Project 1

Error Bound Comparisons in Numerical Differentiation

Govher Sapardurdyyeva
South Dakota State University
Math 374 - Scientific Computation
Jung-Han Kimn
February 14, 2025

Abstract

This project analyzes the difference between truncation and rounding error for the given formulas. We investigate how truncation and rounding errors affect the accuracy of these methods, focusing on the impact of h value on the total error. Through computational analysis using Python, we demonstrate that the central difference method offers better accuracy due to its lower truncation error. Additionally, we identify the optimal h value that minimizes total error, balancing both truncation and rounding errors. Our findings show that while both methods suffer from rounding error at very small h value, the central difference method handles it more effectively.

Introduction

We will begin by evaluating the formulas and deriving the truncation and rounding error equations. This will allow us to compute these errors using computational technology. To accomplish this, I will use Python and its various packages, which provide efficient tools for numerical computation and error analysis. Once the calculations are performed, I will explain the graphs and tables generated by the computer, which will visually represent the errors and help us better understand their behavior as the values change. These visualizations will offer insights into how the errors evolve and provide a clearer comparison between the different formulas.

Truncation Error

Truncation errors occur when an approximation is used instead of an exact mathematical procedure. This happens when an infinite series, an integral, or a differential equation is approximated by a finite number of terms or steps.

In this project we were given two different formulas to approximate:

$$f'(x) = \frac{f(x+h)-f(x)}{h}$$

and

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

for *f(x) = sin(x)* at *x = 1*, as $h \to 0$.

We start by expanding the expression $f\,(\,sinx{+}h\,)$ at x = 1 using Taylor series:

$$f(sin(1) + h) = sin(1) + hcos(1) - \frac{h^2}{2}cos(1) + ...$$

by rearranging the terms we end up with:

$$E_{trunc} \approx -\frac{1}{2}cos(1)\ +.... \textbf{ (1)}$$

Next, we find the truncation error using the second formula. We start by expanding it in the Taylor series. Considering our previous expansion for *f (sin(x) + h),* now let's expand *f(sin(x) - h)* at x = 1.

$$f(sin(1)\ -\ h)\ =\ sin(1)\ -\ hcos(1)\ -\frac{h^2}{2}sin(1)\ +...$$

By subtracting the terms, we find the truncation error bound for the central difference method:

$$E_{trunc} \approx -\frac{h^3}{6}cos(1) \textbf{ (2)}$$

The first formula for truncation error includes both linear and nonlinear terms, whereas the second formula only involves nonlinear terms. This indicates that the second formula converges faster as h→0, meaning it has a smaller error and is more accurate in terms of truncation.

Rounding Error

Rounding errors occur because computers have limited capacity to represent numerical values, both in terms of magnitude and precision. This limitation is a result of the finite number of bits used to store numbers. Machine epsilon represents the smallest difference between two representable floating-point numbers and helps us quantify the rounding error. Using this, we can analyze how the rounding error behaves for both formulas as it depends on the precision of the computer's representation of numbers. By using machine epsilon, we can observe how rounding errors for the first and second formulas behave:

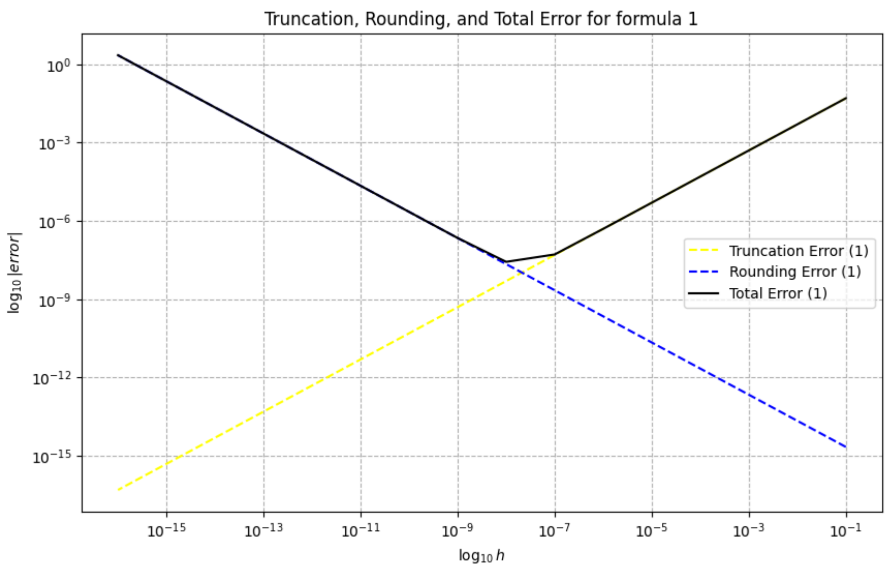
$$E_{round} \approx \frac{\varepsilon}{h} \quad \textbf{(1)}$$

