Problem 1

The integral to calculate the electric field a distance z away can be expressed as:

$$E(z) = \frac{R^2\sigma}{2\epsilon_0}\int_0^\pi \frac{(z - R\cos\theta)\sin\theta}{(R^2 + z^2 - 2Rz\cos\theta)^{3/2}}d\theta, \tag{1}$$

where R is the radius of the shell and σ is the surface charge density.

When running this, I found that my integrator was unable to deal with the singularity but would instead run into a `RecursionError` where it was unable to satisfy the tolerance set (since the value it was trying to compute was ∞). To compensate, I skipped over this value and set the value of the electric field at the singularity to 0. On the other hand, `scipy.integrate.quad` did not care about the singularity (i.e. it didn't raise any warnings or errors). Instead, it assumed the value was the average between it's neighbouring points. I've plotted some trial results below.

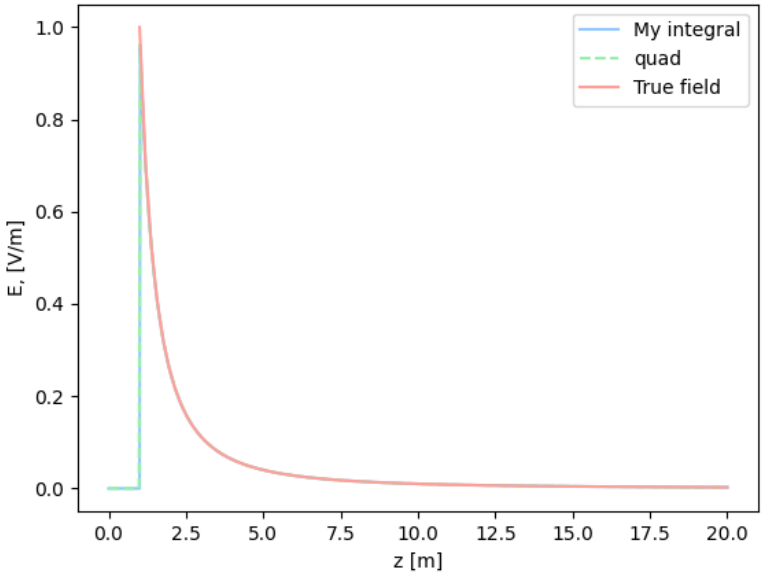

Figure 1: Electric Field as a function of distance from the center of the shell with surface charge density $\sigma = 1$ and radius $R = 1$. Assumed $\epsilon_0 = 1$ for simplicity. At $z = R = 1$, `scipy.integrate.quad` assumed $E(1) = \frac{E(1.01)-E(0.99)}{2} = 0.5$ V/m.

Problem 2

No explanations needed for this problem. Everything is provided in the code.

Problem 3

a)

Consider the linear re-mapping:

$$f : x \mapsto ax + b \equiv x' \tag{2}$$

for $x' \in (-1, 1)$. To determine a and b, we first notice that we'd like the endpoints of our desired fit interval, $x \in (0.5, 1)$, to map to :

$$0.5 \mapsto -1,$$
$$1 \mapsto 1,$$

respectively. Plugging in, we obtain the following set of linear equations:

$$0.5a + b = -1$$
$$a + b = 1$$

which is solved for $a = 4$ and $b = -3$. This provides us a way of re-scaling our x values to be within $[-1, 1]$ where the Chebyshev polynomials are defined.

b)

Suppose we have a positive number x. We can express this number as:

$$x = M \times 2^y, \tag{3}$$

where M is the mantissa which lies in the open interval (0.5,1) and y is a signed integer. Next, recall the logarithm change of basis rule:

$$\log_b a = \frac{\log_d a}{\log_d b}. \tag{4}$$

In our case, we wish to compute $\ln(x)$, so the above becomes:

$$\log_e x = \frac{\log_2 x}{\log_2 e}. \tag{5}$$

However, notice that by writing x as in equation (3), we see that:

$$\begin{aligned}\log_2 x &= \log_2(M \times 2^y) \\ &= \log_2 M + \log_2 2^y \\ &= \log_2 M + y.\end{aligned}$$

Lastly, utilizing the fact that we know that $\log_2 e \approx 1.4426950408889634$, the natural logarithm for any positive x can be computed as:

$$\log_e x = \frac{\log_2 M + y}{1.4426950408889634}, \tag{6}$$

where M is the mantissa and y is the exponent.

Bonus

See `problem_3_bonus.py`. I repeated the above process but fitting the Legendre polynomials and comparing them to the Chebyshev results. I noticed that the errors were comparable to one another after running the code for multiple values of x. The print statement provides the estimated max error (computed by summing over the truncated terms) as well as the true error. Both groups of polynomials performed equally as well.