PS1 Comp Phys

1)(a) Consider the derivatives with transient points $x \pm \delta$ & $x \pm 2\delta$:

$$\bar{f}'(x) \equiv \frac{f(x+\delta) - f(x-\delta)}{2\delta} = \frac{1}{2\delta}\left[f(x) + f'(x)\delta + f''(x)\delta^2/2! + f'''(x)\delta^3/3! + O(\delta^4) - \left(f(x) - f'(x)\delta + f''(x)\delta^2/2! - f'''(x)\delta^3/3! + O(\delta^4) \right) \right]$$

$$= \frac{1}{2\delta}\left[2f'(x)\delta + 2f'''(x)\delta^3/3! + O(\delta^4) \right]$$

$$= f'(x) + \frac{1}{6}f'''(x)\delta^2 + O(\delta^4)$$

$$\tilde{f}'(x) \equiv \frac{f(x+2\delta) - f(x-2\delta)}{2(2\delta)} = \frac{1}{4\delta}\left[2f'(x)(2\delta) + 2f'''(x)(2\delta)^3/3! + O(2\delta)^4 \right]$$

$$= f'(x) + \frac{2}{3}f'''(x)\delta^2 + O(2\delta)^4 \Rightarrow f'''(x) = \frac{3}{2\delta^2}\left[\tilde{f}'(x) - f'(x) - O(2\delta)^4 \right]$$

Plugging $f'''[\tilde{f}'(x)]$ in $\bar{f}'(x)$ gives:

$$\bar{f}'(x) = f'(x) + \frac{1}{6}\left(\frac{3}{2\delta^2}\left[\tilde{f}'(x) - f'(x) - O(2\delta)^4 \right] \right)\delta^2 + O(\delta^4)$$

$$= f'(x) + \frac{1}{4}\tilde{f}'(x) - \frac{1}{4}f'(x) - \underbrace{\frac{1}{4}O(2\delta)^4 + O(\delta^4)}_{g[\delta^4]}$$

$$\Rightarrow \bar{f}'(x) = \frac{3}{4}f'(x) + \frac{1}{4}\tilde{f}'(x) + g[\delta^4]$$

$$\Rightarrow f'(x) = \frac{4}{3}\left[\bar{f}'(x) - \frac{1}{4}\tilde{f}'(x) + g[\delta^4] \right] = \frac{4}{3}\bar{f}'(x) - \frac{1}{3}\tilde{f}'(x) + \tilde{g}[\delta^4]$$

$$= \frac{4}{3}\left(\frac{f(x+\delta) - f(x-\delta)}{2\delta} \right) - \frac{1}{3}\left(\frac{f(x+2\delta) - f(x-2\delta)}{4\delta} \right) + \tilde{g}[\delta^4]$$

$$= \frac{4f(x+\delta) - 4f(x-\delta) - f(x+2\delta) + f(x-2\delta)}{12\delta} + \tilde{g}[\delta^4]$$

$\tilde{g}[\delta^4]$: normalized $O(\delta^4)$ functional

b) Now we wish to construct the error function $E(\delta)$ for our derivative

↳ First we have the theoretical error: E_T:

$$E_T(\delta) = \tilde{f}'(x) - f'(x) = \frac{4}{3}\bar{f}'(x) - \frac{1}{3}\tilde{f}'(x) + \tilde{g}[\delta^4] - f'(x)$$

$$= \frac{4}{3}f'(x) + \frac{2}{9}f'''(x)\delta^2 + \frac{1}{90}f^{(5)}(x)\delta^4 + O(\delta^5) - \frac{1}{3}f'(x) - \frac{2}{9}f'''(x)\delta^2$$

$$- \frac{4}{45}f^{(5)}(x)\delta^4 + O(2\delta)^5 + \tilde{g}[\delta^4] - f'(x)$$

$$= \underbrace{-\frac{7}{90}f^{(5)}(x)\delta^4}_{\text{take modulus}} + \underbrace{O(\delta^5) + O(2\delta)^5 + \tilde{g}[\delta^4]}_{j[\delta^4,\delta^5]}$$

$j[\delta^4,\delta^5]$ — $O(\delta^4), O(2\delta)^4, O(\delta^5), O(2\delta)^5$ functional

$$\Rightarrow E_T(\delta) \longrightarrow |E_T(\delta)| = \frac{7}{90}\left|f^{(5)}(x)\delta^4\right| + j[\delta^4,\delta^5]$$

Next we have the roundoff error: which is proportional to $f(x)/\delta$ w/ a prefactor ε st $\varepsilon \ll 1$:

$$E_R(\delta) \propto f(x)/\delta, \varepsilon \Rightarrow E_R(\delta) \simeq \varepsilon\left|f(x)/\delta\right|$$

Thus the error function we wish to minimize is the sum of the errors squared (variances) (dropping $j[\delta^4,\delta^5] \ll 1$):

$$E(x;\delta) = |E_T(\delta)|^2 + |E_R(\delta)|^2 = \frac{49}{8100}(f^{(5)}(x))^2\delta^8 + \varepsilon^2 f^2(x)/\delta^2$$

$$\Rightarrow 0 = \partial_\delta E(\delta) = \frac{98}{2025}(f^{(5)}(x))^2\delta^7 - 2\varepsilon^2 f^2(x)/\delta^3$$

$$\Rightarrow \frac{98}{2025}(f^{(5)}(x))^2\delta^7 = \frac{2\varepsilon^2 f^2(x)}{\delta^3} \Rightarrow \delta^{10} = \frac{2025}{98}\frac{2\varepsilon^2 f^2(x)}{(f^{(5)}(x))^2}$$

$$\Rightarrow \delta_0 = \sqrt[10]{\frac{2025}{196}}\sqrt[5]{\frac{\varepsilon f(x)}{f^{(5)}(x)}}$$ } Optimal δ

Now we wish to see if our δ_0 results in a proper estimation for $f(x) = e^x,\ e^{x/100}$

$f(x) = e^x$: $f^{(5)}(x) = f(x) = e^x \Rightarrow \delta_0(x) = \sqrt[10]{\frac{2025}{196}}\sqrt[5]{\frac{\varepsilon e^x}{e^x}}$

