PSet4 P4 KTX887

October 18, 2024

[1]:
```python
#Homework set 4, problem 4
#Aaron Weymouth
#KTX887
import sympy as sp
```

0.1 Question 4

Consider a model of a relativistic jet motivated by Blandford and Znajek (1977). The jet is stationary (meaning there is no change over time) and comprises a helical magnetic field threading the horizon of a black hole at its base, extending symmetrically in a paraboloid shape proportional to z. Moreover, let's assume the jet is self-similar in that at every value of xi (which is defined as s squared divided by z), the jet velocity, electric, and magnetic fields are determined by the magnetic flux Phi(s,z) threading the surface of the paraboloid jet and the angular speed of field lines Omega(s,z) as they emerge from the location of the horizon.

0.1.1 a.)

Find the (infinitesimal) contribution dPhi to magnetic flux at constant s through the area 2*pi*s*dz and use it to find Bs in terms of Phi.

HINT: The partial derivative of a function f(x, y) with respect to x is the derivative of f while holding y constant. Use a similar strategy to find Bz in terms of Phi.

0.1.2 b.)

Use Ampere's law (in integral form) to find Bphi, assuming the current within an Amperean loop of radius s is I. Check that the divergence of the magnetic field B in cylindrical coordinates is 0 for this self-similar jet.

HINT: You may want to use a numerical solver such as Mathematica to compute the divergence. Include the output of your code.

0.1.3 c.)

Ohm's law, expressed as E + v cross B = eta*j, has zero resistivity eta in ideal MHD. Use this form of Ohm's law and the fact that the component of v perpendicular to B is v_perp = Omega cross r to express E in terms of Omega in ideal MHD.

0.1.4 d.)

Use Ampere's law (in differential form) to compute j = (js, jphi, jz).

HINT: You may want to use a numerical solver such as Mathematica to compute any curls.

0.1.5 e.)

Find a differential equation for I(xi) by expanding rho*E + j cross B to first order in z/xi.

HINT: You may use a numerical solver.

[2]:
```
#Part A
#symbols
s, z = sp.symbols('s z')
Phi = sp.Function('Phi')(s, z)

#differential element volume
dA_s = 2 * sp.pi * s * sp.symbols('dz')  #area element in s-direction (2pi s dz)
dA_z = 2 * sp.pi * s * sp.symbols('ds')  #area element in z-direction (2pi s ds)

#def flux through surface
dPhi_s = sp.symbols('Bs') * dA_s
dPhi_z = sp.symbols('Bz') * dA_z

#solve for Bs in terms of Phi(s, z)
Bs = (1 / (2 * sp.pi * s)) * sp.diff(Phi, z)
Bz = (1 / (2 * sp.pi * s)) * sp.diff(Phi, s)

#display
print('The value for B_s is:')
display(Bs)
print('The value for B_z is:')
display(Bz)
```

```
The value for B_s is:
```

$$\frac{\frac{\partial}{\partial z}\Phi(s,z)}{2\pi s}$$

```
The value for B_z is:
```

$$\frac{\frac{\partial}{\partial s}\Phi(s,z)}{2\pi s}$$

[3]:
```
#define vars
s, z, mu_0 = sp.symbols('s z mu_0')
I_s = sp.Function('I')(s)  # Define I(s) as a function of s

#ampere law in integral form to solve for B_phi
B_phi = (mu_0 * I_s) / (2 * sp.pi * s)
```

```
#mag field components
Phi = sp.Function('Phi')(s, z)
B_s = (1 / (2 * sp.pi * s)) * sp.diff(Phi, z)
B_z = (1 / (2 * sp.pi * s)) * sp.diff(Phi, s)

#divb
div_B = (1/s) * sp.diff(s * B_s, s) + sp.diff(B_z, z)

#display
print('B_phi is : ')
display(B_phi)
print('The divergence of B is : ')
print(div_B.simplify())
```

```
B_phi is :
```

$$\frac{\mu_0 I(s)}{2\pi s}$$

```
The divergence of B is :
Derivative(Phi(s, z), s, z)/(pi*s)
```

[4]:
```
#define vars
s, z, mu_0, I = sp.symbols('s z mu_0 I')

#ampere law in integral form to solve for B_phi
B_phi = (mu_0 * I) / (2 * sp.pi * s)

#mag field components
Phi = sp.Function('Phi')(s, z)
B_s = (1 / (2 * sp.pi * s)) * sp.diff(Phi, z)
B_z = (1 / (2 * sp.pi * s)) * sp.diff(Phi, s)

#div b
div_B = (1/s) * sp.diff(s * B_s, s) + sp.diff(B_z, z)

#display
div_B.simplify()
#treating phi as a function of s and z
```

