Lab 5: Potential Difference

OBJECTIVES

In this lab you will

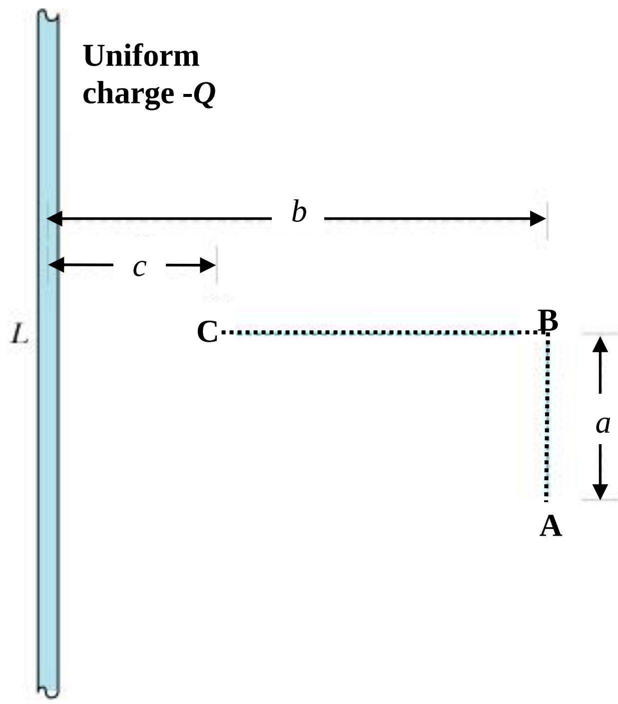
- Create a program to graph potential differences
- Generate a graph of potential difference as a marker moves along various paths
- Explain qualitatively the graphs you generate

Calculating the potential difference along paths near realistic charge distributions is difficult. It is particularly hard if the electric field varies significantly in the region. You will write a VPython program to calculate and display the potential difference as a marker moves along several paths near a charged rod. You will then write brief explanations of the graphs you generated.

1) Warm-Up Problem

Problem (1)

A long rod of length L carries a uniform charge –Q. Calculate the potential difference $V_A - V_C$. All of the distances are small compared to L. Explain your work carefully.



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

2) Potential Difference: Near a Charged Rod

You will use your code from Lab 4 which already contains the code to create a rod.

a) **Open your code from Lab 4 and save it under a new name so you will still have access to your Lab 4code.**

Most of the code from Lab 4 is still useful. The modifications to this code are tricky, so be careful. In order to calculate the change in potential over a large path, you will sum up the small changes in potential over short segments of the path. You will treat the electric field as constant over each small interval of the path. You will need to calculate the electric field in that small region. This electric field calculation will involve summing over all the contributions to the electric field that the small charge segments of the rod make. You will need to put the loop which finds the electric field inside another loop which finds the potential.

b) **Add the following two lines of code to your setup statements. The statements should now be:**

```
from visual import *
from visual.graph import *
from __future__ import division
scene = display(x=0, y=0, width=600, height = 600)
graph = gdisplay(x=600, y=0, width=400, height=300)
```

The second line preps VPython to make a graph. The fourth and fifth lines simply arrange the display windows so they do not overlap.

Do the following in your define constants section.

c) **Set N equal to 100. This will make it look and act like a line of charge.**

d) **Remove the definition of obslocation.**

e) **Define a time interval “deltat”.**

```
deltat=0.001
```

The choice of time interval is a tradeoff between how accurate you want your program to be and how fast you want it to run. If you feel your program is going too quickly, you can always shrink the time interval, however, if you feel your program is running to slowly you can only extend the time interval up to a certain point before you start affecting the accuracy of the program (You should be familiar with this from last semester or Lab 1).

Do the following in your initial values section.

f) **Set the initial value of a time variable, “t”, to be zero.**

g) **Set the initial value of our total potential difference, “deltaVtotal”, to be zero.**

You should already have code for the x-coordinate and Enet.

