Lab 1: Numerical Errors and Simple Numerical Integration

Umi Yamaguchi
September 30, 2024

Background

This lab explores numerical errors, particularly roundoff errors arising from floating-point arithmetic. Precision limitations in computers can lead to significant deviations in calculations, especially in tasks like computing standard deviation, evaluating polynomials, and performing numerical integration. We will investigate these errors and their effects on calculations using two methods of computing standard deviation, evaluating roundoff error in polynomial expressions, and comparing the accuracy of numerical integration methods.

1. Exploring numerical issues with standard deviation calculations

In this question, we compare two methods of calculating standard deviation—one that requires two passes through the data and another that uses a one-pass formula. The objective is to explore how these methods differ in terms of numerical accuracy and identify any potential issues with the one-pass approach.

a)

We calculate the standard deviation of a dataset using two methods: the standard two-pass method and the one-pass method. The true value is computed using numpy.std(array, ddof=1).

Two-pass method (equation 1):

$$\sigma = \sqrt{\frac{1}{n-1}\sum_{i=1}^{n}(x_i - \bar{x})^2}$$

where $\bar{x}$ is the mean of the dataset and σ is the standard deviation. This method requires calculating the mean first, followed by the sum of squared deviations.

One-pass method (equation 2):

$$\sigma = \sqrt{\frac{1}{n-1}\left(\sum_{i=1}^{n} x_i^2 - n\bar{x}^2\right)}$$

This formula allows for calculating the standard deviation in a single pass through the data.

We compare the results from both methods by calculating the relative error:
$$\text{Relative error} = \frac{\sigma_{\text{calculated}} - \sigma_{\text{true}}}{\sigma_{\text{true}}}$$

To ensure numerical stability, we implement a check to prevent taking the square root of a negative number in the one-pass method, with a stopping condition if such an error occurs.

b)

Using Michelsen's speed of light dataset (`cdata.txt`), we calculate the standard deviation using both methods. The relative error for the two-pass method is 0.0. In contrast, the one-pass method exhibits a small relative error of around -2.287×10^{-9}. We observe infinitesimally small differences, approximately 1×10^{-9}, with the two-pass method showing no noticeable error. One-pass method is less accurate due to numerical precision issues, indicating that it has a larger relative error in magnitude.

c)

We generate two normally distributed sequences with means of 0 and 1×10^7, respectively. The two-pass method shows no significant error for either case. However, the one-pass method's error increases to approximately 1×10^{-3} for the larger mean, demonstrating that when the dataset involves large numbers, the one-pass method suffers from precision loss due to the subtraction of nearly equal large numbers.

d)

We apply a simple workaround to mitigate numerical precision issues in the one-pass method. Instead of directly using the one-pass formula, we first compute the mean of the dataset and subtract this mean from each data point. This centring step ensures that the values involved in subsequent calculations are smaller and more stable, reducing the risk of catastrophic cancellation when dealing with large numbers. After this, we calculate the sum of squared differences from the mean. Finally, the standard deviation is computed using the adjusted data. By subtracting the mean first, we avoid directly summing large values that may cancel each other out, thus minimizing the loss of precision. This adjustment makes the method more numerically stable and accurate, particularly for datasets with large means or values.

2. Exploring roundoff error

In this question, we examine the effects of roundoff error when calculating the polynomial $p(u) = (1-u)^8$ and comparing it with its expanded form $q(u)$:

$$q(u) = 1 - 8u + 28u^2 - 56u^3 + 70u^4 - 56u^5 + 28u^6 - 8u^7 + u^8$$

Although $p(u)$ and $q(u)$ are mathematically equivalent, they can produce different numerical results when evaluated on a computer, particularly for values of u close to 1. This is due to roundoff errors, which arise from the limitations of floating-point arithmetic.
We will use standard error statistics and fractional error to analyze these differences. The error constant $C \approx 10^{-16}$ represents machine precision, and for a sum of N terms $x = x_1 + x_2 + \cdots + x_N$, the standard error σ is given by:

$$\sigma = C\sqrt{\overline{x^2}}\sqrt{N}$$

where $\overline{x^2}$ is the mean of the squared terms and where $C \approx 10^{-16}$.

