VPython Class 4: Dynamics of Charged Particles in Electric and Magnetic Fields

1. Introduction

In the previous exercises you should have:

- displayed electric field vectors (as Vpython `arrow` objects) at an observation point $\vec{r}_{\text{obs}}$, due to a point charge at an arbitrary source point $\vec{r}_{\text{source}}$;
- displayed magnetic field vectors (as Vpython `arrow` objects) at an observation point $\vec{r}_{\text{obs}}$, due to a small segment of current at an arbitrary source point $\vec{r}_{\text{source}}$;
- displayed field vectors at an observation point that result from the superposition of fields of multiple point-like sources; and
- learned how to define and use Python functions.

In this exercise we will simulate the motion of charged particles in given electric and magnetic fields. We will use Newton's second law, along with the *stepping equation* technique you learned in PHYS 211, to determine the motion.

2. Getting started

Reminder:

- `All Programs` → `Python 3.2` → `IDLE (Python GUI)`
- From the Python *shell*, select `File` → `New Window`
- Now you've got the IDLE window in which we'll write our scripts. If you liked my background with the transparent planes, open the file `background.py` that you can find in the PHYS 211E folder in our departmental public netspace. Save this with a new name in your space.

  Otherwise, just start your own program. The first line of your program should *always* (unless otherwise stated) be:

```
from visual import *
```

3. Newton's second law and the *stepping equations*

You are all familiar with Newton's second law: $\vec{F}_{\text{net}} = m\vec{a}$. I want to recast that slightly as

$$\vec{a} = \frac{1}{m}\vec{F}_{\text{net}}, \tag{1}$$

which is really a second-order differential equation:

$$\frac{d^2\vec{r}}{dt^2} = \frac{1}{m}\vec{F}_{\text{net}}(\vec{r}, \vec{v}). \tag{2}$$

Some times it is easy to solve this differential equation, as in the case of constant forces, or linear restoring forces, and we get a nice function $\vec{r}(t)$ that tells us where the particle is at time t, and how fast it's going ($d\vec{r}/dt$). In other cases it's not so easy. That's where computers come into play.

As in PHYS 211, we are going to consider a sequence of very short time intervals Δt — so short that the force can be considered to be constant during this interval. Constant force means constant acceleration, giving the *stepping equations*,

$$\vec{r}_{\text{new}} \simeq \vec{r}_{\text{old}} + \vec{v}_{\text{old}}\,\Delta t \tag{3}$$

$$\vec{v}_{\text{new}} \simeq \vec{v}_{\text{old}} + \vec{a}\,\Delta t. \tag{4}$$

These are only valid during the very short time interval, and we have to repeat this process many times to get the motion over a longer interval.

Make sure you figure out how you would program these two equations for an object named `particle` with `pos`, `vel`, and `mass` attributes, and a function that gives the force.

4. First dynamics simulation

Let's start with a simple example: a charged particle moving in a uniform magnetic field.

- Define a named `sphere` object with attributes for position, velocity, charge, and mass, for example,

```
charge1 = sphere(pos=vector(0,2.,0.),radius=0.2, color=color.red,
                                 vel=vector(3.,0.,0.), mass=1., charge=1.)
```

- Define a function that returns the *vector* magnetic field, given a `vector` position as an input. Lets start with a magnetic field of magnitude 1 in the z-direction:

```
def b(pos):
   ...
   ...
```

  It may seem like overkill to define a function for the constant field, but defining this as a function makes it easy to generalize to more complicated fields later.

- Define a function that returns the *vector* force, given a particle name as its input:

```
def f(particle):
   ...
   ...
```

- Define a value for the time step Δt. Let's start with

```
dt = 0.01
```

- As in the first Vpython exercise, we want to animate the motion of the charged particle. As you may recall, we did that with a `while` loop in which we updated the `pos` attribute of our ball. In this exercise well will be using the stepping equations to update the `pos` and `vel` attributes of the `charge1` object:

```
while True:
    rate(100)
    ...
    ...
```

- You can turn on a particle track if you like. Before the `while` loop, add the line

```
charge1.trail = curve(color=charge1.color)
```

  and within the loop add the line

```
charge1.trail.append(pos=charge1.pos)
```

- Investigate the motion you see:
  - Is it what you expect? (You should be quantitative here.)
  - What happens if you change the field strength?
  - What happens if you change the magnitude of the initial velocity of the particle?
  - What happens if you change the direction of the initial velocity in the x-y plane?
  - What happens if you give the initial velocity a small component in the z-direction?
  - What happens if you change the charge or mass of the particle?

5. Additional things to do

- Add a constant electric field. See what happens when the electric field is perpendicular to the magnetic field. What happens when the field is parallel to the magnetic field.
- You may have noticed that the simulation doesn't give you exact circles. There is one pretty obvious thing you should try to improve the situation. But there are also better *stepping equations*. So far, we have been using what is known as the Euler method. Look up the Euler-Richardson method and use it in your program — you should see a marked improvement.
- For the electric field, use that of a point positive charge at the origin. (This should look familiar!) I don't know how to describe the general motion analytically (I haven't spent much time trying), but it's easy to see what happens using a computer.

6. Submit your work

For your submitted scripts please use the following filename format:

`vpXX_yourlastname.py`,

where XX is the two-digit assignment number (in this case 04) and "_" is the underscore character. For example, a good filename would be: `vp01_ligare.py`

Once you've saved a copy of your code with this filename you can submit it by copying it into my drop box (`facultystaff/m/mligare`). (You can't save it directly there, but you can copy or drag the file into it.)