Electric Field of Point Charges and Dipoles

Objectives:

In this lab you will:

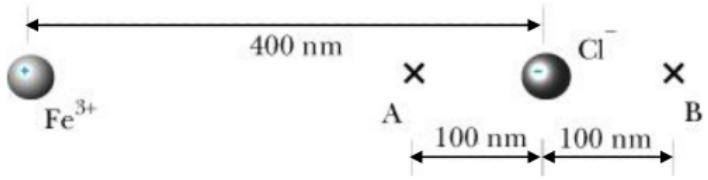
- Practice calculating Coulomb Electric fields.
- Learn to display the Electric field of a single particle using VPython.
- Learn to display the Electric fields of multiple particles using Vpython.
- Simulate a dipole using VPython.
- Use vectors to map the electric field pattern of the dipole in 3D.

Electric fields created by single particles are both simple to envision and simple to calculate. However, the electric fields created by an arrangement of particles are much harder to visualize and extremely tedious to calculate. VPython is an excellent tool for creating complex arrangements of particles and modeling their electric fields. VPython also provides the added benefit of visualization in 3D.

1: Warm-Up Problem 1

In the diagram below, an Fe^{3+} ion is located 400 nm from a Cl ion. (Recall that 1 nm is 10^{-9} m).

1. **What is the electric field, $\vec{E}_A$, at location A, 100 nm to the left of the Cl ion?**
2. **What is the electric field, $\vec{E}_B$, at location B, 100 nm to the right of the Cl ion?**



Checkpoint 1: Ask an instructor to check your work for credit.
You may proceed while you wait to be checked off.

2: Electric Field Due to a Single Particle

- Open IDLE for VPython. As always, you should begin the code with the following lines:

```
from __future__ import division
from visual import *
```

The extent to which you comment your code is up to you, but you need to at least define these four sections in the following order: Define Constants, Set Initial Values, Create Objects, and Calculations.

- **Insert these titles now.**
- **Define the following two constants:**

```
e = 1.6e-19
oofpez = 9e9
```

The first of these constants is the fundamental unit of charge and has units of Coulombs. The second constant is $\frac{1}{4\pi\epsilon_0}$ and has units of Nm^2 / C^2.

- **Also in the Define Constants section, add the following line:**

```
scalefactor = 1
```

The purpose of the scale factor will be explained later in this lab.

- **In the create objects section, type code to create the first particle (the particles will represent atoms), give the atom the charge of 3 excess protons, and create the arrow you will use to visualize the E field:**

```
atom1 = sphere(pos = vector(-300e-9,0,0), radius = 10e-9, color = color.red)
atom1.q = 3*e
Earrow = arrow(pos = vector(0,0,0), axis = vector(0,0,0), color = color.cyan)
```

As you may recall from Phys 172, there are three characteristics to a sphere: its position (pos), radius, and color. There are also three characteristics to an arrow: the position of its tail (pos), the vector property of the arrow (axis), and its color. This arrow initially has no magnitude. The position of the tail of the arrow represents the Observation Location, the location at which the electric field will be calculated. From the code for Earrow, it should be evident that the Observation Location has been chosen to be the origin (i.e. We are going to display the Electric Field produced

by atom1 at the origin) Now you can begin your calculations to make VPython calculate the electric field at the tail of the arrow. You will use Coulombs law to find the electric field:

$$\vec{E} = \frac{1}{4\pi\epsilon_0}\frac{q}{r^2}\hat{r}$$

When calculating the electric field at an observation location, the relative position vector, $\overrightarrow{r}$, always points from the source particle to the observation location. The two vector positions needed to calculate the relative position vector are denoted in VPython as:

atom1.pos
Earrow.pos

- **In your program, complete the code that correctly calculates the relative position vector. Name this vector r1.**

Now to find the magnitude of r1 in VPython, you need to use the mag().