$$\Rightarrow \delta_0 = \sqrt[10]{\frac{2025}{196}}\sqrt[5]{\varepsilon}$$

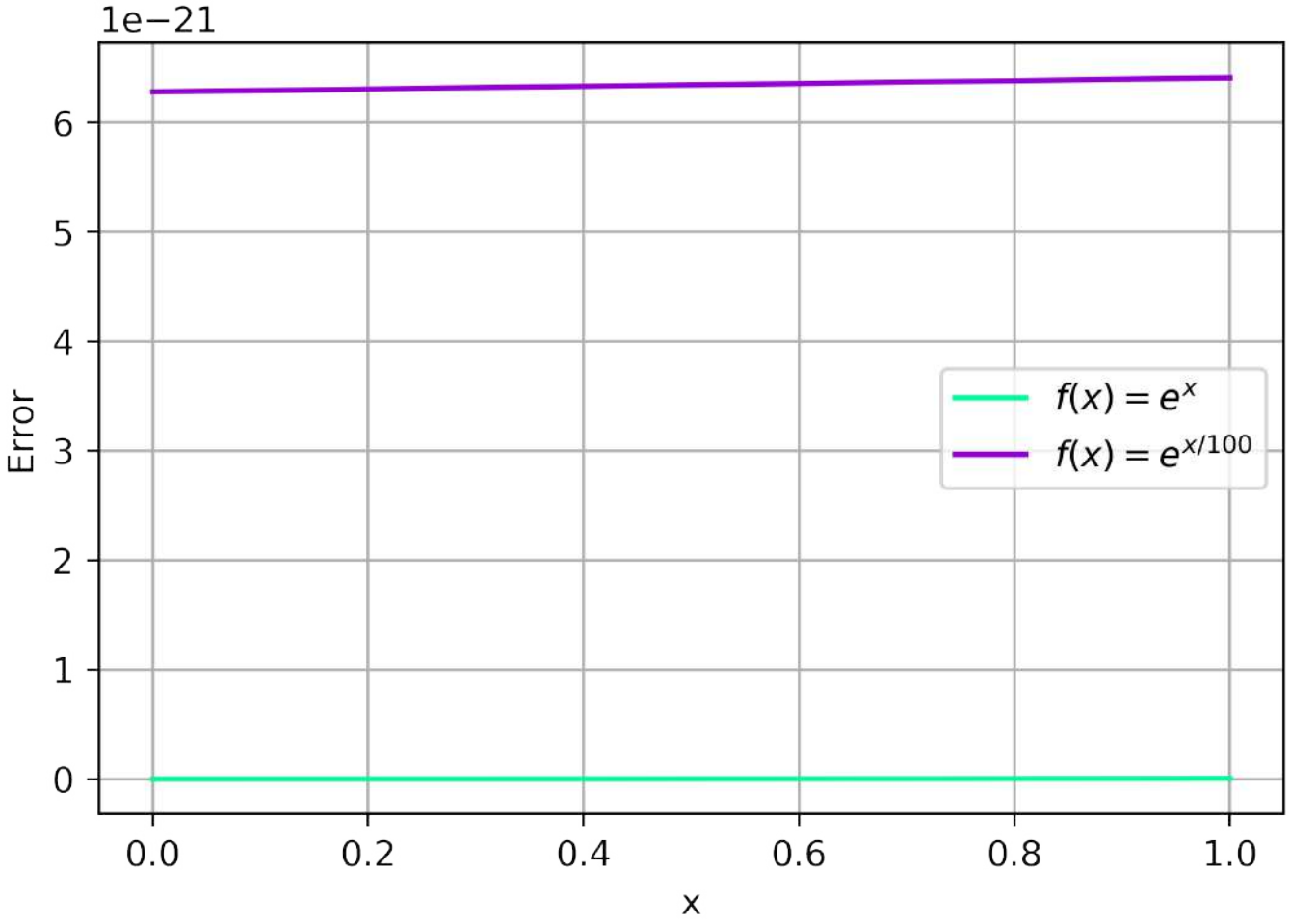
$$\Rightarrow E(x;\delta_0) = \frac{49}{8100}e^{2x}\delta_0^8 + \varepsilon^2 e^{2x}/\delta_0^2$$

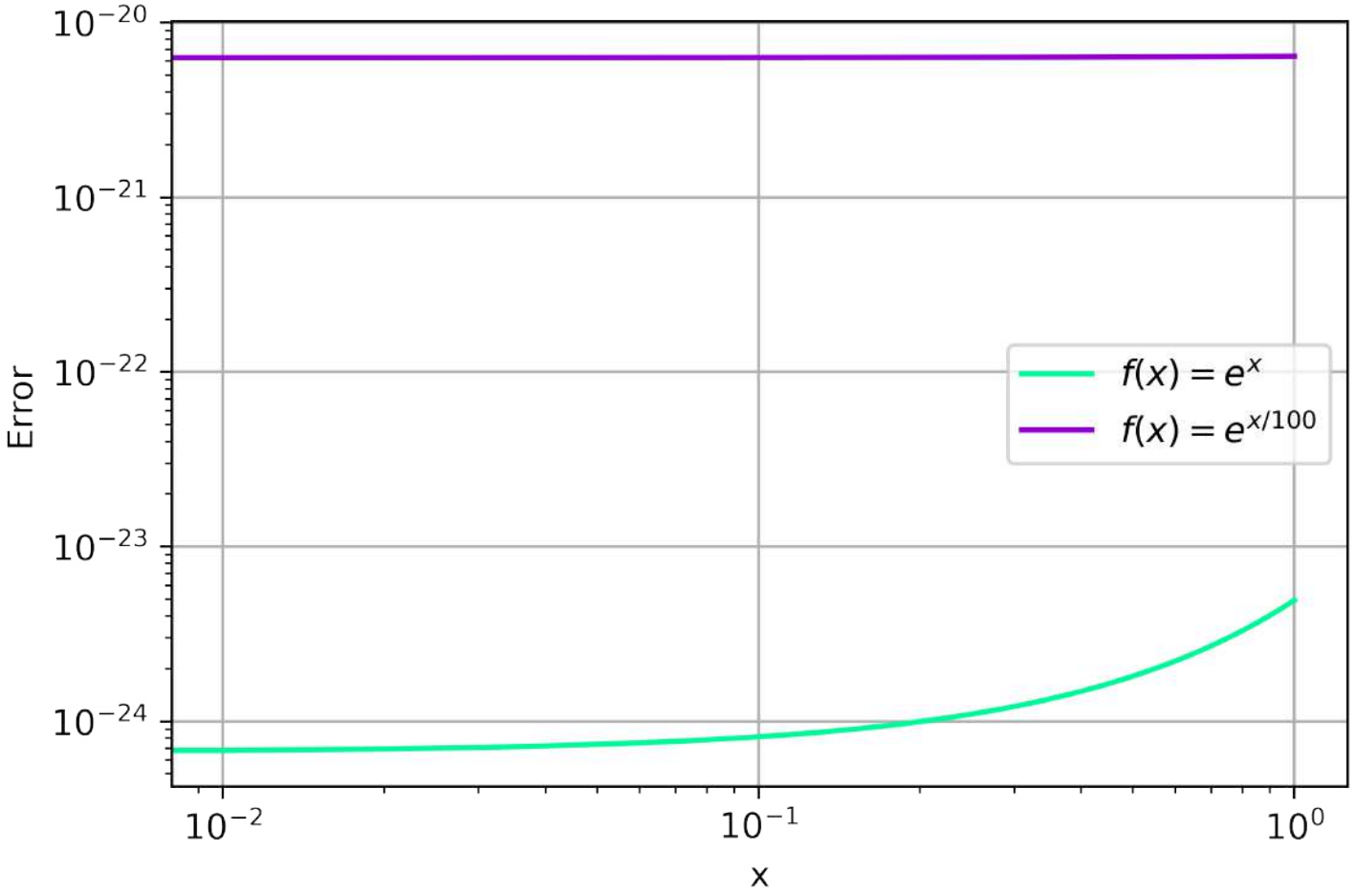
$f(x) = e^{x/100}$: $f^{(5)}(x) = (\tfrac{1}{100})^5 e^{x/100} \Rightarrow \delta_0(x) = \sqrt[10]{\frac{2025}{196}}\sqrt[5]{\frac{\varepsilon}{100^5}\frac{e^{x/100}}{e^{x/100}}}$

$$\Rightarrow \delta_0' = \sqrt[10]{\frac{2025}{196}}\frac{\sqrt[5]{\varepsilon}}{100}$$

$$\Rightarrow E(x;\delta_0') = \frac{49}{8100}\left(\left(\tfrac{1}{100}\right)^5 e^{x/100}\right)^2\delta_0'^8 + \varepsilon^2 e^{x/50}/\delta_0'^2$$