$$E_{round} \approx \frac{\varepsilon}{2h} \quad \textbf{(2)}$$

Analysis

By using the pseudocode from *Numerical Mathematics and Computing* (Chapter 1.1, page 12) as a reference, Python was employed to compute the truncation and rounding errors. By translating the pseudocode into Python, I could directly apply the formulas, compute the errors, and generate results that demonstrate how the errors behave under various conditions.

In Figure 1, we can see that the intersection of the truncation and rounding errors represents the optimal value for h. This point is significant because it minimizes the total error, balancing both truncation and rounding errors. As $h \to 0$, truncation error decreases, but rounding error increases due to the limitations in precision.

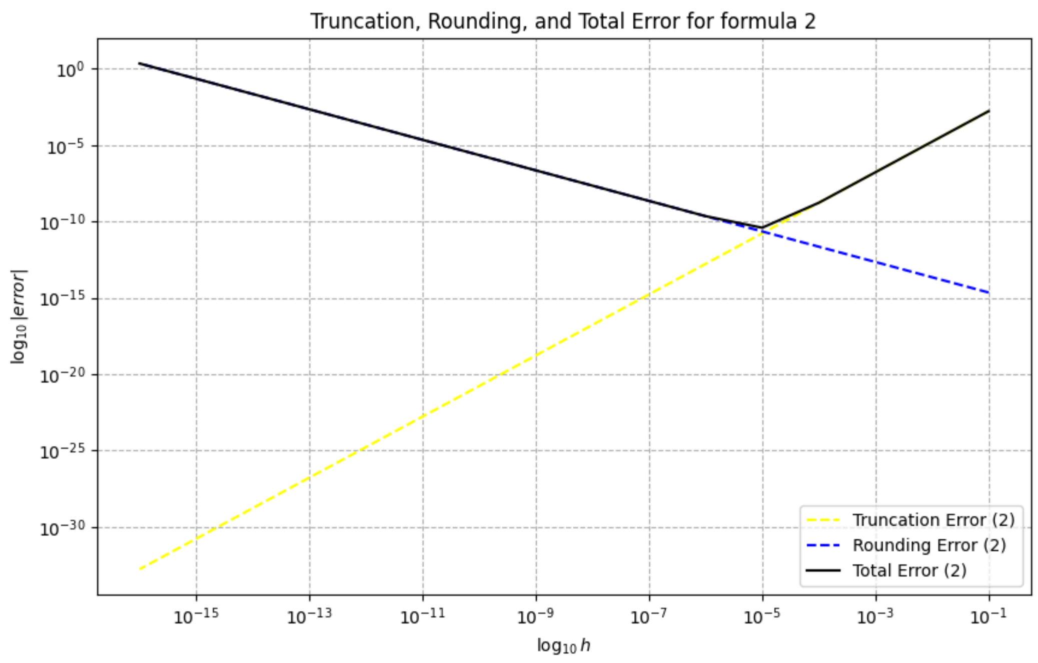


(Figure 1)

By looking at the intersection at the figure we can see that the optimal value of h lies between $1 \cdot 10^{-5}$ *and* $1 \cdot 10^{-8}$.

Figure 2 displays a similar analysis for Formula 2, but it shows that the total error remains lower compared to the method in formula 1. This confirms that Formula 2 offers better accuracy. The second difference method, in particular, provides better results for the same ***h*** value, making it the preferred choice for numerical differentiation.

As ***h*** becomes larger, truncation error becomes the dominant factor, leading to inaccurate results. On the other hand, for very small values of ***h***, rounding error increases exponentially, which can cause numerical differentiation to become unstable.



(Figure 2)

Conclusion

In this project, we looked at numerical differentiation errors and came to a few key conclusions. The first formula for the difference method is better because it has lower truncation error, making it a more accurate choice for numerical differentiation. We also found that if the ***h*** value is too large, truncation error becomes larger and leads to inaccurate results. On the other hand, if the ***h*** value is too small, rounding error increases, which makes the calculations less accurate.

Both methods experience problems when the ***h*** value is very small, but the second formula handles it slightly better. While rounding error grows exponentially as ***h*** becomes smaller, the second method remains more stable and accurate compared to the first one. This makes it a more reliable choice when working with small values of ***h***, as it reduces the impact of rounding error.