- **Write a line of code to calculate the magnitude of the relative position vector. Name it r1mag.**

You can find a unit vector by dividing a vector by its magnitude.

- **Write a line of code to calculate the unit vector in the direction of the relative position vector. Name it r1hat.**

At this point, you have calculated r1, r1mag, and r1hat. These quantities are needed to calculate the Electric Field vector at the observation location.

- **Write down the symbolic algebraic equation used to calculate the electric field vector then convert it to VPython. Use the name E1 for the electric field vector.**

Initially, the arrow you created was temporarily assigned an axis = vector(0,0,0). In order to make this arrow represent the Electric Field, E1, the axis of this arrow must be reassigned to be equal to E1.

- **Enter the following line of code:**

Earrow.axis = E1

- **Run the program.**

You will notice that all you see is an arrow. You may recall from Physics 172 that scale factors were used as so to scale vectors that did not have units of length:

$$scalefactor = \frac{Desired\ magnitude}{Current\ magnitude}$$

The desired magnitude should be inferred from sizes of and distances between the objects in your system. In the current situation, we have a sphere located 300 nm from the origin having a radius of 10 nm. Therefore, in order to accommodate an arrow within the same screen, it should have a length with order of magnitude between ten and one hundred nanometers. (e.g. 100 nm).

- **To find the current magnitude (The magnitude of E1), add the following print statement to the end of your code:**

```
print ("The magnitude of E1 field =", mag(E1),"N/C")
```

- **Now calculate an appropriate scalefactor, and define scalefactor to be as such in the Define Constants section of your code.**

You MUST enter the scale factor as a number. If you enter a variable, the value of your scale factor may change during the program run and this will corrupt the physics.

- **In your code above, change Earrow.axis = E1 to instead be Earrow.axis = scalefactor*E1.**
- **Run your program, and check to make sure everything looks correct.**

You can tweak your scale factor to make your arrow slightly larger or smaller as necessary.

Checkpoint 2: Ask an instructor to check your work for credit. You may proceed while you wait to be checked off.

3: Electric Field Due to Several Atoms

You will now create an additional charged atom and display the net electric field at various observation locations. When introducing multiple particles, there may be several objects referencing the observation location. If one wishes to move the observation location, then that quantity may have to be changed in multiple places. It is convenient to define the observation location separately, and from then on simply reference that variable. This affords you the ease of varying the observation location while only changing one quantity.

- **In the Set Initial Values section, enter the following code:**

Obslocation = vector(0,0,0)

- **Then, where you previously defined Earrow, change the position of the arrow to be at the observation location. This line of code should now look like the following:**

Earrow = arrow(pos = Obslocation, axis = vector(0,0,0), color = color.cyan)

- **Immediately after the lines of code creating the first atom and giving it a charge, add code to create another atom. Name this sphere "atom2" and locate it at (100e-9 0, 0).**

- **Give the new particle a charge of an electron, -e.**

You will need to use the Superposition Principle to find the net electric field. Therefore, you need to calculate the electric field contribution from each particle individually at the observation location and then by adding the fields together, the net Electric Field can be obtained. You can perform each step of the calculation in groups, labeling each variable with its appropriate number, e.g.:

r1hat = r1/r1mag
r2hat = r2/r2mag

To calculate the magnitude of each relative position vector, you can use the mag command built into VPython:

r1mag = mag(r1)
r2mag = mag(r2)

- **After you calculate both electric fields, add a line of code to sum up the fields and find the net E field. Name the net field "Enet".**

- **Change your print statement to print out the net electric field:**

print ("Enet =", Enet,"N/C")

- **Lastly, change the arrow axis to represent the net electric field.**

Earrow.axis = scalefactor*Enet

The observation location should still be at (0, 0, 0) m.