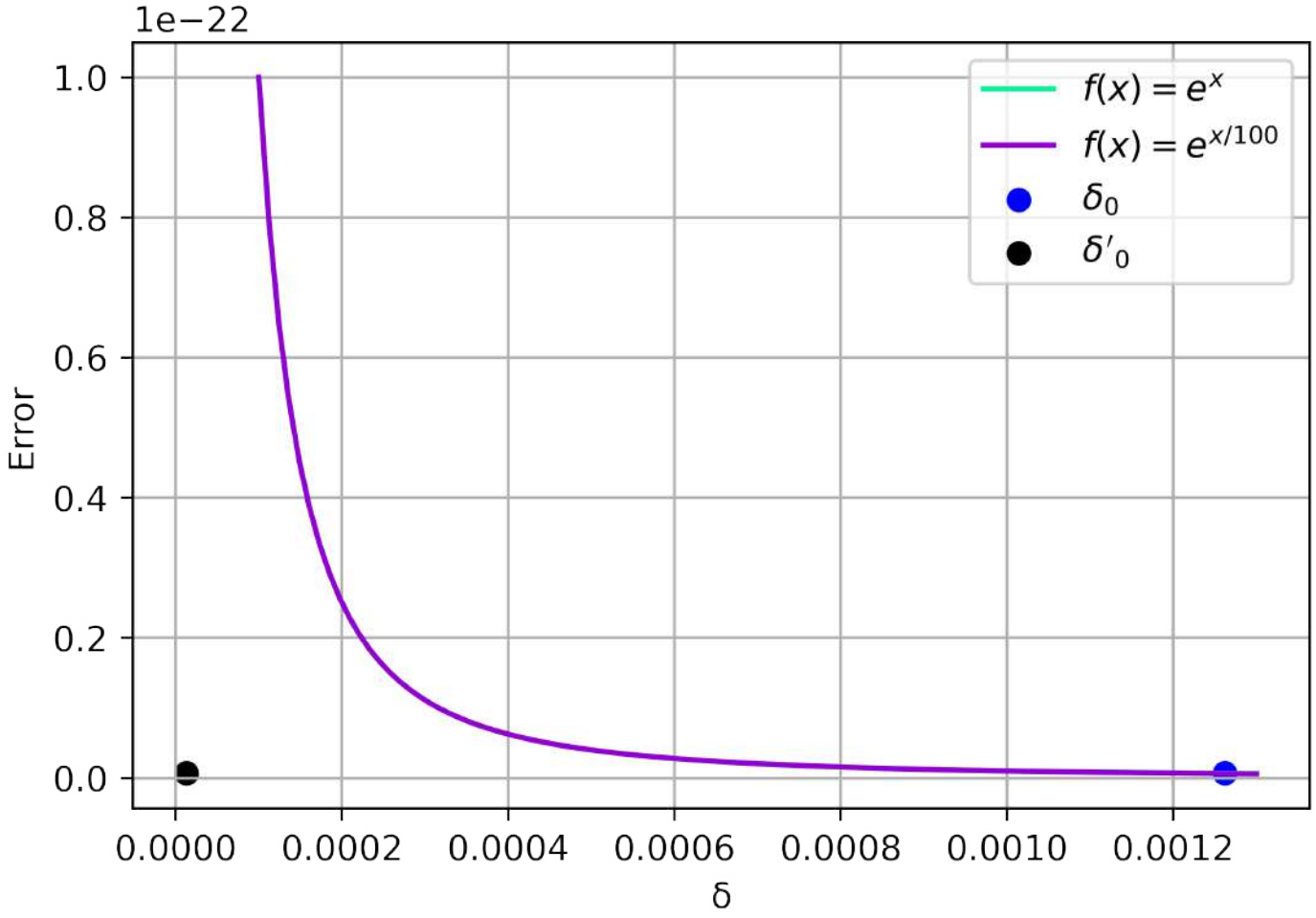
To see how good our approximation is, we shall plot $E(x;\delta)$ for $x \in [0,1]$ For $\varepsilon \sim 10^{-15}$ for normal & logarithmic axes:


1e−21
6
5
4
3
2
1
0
Error
0.0
0.2
0.4
0.6
0.8
1.0
x
$f(x) = e^x$
$f(x) = e^{x/100}$


10^{-20}
10^{-21}
10^{-22}
10^{-23}
10^{-24}
Error
10^{-2}
10^{-1}
10^{0}
x
$f(x) = e^x$
$f(x) = e^{x/100}$

As can be seen above, we have $E(x;\delta) \sim O(10^{-21})$ which is a sufficient estimate. ($O(10^{-24})$ & $O(10^{-20})$ for the loglog plot).

We can also fix $x = 0$ & plot for different δ to see how our approximation holds:


1e−22
1.0
0.8
0.6
0.4
0.2
0.0
Error
0.0000
0.0002
0.0004
0.0006
0.0008
0.0010
0.0012
δ
$f(x) = e^x$
$f(x) = e^{x/100}$
δ_0
δ'_0

Once again we have $O(10^{-22})$ for error which is sufficient.

↳ We can see δ_0 results in minimal error while δ_0' does not

Let $\partial x \equiv h$

2) Consider $f'(x) = \frac{f(x+h) - f(x-h)}{2h} + O(h^2)$

Proceeding analogously from what was done in 1)b) & in class, for $\varepsilon \ll 1$ we can write the error function:

$$E(h) = \frac{h^2}{6} f''' + \varepsilon f / h$$

We can then again minimize this wrt h to get:

$$0 = \partial_h E\Big|_{h=h_0} = \frac{1}{3} h_0 f''' - \varepsilon f / h_0^2 \Rightarrow \frac{1}{3} h_0 f''' = \varepsilon f / h_0^2 \Rightarrow h_0 = \sqrt[3]{\frac{3\varepsilon f}{f'''}}$$

For this estimate to work we need an expression for $f'''(x)$, so we will once again expand:

$$f(x \pm h) = f(x) \pm f'(x)h + \frac{1}{2} f''(x) h^2 \pm \frac{1}{6} f'''(x) h^3 + O(h^4)$$
$$f(x \pm 2h) = f(x) \pm 2f'(x)h + 2f''(x) h^2 \pm \frac{4}{3} f'''(x) h^3 + O(h^4)$$

$$\Rightarrow f(x+h) - f(x-h) = 2f'(x)h + \frac{1}{3} f'''(x) h^3 + O[h^4] \quad : A$$
$$\Rightarrow f(x+2h) - f(x-2h) = 4f'(x)h + \frac{8}{3} f'''(x) h^3 + O[h^4] \quad : B$$

$$\Rightarrow B - 2A : f(x+2h) - f(x-2h) - 2f(x+h) + 2f(x-h) = 2 f'''(x) h^3 + O[h^4]$$

$$\Rightarrow f'''(x) = \frac{f(x+2h) - f(x-2h) - 2f(x+h) + 2f(x-h)}{2h^3} + \underbrace{O[h^4]}_{\text{Dropped in code}}$$

Notice!: $f'''(x)$ has an implicit h dependance $f(x \pm nh)$, $n \in \mathbb{Z}$ & so when trying to get an optimal h we run into:

$$h_0 \sim \underbrace{\left(f(x+2h_0) - \dots \right)^{-1/3}}_{\text{We cannot get an explicit expression for } h_0!}$$

What do we do? Panic? Ok, then what? For h_0 within the argument of $f(x \pm nh)$ in $f'''(x)$ we will take a small enough h