[4]:

$$\frac{\frac{\partial^2}{\partial z \partial s}\Phi(s, z)}{\pi s}$$

[5]:
```
#problem 4 part b
s, z = sp.symbols('s z')

#define Phi(s, z) as a function of s and z
Phi = sp.Function('Phi')(s, z)
```

```
#magnetic field components
B_s = (1 / (2 * sp.pi * s)) * sp.diff(Phi, z)  #Bs
B_z = (1 / (2 * sp.pi * s)) * sp.diff(Phi, s)  #Bz

#calculate div in cylindrical
div_B = (1/s) * sp.diff(s * B_s, s) + sp.diff(B_z, z)

#simplify
div_B_simplified = div_B.simplify()

#show div
print('The divergence of B is : ')
display(div_B_simplified)

#check if the divergence simplifies to show that mixed partial derivative must␣
 ↪be zero
mixed_partial = sp.diff(Phi, s, z)  #this is the second mixed partial␣
 ↪derivative of Phi

print('\nFor divergence to be zero, the following mixed partial derivative must␣
 ↪be zero:')
display(mixed_partial)

#subsitute a simple phi(s,z) to show it vanishes.
Phi_sub = s**2 + z**2  #example function
div_B_sub = div_B.subs(Phi, Phi_sub).simplify()

print('\nSubstitute Phi(s, z) = s**2 + z**2:')
display(div_B_sub)
```

The divergence of B is :

$$\frac{\frac{\partial^2}{\partial z \partial s}\Phi(s,z)}{\pi s}$$

For divergence to be zero, the following mixed partial derivative must be zero:

$$\frac{\partial^2}{\partial z \partial s}\Phi(s,z)$$

Substitute Phi(s, z) = s**2 + z**2:

0

[6]:
```
#Part C
#vector symbols
Omega_x, Omega_y, Omega_z = sp.symbols('Omega_x Omega_y Omega_z')
```

```
r_x, r_y, r_z = sp.symbols('r_x r_y r_z')
B_x, B_y, B_z = sp.symbols('B_x B_y B_z')

#vectors
Omega = sp.Matrix([Omega_x, Omega_y, Omega_z])  # Angular velocity
r = sp.Matrix([r_x, r_y, r_z])  # Radial vector
B = sp.Matrix([B_x, B_y, B_z])  # Magnetic field

#dots
Omega_dot_B = Omega.dot(B)
Omega_dot_r = Omega.dot(r)

#cross for v_perp = Omega x r
v_perp = Omega.cross(r)

#ohm in ideal MHD: E = - (v_perp x B)
E_vector = -v_perp.cross(B)

#simplify/factor
E_simplified = -(Omega_dot_B * r - Omega_dot_r * B)
E_simplified
```

[6]:

$$\begin{bmatrix} B_x\left(\Omega_x r_x + \Omega_y r_y + \Omega_z r_z\right) - r_x\left(B_x\Omega_x + B_y\Omega_y + B_z\Omega_z\right) \\ B_y\left(\Omega_x r_x + \Omega_y r_y + \Omega_z r_z\right) - r_y\left(B_x\Omega_x + B_y\Omega_y + B_z\Omega_z\right) \\ B_z\left(\Omega_x r_x + \Omega_y r_y + \Omega_z r_z\right) - r_z\left(B_x\Omega_x + B_y\Omega_y + B_z\Omega_z\right) \end{bmatrix}$$

[7]:

```
#part D

#vars
s, z, mu_0 = sp.symbols('s z mu_0')
Phi = sp.Function('Phi')(s, z)
I = sp.Function('I')(s)

#mag field components
B_s = (1 / (2 * sp.pi * s)) * sp.diff(Phi, z)
B_phi = (mu_0 * I) / (2 * sp.pi * s)
B_z = (1 / (2 * sp.pi * s)) * sp.diff(Phi, s)

#b in cylindrical
B = sp.Matrix([B_s, B_phi, B_z])

#curl b compute
curl_B = sp.Matrix([
    (1/s) * sp.diff(s*B_z, sp.symbols('phi')) - sp.diff(B_phi, z),
    sp.diff(B_s, z) - sp.diff(B_z, s),
    (1/s) * sp.diff(s*B_phi, s)
])
```

```
#current density j = (1/mu_0) * curl(B)
j = (1 / mu_0) * curl_B

#show
display(j)
```

$$\left[\begin{matrix}0\\ \frac{-\frac{\frac{\partial^{2}}{\partial s^{2}}\Phi(s,z)}{2\pi s}+\frac{\frac{\partial^{2}}{\partial z^{2}}\Phi(s,z)}{2\pi s}+\frac{\frac{\partial}{\partial s}\Phi(s,z)}{2\pi s^{2}}}{\mu_{0}}\\ \frac{\frac{d}{ds}I(s)}{2\pi s}\end{matrix}\right]$$

