PHYS 512 Problem Set 2

Teophile Lemay, 281081252

1

The electric field at a point near a uniformly charged spherical shell can be calculated by integrating over infinitesimal rings to make up the sphere. The electric field due to each ring is calculated starting from Coulomb's law $E = \frac{kQ}{r^2}$. Using the variables from the ring diagram in Figure 1, integrating over the circumference of the ring, given a uniform line charge density ρ gives $E = \frac{k2\pi\lambda R'}{r^2}$. However, due to the radial symmetry of the ring, only the z component of the electric field will remain for a point along the ring's axis so

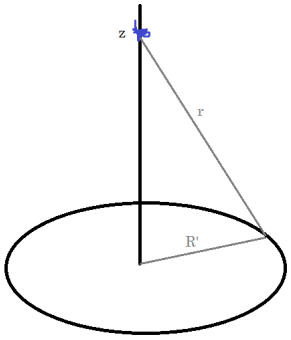
$$E_{ring} = \frac{k2\pi\rho R'}{r^2}\cos\theta = \frac{k2\pi\rho R'}{r^2}\frac{z}{r} = \frac{k2\pi\rho R' z}{r^3} \ .$$

Finally, r can be expressed in terms of z and R':

$$E_{ring} = \frac{k2\pi\rho R' z}{(z^2 + R'^2)^{3/2}} \ .$$

Now, the integral over the sphere can be evaluated by taking the sum of rings making up the sphere for R' the

Figure 1: Charged ring diagram



radius of each infinitesimal ring and z' the height of the point relative to each ring. Using an azimuth angle θ:

$$R' = \frac{R}{\sin(\theta))}$$

$$z' = z + R\cos\theta$$

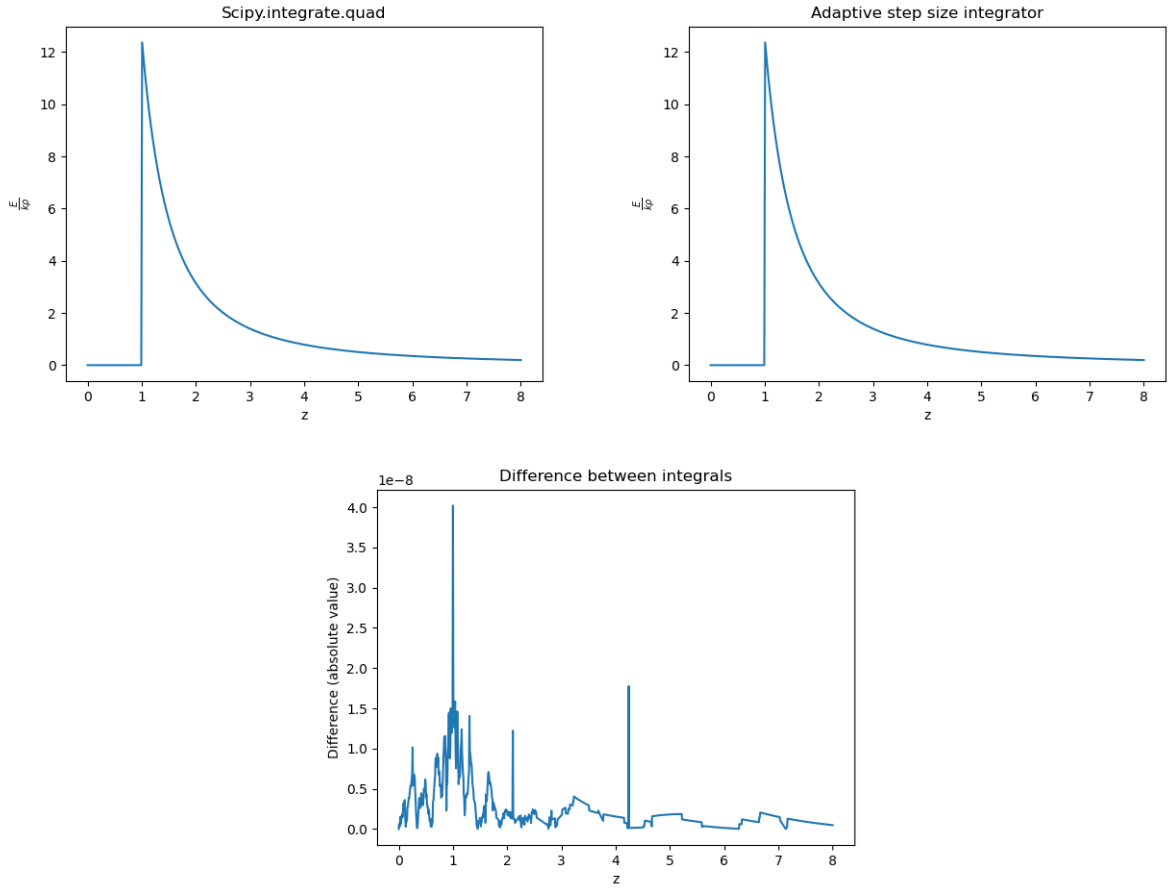
$$E_{ring} = E_{ring}(\theta, z, R) = \frac{k2\pi\rho R\sin\theta(z + R\cos(\theta)}{((z + R\cos\theta)^2 + (R\sin\theta)^2)^{3/2}}$$

$$E_z = \int_0^\pi d\theta E_{ring}(\theta, z, R)$$

Figure 2 shows the results of the numerical integration of the electric field at a point relative to the charged sphere using SciPy's "scipy.integrate.quad" function and my adaptive step size integrator (see problem 2). For simplicity,

the integrals were calculated for a sphere of radius 1, and the outputs are in units of $E/k\rho$ since the charge of the sphere is arbitrary. As shown in the plots of E and the difference between the functions, both numerical integrators produce very similar results. Indeed, the maximum difference between the results is on the order of 10^{-8}, and occurs at $z = 1$, where there should be a singularity in E at the surface of the sphere. Despite the maximum difference occurring at the expected singularity, both functions perform nearly identically, simply producing a sharp peak around $z = R = 1$.

