),zeros(N)]         #dphi vector
dphi_abs = 2*np.pi/N                                       #dphi magnitude
dl = radius*dphi                                           #using that dl = rdphi
I = c_[(-abs(cos(phi))*sin(phi))*4*np.pi/mu,abs(cos(phi))*cos(phi)*4*np.pi/mu,zeros
I_plot = I/(4*np.pi/mu)     #scaling all I factors to get better quiver
```

Script used to plot the current elements and quiver:

```
#Plot current elements
title("Centres of elements of broken wire")
xlabel(r"$x\longrightarrow$")
ylabel(r"$y\longrightarrow$")
plot(rl[:,0],rl[:,1],'bo',markersize=3)
savefig("Current elements.jpg")
close()

#Plot the quiver for I vs r
title("Quiver plot for current")
xlabel(r"$x\longrightarrow$")
ylabel(r"$y\longrightarrow$")
quiver(rl[:,0],rl[:,1],I_plot[:,0],I_plot[:,1],color ='blue',scale=30)
grid()
savefig("Current Quiver Plot.jpg")
close()
```

The plots we get from the same are:

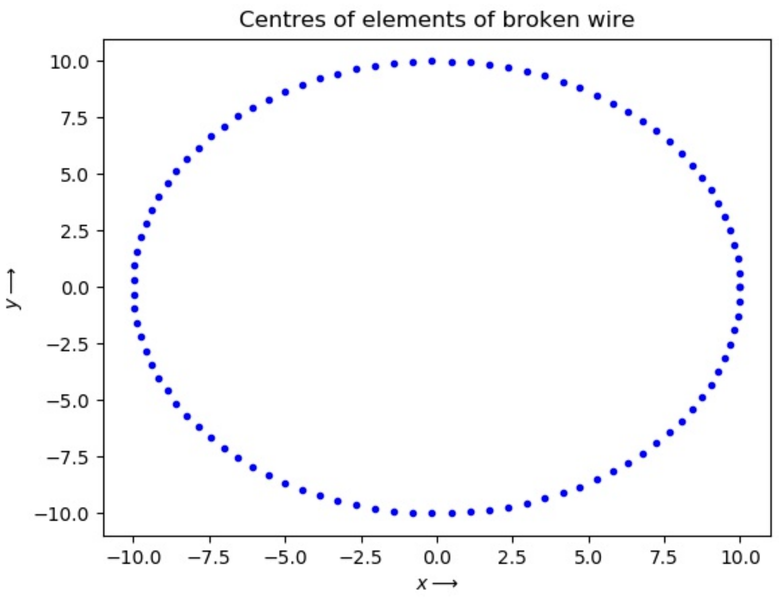

Figure 1: Current Elements

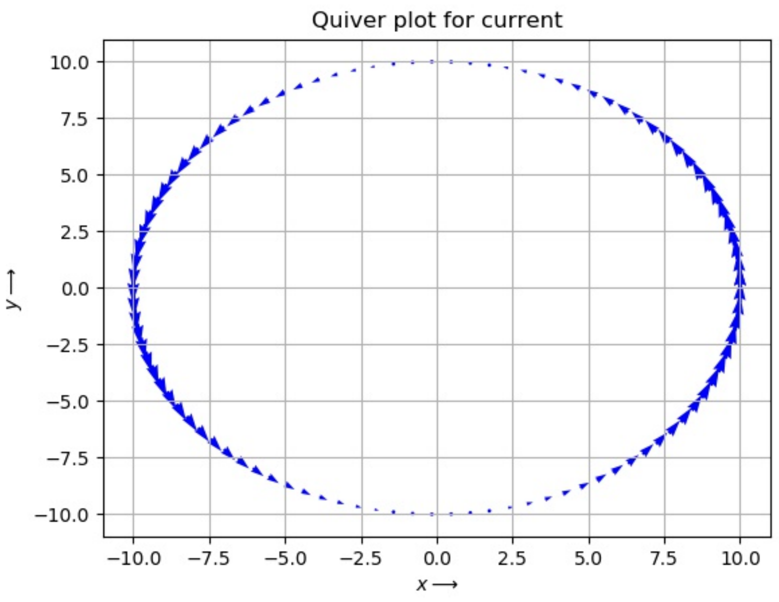

Figure 2: Quiver Plot

Solving for Magnetic Vector Potential and eventually Magnetic Field along Z-axis

As explained in Pseudo Code, we will go through the calculations for A, Bz. First, we will make the 2 meshgrids required for each approach as follows:

```
#Make a meshgrid of 3,3,1000 with x,y centred at 0 whereas z ranging from 1 to 1000
xi = 3
yi = 3
zi = 1000
x=np.linspace(-pow(10,-5),pow(10,-5),xi)
y=np.linspace(-pow(10,-5),pow(10,-5),yi)
z=np.linspace(1,zi+1,zi)
#Indexing = 'ij' makes meshgrid of form x,y,z
xx,yy,zz = np.meshgrid(x,y,z,indexing='ij')
#General vector r for any position of size 3,3,1000
r = np.stack((xx,yy,zz),axis=3)
#General vector r for any position, copied 100 times and reshaped
r_vectorization = np.tile(r,100).reshape(3,3,1000,100,3)
```

Solve for A using for loop approach, shape of R in these calculations is (3,3,1000) as calculated from calc(l):

```
#Bz using for loop using calc(l) function only for Dynamic case
def Bz_by_for_loop():
    dAx = []
    dAy = []
    for l in range(N):
        R = calc_l(r,rl,l)
        dAx.append((mu/(4*np.pi))*I[l][0]*exp(-1j*k*R)*radius*dphi_abs/(R))
        dAy.append((mu/(4*np.pi))*I[l][1]*exp(-1j*k*R)*radius*dphi_abs/(R))

    #Ax,Ay given by sum of dAx,dAy respectively
    Ax = np.sum(dAx, axis=0).reshape(3,3,1000)
    Ay = np.sum(dAy, axis=0).reshape(3,3,1000)

    #Since Bz is curl of A, its given as follows
    Bz = (Ay[2,1,:]-Ay[0,1,:]-(Ax[1,2,:]-Ax[1,0,:]))/(2*pow(10,-5))
    return Bz
```

Solve for A using vectorization, shape of R in these calculations is (3,3,1000,100) as calculated from calc(all):

```
#Function calc_all extended from calc(l) to find R = |r-rl| for all l
def calc_all(r,rl):
    '''
    Extra dimension is added since this function stores the value of |r-rl| for
    all of the 100 rl's corresponding to the 3,3,1000 points in the meshgrid
    '''
    return norm(r-rl,axis=-1)
```

```
#Bz using vectorization using calc_all extended from calc(l)
def Bz_by_vectorization():
    R = calc_all(r_vectorization,rl)

    #Taking sum of -1th elements in array since it contains dl element of sum
    Ax = np.sum((mu/(4*np.pi))*I[:,0]*exp(-1j*k*R)*radius*dphi_abs/(R),axis=-1)
    Ay = np.sum((mu/(4*np.pi))*I[:,1]*exp(-1j*k*R)*radius*dphi_abs/(R),axis=-1)

    #Finding Potential Fields for static case. No e^(-jkR) factor in this summation
    Ax_st = np.sum((mu/(4*np.pi))*I[:,0]*radius*dphi_abs/(R),axis=-1)
    Ay_st = np.sum((mu/(4*np.pi))*I[:,1]*radius*dphi_abs/(R),axis=-1)

    #Taking Bz as the curl = dAy/dx - dAx/dy
    Bz = (Ay[2,1,:]-Ay[0,1,:]-(Ax[1,2,:]-Ax[1,0,:]))/(2*pow(10,-5))

    #Similar calculations of Bz for static current
    Bz_st = (Ay_st[2,1,:]-Ay_st[0,1,:]-(Ax_st[1,2,:]-Ax_st[1,0,:]))/(2*pow(10,-5))

    return Bz, Bz_st
```

Now we will plot the 2 distinct Bz obtained by the 2 approaches in the same plot using subplots to compare, as expected, the results are identical since the 2 methods are essentially the same, vectorization is the more efficient way of implementation since it replaces the for loop and np.sum by a single np.sum

```
Bz, Bz_static = Bz_by_vectorization()
Bz1 = Bz_by_for_loop()

