Union College **Winter 2013**

Physics 121 Lab #4:

Numerical Calculations of the Magnetic Field from a Moving Charge

In Lab #3 you gained a little experience about using computers to calculate the electric field in complex situations (e.g. the E-field due to a rod of continuous charge at a random position). Computer simulations can also be used to make theoretical time sequences (i.e. movies) and hence to reveal how a system may change with time. In lecture we've now introduced the Biot-Savart Law, which describes the magnetic field due to a moving charge, and so we can now write VPython code to show a moving charge of varying magnitude, moving at varying speed, and calculate and display the magnetic field at various positions.

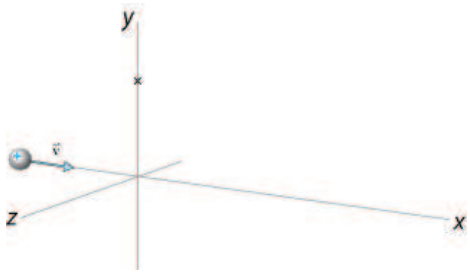
Before getting started, go to the course Nexus page and download the VPython program "Biot-Savart.py." In this program there is a start to the code involving a proton moving at some constant velocity in the +x direction along the x-axis and starting at some initial position. There is also a start for calculating and displaying the magnetic field of the proton at a location in the yz plane at some distance from the x-axis.

1 Planning and Predicting

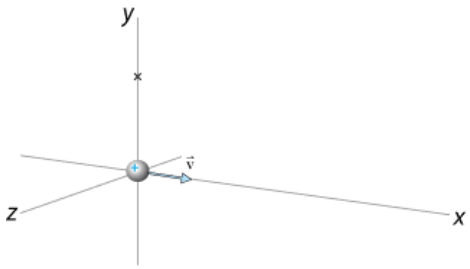
Consider the three figures below in which a proton starts at $x_i =< -2x10^{-10}, 0, 0 >$ m and moves with velocity $v =< 4x10^4, 0, 0 >$ m/s.

- Calculate by hand, using the Biot-Savart Law, the magnetic field (magnitude and direction) at the observation location, $r_{obs} =< 0, 8x10^{-11}, 0 >$ m for the three "snapshots" shown here. Draw arrows to represent the magnetic field in each case.

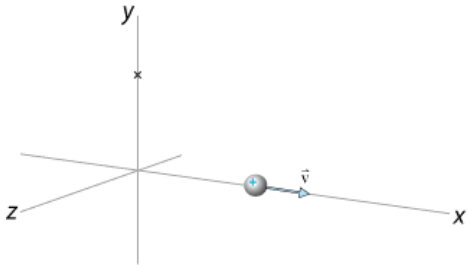
Time 1



Time 2



Time 3



2 Get Familiar with the Program

Read through the program to see how it is organized and what you will need to calculate. First note the basic structure of the program. It is divided into four sections:

- `Constants`: section where any important constants are defined.
- `Objects`: section where visible 3D objects (like spheres and arrows) are created.
- `Initial Values`: the starting values of important variables are set here.
- `Loop`: where the physics calculations are done. The program involves stepping forward in time, so the calculations needed to repeated for each successive time step, and hence are done in a "Loop."

3 Making the Proton Move

- In the objects section, change the initial position of the proton to match the value used in your prediction calculations (above). Note that the proton's initial position is given as ??e-10 m, so the right order of magnitude is set but you need to replace the question marks with numbers. Also, the proton's radius is set to 1e-11 – this is much larger than the real radius of the proton, but it allows us to see the sphere.
- In the `Initial Values` section, enter the proton's velocity.

Note that the "`Loop`" section is started by the command:

```
while proton.x <
```

meaning that all the following steps and calculations in the indented section will be continuously redone for each time step as long as the x-position of the proton is less than whatever value is put in after the '<' sign. (To start, all these calculation lines are commented out with the # sign.)

The program calculates the position of the object and displays it at that position. The program then calculate where the object is a very short time '`deltat`' (i.e. Δt) later, and changes its position to the new location and in the next frame displays it there. In vector terms this is

$$\vec{r}_f = \vec{r}_i + \vec{v}_{\text{avg}}\Delta t, \text{ because } \vec{v}_{\text{avg}} = \frac{\Delta\vec{r}}{\Delta t}$$

In VPython, this translates to a statement like:

```
particle.pos = particle.pos + velocity*deltat
```

Note that this line is found at the end of the Loop (and at the very end of the program).

This statement is similar to the Erod = Erod + dE in Lab 3. It is not an equation (if it were, it could only be true if velocity*deltat = 0). This is an update statement, or a re-assign statement. The value on the left is assigned to be equal to the value on the right (even if it ends up changing one of the values on the right). It does this calculation once and then moves on. This line, in effect, is what moves the particle, and hence makes a movie...as you'll now see.

- Change the ?? in the while statement to set the final position of the proton to be in the +x direction the same distance from the origin as the initial position.