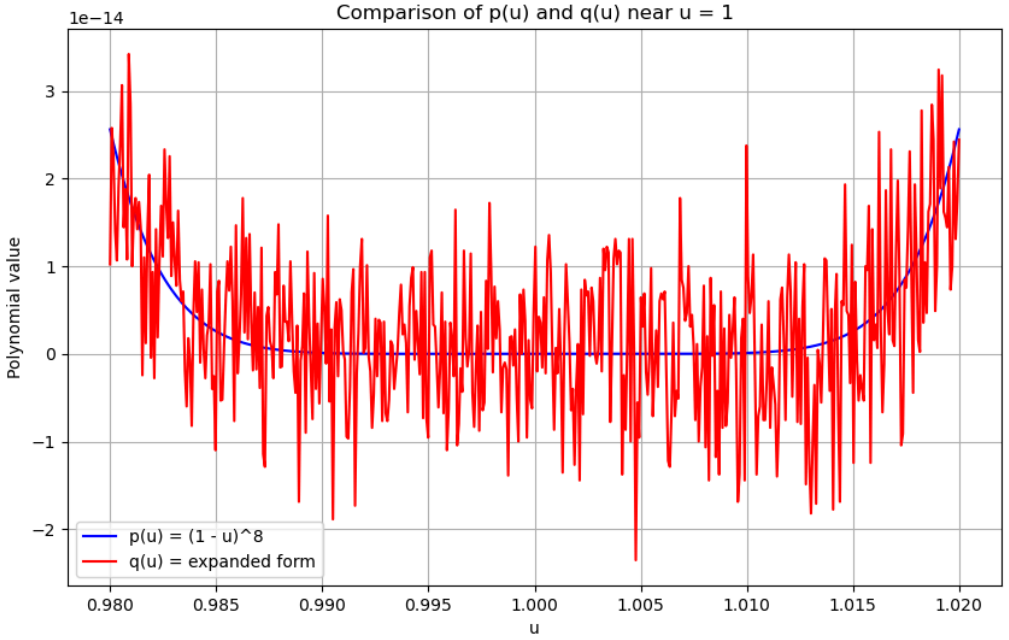
The fractional error f, which measures the error relative to the computed result, is given by:

$$f = \frac{\sigma}{\sum_i x_i} = \frac{C\sqrt{\overline{x^2}}}{\sqrt{N} \cdot \overline{x}}$$

This analysis will show how roundoff errors become more significant as u approaches values near 1, and we will quantify these effects using the above formulas.

a)

We calculate and plot $p(u) = (1-u)^8$ (blue line) and its expanded form $q(u)$ (red line) for the range $0.98 < u < 1.02$. Although these two expressions are mathematically equivalent, they produce different numerical results due to roundoff errors, particularly as u approaches 1.