[8]:
```
#part E
"""
Asked to find diffeq for current where xi is s**2/z. Expand roh E + jxB to␣
 ↪first order..small approximation.
We know rohE is charge density time electric field and jxB is cross of current␣
 ↪density and mag.
"""

# Define variables
s, z, xi, mu_0 = sp.symbols('s z xi mu_0')
rho, I = sp.symbols('rho I', cls=sp.Function)  # Charge density and current
Phi = sp.Function('Phi')(s, z)  # Magnetic flux
B_s, B_phi, B_z = sp.symbols('B_s B_phi B_z')  # Magnetic field components
j_phi, j_z = sp.symbols('j_phi j_z')  # Current density components
E_s, E_phi, E_z = sp.symbols('E_s E_phi E_z')  # Electric field components

# Define B and j as vectors
B = sp.Matrix([B_s, B_phi, B_z])
j = sp.Matrix([0, j_phi, j_z])

# Define the electric field vector
E = sp.Matrix([E_s, E_phi, E_z])

# Expression for rho * E + j x B
expression = rho(s) * E + j.cross(B)

# Expand each component of the matrix to first order in z/xi
expansion = expression.applyfunc(lambda expr: expr.subs(s**2/z, xi).series(z/
 ↪xi, n=2))

# Simplify the expansion
simplified_expansion = expansion.applyfunc(lambda expr: expr.simplify())

simplified_expansion
```

[8]:

$$\begin{bmatrix} -B_\phi j_z + B_z j_\phi + E_s \rho(s) \\ B_s j_z + E_\phi \rho(s) \\ -B_s j_\phi + E_z \rho(s) \end{bmatrix}$$

0.2 Expanded Expression

The expansion of rho E + j * B to first order in (z/) as shown above

0.2.1 First Row:

- **-B_phi j_z + B_z j_phi + E_s rho(s)**

Only component in the s dir (radial/cylindrical

- **-B_phi j_z**: force density cross B_phi and axial J_z.
- **B_z j_phi**: force density from B_z with current density from azimuthal j_phi. radial force makes sense
- **E_s rho(s)**: cleary electric field in s multiplied by charge density coloumb force acting radially.

0.2.2 Second Row:

- **B_s j_z + E_phi rho(s)**

Component of expression in phi/azimuthal

- **B_s j_z**: interaction of radial magnetic B_s and axial current dens. j_z
- **E_phi rho(s)**: azimuth electric field by charge density – electric force in azimuth dir

0.2.3 Third Row:

- **-B_s j_phi + E_z rho(s)**

resultant of z dir/axial

- **-B_s j_phi**: force generated by interaction of B_s and j_phi axial force made. Lorentz??
- **E_z rho(s)**: clearly the coulomb force along z

0.2.4 Overall

Yep pretty much makes perfect sense. Looking at it by direction:

- **Radial (s) direction**: influence by axial/azimuthal mag field and radial e field
- **Azimuthal (phi) direction**: influenced by radial mag and azimuth e-field
- **Axial (z) direction**: radial magnetic and axial electric field

[9]:
```
#unsure on if this is what you're asking but i'm going to make up an I(xi) that
↪relates current density to j_phi and j_z
# Redefine the current density components j_phi and j_z in terms of I(xi)

I_xi = sp.Function('I')(xi)
```

```
#j_phi and j_z are related to the derivative of I w/respect to xi
j_phi = sp.diff(I_xi, xi)
j_z = I_xi / xi

#sub in
new_expression = simplified_expansion.subs({sp.symbols('j_phi'): j_phi, sp.
 ↪symbols('j_z'): j_z})

#simplify
new_expression_simplified = new_expression.applyfunc(lambda expr: expr.
 ↪simplify())

# Now let's isolate the terms and find the differential equation governing I(xi)
new_expression_simplified
```

[9]:

$$\left[\begin{matrix}- \frac{B_{\phi} I{\left(\xi \right)}}{\xi} + B_{z} \frac{d}{d \xi} I{\left(\xi \right)} + E_{s} \rho{\left(s \right)}\\\frac{B_{s} I{\left(\xi \right)}}{\xi} + E_{\phi} \rho{\left(s \right)}\\- B_{s} \frac{d}{d \xi} I{\left(\xi \right)} + E_{z} \rho{\left(s \right)}\end{matrix}\right]$$

[10]:

```
#0,0 and 2,0
eq_1 = new_expression_simplified[0, 0]
eq_2 = new_expression_simplified[1, 0]
eq_3 = new_expression_simplified[2, 0]



display(eq_1, eq_2, eq_3)
```

$$- \frac{B_{\phi} I{\left(\xi \right)}}{\xi} + B_{z} \frac{d}{d \xi} I{\left(\xi \right)} + E_{s} \rho{\left(s \right)}$$

$$\frac{B_{s} I{\left(\xi \right)}}{\xi} + E_{\phi} \rho{\left(s \right)}$$

$$- B_{s} \frac{d}{d \xi} I{\left(\xi \right)} + E_{z} \rho{\left(s \right)}$$

[11]:

```
#solve for I(xi)
sol_1 = sp.dsolve(eq_1, I_xi)

#solve for I(xi)
sol_3 = sp.dsolve(eq_3, I_xi)

#general solution
display(sol_1, sol_3)
```

$$I{\left(\xi \right)} = \frac{C_{1} \left(B_{\phi} - B_{z}\right) e^{\frac{B_{\phi} \log{\left(\xi \right)}}{B_{z}}} + E_{s} \xi \rho{\left(s \right)}}{B_{\phi} - B_{z}}$$

$$I(\xi) = C_1 + \frac{E_z \xi \rho(s)}{B_s}$$

[11]:

```

```