PHYS 512 Assigment 1

Pauline Pestre - 260631905

September 2020

Problem 1:

(a) Writing the Taylor series expansions of the known terms:

$$
\begin{aligned}
f(x+\delta) &= f(x) + \delta f'(x) + \frac{\delta^2}{2} f''(x) + \frac{\delta^3}{6} f''(x) + \mathcal{O}(\delta^4) + \mathcal{O}(\delta^5) \\
f(x-\delta) &= f(x) - \delta f'(x) + \frac{\delta^2}{2} f''(x) - \frac{\delta^3}{6} f''(x) + \mathcal{O}(\delta^4) + \mathcal{O}(\delta^5) \\
f(x+2\delta) &= f(x) + 2\delta f'(x) + \frac{(2\delta)^2}{2} f''(x) + \frac{(2\delta)^3}{6} f'''(x) + \mathcal{O}(\delta^4) + \mathcal{O}(\delta^5) \\
f(x-2\delta) &= f(x) - 2\delta f'(x) + \frac{(2\delta)^2}{2} f''(x) - \frac{(2\delta)^3}{6} f'''(x) + \mathcal{O}(\delta^4) + \mathcal{O}(\delta^5)
\end{aligned}
$$

We get a first estimate of the derivative of f between $x \pm \delta$:

$$
\begin{aligned}
f'_\delta(x) &= \frac{1}{2\delta}\Big(f(x+\delta) - f(x-\delta)\Big) && (1) \\
&= \frac{2}{2\delta}\Big(\delta f'(x) + \frac{\delta^3}{6} f'''(x) + \mathcal{O}_\delta(\delta^5)\Big) \\
&= f'(x) + \frac{\delta^2}{6} f'''(x) + \frac{1}{\delta}\mathcal{O}_\delta(\delta^5) && (2)
\end{aligned}
$$

where the odd order terms of the Taylor series have cancelled. Similarly, for the derivative between $x \pm 2\delta$:

$$
\begin{aligned}
f'_{2\delta}(x) &= \frac{1}{4\delta}\Big(f(x+2\delta) - f(x-2\delta)\Big) && (3) \\
&= \frac{2}{4\delta}\Big(2\delta f'(x) + \frac{(2\delta)^3}{6} f'''(x) + \mathcal{O}_{2\delta}(\delta^5)\Big) \\
&= f'(x) + \frac{2\delta^2}{3} f'''(x) + \frac{1}{2\delta}\mathcal{O}_\delta(2\delta^5) && (4)
\end{aligned}
$$

Combining (2) and (4) cancels the third order term and gives our new estimate of the derivative:

$$
\begin{aligned}
4f'_\delta - f'_{2\delta} &= 3f'(x) \\
f'(x) &= \frac{4}{3} f'_\delta - \frac{1}{3} f'_{2\delta} && (5)
\end{aligned}
$$

Plugging in (3) and (4):

$$
\begin{aligned}
f'(x) &= \frac{4}{3}\frac{1}{2\delta}\Big(f(x+\delta) - f(x-\delta)\Big) - \frac{1}{3}\frac{1}{4\delta}\Big((f(x+2\delta) - f(x-2\delta)\Big) \\
&= \frac{2}{3\delta}\Big(f(x+\delta) - f(x-\delta)\Big) - \frac{1}{12\delta}\Big(f(x+2\delta) - f(x-2\delta)\Big)
\end{aligned}
$$

(b) First we need the fifth order term:

$$\frac{4}{3}\frac{1}{\delta}\mathcal{O}_{\delta}(\delta^5) - \frac{1}{3}\frac{1}{2\delta}\mathcal{O}_{2\delta}(\delta^5) = \frac{4}{3}\frac{1}{\delta}\frac{\delta^5}{120}f^{(5)}(x) - \frac{1}{3}\frac{1}{2\delta}\frac{(2\delta)^5}{120}f^{(5)}(x) \tag{6}$$

$$= -\frac{\delta^4}{30}f^{(5)}(x) \tag{7}$$

Subtracting the values in (1) and (3) leads to round off error of order $\frac{\epsilon f(x)}{\delta}$ where ϵ is the machine precision. The total error is optimized when these cancel each other, so

$$\frac{\epsilon f(x)}{\delta} = \frac{\delta^4}{30}f^{(5)}(x)$$

$$\delta = \left(\frac{30\epsilon f(x)}{f^{(5)}(x)}\right)^{\frac{1}{5}}$$

Problem 2:

Using interp1d, the code problem2.py returns a function TofV(V) that returns the interpolated termperature.

```
yourvalue=1.2
if yourvalue>np.max(V0) or yourvalue<np.min(V0):
    print(''Hmm that's not right")
else:
    print(''At your value %s V, the temperature is %.1f K"
    %(yourvalue, TofV(yourvalue)))
```

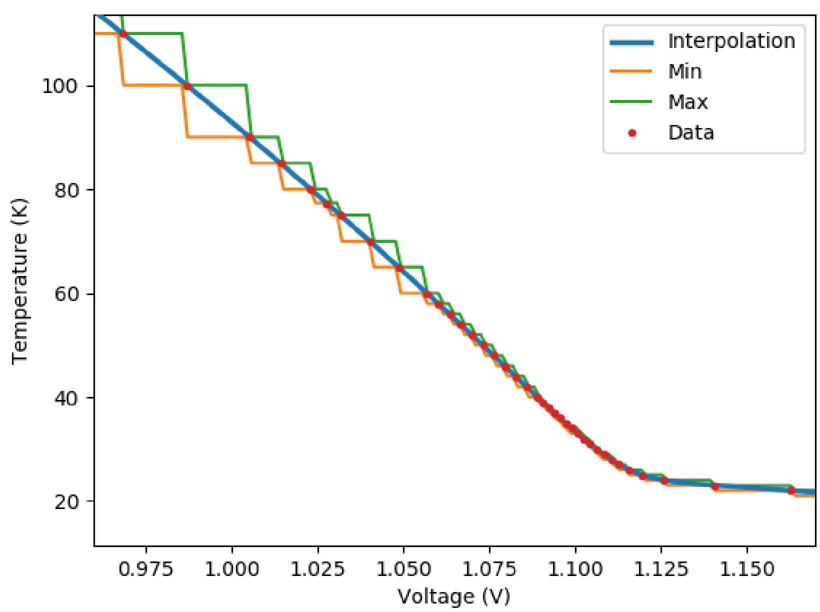


Figure 1: Temperature as a function of voltage, with cubic spline interpolation. The error at each point can be estimated from the difference between the two neighboring data points.

Problem 3:

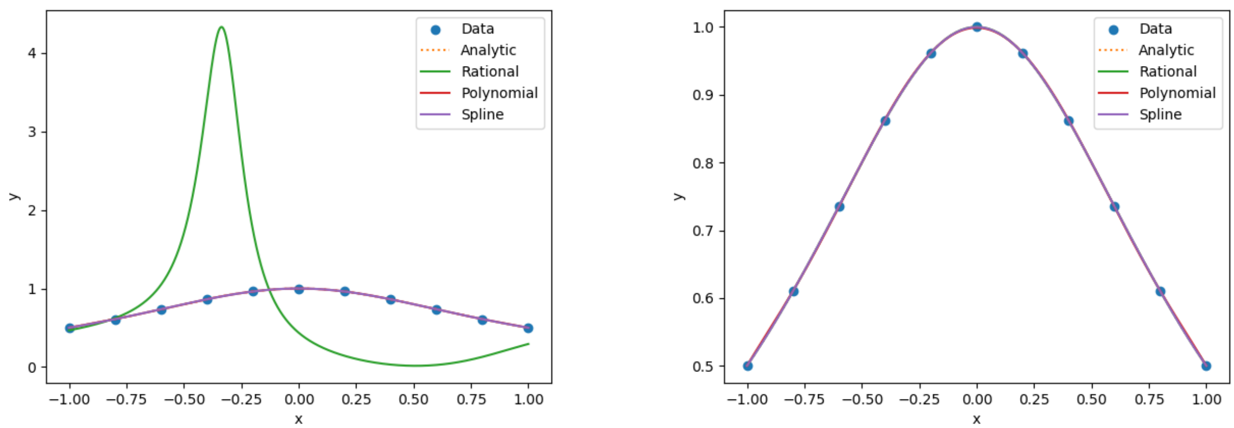
In the attached code problem3.py, I used 11 points to fit to. For cosine, the standard deviation of the difference of each fit from the true value prints:

```
Rational: 3.1818815673905338e-09
Polynomial: 9.926766094087246e-10
Spline: 3.141686973501803e-05
```

