VPython Class 3: Teaching a Computer to Superpose: Fields, Functions, and the dreaded $\hat{r}$

1. Introduction

In our study of electric fields, we start with a single point charge source q_{source} located at some position $\vec{r}_{\text{source}}$, and we want to know the field created by this source at some observation point $\vec{r}_{\text{obs}}$. The relationship between the vectors is illustrated in Fig. 1. Note that this figure uses the conventions for axis labeling that is consistent with that of the default view in VPython; in many textbooks, the default view has z going up the page, y to the right, and x coming out of the page.

The vector displacement between the source and the observation point is given by

$$\vec{r} \equiv \vec{r}_{\text{obs}} - \vec{r}_{\text{source}}. \tag{1}$$

Coulomb's law then gives the field at $\vec{r}_{\text{obs}}$ due to the source at $\vec{r}_{\text{source}}$:

$$\vec{E} = \frac{kq}{r^2}\,\hat{r}, \tag{2}$$

where $\hat{r}$ is the unit vector in the direction of $\vec{r}$.

Your first task will be to write a Python program that will display a point charge at some position $\vec{r}_{\text{source}}$ along with an arrow representing the electric field at some other point $\vec{r}_{\text{obs}}$. As a specific simple, consider a point charge $q_{\text{source}} = 3$ C on the x-axis with coordinates (3,0,0), and suppose I ask for the field at position with coordinates (3,2,0). Your Python program should display something like Fig. 2, with an an arrow with its tail at the observation point, pointing away from the source, and a length proportional to $kq_{\text{source}}/r^2 = 3k/2^2 = 3k/4$ (except your output will be much prettier, and you will be able change your perspective, and zoom in and out). In this exercise we will set $k = 1$.

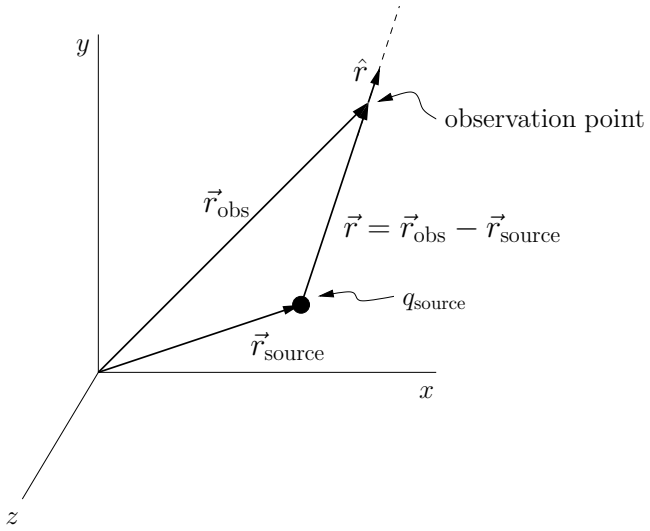

Figure 1: Vectors associated with the positions of a point charge source and the observation point for an electric field.

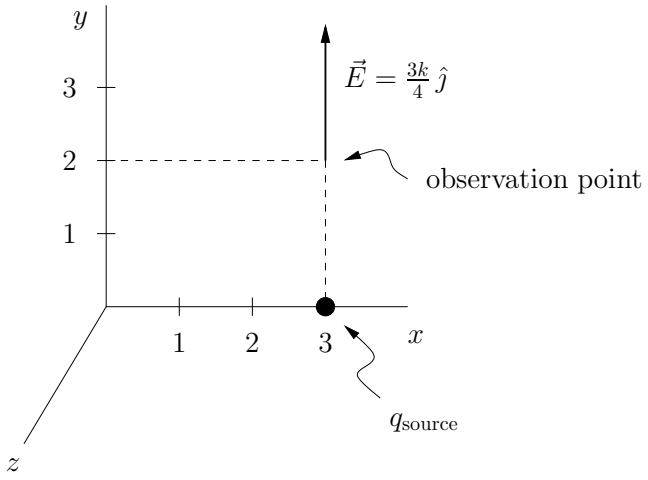

Figure 2: Example output. A source charge is located on the x-axis with coordinates (2,0,0), and we show the field vector $\vec{E}$ with its tail at the observation point with coordinates (3,2,0).

2. Reminder about Python Jargon: Objects and Attributes

Before starting on the ultimate task of displaying field vectors, lets review some commands we used last time and learn how to extend them, while introducing some of the language of Python.

