PHYS 512 Assignment 1

Gabrielle Mitchell
400023877

Due Date: Sept. 18, 2020.

Problem 1

a)

See 1a.pdf

b)

See 1a.pdf
Refer to `A1_Q1_1.py` to output to see the error (third column). The output for $f(x) = exp(x)$ is :

```
-7.0 2.718281828887707 4.286619947890813e-10
-6.5 2.718281828421743 3.730216135977571e-11
-6.0 2.7182818282585806 2.0046453386157737e-10
-5.5 2.71828182849196 3.291500405566694e-11
-5.0 2.7182818284769237 1.787858749935367e-11
-4.5 2.7182818284697245 1.0679457318474306e-11
-4.0 2.71828182845916 1.1501910535116622e-13
-3.5 2.7182818284587236 3.2152058793144533e-13
-3.0 2.7182818284585317 5.133671265866724e-13
-2.5 2.71828182844997 9.07496300328603e-12
-2.0 2.718281827552945 9.061000838528344e-10
-1.5 2.7182817378388666 9.062017847227821e-08
-1.0 2.71827275672649 9.071732554932765e-06
```

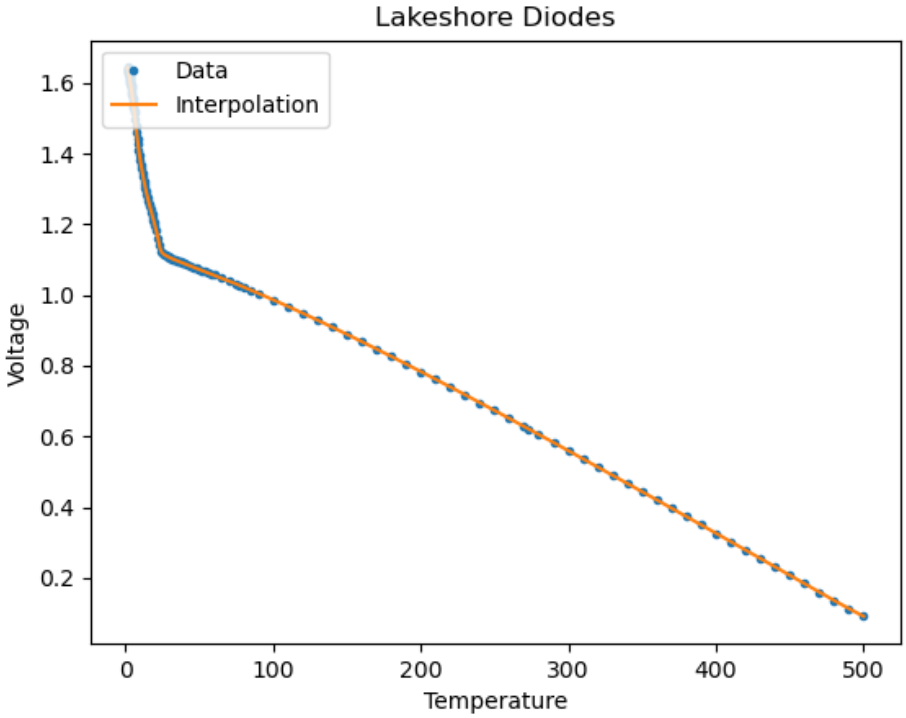
We can see that the error is minimized between $\delta = 10^{-3}$ as expected from 1a.pdf.
For $f(x) = exp(0.01x)$, the output is:

```
-7.0 0.010100501361224682 0.010100501260219665
-6.5 0.010100501885185995 0.010100501784180978
-6.0 0.010100501638780438 0.010100501537775421
-5.5 0.010100501692090156 0.010100501591085139
-5.0 0.010100501666536013 0.010100501565530997
-4.5 0.01010050167278057 0.010100501571775554
-4.0 0.010100501670606832 0.010100501569601815
-3.5 0.010100501670732584 0.010100501569727567
-3.0 0.010100501670754861 0.010100501569749844
-2.5 0.010100501670820355 0.010100501569815339
-2.0 0.010100501670836278 0.010100501569831262
-1.5 0.010100501670843762 0.010100501569838745
-1.0 0.010100501670840164 0.010100501569835147
```

The error is large and almost the same for all of the spacings. This is likely due to the roundoff error accumulating from the 0.01 in the exponent.

Problem 2

Refer to `A1_Q2_1.py`. I interpolated using cubic splines.