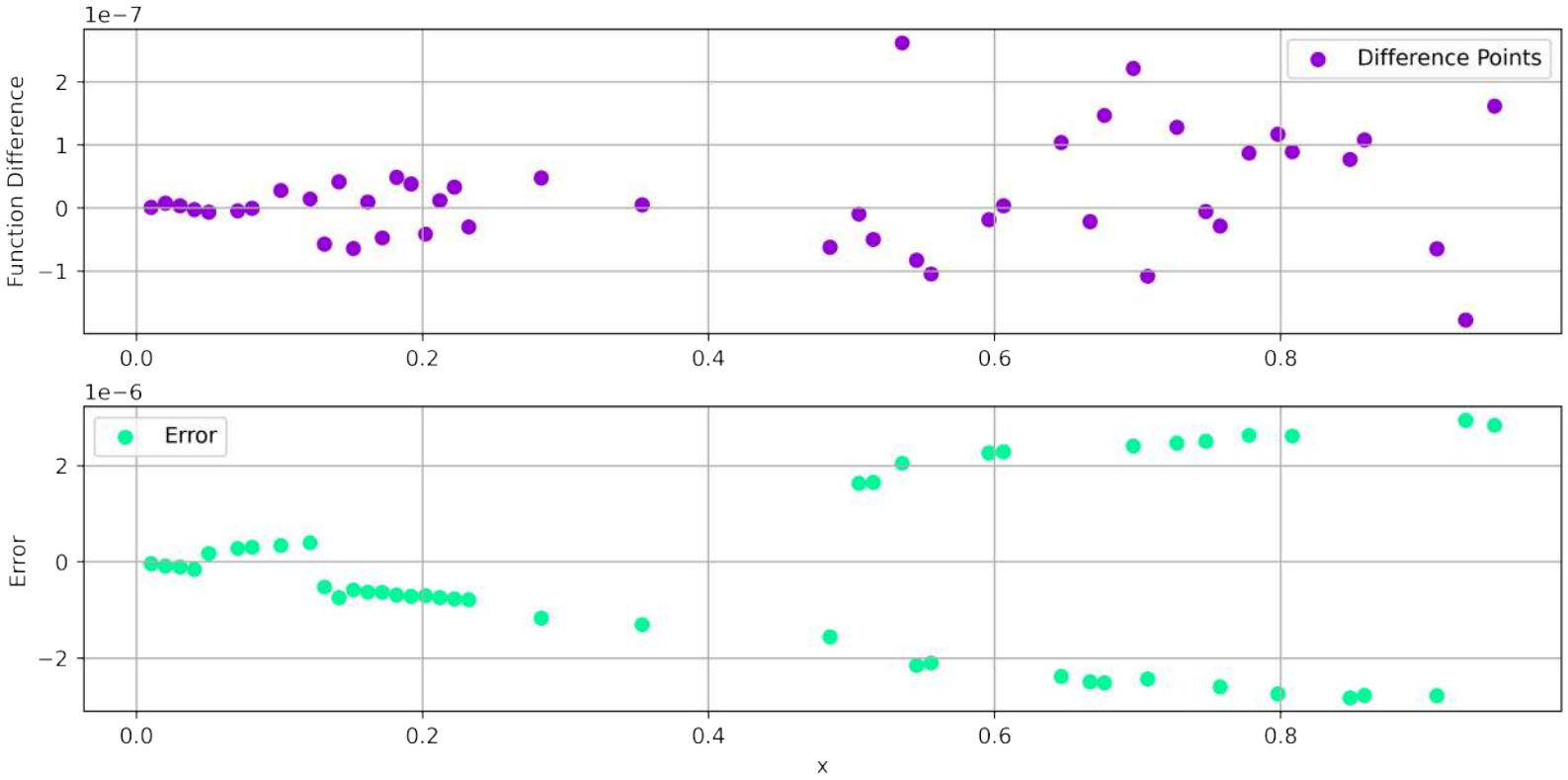
↳ The error in h doesn't matter since it doesn't affect the answer directly. We call this fixed small h, γ:

$$h_0(x) = \sqrt[3]{\frac{3\varepsilon f(x)}{f'''(x)}} = \left[3\varepsilon f(x) \cdot \left\{ \frac{2\gamma^3}{f(x+2\gamma) - f(x-2\gamma) - 2f(x+\gamma) + 2f(x-\gamma)} \right\}^{-1} \right]^{1/3},$$

where $\mathbb{R} \ni \varepsilon, \gamma \ll 1$. With the above expression for the optimal h (dx), we can write a numerical differentiator.

Now to test this, we choose for example $fun = np.sin$ & plot the difference between our differentiator & the actual derivative ($ndiff(x) - np.cos(x)$)

↳ We also plot the error $E(x)$ in the subplot under:


1e−7
Function Difference
Difference Points
2
1
0
−1
0.0
0.2
0.4
0.6
0.8
1e−6
Error
Error
2
0
−2
0.0
0.2
0.4
0.6
0.8
x

As we can see, the difference between our derivative & the analytic derivative is of $O(10^{-7})$ & the error is of order $O(10^{-6})$

↳ Note: To get the error at x you must run
ndiff(fun, x, True)[1]. The [1] is because when
full == True, ndiff returns (derivative, error)

Moreover we can make ndiff take in arrays by differentiating between points & arrays via:

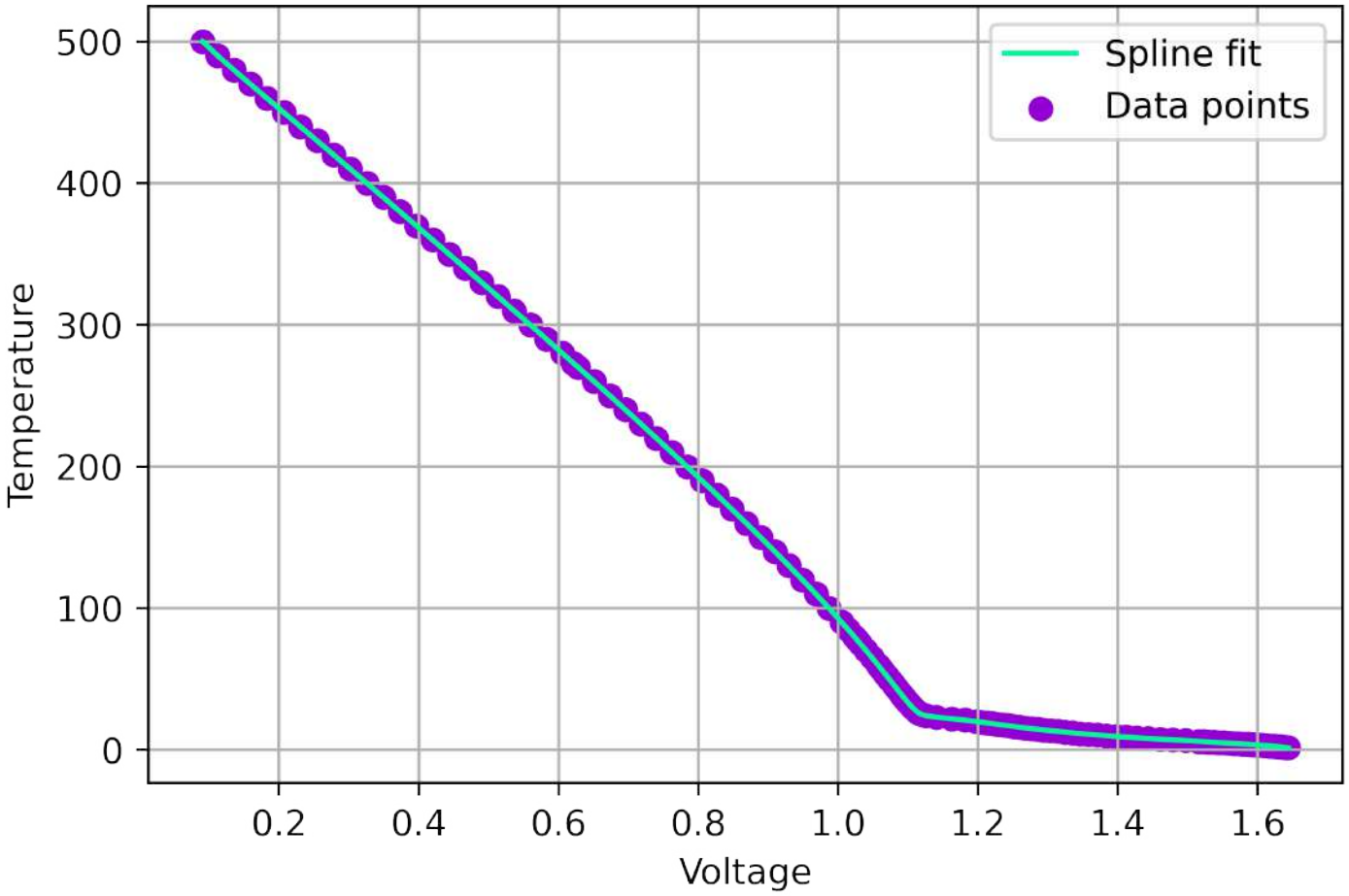
if np.isscalar(x) == True

Turn all you need to do is act ndiff on the array x. I have not done it as I am behind in string lectures.

3) $I = 10\mu A = 10(10^{-9})A = 10^{-10}A$

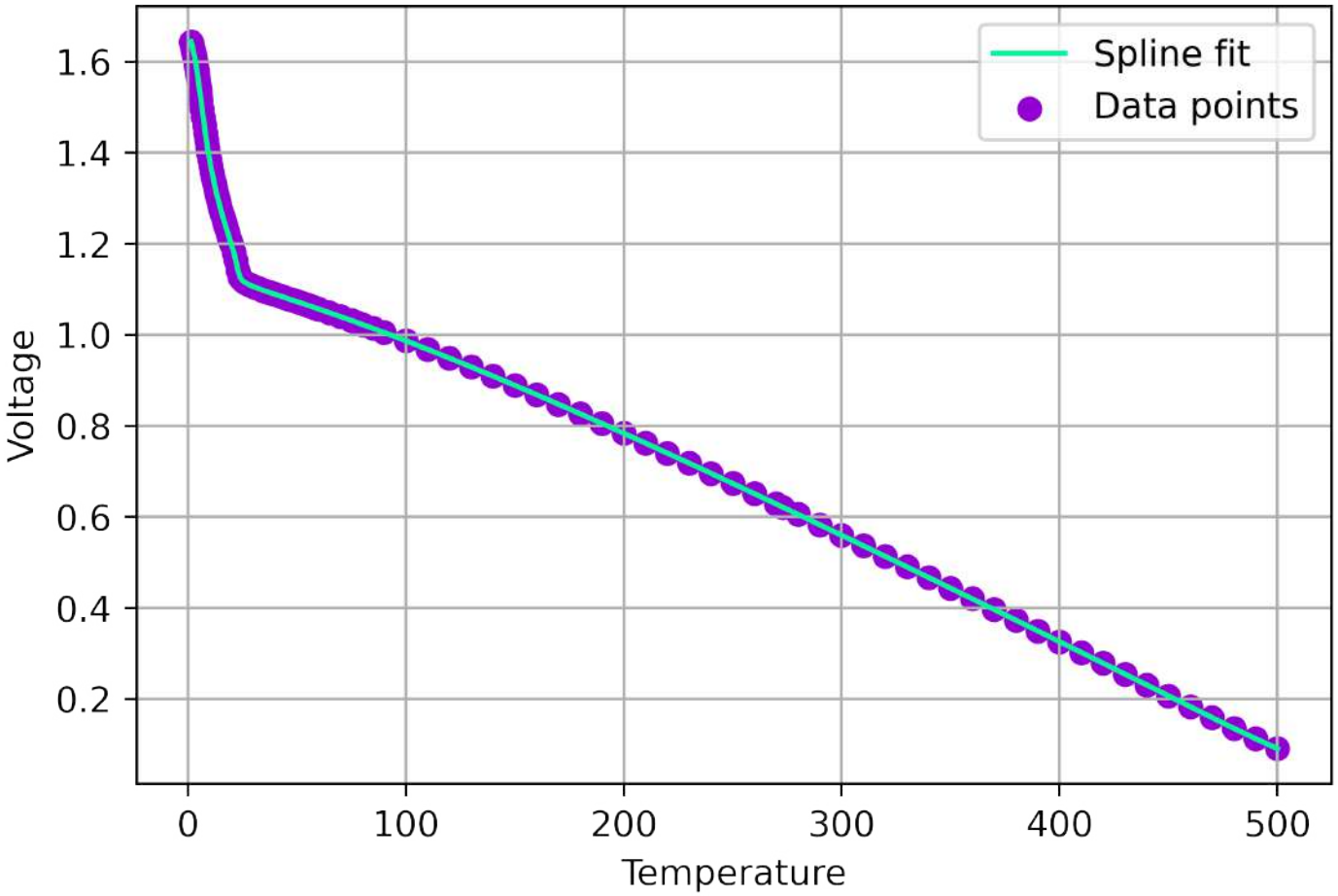
For the first part of the question, to have a routine that takes in an arbitrary V & returns a T, we fit a cubic spline through the points

↳ This allows us to interpolate any values between the data points & the fit looks like:

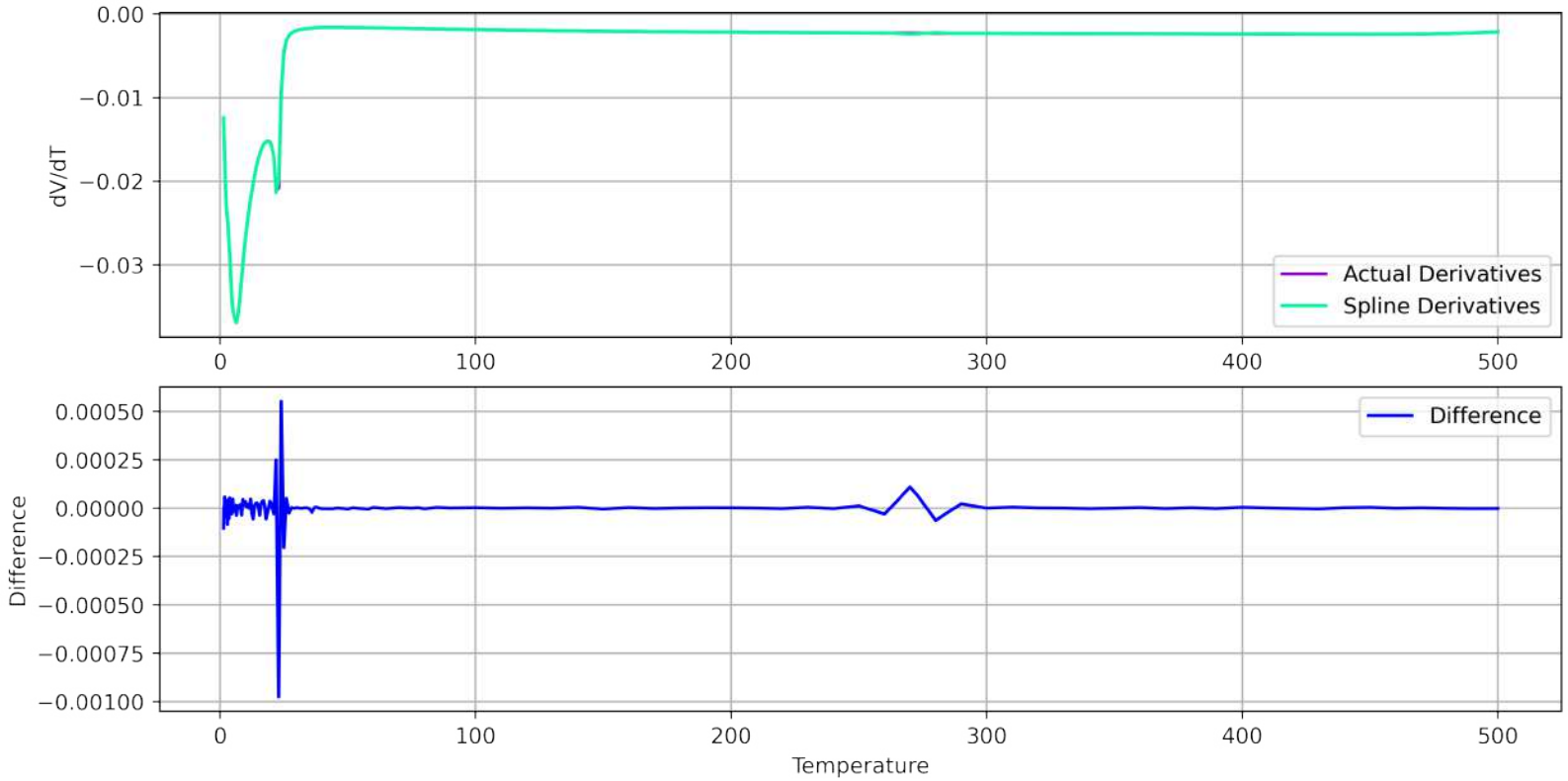

Spline fit
Data points
Temperature
Voltage
500
400
300
200
100
0
0.2
0.4
0.6
0.8
1.0
1.2
1.4
1.6

Now for the estimate I propose a possibly strange approach. The error will be a function of the derivative error (since we have $\partial V/\partial T$) & the error in the interpolat

First, we want the error in the derivative. Initially, we must invert our $T(V)$ interpolation to $V(T)$ since we have $\partial V/\partial T$. This doesn't change the error & looks like:


Spline fit
Data points
Voltage
Temperature
1.6
1.4
1.2
1.0
0.8
0.6
0.4
0.2
0
100
200
300
400
500

Now, we will compute the derivative of the spline at each point & then plot the difference of the spline derivative to the actual derivative values dV/dT & this will be our derivative error. We plot the different dV/dT & below it the difference:


0.00
−0.01
−0.02
−0.03
dV/dT
Actual Derivatives
Spline Derivatives
0
100
200
300
400
500
0.00050
0.00025
0.00000
−0.00025
−0.00050
−0.00075
−0.00100
Difference
Difference
0
100
200
300
400
500
Temperature

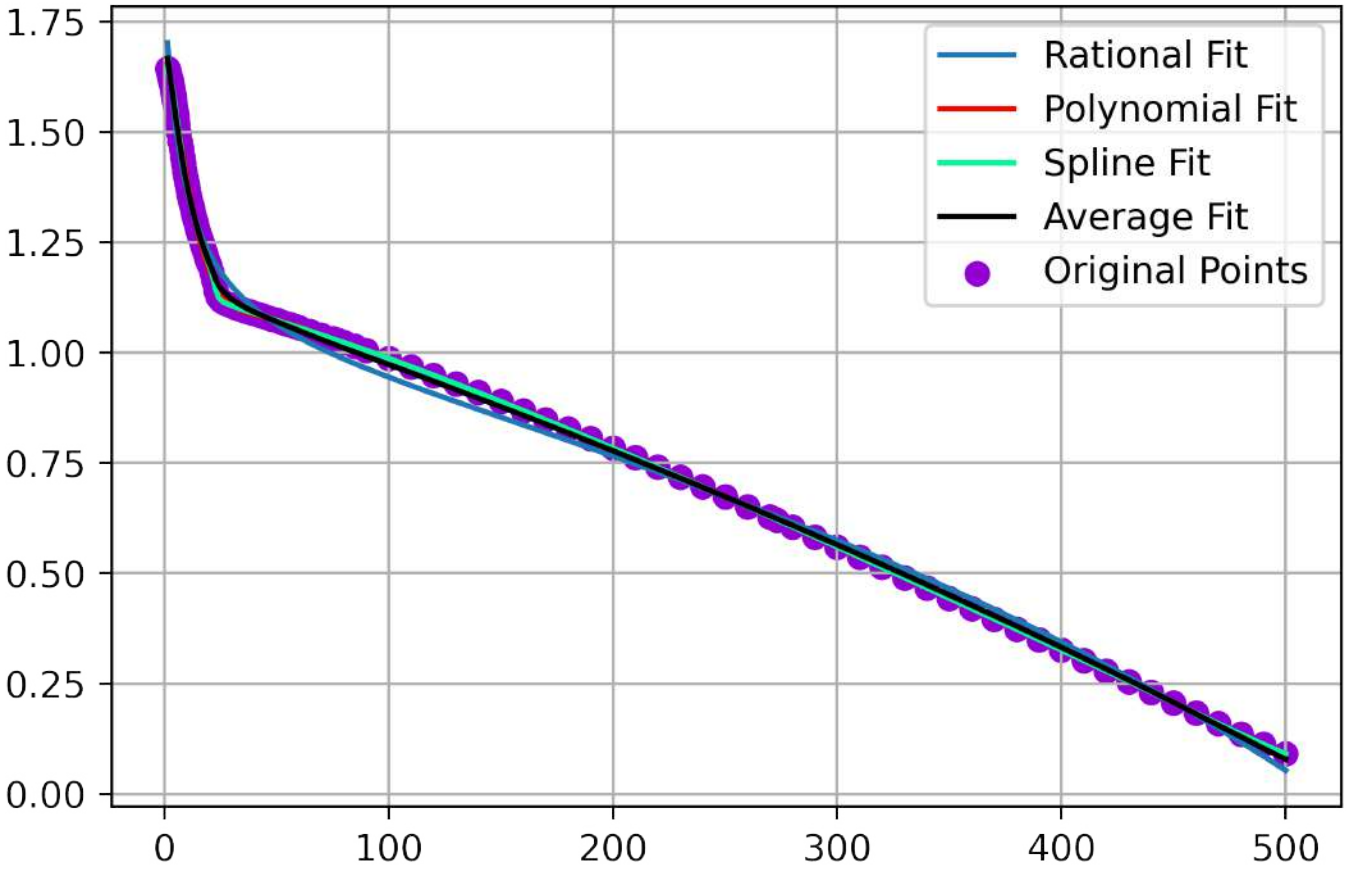
As we can see above, the derivative of the spline follows closely with the actual derivative values

↳ The difference (derivative error) is of $O(10^{-4})$ which is decent.