Figure 2: Electric Field at point z relative to center of charged sphere of radius 1



2

My adaptive step size integration function uses Simpson's rule. For the given interval $[a, b]$, the midpoint $m = (a+b)/2$ is calculated and the function is evaluated at the start, midpoint, and end of the interval. These values are used to compute the integral from a to b using Simpson's rule. Next, the secondary midpoints ($m' = (a+m)/2$ and $m'' = (m+b)/2$) are calculated and the function is also evaluated at these points. With these points, the integral is calculated again using Simpson's rule, but with two smaller intervals, and compared to the previous result. If the difference of the two intervals is within a set tolerance ($\epsilon = 10^{-8}$ was chosen arbitrarily), then the integral is returned (the two interval result is always taken since error for Simpson's rule is proportional to dx^5 so smaller intervals will produce smaller error).

If the resulting integrals do not match, my function uses recursion to call itself over each of the sub-intervals:

$$\int_a^b dx f = \int_a^m dx f + \int_m^b dx f$$

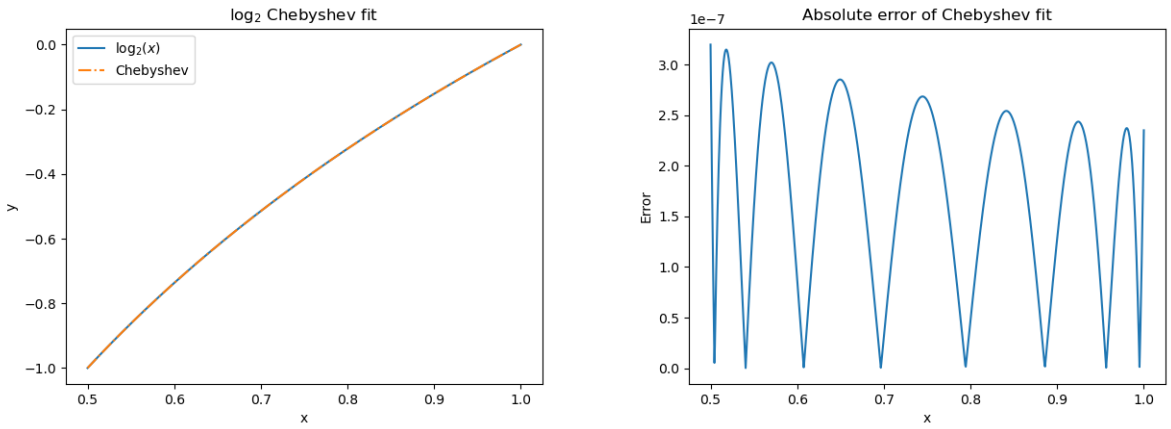
The recursion is limited by the size of the sub-intervals made by the function. If dx ever becomes smaller than 10^{-15} for any three point interval, then rounding error becomes a serious consideration and the function stops.

3

3.1 a)

In order to fit a Chebyshev polynomial to $\log_2$ on the interval $[0, 5, 1]$, I sampled the function at 51 evenly spaced points on the interval. A Chebyshev polynomial of order 25 was chosen arbitrarily to fit the sampled points using "numpy.polynomial.chebyshev.chebfit". Only the first 8 terms were required to model $\log_2$ with accuracy greater than 10^{-6} (see figure 3). The coefficients used were also saved to a text file so that they could be re-used in the second part of the question. Code for this section is found in "Q3_chebyshev_log.py"

Figure 3: Chebyshev polynomial fit of $\log_2$



3.2 b)

The natural logarithm of any real, positive value can be easily expressed in terms of $\log_2$ using logarithm rules:

$$\ln(x) = \frac{\log_2(x)}{\log_2(e)} \ .$$

Next, due to the nature of floating point numbers, I only need to be able to evaluate $\log_2(x)$ for $x \in [0.5, 1]$ since every positive number is represented in terms of a mantissa multiplied by 2 to the power of some exponent. Breaking down x in this way gives

$$\ln(x) = \frac{\log_2(\text{mantissa} \cdot 2^{\text{exponent}})}{\log_2(e)} = \frac{\log_2(\text{mantissa}) + \text{exponent} \cdot \log_2(2)}{\log_2(2)} = \frac{\log_2(\text{mantissa}) + \text{exponent}}{\log_2(e)} \ .$$

For all positive real numbers, the mantissa is within the range $[0.5, 1]$ which means that the coefficients calculated from my previous Chebyshev fit will suffice. As shown in figure 4, this method produces a very good model of the natural logarithm, with absolute error less than 10^{-6} over the tested interval. The code for this section, including the "mylog2" function is found in the file named "Q3_chebyshev_ln.py".

Figure 4: Natural logarithm using Chebyshev fit of $\log_2$