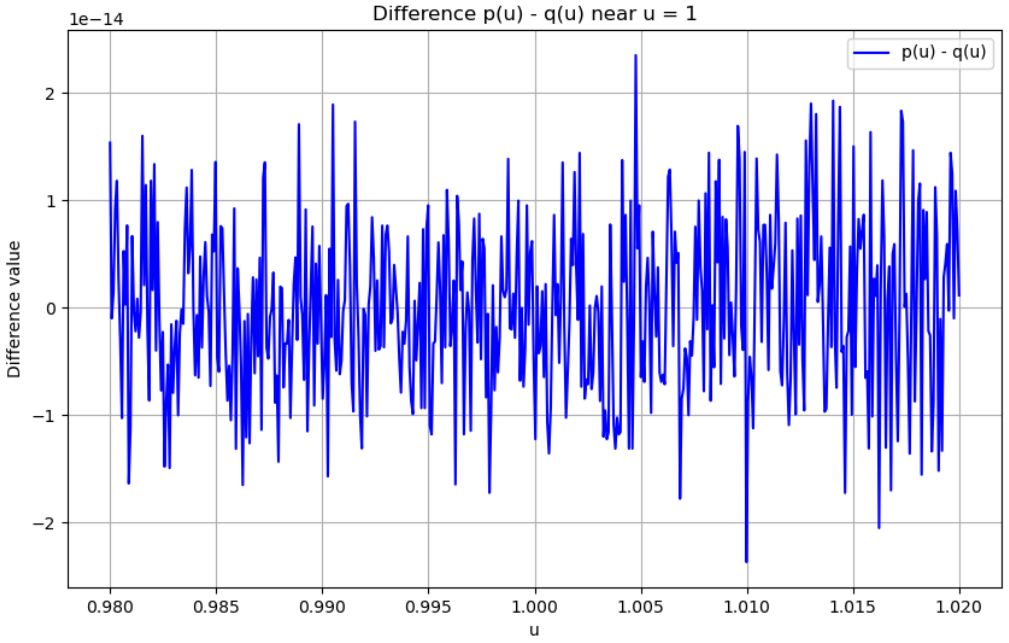


From the graph, we can observe that the expanded form $q(u)$ exhibits significantly more noise compared to the smooth curve of $p(u)$. This increased noise in $q(u)$ is due to the cancellation of

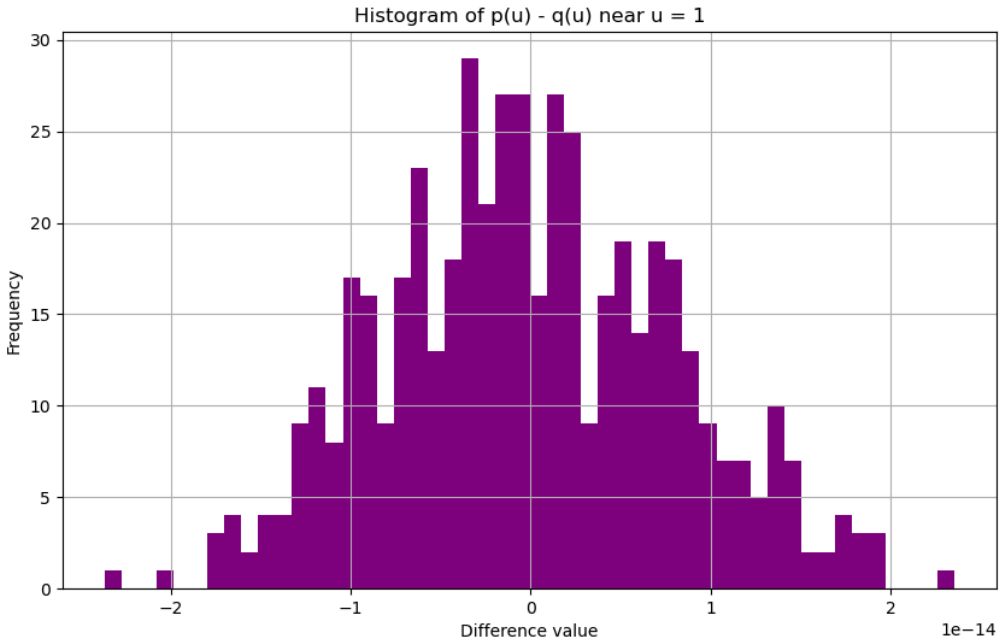
large terms with opposite signs, which amplifies roundoff errors. In the expanded form, terms like $1 - 8u + 28u^2$ involve large numbers that nearly cancel each other out as u approaches 1. However, floating-point arithmetic is not precise enough to handle these cancellations accurately, leading to oscillations and large fluctuations in the computed values. This behaviour demonstrates how roundoff errors dominate the computation of $q(u)$ near $u = 1$, making the expanded form highly unstable in this region.

b)

To further explore the numerical differences between the two functions $p(u) = (1 - u)^8$ and its expanded form $q(u)$, we plot $p(u) - q(u)$ and generate a histogram of the differences for values of u near 1. These plots allow us to understand the nature of the roundoff errors that arise when evaluating these two forms on a computer.