#Plotting absolute value of Bz vs z in loglog scale for the for loop and vectorizat
fig,axs = subplots(2)
fig.suptitle("Bz using i) for loop ii) Vectorization")
axs[0].loglog()
axs[0].grid()
axs[0].set_xlabel("log(z)")
axs[0].set_ylabel("log(Bz)")
axs[0].plot(z,abs(Bz1))
axs[1].loglog()
axs[1].grid()
axs[1].set_xlabel("log(z)")
axs[1].set_ylabel("log(Bz)")
axs[1].plot(z,abs(Bz))
savefig("Different computations for B(z).jpg")
close()
```

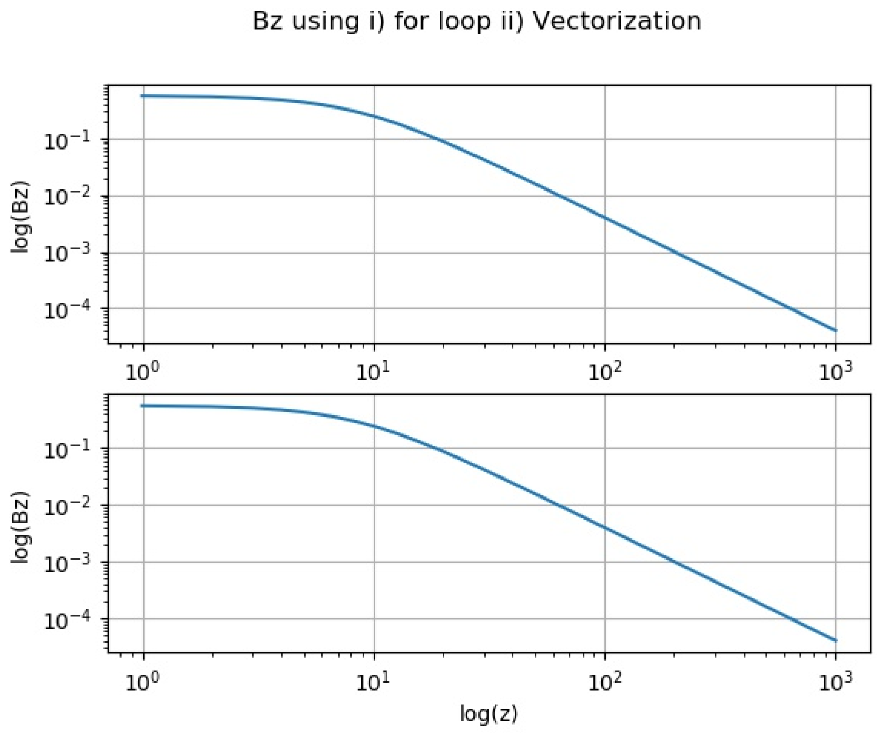

Figure 3: Bz using 2 approaches

Best Fit to the graphs

To obtain a best fit to cz^b for abs(B(z)) is the final aim of this assignment, as explained in pseudo-code, it is equivalent to fitting $log(abs(B(z))$ with $blog(z) + log(c)$.
We do this using lstsq method of python, note that we will do a fit for B(z) only after first few values to get a good value since intially the graph is constant and becomes of the form $blog(z) + log(c)$ after some small value of z. I will take this value as 10, i.e will fit from z = 10 to 1000

```
#Finding least error fit and plotting fit vs actual curve in loglog scale
def least_error_fit(Bz,z):
    '''
    Taking elements from z>10 onwards since it can be seen from graph of actual
    curve that it becomes linear in log scale after certain value,
    This is done to get the best possible fit for the curve by considering as
    much domain of z as possible. The c1 we get is actually log(c) if we
    have fit to cz^b since we actually fit log(Bz) to b*log(z) + log(c),
    so c = e^c1 so we return exp(c)
    '''
    y = log(abs(Bz))[10:]
    x = log(z[10:])
    A = c_[x,ones(1000)[10:]]
    b,c = lstsq(A,y)[0]
    return b,exp(c)
```

Plotting actual value vs best fit

```
#Plotting fits for Dynamic case
b1,c1 = least_error_fit(Bz,z)
print("c = "+str(c1)+" and b = "+str(b1)+", Bz is fitted to cz^b for Dynamic case")

x = log(z[10:])
title("Actual Curve and Linear Fit vs z for Dynamic case in loglog")
xlabel("log(z)")
ylabel("log(|Magnetic Field|)")
loglog()
plot(z,abs(Bz), label="Actual Curve")
plot(exp(x),c1*exp(b1*x),label="Linear Fit")
grid()
legend()
savefig("Actual vs Fit for Dynamic.jpg")
close()

#Plotting fits for Static case
b2,c2 = least_error_fit(Bz_static,z)
print("c = "+str(c2)+" and b = "+str(b2)+", Bz is fitted to cz^b for Statics case")