- **Before you run your code, can you predict what the Enet will be? Draw a diagram labeling the locations of the atoms and their relative charges. Does this diagram look familiar?**
- **Run your code, and adjust your scale factor using your printed value of Enet.**
- **Compare the diagram and value for Enet to part (1) of the Warm-Up problem, do your values of Enet agree?**
- **Use your program to find an answer to part (2) of the Warm-Up problem. Where must you move the observation location to?**

Checkpoint 3: Ask an instructor to check your work for credit. You may proceed while you wait to be checked off.

4: Problem 2

Charge 1 (12e) is located at (-500, 0, 0) nm. Charge 2 (5e) is located at (600, 0, 0) nm. The observation location is (0, 0, 0) m.

(a) Where would you place an electron so that the net E field at the observation location would be 0? Hint: You may use your code to find the E- field produced from charges 1 and 2 at the origin. Then, what must the Electric Field of the electron be (at the origin) in order for the net E-Field to be zero?

(b) Add to your code an electron at the location you found in part (a). Run your code. Is the electric field zero at the origin? If not, why?

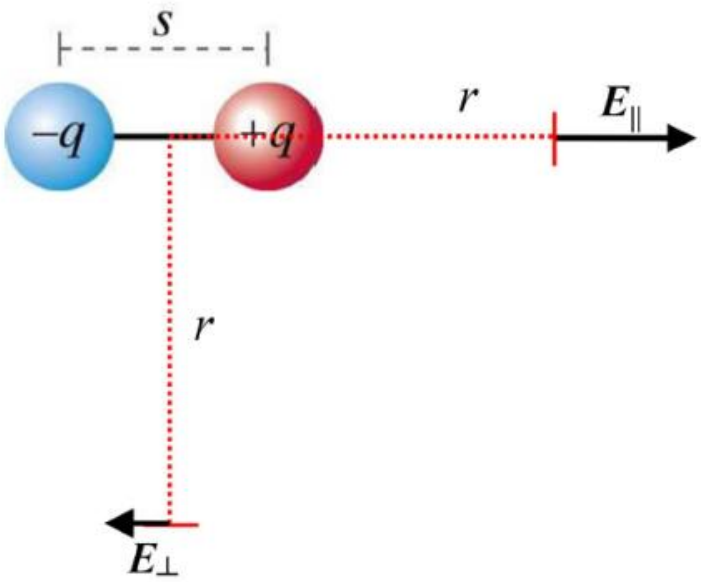
Checkpoint 4: Ask an instructor to check your work for credit. You may proceed while you wait to be checked off.

5: Electric Field of a Dipole

From investigating the electric field of a dipole in lecture, you have learned very simple, yet powerful formulas for calculating the E field of a dipole:

Along the *dipole axis*: $|\vec{E}_{\parallel}| = \frac{1}{4\pi\epsilon_0}\frac{2qs}{r^3}$

Perpendicular to the dipole axis: $|\vec{E}_{\perp}| = \frac{1}{4\pi\epsilon_0}\frac{qs}{r^3}$



To calculate the electric field at any location other than these two axes, you must use the superposition principle. Using VPython, you will create a program that allows easy calculation of the electric field at any point in space. Then, by having VPython display arrows representing the E- field at multiple locations, you will be able to visualize the E- field pattern of an electric dipole.

6: Electric Field at one location

You will now remove several lines and variables from your code in order to create a framework for the remainder of the lab.

- **Under the "Create Objects" section, remove any additional objects so that only atom1 and atom2 remain. Move the code creating Earrow to the bottom of the program.**
- **Towards the bottom of your code, eliminate the line that assigns Enet to be the arrow's axis.**
- **Within your "Calculations" section, retain all calculations for E1 and E2 (the calculations pertaining to atom1 and atom2), but remove any additional calculations. Keep the calculation for Enet, but it should only contain E1 and E2.**

Now you can begin constructing the new program.