Data Table

h value	**Truncation 1**	**Truncation 2**	**Rounding errors**	**Total 1**	**Total 2**
$1 \cdot 10^{-1}$	$5.00 \cdot 10^{-2}$	$1.67 \cdot 10^{-3}$	$2.22 \cdot 10^{-15}$	$5.00 \cdot 10^{-2}$	$1.67 \cdot 10^{-3}$
$1 \cdot 10^{-2}$	$5.00 \cdot 10^{-3}$	$1.67 \cdot 10^{-5}$	$2.22 \cdot 10^{-14}$	$5.00 \cdot 10^{-3}$	$1.67 \cdot 10^{-5}$
$1 \cdot 10^{-3}$	$5.00 \cdot 10^{-4}$	$1.67 \cdot 10^{-7}$	$2.22 \cdot 10^{-13}$	$5.00 \cdot 10^{-4}$	$1.67 \cdot 10^{-7}$
$1 \cdot 10^{-4}$	$5.00 \cdot 10^{-5}$	$1.67 \cdot 10^{-9}$	$2.22 \cdot 10^{-12}$	$5.00 \cdot 10^{-5}$	$1.67 \cdot 10^{-9}$
$1 \cdot 10^{-5}$	$5.00 \cdot 10^{-6}$	$1.67 \cdot 10^{-11}$	$2.22 \cdot 10^{-11}$	$5.00 \cdot 10^{-6}$	$3.89 \cdot 10^{-11}$
$1 \cdot 10^{-6}$	$5.00 \cdot 10^{-7}$	$1.67 \cdot 10^{-13}$	$2.22 \cdot 10^{-10}$	$5.00 \cdot 10^{-7}$	$2.22 \cdot 10^{-10}$
$1 \cdot 10^{-7}$	$5.00 \cdot 10^{-8}$	$1.67 \cdot 10^{-15}$	$2.22 \cdot 10^{-9}$	$5.22 \cdot 10^{-8}$	$2.22 \cdot 10^{-9}$
$1 \cdot 10^{-8}$	$5.00 \cdot 10^{-9}$	$1.67 \cdot 10^{-17}$	$2.22 \cdot 10^{-8}$	$2.72 \cdot 10^{-8}$	$2.22 \cdot 10^{-8}$
$1 \cdot 10^{-9}$	$5.00 \cdot 10^{-10}$	$1.67 \cdot 10^{-19}$	$2.22 \cdot 10^{-7}$	$2.23 \cdot 10^{-7}$	$2.22 \cdot 10^{-7}$
$1 \cdot 10^{-10}$	$5.00 \cdot 10^{-11}$	$1.67 \cdot 10^{-21}$	$2.22 \cdot 10^{-6}$	$2.22 \cdot 10^{-6}$	$2.22 \cdot 10^{-6}$
$1 \cdot 10^{-11}$	$5.00 \cdot 10^{-12}$	$1.67 \cdot 10^{-23}$	$2.22 \cdot 10^{-5}$	$2.22 \cdot 10^{-5}$	$2.22 \cdot 10^{-5}$
$1 \cdot 10^{-12}$	$5.00 \cdot 10^{-13}$	$1.67 \cdot 10^{-25}$	$2.22 \cdot 10^{-4}$	$2.22 \cdot 10^{-4}$	$2.22 \cdot 10^{-4}$
$1 \cdot 10^{-13}$	$5.00 \cdot 10^{-14}$	$1.67 \cdot 10^{-27}$	$2.22 \cdot 10^{-3}$	$2.22 \cdot 10^{-3}$	$2.22 \cdot 10^{-3}$
$1 \cdot 10^{-14}$	$5.00 \cdot 10^{-15}$	$1.67 \cdot 10^{-29}$	$2.22 \cdot 10^{-2}$	$2.22 \cdot 10^{-2}$	$2.22 \cdot 10^{-2}$
$1 \cdot 10^{-15}$	$5.00 \cdot 10^{-16}$	$1.67 \cdot 10^{-31}$	$2.22 \cdot 10^{-1}$	$2.22 \cdot 10^{-1}$	$1.67 \cdot 10^{-1}$
$1 \cdot 10^{-16}$	$5.00 \cdot 10^{-17}$	$1.67 \cdot 10^{-33}$	$2.22 \cdot 10^{0}$	$2.22 \cdot 10^{0}$	$1.67 \cdot 10^{0}$

References

Cheney, W., & Kincaid, D. (2018). *Numerical mathematics and computing* (7th ed.). Cengage Learning.