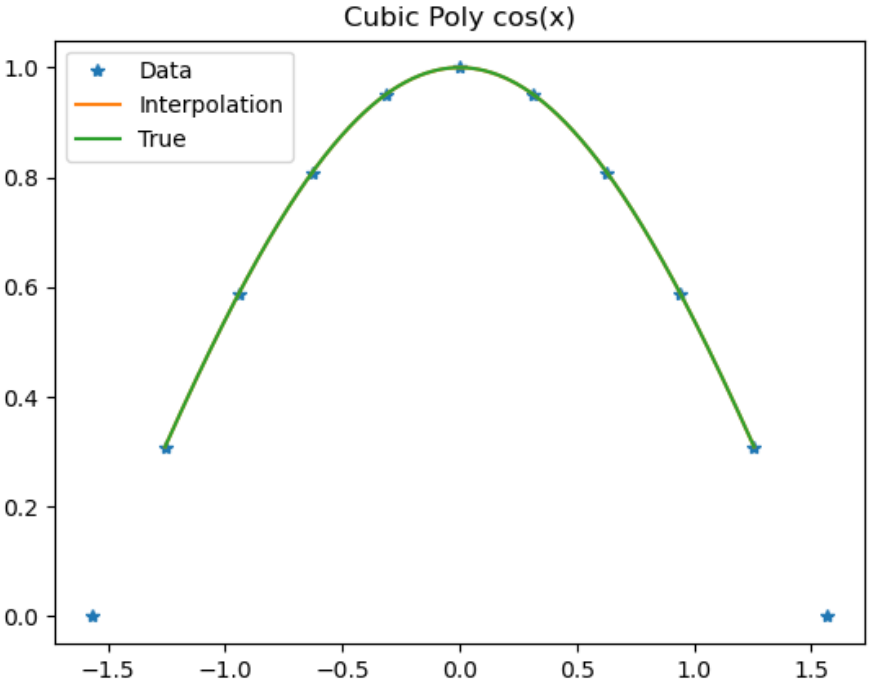
I estimated the error by doing same algorithm, but with half as many points (i.e. used every other data point), then finding the difference between those values and the values from the first higher resolution interpolation. Then I took the largest of those differences to be the error upper bound. The output for the upper bound on the error estimate is:

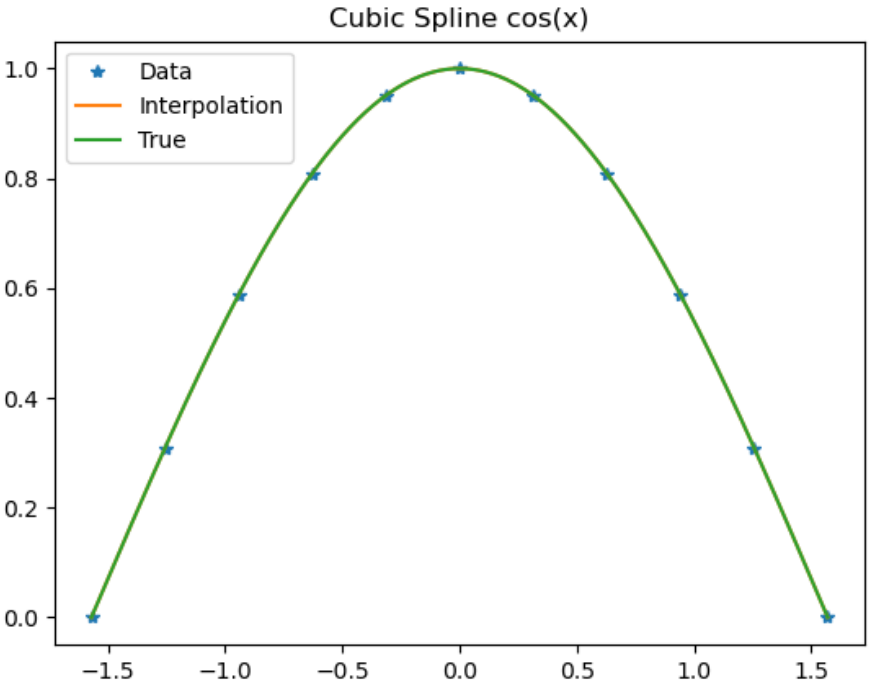
```
my error estimate is  0.0015828650664011512
```

Problem 3

Refer to `A1_Q3_1.py` for the cubic polynomial and the cubic spline interpolations of $cos(x)$.
The error outputs are:

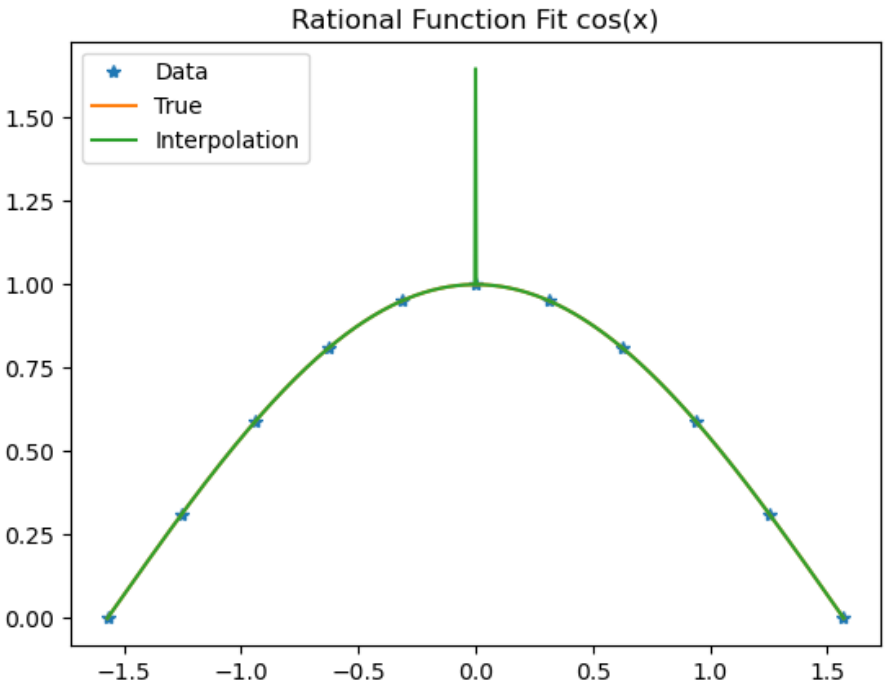
```
my spline rms error is  3.142471741294897e-05
my cubic poly rms error is 6.185519366744189e-05
```





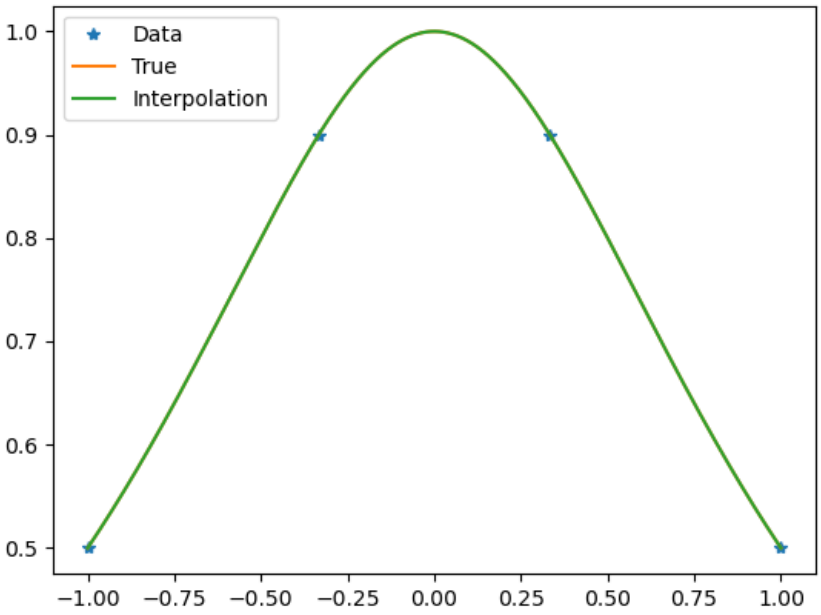
Refer to `A1_Q3_2.py` for the rational function interpolation of $cos(x)$.
The error output is:

```
my ratfit rms error is  0.020434665679561884
```



Refer to `A1_Q3_3.py` for the rational function interpolation of the Lorentz function.
For $n = 2, m = 3$ The output gives the error and (p,q) coefficients:

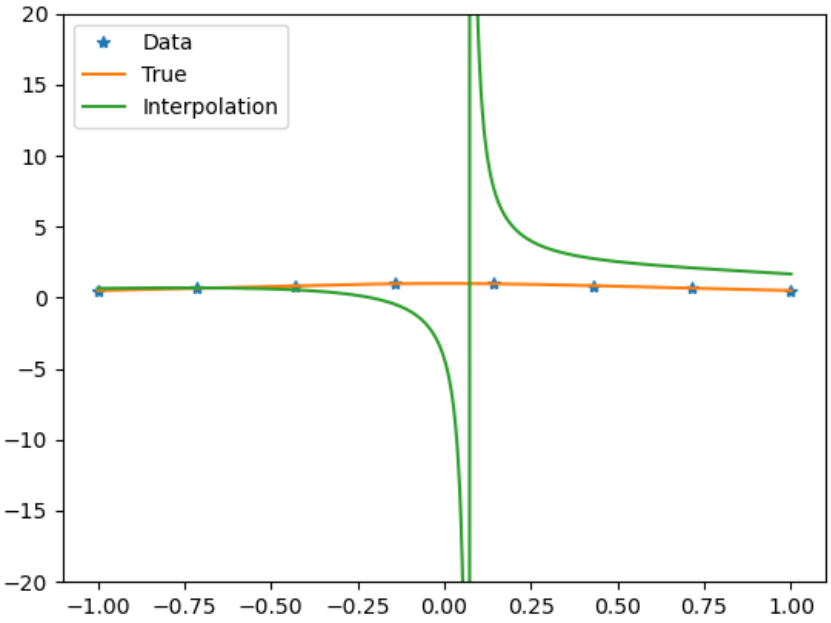
```
my ratfit rms error is  1.1953346212774387e-16
[1.0000000e+00 4.4408921e-16] [8.8817842e-16 1.0000000e+00]
```



For $n = 4, m = 5$ The output gives the error and (p,q) coefficients:

```
my ratfit rms error is  21.51833115012628
[ -4.29862069 -20.          2.          3.96729867] [-14.   4.  -2.   0.]
```

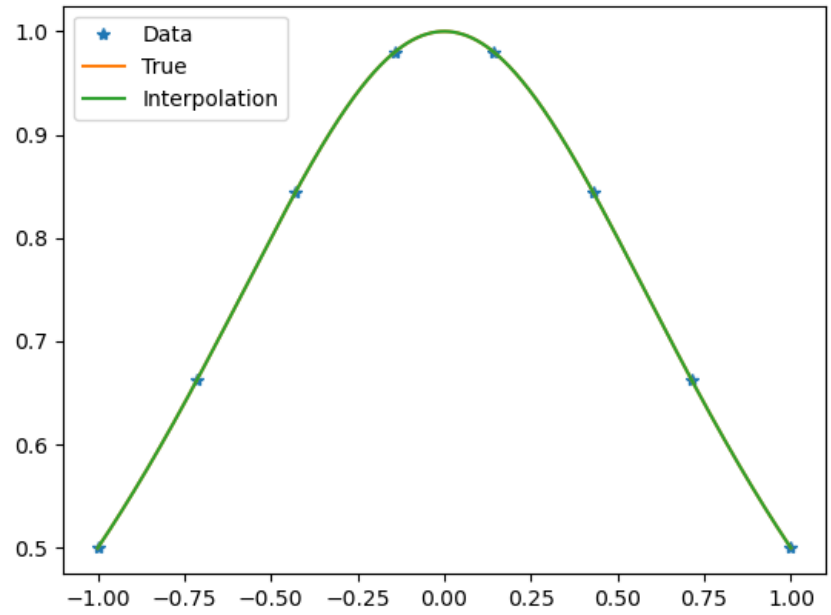
The Lorentz function written as a rational function only needs one coefficient in the numerator (for the constant term) and two coefficients in the denominator (for the constant and the quadratic term). In the algorithm, the number of points is equal to the number of coefficients. This is why the error is smaller with smaller n and m (i.e. less points).



When you use np.linalg.pinv the output is:

```
my ratfit rms error is  3.1731033732751607e-16
[ 1.00000000e+00  1.33226763e-15 -3.33333333e-01 -1.77635684e-15] [ 1.77635684e-15  6.66666667e-01 -8.88178420e-16
```

The error and the coefficients got much smaller. I don't really understand what happened. The function had poles so the pinv must've been able to deal with that correctly?



Problem 4

The integral we are supposed to evaluate is:

$$E_z = \frac{2\pi R^2 \sigma}{4\pi\varepsilon_0} \int_{-1}^{1} \frac{z - Ru}{(R^2 + z^2 + 2Rzu)(3/2)} du$$

I took $R = 1$ for simplicity.
Refer to `A1_Q4_1.py`. My integrator was based on Simpsons rule.
The analytic solution to the integral is:

$$E_z = \frac{2\pi R^2 \sigma}{4\pi\varepsilon_0 z^2} \left(\frac{z - R}{|z - R|} - \frac{-z - R}{|z + R|} \right)$$

So my integrator gave the right behaviour away from the shell wall. My integrator has a divergence at the shell wall ($z = R = 1$), but the scipy integrator has no such divergence.