Phys 232 – Lab 6 Ch 19 The Electric Field of a Surface-Charge Gradient

Objectives

In this lab you will do the following:

- Computationally model the electric field due to a linear surface-charge gradient along a cylindrical wire.

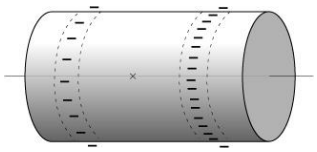
Describe the following attributes of a *metal wire* in steady state vs. equilibrium:

	Metal wire in steady state (current flowing)	Metal wire in Static equilibrium
Location of excess charge		
Motion of mobile electrons		
E inside the metal wire		

Background

In Chapter 19, you saw that the electric field inside a current-carrying wire must be parallel to the wire for current to flow along the wire; it must also be of constant magnitude for the current strength to be constant (assuming the wire has constant diameter and conductivity.) In addition, you know that any excess charges (the sources of electric field) will be on the surface of a wire. That begs the question 'what kind of surface charge distribution produces a uniform electric field along the length of a wire (and thus drives a constant current)?'

As you'll demonstrate in your simulation, a surface-charge density that changes linearly along the length of a cylindrical wire can provide that uniform electric field parallel to the wire's length.



This figure roughly illustrates such a surface charge density: the ring of charge on the right is twice as negative as that on the left; you can imagine yet another ring of charge further to the right and three times as negative as the first, etc. In your program, you'll make several such rings (of increasing charge density) down the length of a cylinder/wire.

Recall that to find the electric field due a charge distribution, you must divide the charge into small pieces and treat each piece as a point charge. The net electric field is the vector sum of the electric fields due to all of the pieces.

You may recall that in our previous lab you'd started with an old program for the electric field of a single ring of charge, and you'd modified it to provide the magnetic field of a stack of rings of

current (a solenoid.) This week, you'll start with the same old program (electric field of a single ring of charge) and modify it to provide the *electric* field of a stack of *charged* rings with varying charge.

I. Modifying your Program

For your convenience, copies of RingE.py and Solenoid.py are available in WebAssign to download. I'd suggest downloading and resaving RingE.py as WireE1.py and modifying it for this lab, but also downloading and opening Solenoid.py so you can look at it to get good ideas about how to do some of the things you'll need to do for this lab.

A. Uniformly Charged wire

As a first step toward simulating a wire of linearly-varying charge density, modify your code to simply produce a *uniformly-charged* wire (stack of rings with the same charge.) You'll see that the field it produces *wouldn't* drive a uniform current, but this version of the program is a good step in developing your program. More specifically,

- the length (`L`) of the wire (use 2 cm)
- make sure the wire is centered on the origin and stretches along the y axis
- the radius (`R`) of the wire (use 2 mm)
- the number of rings (`Nrings`) into which the cylinder should be divided (start with 11)
- the charge (`Q`) of each ring (use 9e-7 / Nrings)
- an `Escale` around e-5 should be good (feel free to tweak it a bit)
- Narrow the range of observation locations to run between x = 1.25*R and -1.25*R and between y =2*R and -2*R so you focus on the mid-section of the wire. This is so we don't bother looking at the unrealistically-disconnected ends of this simulated wire (in a real circuit, the ends would necessarily be connected to something else.)

By referencing the Solenoid program, you should be able to modify the old `RingE` program satisfactorily.

When you are satisfied, check with a neighboring group and then upload your program.

B. Wire of Linearly-Varying Charge

Now you'll want to make some changes to smoothly vary the charge from one ring to the next. Along the way, you'll meet a couple of new programming tools and strategies.

- Save your program as WireE2.py.

Assigning Different Rings Different Charges

Just as each ring has a different position that's recorded as that ring's `pos` attribute, each ring should also have a different charge, so you'll give the ring a new attribute, `Q`. While the ring object has some built-in attributes (`pos`, `axis`, `color`, and `thickness`), you can give it additional attributes such as, in this case, `Q` for charge.

- First, up in the `constants` section of your program, define `Qfirst` = 9e-7 / Nrings, `Qlast = -Qfirst`, and `deltaQ = (Qlast - Qfirst)/Nrings`. The latter will be the difference between charges on consecutive rings.
- Just above the loop where you create the stack of rings, define `Qtemp = Qfirst`

- *Inside* the `source = ring(…)` add `Q = Qtemp`.
- Below that, still inside the loop, add a line to update `Qtemp`'s values for use when you create the next ring: `Qtemp = Qtemp + deltaQ`.

Charge-dependent Ring Colors

One merely-aesthetic change to your program: it would be great if each ring's color somehow represented its charge. There are two ways to specify the color of an object in Python: by assigning one of the pre-defined colors (`color =color.red, color =color.cyan`, etc.) as we've done up until now, or by specifying how strongly Red (`color = (1,0,0)`), Green (`color=(0,1,0)`), and Blue (`color = (0,0,1)`) it is on a scale of 0 to 1.

- Within the `source = ring(…)` line, define the color as `color = (abs(Qtemp/Qfirst),1-Qtemp/Qfirst,0)`. That will make the first ring purely red, the middle ring purely green, and the last ring yellow.

Field due to Varying Charges

Now, down in the `calculations` section of your program, you'll want to change a couple of lines to make use of each ring's charge when determining its contribution to the total electric field. You probably have a "`for theta in sourcethetas:`" loop nestled inside a "`for y in sourceys:`" which is itself nestled inside a "`for obsloc in obslocations:`" loop (or something to this effect). You also probably have a line that reads "`dQ = Q/Ns`" (or something to this effect) which defines the amount of charge on each differentially-small wedge of each ring. Here's what you'll want to change about these lines.

- "`for y in sourceys:`" should change to "`for source in sources:`"
- "`dQ = Q/Ns`" should change to "`dQ = source.Q/Ns`" and should be positioned just under that "`for source in sources:`" line (and indented so it's in the loop.)
- Where you identify the source's location as something `like vector(R*cos(theta), y, R*sin(theta))`, change the "`y`" to "`source.y`".

What these changes achieve is for each ring or "`source`" that makes up your wire or "`sources`" list, you use *its* charge, `source.Q`, and *its* elevation, `source.y`, when calculating its contribution to the electric field.

To make sure you've got enough rings to realistically model the wire you should print out a representative value of E's magnitude and increase the number of rings until this value minimally changes. More specifically,

- at the bottom of your program, but just inside the loop over observation locations add the lines

```
if obsloc == vector(0,0,0):
  print(mag(E))
```

with the 11 rings, what is the value that's printed?