Here n=5, m=7 for the rational function. The polynomial is of order 10, the highest order that's not ill-conditioned. Lower orders, down to 4 give reasonable errors too.

For a Lorentzian, the rational function fit goes crazy. The errors are:

```
Rational: 1.0461493130109722
Polynomial: 0.0010212692604544447
Spline: 0.00010660659657017622
```



(a) (b)

Figure 2: The rational function fit fails when using np.linalg.inv (a) and succeeds using np.linalg.pinv (b).

Looking at the matrix computed for the fit, the determinant is very close to zero. This can't be properly inverted by linalg.inv, since the matrix is singular, but .pinv returns the pseudo inverse. The resulting error is then:

```
Rational: 2.419959220299023e-16
Polynomial: 0.0010212692604544447
Spline: 0.00010660659657017622
```

Problem 4:

There's a singularity in the integral at z=R. This point doesn't stop quad, but my integrator couldn't converge. If the function is changing too quickly, the difference between the adjacent values will never be within the tolerance (here set to $1e-5$). I added a try/accept to replace values that don't converge with infinity.

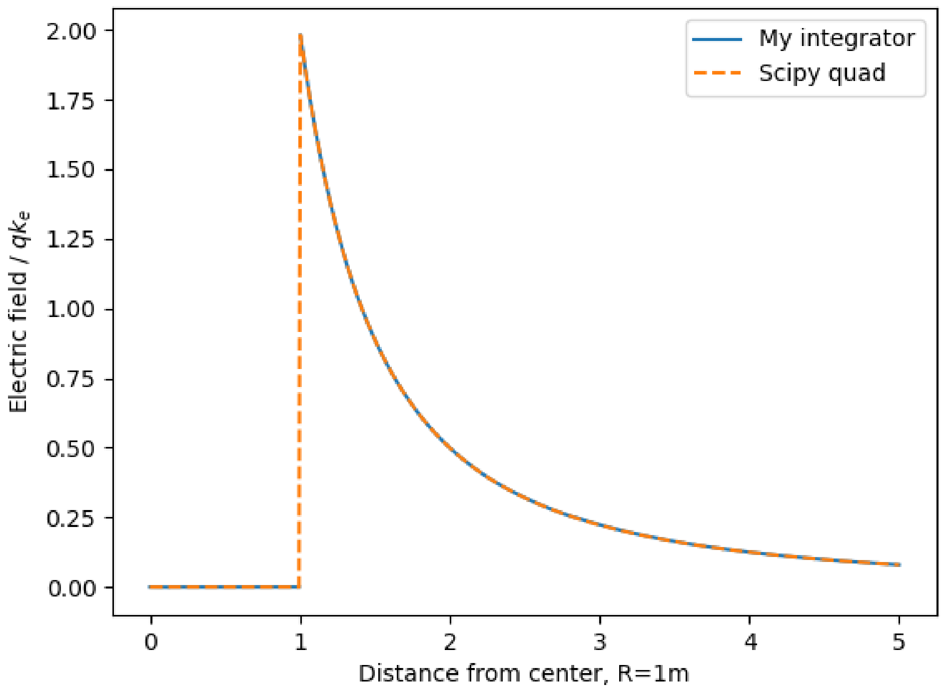


Figure 3: Electric field at a distance z from the center of a spherical shell.