Motion of a charged particle in a magnetic field

1 Purpose

You will write a VPython program to simulate the motion of a charged particle immersed in a magnetic field. **Start with the program shell that is provided**. The program shell draws a "floor" and displays a uniform magnetic field, which is initially set to $\langle 0, 0.2, 0 \rangle$ T.

2 Display a Moving Proton

In this section, you will generate the code to display a proton (shown as a sphere) moving in a straight line. You will not add the effects of the magnetic field until section 3.

1. Create a sphere named `particle` of appropriate radius representing a proton at location $\langle 0, 0.15, 0.3 \rangle$. Give it an initial velocity of 2×10^6 m/s ($v << c$) in the -x direction, by defining an appropriate vector quanitity:

   ```
   velocity = vector(## you fill this in)
   ```

2. After creating the proton, add the following line of code to prepare for leaving a trail later:

   ```
   trail = curve(color=particle.color)
   ```

3. Write a loop to move the proton in the direction of its velocity, by following the instructions below. If this is a very familiar task to you, you can do it now and move to step 4.

   In a program, you calculate the position of the object, and it is displayed in that position. You calculate where the object will be a very short time `deltat` later, and change its position to the new location, so it is displayed in the new location. You will use the relation between velocity (a vector) and position (a vector) to calculate the position of a moving particle at successive times, separated by a short time step Δt. Remember that, in vector terms:

   $$\vec{r}_f = \vec{r}_i + \vec{v}\Delta t \text{ (because } \vec{v} = \tfrac{\Delta \vec{r}}{\Delta t})$$

   In VPython, this translates to a statement like:

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

   where `velocity` is the vector quantity which you have initialized earlier in the program.

   This statement may look odd to you if you have not had much programming experience. The equals sign here has the meaning of "assignment" rather than equality. This instruction tells VPython to read up the old position of the particle, add to it the quantity, and store the new position as the current position of the particle. This will automatically move the particle to the new position.

4. Read the program shell: What is the value of `deltat` that is set in the program shell?

5. Inside the loop, update the position of the proton by using its velocity:

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

6. For this section, you are only generating the code to display a particle moving with constant velocity.

- In section 3 you will add the effects of the magnetic field, so the particle's velocity may no longer be constant. So far you have updated the particle's position; *think about how you might update the particle's velocity*.
- **DO NOT update the velocity until section 3**. (HINT: consider the momentum principle, $\Delta\vec{p} = \vec{F}_{net}\Delta t$, where $\vec{p} \approx m\vec{v}$. *Think about how could you adapt the code used to update position to update momentum.*)
- Recall that $\Delta\vec{p} = \vec{F}_{net}\Delta t$ can be written as:

$$\vec{p}_f = \vec{p}_i + \vec{F}_{net}\Delta t$$

7. Inside the loop, add this line after updating the position:

```
trail.append(pos=particle.pos)
```

8. Run your program. The proton should move in a straight line and leave a trail.

3 Add the Force due to the Magnetic Field

Now you will add the force of the magnetic field on the moving, charged particle.

1. Before the start of the loop, initialize the particle's charge, mass, and momentum, where $\vec{p} \approx m\vec{v}$ (since $v << c$).
2. **Inside the loop**, before the line updating the proton's position, you need to update the particle's velocity.
3. How did you decide to update the particle's velocity in section 2, part 6? **Answer on a piece of paper**.
   - What will you need to calculate in order to update the particle's velocity?
   - You may decide to find the (vector) force of the magnetic field on the moving, charged particle. A cross product $\vec{A} \times \vec{B}$ can be calculated in VPython as `cross(A,B)`, if `A` and `B` are vector quantities.
4. Complete the loop in the program shell with the plan that you checked above.
5. Run your program and observe the motion.

4 Numerical Experiments

Do the following experiments and **answer the questions on a piece of paper**. If the program runs too fast, decrease `deltat`. You can also use a `rate` statement to slow down the animation.

Answer the following questions on paper.

1. Describe the path of the proton's motion. Explain qualitatively in terms of force and momentum.
2. Derive the radius of the path in terms of the known variables, m, v, q, and B.
3. What is the approximate radius of the path in your simulation? How does it compare to the value predicted in step 2?

4. What happens to the radius of the path if you double the initial speed? Halve the initial speed?

5. Reset the speed to 2×10^6 m/s. Change the while statement to `while t < 3.34e-7`: and run the program. The proton should make just one complete orbit. If not, review your calculations.

6. Double the initial speed. How far around will the proton get in the time 3.34×10^{-7} seconds? Try it. What do you find? This striking behavior is the basis for the cyclotron.

7. Change the proton's initial velocity to be in the same direction as the magnetic field. What kind of path does the proton follow? Why?

8. Restore the velocity to its original value, and change the while statement to do 5 complete circles. Add a significant +y component to the proton's velocity. Now what kind of a path does the proton follow? Why?

9. Change the proton to an antiproton, a particle with the same mass as the proton, but a charge of $-e$. What happens to the motion of the particle? Why?

Last edited: Tuesday 22nd May, 2012 at 4:01pm