Zoe Kearney

Problem set 2

Phys 512
September 25, 2021

Problem 1

To set up this equation we can use the following integral which exhibits the electric field at a point from a thin shell:

$$\vec{E_z} = \frac{1}{4\pi\epsilon_0}(2\pi R^2\sigma)\int_{-1}^{1}\frac{z - Ru}{(R^2 + z^2 - 2Rzu)^{3/2}}du$$

Where z is the distance to the point, R is the radius of the shell and u is $cos\theta$. First we can solve this using a Simpson's integrator.

Then we can solve using quad from scipy. For both methods, re-doing the integration for multiple values of z relative to R allows us to see the behavior of the electric field with respect to R, and more importantly what happens when $z = R$ and then flips from being less than R to greater than. As seen in the final graphs where E is plotted versus z, we can see that the shape of the function is what we would expect. Using Gauss' law and thinking about what the E field would be inside and outside the shell, it makes sense that the E field is 0 inside the sphere since there is no enclosed charge. Then, since the enclosed charge is no longer changing once we are outside of the sphere, the magnitude of the E field will only depend on the inverse value of z^2 therefore creating the falloff we see where the point where E is the greatest is at the surface when z is the smallest without being smaller than R. This point is a discontinuity so I took the limit of a value very close to R for z to show the behavior graphically.

Problem 2

See code. In order to reduce the function calls we have to save the old x array and associated y array. After the first run where extra= None the x array and y array are initialized. Each time the integrator is called after that it is able to take the previous x values and y values and compare which x values are repeated in the new x array. In the new y array, which I initialize as nans to begin with then is filled with any y values associated with repeated x values. The remaining components in the new y array that are still nan I then re calculated the y by calling the function. The code should then append that x and y run to the next tuple passed through extra. It is still not quite working like I would expect as the extra values are not properly concatenating with the previous ones and so the extra tuple doesn't have all x values to check for repeats rather just some. It is still at comparable with what was done in class, but by including all previous x and y values would reduce the function calls even further.

Problem 3

See code.