- Then, run the program and verify that the proton moves in a straight line in the +x direction.

4 Calculating the Magnetic Field

Now you need to get the program to correctly calculate the magnetic field.

- Input the observation position you used in the "Planning and Predicting" (on page 2 of this handout) and uncomment the lines for robs1 and barrow1 before the Loop.

Now, you'll also need to input in the loop the statement for calculating the magnetic field at robs1. Note that the Loop contains two sets of four-commented lines. The first set describes the steps needed while the second set are the lines that will do the calculation.

- Uncomment (by deleting the #'s) the second set of lines.

Now, you need to put in the correct statements. Three of these lines are already done for you. (Note: since robs1 is a vector, it doesn't have a "pos attribute," and so to point to the 1st observation position one simply puts in "robs1," whereas "proton" is an "object" and so to point to its position we need to put in "proton.pos". Also, in the second line, VPython provides an easy way of calculating the magnitude of a vector ("mag(robs1)"). Or, as we did in Lab #3, one could just as easily use "robs1.mag.")
The third line is where the actual calculation of the magnetic field occurs. This involves a cross product, which, in principle, seems like a horrible task. However, VPython provides an easy way of doing this as well. (Just about any vector operation has a defined routine in VPython). To perform $\vec{A} \times \vec{B}$ in VPython, one types `cross(A,B)`.

- Using this command for a cross product, along with the notation for inputting the magnitude of a vector, and the already defined "rhat", type in an equation for calculating the *vector* "B1" using the Biot-Savart Law. (Remember that constants are defined at the beginning of the program).

Now, when you run the program you may discover that you need to change the size of the arrow. If so, you can do this using the "`scale`" in the `Constants` section. In the Loop, the axis of the arrow is changed with the statement:

```
barrow.axis = B1*scale.
```

- Run the program. Does your magnetic field vector behave as expected? Does it have the correct direction? Check direction by using the right hand rule. Are its direction *and* relative magnitude at the three time steps consistent with the arrows you drew in the Figures on page 2?

5 Print Values

The animated visual with the magnetic field arrow demonstrates how the magnetic field *changes*. But, it does not provide a check on the absolute value of the magnetic field. This is easy to accomplish using `Print` statements.

- At the end of the Loop (after the `particle.pos = particle.pos + velocity*deltat`line), insert a new line, with no indent, that says:

```
print B1.mag.
```

- Since this print statement is not indented it is outside the while loop and so the value of B that

will be printed is the last calculated value, with the proton at its final position, as given in the while statement. So, to compare with your analytical solution, change the final position of the proton in the while statement to be the origin. Now, when you run the program you will get a numerical output for the magnitude of the magnetic field at the observation position when the proton is at the origin.

- Does the magnitude of the magnetic field match what you expected according to the Biot-Savart Law?

6 Adding Observation Locations

- Add three more observation locations, all with x=0, but different y and z coordinates, so that the four locations are at the corners of a square surrounding the path of the particle. Include lines to do the calculations for these three new positions identical to what was done with robs1 (you'll need lines for the new barrow's, r's, rhat's, and B's.)
- For all the following, leave the final position of the proton at the origin.

Your program should now display the magnetic field vectors simultaneously at all these locations as the particle moves.

- Run the program. Do the field arrows behave as expected? Are they in the correct directions?
- Set "scale" to a value that makes the arrows representing magnetic field large, but not so large that they extend offscreen.

7 Testing the Dependencies

- Change the sign of the charge of the particle—turn it into an electron. What happens to the magnetic field vectors?

- Change the sign of the particle's velocity—make it go in the -x direction. You'll also need to change its starting position to be on +x axis (so that it approaches the origin while moving in the -x direction). You'll also need to change the `while`statement–since the particle is starting in the +x direction, you now want the calculations in the Loop to occur while the particle's x-position is ¿ 0 (not ¡ 0). Now with both the sign of the charge and the sign of the velocity switched, how does the magnetic field compare to what it was with an electron moving in the +x direction?

- Change the magnitude of the charge—turn it into an alpha particle. Does the magnitude of the B-field change as expected (check the numerical output)?

- Double the particle's speed. Does the magnitude of the B-field change as expected?

- Change the position of one of the observation locations so that it is twice as far from the origin. What happens to the magnetic field at this location?

Save your program as "Phy121_lab4_*yourname*.py" before exiting.

8 Your Report

Send your final program to your instructor and turn in answers to the following **questions**.

1. Outline, briefly, how an animation is accomplished with VPython.
2. Comment on the ease with which the change in magnetic field as the proton moved was demonstrated with a computer animation, in comparison to doing calculations by hand.
3. Comment on the dependences of the magnitude and direction of the magnetic field on:
   (a) sign of the charge;
   (b) direction of velocity;
   (c) magnitude of the charge;
   (d) speed of the charge;
   (e) distance of the observation point (for fixed θ);
4. Was it easy or difficult to test all these dependencies using the code?