The command

```
particle = sphere(pos=vector(0,0,0), radius=0.5, color=color.red,
                  charge = 5)
```

creates a sphere *object* (or equivalently, an *object* in the sphere *class*) whose name is `particle`. We could give it any name we want (e.g., `Fred` or `Ginger`). The name is used as a way to identify the object that we just created, and we can create other objects in the same class class with other names.

In creating the sphere above, we've given it certain *attributes*, namely a position, a radius, and a color. The visual module automatically "knows" what to do with these attributes — in this case, it draws the sphere centered at the given position, with the given radius and color. We have also added another attribute to the object "by hand," and called it `charge`; this isn't a VPython attribute.

3. The $\hat{r}$ vector

Now let's get down to the business of calculating and displaying field vectors. We'll start in this section by displaying $\hat{r}$ vectors.

In the rest of this section I'm not going to give you explicit commands to type. Rather, I would like you to try to use resources at your disposal, including the handouts for VPython Classes #1 and #2, and any programs you wrote last week, or the online information at `http://vpython.org/contents/docs/primitives.html`. None of this should take much time. If you get bogged down, just ask, and I'll give you much more explicit instructions.

- Create a *sphere object* at $(3, 0, 0)$ representing a *source* charge with $q = +4$. Let's all agree to give this sphere object the name `source`.

- The `source` object has several attributes. Which of these attributes corresponds to the vector $\vec{r}_{\text{source}}$? Put a temporary `print` statement into your script to check and make sure that you know how to get the vector $\vec{r}_{\text{source}}$ in terms of your Python variables. (Hint: `source` is not itself equal to $\vec{r}_{\text{source}}$.)
- Define an observation point at (3,2,0) as a Python *vector* with the name `obs`. This observation point serves as $\vec{r}_{\text{obs}}$. (There is no real need to define this as a sphere object; it's just a point, with no radius, color, or charge attributes. But if you want to display it with a colored sphere, that's ok.)
- Determine a Python expression for the vector $\vec{r}$ in terms of your `source` and `obs`; I suggest calling this `r`.
- Determine a Python expression for the vector $\hat{r}$ in terms of your `source` and `obs`; I suggest calling this `rhat`.
- Have VPython draw an *arrow object* at the position `obs` that represents $\hat{r}$. (I use the attribute `shaftwidth=0.1`). Do you get the arrow you expect?
- Change your observation point. Does the arrow representing $\hat{r}$ change appropriately?
- Change the position of your source charge. Does the arrow representing $\hat{r}$ change appropriately?

4. Visualizing electric field vectors

Modify your previous work so that your program gives the field vector at your observation point. (You may set $k = 1$.) If you use the previously defined `r` and `rhat`, it should take a very modest modification to go from an arrow representing $\hat{r}$ to an arrow representing $\vec{E}$. When you have this working, check the effect of varying:

- the source charge (magnitude and sign),
- the position of the source charge, and
- the observation point.

Does the the displayed field vector change appropriately?

5. The electric field from multiple sources

Now consider two charges, illustrated in Fig. 3, like those we studied on the second day of class. Use your electric field function to display the total field vector at the point (2,0,0). Check the field vector at some other points, like $(4, 0, 0)$, $(0, 2, 0)$, $(0, -2, 0)$, ...

6. Submit your work

Make sure to submit your work to my dropbox in `facultystaff/m/mligare` with the filename `vp03_lastname.py` when you're done.

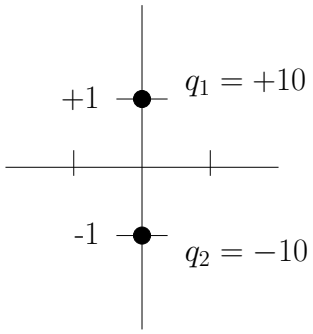

Figure 3: Two charges

7. Extras

- Define a function which takes as inputs the your `source` obect and your observation point, and returns a displayed `arrow` object.
- Reconsider the two charge example, and bump up the charges to ± 30. Now create field vectors at 16 equally spaced points around a circle of radius 4. Then add field vectors at equally spaced points around a circle of radius 6. Think about the field lines that would represent the same vector field.
- Generalize the two charge example to situations in which there are many more sources. What features of Python will make this an easy task. Think about ways in which you might numerically approximate an integral for an line of charge.
- ...