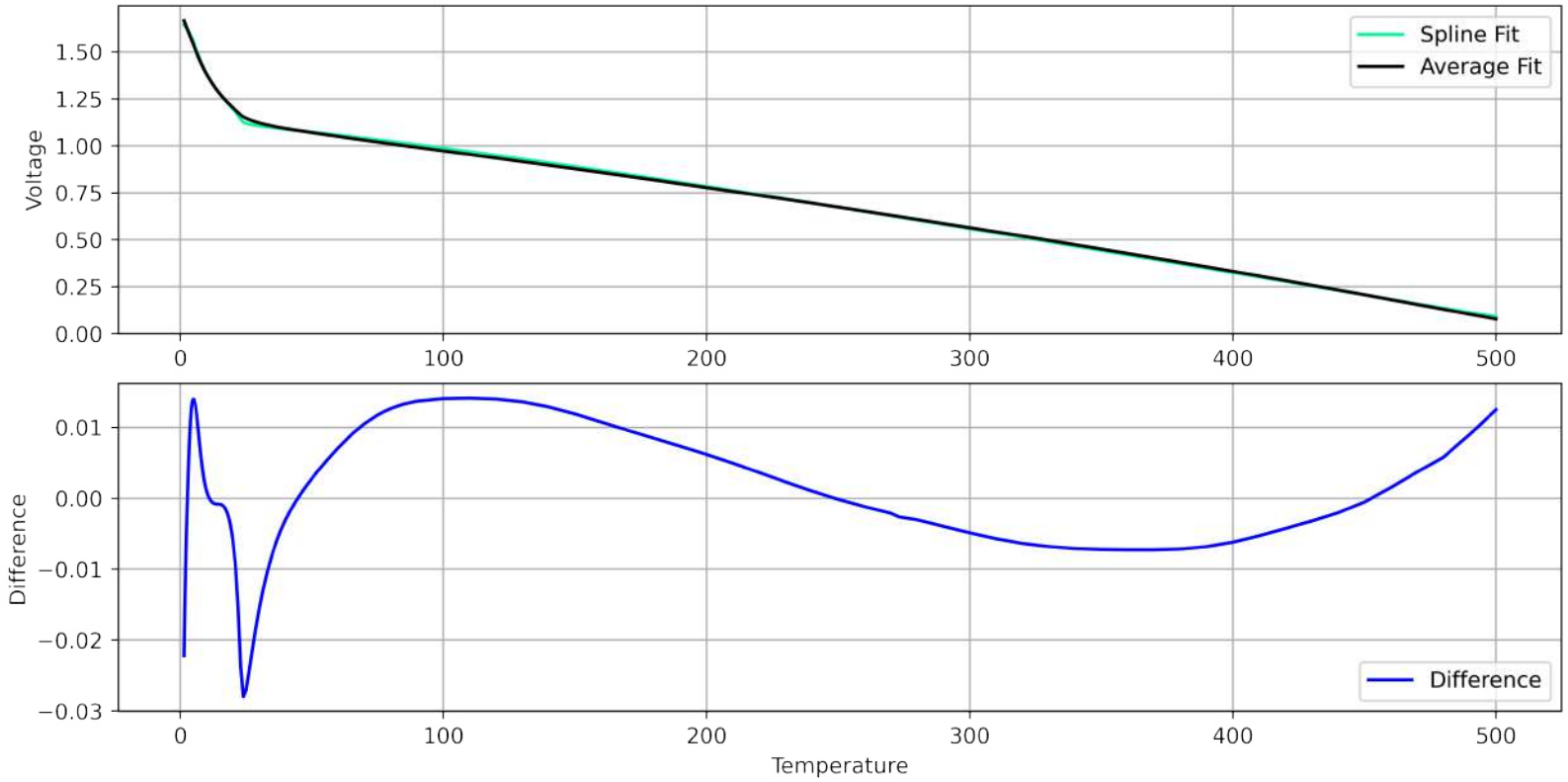
Next, we would like an estimate for the error in our spline fit values. To do so we take a somewhat naive approach.

↳ Our 'true' value will be the average of a spline, polynomial & rational function fits & our error will be the difference between the 'true' values & the spline fit.

The following is a plot of all the different fits & the average of the fits ('true value'):


Rational Fit
Polynomial Fit
Spline Fit
Average Fit
Original Points
1.75
1.50
1.25
1.00
0.75
0.50
0.25
0.00
0
100
200
300
400
500

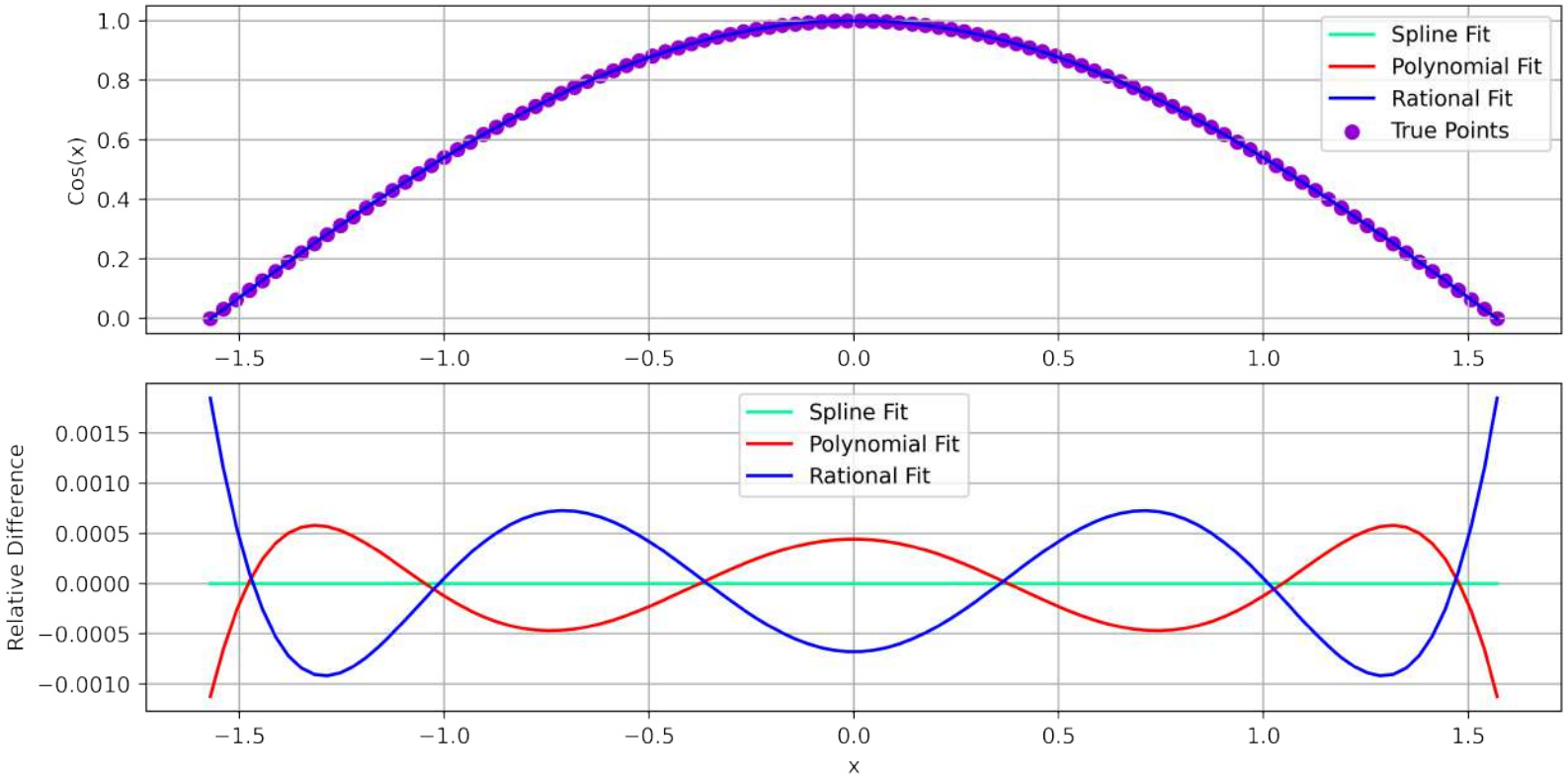
Finally, we plot just the spline fit, average fit in one subplot, & in the subplot below we plot the difference between the spline & average plot (spline fit error):