- **Change the locations of atom1 and atom2 to be < 0.5e-9, 0, 0 > m and < -0.5e-9, 0, 0 > m, respectively.**
- **Give atom1 and atom2 a charge of -e and +e, respectively (this is opposite of the diagram, but that's ok).**
- **Make the negatively charged atom blue and the positively charge atom red.**
- **Change the radius of each atom to be 1 Angstrom, 1e-10 m. (This is more reasonable for the size of actual atoms).**
- **Change the observation location to < 1.8e-9, 1e-9, 0 > m.**

Obslocation = vector(1.8e-9,1.0e-9,0)

Since we have moved the arrow to the bottom of the screen, we now need to change how we calculate the relative position vectors r1 and r2 so they only reference the observation location directly

- **Change the lines that calculate the relative position vectors to:**

r1 = Obslocation - atom1.pos
r2 = Obslocation - atom2.pos

- **In the line of code that creates the arrow, change the axis to Enet times your scale factor. Use your answer to part (2) of the Warm-Up problem and the sizes/positions of the objects within the system to determine an appropriate scale factor.**

Earrow = arrow(pos = Obslocation, axis = Enet*scalefactor, color = color.orange)

- **Set the scale factor to the appropriate value. Run the code and draw what you see.**

Checkpoint 5: Ask an instructor to check your work for credit. You may proceed while you wait to be checked off.

7: Adding More Observation Locations

To see the electric field pattern of this dipole, you will extend your program to calculate the electric field at many different locations, all of which will lie on circles centered on the dipole. The circles will all have a radius of 2e-9 m. Instead of copying and pasting the code many times, you will place your calculations within a loop. During each iteration of the loop, VPython will create an arrow representing the Electric field at a different observation location. An easy and efficient method of moving the observation locations around a circle (of fixed radius) is to use polar coordinates.

- **In your "Set Initial Values" section, set the initial value of an angle, theta, to 0. Also, set the radius, r, to be 2e-9 m.**

```
theta = 0
r = 2e-9
```

- **Place the entirety of your "Calculations" section into a while loop that terminates when theta is 2π . Type the following before your calculations and indent all of your calculations:**

```
while theta <2*pi:
```

- **Since you are changing the observation location during each iteration, and since each calculation is dependent on the observation locations, you must add the following line of code to your loop, just beneath the while statement:**

```
Obslocation = vector(?,?,0)
```

Using theta and r, replace the question marks above with the appropriate polar values. VPython has the sine and cosine functions built-in, i.e., sin() and cos() are valid functions.

- **Add a print statement for the observation location just above the print statement for Enet:**

```
print ("The observation location is", Obslocation, "m")
print ("Enet =", Enet, "N/C")
```

- **At the very end of your loop, calculate the new value of theta with the following line:**

```
theta = theta + pi/6
```

This will allow the loop to iterate 12 times before the value of theta meets or exceeds 2pi.

- **Run your program.**

It should calculate and display the electric as arrows at 12 locations on a circle surrounding the dipole. If needed, adjust your scale factor.

- **Look at your display. Does it make sense? Do the magnitudes and directions of the 12 orange arrows make sense?**
- **Compare the two vectors along the dipole axis with the vector directly above and the vector directly below the dipole center. What is the relationship between their magnitudes?**

Checkpoint 6: Ask an instructor to check your work for credit. You may proceed while you wait to be checked off.

8: Adding Even More Locations

- Currently, you've plotted the electric field at 12 locations around a circle in the x-y plane using polar coordinates. Now plot the electric field along the surface of a sphere with radius r. (Hint: use spherical coordinates and place the original loop inside another while for the angle phi. You'll have to place appropriate bounds on phi.)
- Run your program
- Now plot the electric field everywhere inside a sphere with radius 5r. (Hint: place booth loops inside another loop that iterates r from 2e-9 to 10e-9.)
- Run your program
- Does your plot make sense? Does the magnitude of the arrows follow the distance dependence for the perpendicular and parallel axis given at the start of the lab? Does the electric field have the appropriate symmetry given that it came from a dipole?

Checkpoint 7: Ask an instructor to check your work for credit.