Do the following in your create objects section.

h) Enter the following line of code to create a graph.

```
Vgraph=gcurve(color=color.cyan)
```

This creates a graph called “Vgraph” which has a cyan data curve. This line does not tell VPython what to put on the graph.

i) Create a marker called “marker” so as you calculate the potential change along the path, you can see where you are. This marker should be a red sphere with radius 0.05 m initially located at <0, 0.15, 0> m.

j) Give the marker a velocity by defining “marker.v” to be <0, 0.5, 0> m/s

Do the following in your calculations sections.

k) Remove the arrow and all the print statements.

l) Put your entire calculations section in a while loop that stops once the y-coordinate of the marker’s position is greater than 1.15 m.

Your calculations section should now look like this:

```
while marker.pos.y<1.15:
    x=(deltax-L)/2
    while x < L/2:
        r=obslocation-vector(x,0,0)
        rmag=mag(r)
        rhat=r/mag(r)
        E=oofpez*deltaq/rmag**2*rhat
        Enet=Enet+E
        x=x+deltax
```

Notice how the part which calculates the electric field is indented twice. We will call the loop involving marker.pos.y the external loop and the loop involving x the internal loop.

You took out obslocation, so you need to change that line. You are interested in the electric field at the location of the marker.

m) In place of obslocation put the marker.pos variable.

VPython will calculate the net electric field at a certain location, but the next time through the loop you will want it to calculate an entirely new electric field. So, you need to zero the Enet variable before the internal loop.

n) Set Enet back to a vector with zero magnitude right before the internal loop starts.

Now you are ready start calculating the potential. The followings lines of code should be after the internal loop, and should only be indented once, so they are part of the external loop but not the internal loop.

You are going to calculate little changes in potential over small segments of the marker's path. On these small segments you are taking the electric field to be constant. Potential change is given by:

$$dV=-\vec{E}\cdot d\vec{l}$$

You could do the dot product calculation manually, however, VPython has a dot product function.

$$\vec{A}\cdot\vec{B}\rightarrow \mathrm{dot}\left(A,B\right)$$

You will also need to figure out the length of the line element i.e. how far the marker traveled during the time interval. This length is simply the speed of the marker times the time interval.

o) Enter a line of code to calculate the potential change over a single time interval. Name that potential change "deltaV".

p) Like you did in finding the net electric field, add this deltaV to the value of deltaVtotal with the following line.

```
deltaVtotal=deltaVtotal+deltaV
```

q) Add one time interval to the time.

```
t=t+deltat
```

r) Use the position update formula to move the marker.

```
marker.pos=marker.pos+marker.v*deltat
```

s) Add a data point to the graph with the following line.

```
Vgraph.plot(pos=(t,deltaVtotal))
```

The graph will then be of potential difference versus time.

t) After the calculations and outside both loops put a print statement to print out deltaVtotal.

u) Run your code.

You should see the marker travel away from the rod as VPython generates a graph.

v) Record the value of deltaVtotal and sketch a copy of the graph into your workspace, be sure to give its axes labels with units and the graph a title.

w) Write an explanation of how you know the general shape of this graph is correct.

x) **Now start the marker at <-1, 0.15, 0> m and move it to <1, 0.15, 0> m with a velocity of <0.5, 0, 0> m/s. BEFORE YOU RUN YOUR CODE, sketch a prediction of the graph's shape.**

y) **Record the value of deltaVtotal and sketch a copy of the graph into your workspace, be sure to give its axes labels with units and the graph a title.**

z) **Write and explanation of how you know the general shape of this graph and the value you received for deltaVtotal are correct.**

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

3) Potential Difference Near a Charged Rod: Path Independence

a) **Copy your entire calculations section and place a copy directly after the original.**

b) **In between the original calculations section and the copy you just made, insert a line which redefines marker.v i.e. changes the velocity of the marker.**

c) **Now make the marker travel from <0, 0.15, 0> m to <1, 1.15, 0> m, and then continue on to <0, 1.15, 0 > m.**

d) **Sketch a copy of the graph in your work space.**

e) **There are two distinct parts to the graph. Write a brief statement explaining the shape of each one.**

f) **Compare this deltaVtotal from this run to that from your first run.**

CHECKPOINT 3: Ask an instructor to check your work for credit.