Spline Fit
Average Fit
Voltage
1.50
1.25
1.00
0.75
0.50
0.25
0.00
0
100
200
300
400
500
Difference
0.01
0.00
−0.01
−0.02
−0.03
Difference
Temperature

As we can see from the plot above, our difference or error is of $O(10^{-2})$ which is pretty poor.

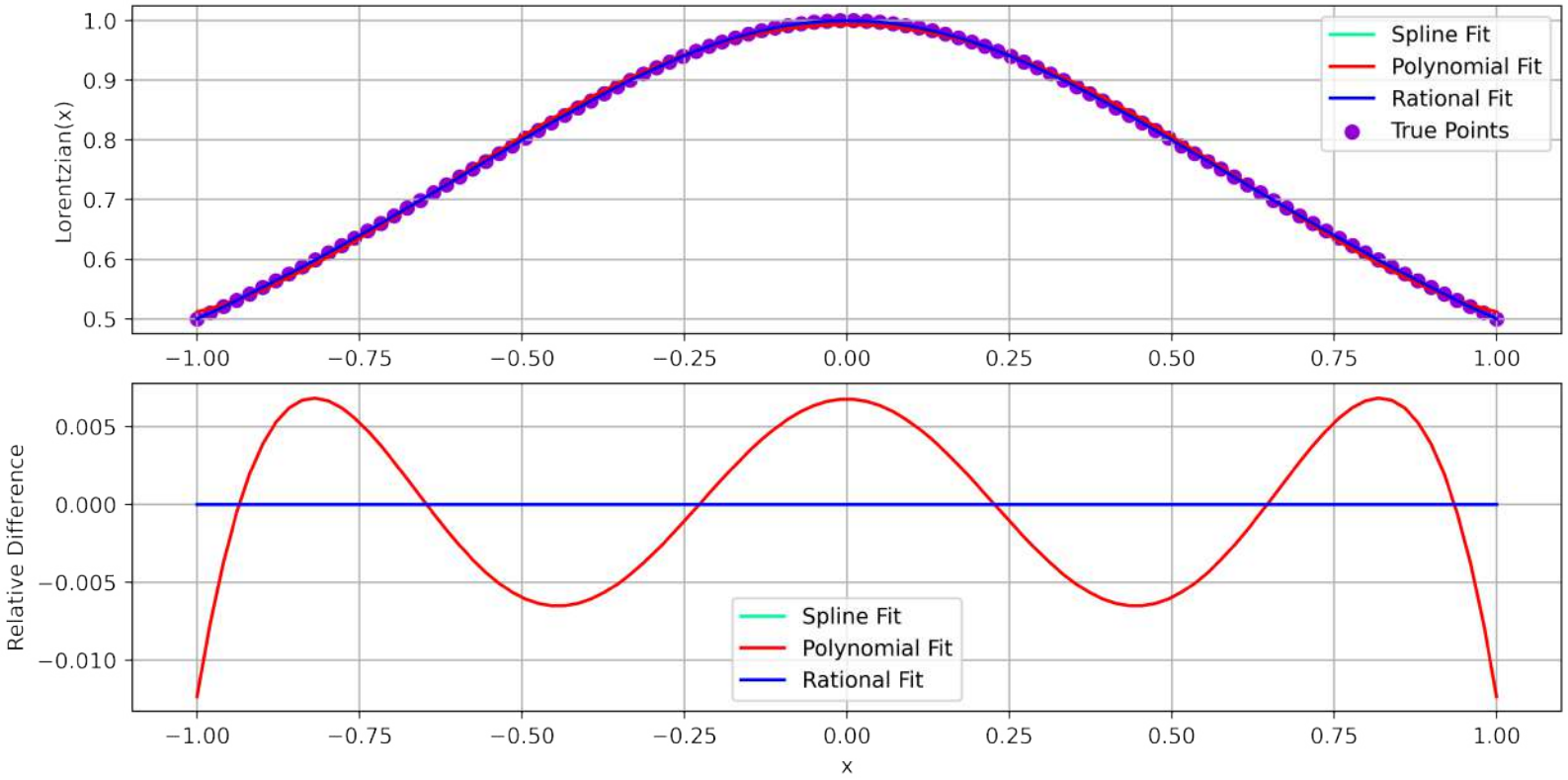
=> From the derivative of the spline error we have $O(10^{-4})$ error while the naive 'true' error has $O(10^{-2})$ error => $\geq O(10^{-3})$ error overall

4) First we fit $\cos x$, $x \in [-\pi/2, \pi/2]$ with the three different fits; plot the fits (top subfigure), and plot the difference between the function values & the fit values (bottom subfigure):


Spline Fit
Polynomial Fit
Rational Fit
True Points
Cos(x)
1.0
0.8
0.6
0.4
0.2
0.0
−1.5
−1.0
−0.5
0.0
0.5
1.0
1.5
Spline Fit
Polynomial Fit
Rational Fit
Relative Difference
0.0015
0.0010
0.0005
0.0000
−0.0005
−0.0010
−1.5
−1.0
−0.5
0.0
0.5
1.0
1.5
x

As can be seen, the spline fit has an extremely small error while the polynomial & rational fit errors oscillate at $O(10^{-3})$

Now we repeat the same for the Lorentzian:


Lorentzian(x)
1.0
0.9
0.8
0.7
0.6
0.5
−1.00
−0.75
−0.50
−0.25
0.00
0.25
0.50
0.75
1.00
Spline Fit
Polynomial Fit
Rational Fit
True Points
Relative Difference
0.005
0.000
−0.005
−0.010
Spline Fit
Polynomial Fit
Rational Fit
x

Here we see the rational & spline fit dominate w/ extremely small error while the polynomial fit error oscillates at $O(10^{-3})$

The error of the Lorentzian with a rational function fit in principle (& practice) is zero as the Lorentzian is a rational function.

↳ What we have agrees with our expectations for higher orders as the rational function still decays for $x \to \pm\infty$ & thus properly fits the Lorentzian.

When we switch from np.linalg.inv to np.linalg.pinv, all the singular eigenvalues ($\lambda_k = 0$) get removed/ignored

↳ Analytically this is an issue as $\det A|_{\lambda_k = 0} \to \infty$, but numerically cutting them out avoids the issue

Looking at p & q, using inv (instead of pinv) results in much larger coefficient values due to inverting a matrix with near singular eigenvalues.

↳ This gives us a pole in the denominator & the error blows up.

↳ This is mitigated by using pinv which removes the singularities