HUIYI WANG
ID: 261-016-753
PHYS 512
Sep 11th 2021

Problem Set 1

1 Problem 1

This problem asked us to calculate the numerical derivative of a function using four points. First, we would Taylor expand all of the terms:

$$f(x-\delta) = f(x) -' (x) + \frac{\delta^2}{2} f''(x) - \frac{\delta^3}{6} f'''(x) + \frac{\delta^4}{24} f^{(4)}(x) - \frac{\delta^5}{120} f^{(5)}(\xi_1) \tag{1}$$

$$f(x+\delta) = f(x) +' (x) + \frac{\delta^2}{2} f''(x) + \frac{\delta^3}{6} f'''(x) + \frac{\delta^4}{24} f^{(4)}(x) + \frac{\delta^5}{120} f^{(5)}(\xi_2) \tag{2}$$

$$f(x-2\delta) = f(x) - 2\delta f'(x) + 2\delta^2 f''(x) - \frac{4\delta^3}{3} f'''(x) + \frac{2\delta^4}{3} f^{(4)}(x) - \frac{4\delta^5}{15} f^{(5)}(\xi_3) \tag{3}$$

$$f(x+2\delta) = f(x) + 2\delta f'(x) + 2\delta^2 f''(x) + \frac{4\delta^3}{3} f'''(x) + \frac{2\delta^4}{3} f^{(4)}(x) + \frac{4\delta^5}{15} f^{(5)}(\xi_4) \tag{4}$$

Here, in order for us to achieve an estimation with the smallest error term: $f^{(5)}$, we would need to factor Equation (1) and (2) by 8, which cancels out the fourth derivative of the terms.

$$\begin{aligned} f(x+2\delta) - 8f(x-\delta) + 8f(x+\delta) - f(x-2\delta) = \\ 12\delta f'(x) + \frac{\delta^5}{15} f^{(5)}(\xi_1) + \frac{\delta^5}{120} f^{(5)}(\xi_2) - \frac{4\delta^5}{15} f^{(5)}(\xi_3) - \frac{4\delta^5}{15} f^{(5)}(\xi_4) \end{aligned} \tag{5}$$

The terms on involving ξ_n on the right side of the equation are the truncation errors and could be estimated by taking the greatest error of all terms of summing up the coefficients, which gives:

$$\left(\frac{\delta^5}{15} + \frac{\delta^5}{15} + \frac{\delta^5}{120} + \frac{\delta^5}{120}\right) \times \max|f^{(5)}(x)| = \frac{2\delta^5}{3} \max|f^{(5)}(x)| \tag{6}$$

Plugging Equation (6) back into Equation (5), and rearranging, we have:

$$f'(x) \leq \frac{f(x+2\delta) - 8f(x-\delta) + 8f(x+\delta) - f(x-2\delta)}{12\delta} + \frac{\delta^4}{18} \max|f^{(5)}(x)| \tag{7}$$

Hence, our estimate of the first derivative should be:

$$\boxed{f'(x) = \frac{f(x+2\delta) - 8f(x-\delta) + 8f(x+\delta) - f(x-2\delta)}{12\delta}} \tag{8}$$

Now, besides the truncation error, we also have a round-off error, ε, which in this case could be written as:

$$E = \frac{f(x+2\delta)\varepsilon_1 - 8f(x-\delta)\varepsilon_2 + 8f(x+\delta)\varepsilon_3 - f(x-2\delta)\varepsilon_4}{12\delta} \tag{9}$$

Taking ε^* as the round off error of $f(x+\delta)$, we can rewrite equation (9) as:

$$E \leq \frac{3\varepsilon*}{\delta}|f(x)| \tag{10}$$

Hence the error term including truncation error becomes

$$E(\delta) \leq \frac{\delta^4}{18}\max|f^{(5)}(x)| + \frac{3\varepsilon*}{\delta}f(x) \tag{11}$$

Notice that δ is not a linear function, but with maxima. Hence, by finding the solution of the first derivative, we could acquire δ that minimize E.

Hence, we have:

$$E'(\delta) = \frac{2\delta^3}{9}|f^{(5)}(x)| - \frac{3\varepsilon^*}{\delta^2}|f(x)| = 0 \tag{12}$$

which gives:

$$\boxed{\delta_{\min} = \sqrt[5]{\frac{27\varepsilon^*|f(x)|}{2|f^{(5)}(x)|}}} \tag{13}$$

Here, we apply our results to the code at $x = 42$, with function $f(x) = e^x$ and $f(x) = 0.01e^x$. And the results from python shows:

```
The derivative of exp(x) at  42 is 1.739274941516718e+18
the fractional error is 2.175037927543144e-12
The derivative of exp(0.01x) at  42 is 0.015219615556183353
the error is 1.9606538614880265e-13
```

In which the fractional error is satisfactory.

2 Problem 2

Here, we could like to evaluate the error using of a two point numerical differentiator. Using the same method as Problem one, we have:

$$E(\delta) = \frac{\delta^2}{6}|f'''(x)| - \frac{\varepsilon^*}{\delta}|f(x)| = 0 \tag{14}$$

which the optimal value of δ to achieve the smallest error would be the zero solution of the first derivative.

$$E'(\delta) = \frac{\delta}{3}|f'''(x)| - \frac{\varepsilon^*}{\delta^2}|f(x)| = 0 \tag{15}$$

which gives:

$$\boxed{\delta_{\min} = \sqrt[3]{\frac{3\varepsilon^*|f(x)|}{|f'''(x)|}}} \tag{16}$$

3 Problem 3

Straightforward, see comment in code

4 Problem 4

When using the cosine function, the polynomial and cubic spline fit generates an error on the order of 10^{-11} and 10^{-12} respectively. However when using the rational function fit, the error term is on the order of 10^{-6}. This discrepancy in error should be cause by the fact that the cosine function could not be approximated well by the divisions of two functions.

When evaluating the Lorentz function, I generated an error on the order of 10^{-9} and 10^{-10} respectively using the poly and cubic spline fit. However, when choosing the rational function fit with $n = 2$ and $m = 2$, we have the error term down to 10^{-16}. If we choose $n = 4$ and $m = 5$, we error increase to the order of 10^{-3}. However, if we use np.linalg.pinv instead, the error term for $n = 4$ and $m = 5$ decreases down to 10^{-16}. This allows python to evaluate a pseudo inverse of the matrices that are singular and non-square. Previously, np.linalg.inv is only used to evaluate non-singular squared matrix and failed for other conditions. The values of p and q becomes extremely close to 0 with the np.linalg.pinv evaluation. And the constant term in the denominator is to ensure we are not dividing by 0.