Computational exercise: The motion of a charged particle in a magnetic field
PH 220

Objective: Write a computer code that reads in the magnitudes, locations, and directions of several line currents, as well as the mass, charge, initial position, and initial velocity of an additional charged particle. Then determine the trajectory of the particle through the fields using Euler's method, for a specified time.

Background: In exercise 4 we investigated the motion of a charged particle in an electric field. In this assignment, we want to do a similar investigation, but with magnetic fields.

As with the electric fields, the fundamental physics behind this problem is straightforward: the force acting on the charged particle is given by

$$\vec{F} = q\vec{v} \times \vec{B}$$

and once we know the force acting on the particle, we can find the acceleration

$$\vec{a} = \vec{F}/m$$

Knowing the acceleration, we can get the velocity and position through time by integration via Euler's method (see exercise 4 for details).

The tricky part with this exercise is finding the magnetic field. In the input file, we'll be specifying a current by it's magnitude, one point in space that the current passes through, and it's direction. How do I find the magnetic field at some arbitrary point in space from this information? If I can do it for one line current, I can do it for many (I just need to superimpose the fields).

First of all, we must make sure that the direction information is normalized. In the input file we'll basically just be specifying three vector components. To normalize these vector components, I need to find the length of the vector, and then divide each component by that length. In other words, if we pass in vector components V_x, V_y, and V_z, the normalized vector components will be

$$u_x = V_x/V$$
$$u_y = V_y/V$$
$$u_z = V_z/V$$

where

$$V = \sqrt{V_x^2 + V_y^2 + V_z^2}$$

Once I know the normalized components of this unit vector, I can find the point on the line current that is closest to my point of interest (i.e. where I would like to know the field). First, consider that we know the coordinates of the point of interest (which I will call x, y, and z). We also know one of the points on the line charge, which coordinates we will call x_I, y_I, and z_I. We'll now consider the vector pointing from the known current point to the point of interest, which I will call $\vec{A}$, and whose components are

$$A_x = (x - x_I)\hat{i}$$
$$A_y = (y - y_I)\hat{j}$$
$$A_z = (z - z_I)\hat{k}$$

The projection of this vector onto the line current will locate the point on the line current closest to the point of interest, i.e. we consider a vector $\vec{B}$ which is

$$\vec{B} = (\vec{A} \cdot \vec{u})\vec{u}$$

This vector points from the known point on the line current to that point on the line current which is closest to the point of interest, whose coordinates I will call x_P, y_P, and z_P. These coordinates are found by

$$x_P = x_I + B_x$$
$$y_P = y_I + B_y$$
$$z_P = z_I + B_z$$

Now that I know the location of this "closest point" on the line current, I will define a vector going from this point to the point of interest ($\vec{r}$), whose components are

$$r_x = (x - x_P)\hat{i}$$
$$r_y = (y - y_P)\hat{j}$$
$$r_z = (z - z_P)\hat{k}$$

We know that the magnitude of the magnetic field at our point of interest is $\mu_0 I/2\pi r$, where r is the length of the vector $\vec{r}$ we just defined. Normally we would get the direction of this field using the right hand rule, although since we know the direction of the current ($\vec{u}$) and the direction of this vector $\vec{r}$, we can get all of the components of the magnetic field at this point using the following:

$$\vec{B} = \frac{\mu_0 I \vec{u} \times \vec{r}}{2\pi r^2}$$

Now that we have the field(s) at that point, finding the force is fairly straightforward.