title("Actual Curve and Linear Fit vs z for Static case in loglog")
xlabel("log(z)")
ylabel("log(|Magnetic Field|)")
loglog()
plot(z,abs(Bz_static), label="Actual Curve")
plot(exp(x),c2*exp(b2*x),label="Linear Fit")
grid()
legend()
savefig("Actual vs Fit for Static.jpg")
close()
```

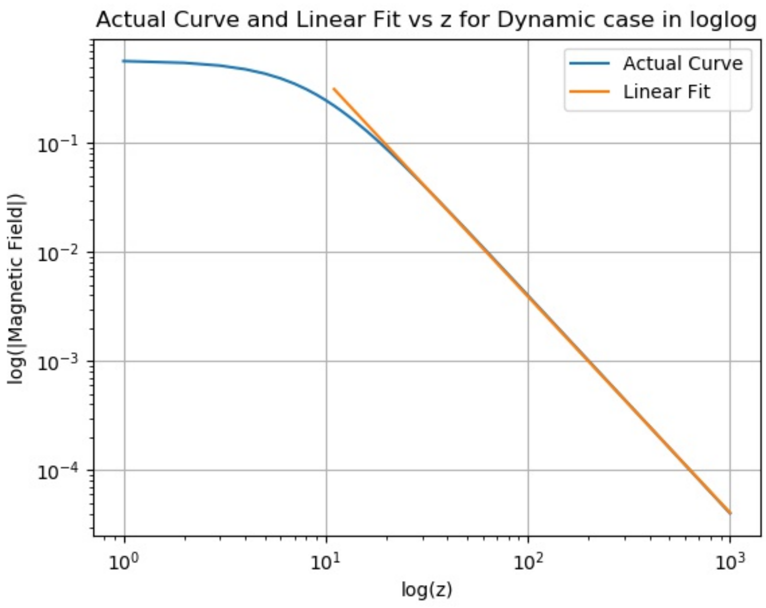

Figure 4: Dynamic case Bz actual vs best fit

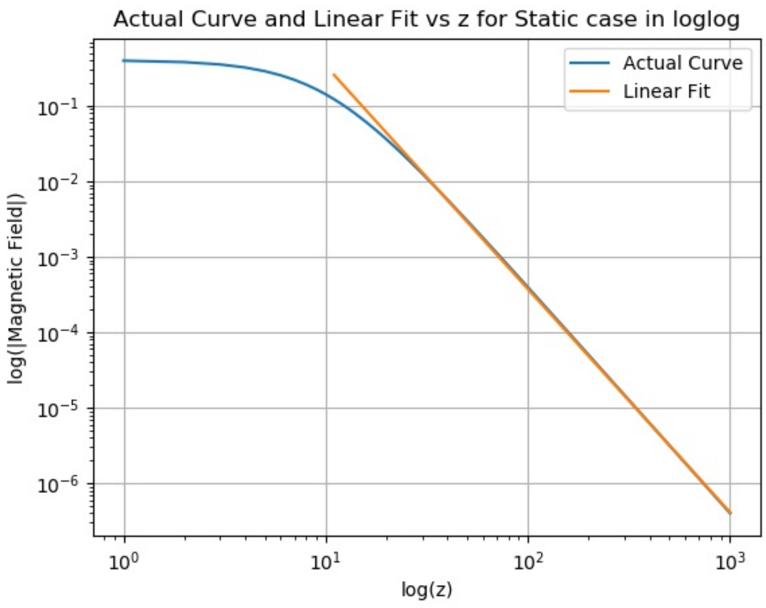

Figure 5: Static case Bz actual vs best fit

Results and Conclusions

The following is the output from the code for best fit values for the 2 cases for which I have solved.

- c = 36.22639047173061 and b = -1.9832272012692649 where Bz is fitted to cz^b for Dynamic case
- c = 309.65427104727644 and b = -2.9581828664779777 where Bz is fitted to cz^b for Statics case

So it is clear that Bz is almost proportional to z^{-2} in Dynamic case and z^{-3} in Static case.

Observations

On the basis of the results obtained, we can make the following observations:

- The magnetic field along Z-axis is proportional to z^{-2} as expected for a time dependant current.
- For a time independant or static current, the Magnetic Field is proportional to z^{-3}.
- Solely looking at the formulas that we have used to solve for the same, the difference between Statics and Dynamics is the Green's theorem term e^{-jkR} which is present in the numerator of Dynamics which clearly contributes to this discrepancy that we have observed
- This can be explained by seeing the Maxwell's equations governing electrical and magnetical fields.
- $\nabla \times \mathbf{H} = \frac{\partial \mathbf{D}}{\partial t} + \mathbf{J}$ is the Equation for Magnetic Field. As one can see, it depends not only Current but also the variation of electrical field.
- The best explanation for the discrepancy in the proportionality can be that since E depends on inverse square of distance, that value dominates the inverse cube relation due to the current flow and hence making B proportional to z^{-2} in dynamic case because z^{-2} dominates z^{-3} for large values of z (In general z¿1), since $\frac{dE}{dt}$ is 0 in static case however, B depends solely on current and is hence proportional to z^{-3} as expected.