Exercise requirements: Write a computer code that does the following:

1. Reads a file specifying the number of line currents, the magnitude of each current (in A), a coordinate (in cm) that the current passes through, and three vector components that specify the direction of the current (which you will have to normalize, as described above). The file then contains the mass (in grams) and the charge (in nC) of the test particle that we want to track, its initial coordinates (in cm), its initial velocity (in cm/s), and finally, the duration and time step to use in the simulation (in seconds). This input file will be formatted as follows. The first line of the input will consist of a single integer specifying how many line currents there are. The next lines, one for each line current, gives the magnitude of the current in Amperes, the known x-coordinate in cm, the known y-coordinate in cm, the known z-coordinate in cm, and then three vector components specifying the direction (which you will normalize). The source charge lines are followed by a line specifying the mass and charge of the particle we want to track, in grams and nC respectively. The next line specifies the initial x-, y-, and z-coordinates of this particle, in cm, and is immediately followed by a line that specifies the x-, y-, and z-components of the particle's initial velocity, in cm/s. The final line gives the duration of the simulation, in seconds, and the time step to use, also in seconds. Here is an example input file.

```
3
 500.000   1.000   0.000   0.000   1.000   1.000   1.000
 500.000   0.000   1.000   0.000   0.000   1.000   1.000
 500.000   0.000   0.000   1.000   0.000   0.000   1.000
1.00E-10   3.000
  -1.000   1.000     1.000
   0.500  -0.500   -0.500
   4.000   1.E-5
```

2. Determines the trajectory of the particle using Euler's method, as explained above.
3. For each time step, writes out the current time in seconds, the coordinates of the particle in cm, and the velocity components of the particle in cm/s, formatted as follows (please note the precision used for each of the numbers!):

```
0.000000  -0.999995   0.999995   0.999995   0.496969  -0.501098  -0.501934
0.000010  -0.999990   0.999990   0.999990   0.493928  -0.502188  -0.503856
0.000020  -0.999985   0.999985   0.999985   0.490877  -0.503271  -0.505767
0.000030  -0.999980   0.999980   0.999980   0.487817  -0.504348  -0.507666
0.000040  -0.999975   0.999975   0.999975   0.484747  -0.505417  -0.509554
0.000050  -0.999971   0.999970   0.999970   0.481667  -0.506479  -0.511430
0.000060  -0.999966   0.999965   0.999965   0.478578  -0.507533  -0.513295
0.000070  -0.999961   0.999960   0.999959   0.475479  -0.508581  -0.515148
0.000080  -0.999956   0.999955   0.999954   0.472371  -0.509621  -0.516990
0.000090  -0.999952   0.999949   0.999949   0.469254  -0.510654  -0.518820
```

Note that since we are simulating the particle's trajectory for 4 second, but with a 0.00001 second time step, your output file will have 400,000 lines.

Validation: You can verify whether your code is working correctly or not by feeding it the example input file above, and seeing whether it reproduces the following results at the appropriate times (note that these lines are taken from the output for the given input file - I did not use a 0.5 second time step!):

```
0.500000  -0.999896   0.925354   1.044994   0.942797  -0.657960  -0.775144
1.000000  -1.004145   0.835444   1.095913   1.537825   0.571101   1.492992
1.500000  -1.000939   0.744737   1.131321  -2.813077   0.717548   2.064194
2.000000  -0.986617   0.759327   1.121112  -5.134280  -1.073331  -2.349831
2.500000  -0.968410   0.907201   1.078908   5.688287  -3.442235  -6.418851
3.000000  -0.979815   0.798397   1.163516   9.890676   5.383537   9.710379
3.500000  -0.896967   0.789131   1.116895 -22.403950   3.501631   7.573653
```

Grading rubric: Your grade on this exercise, out of 50 points total, will be calculated based on the following criteria:

- Your code accurately tracks the projectile for a test input file that I provide. I will check the position of the particle at 5 different times, with two points for each coordinate. (30 points) Note that since everyone is using the same time step, everyone should be getting the same results!
- Your code is well organized and "pretty". (3 points)
- The output is formatted as required. (2 points)
- You have included all of the following comments in your code:
    - A header that describes what the code does and how to use it. (5 points)
    - A description of each of the variables in your code. (5 points)
    - Frequent descriptions of what your code is doing, including any numerical methods that you are employing. (5 points)