The plot of $p(u) - q(u)$ shows oscillations around zero, which indicates the accumulation of roundoff errors. These fluctuations are a result of the instability introduced by the expanded form $q(u)$, where the cancellation of large terms with opposite signs causes precision loss. As expected, the differences between $p(u)$ and $q(u)$ increase as u approaches 1, where the roundoff errors dominate due to the small differences in the terms of $q(u)$.



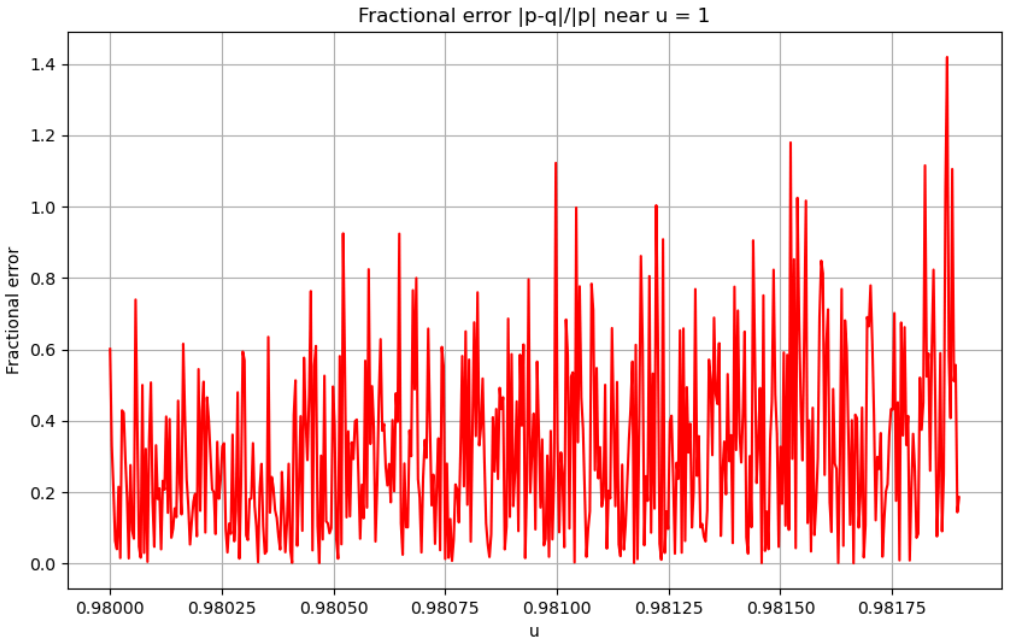
The histogram of $p(u) - q(u)$ provides a more detailed statistical view of the differences. The width of the histogram is directly related to the standard deviation of the roundoff errors, which in this case is approximately 7.986×10^{-15}. This value reflects the random nature of roundoff errors in the computation of the two equivalent expressions. As shown in the histogram, the distribution is relatively wide, indicating that the errors are significant even though the functions are mathematically equivalent. Using the standard error equation we estimate the error to be approximately 1.135×10^{-14}. While this value is not the same as the spread observed in the histogram, it is of the same order of magnitude, confirming that the estimated error provides a reasonable approximation for the actual distribution of the errors. The small discrepancy between the calculated and estimated errors is expected, given the inherent randomness of roundoff errors in floating-point computations.

In calculating N, we count the number of terms in the polynomial expansion of $p(u) - q(u)$, square each term, and take the mean. For simplicity, we let $u = 1$ in this calculation, as this is where the errors are most pronounced.

c)

In this part, we aim to demonstrate that for values of u somewhat greater than 0.980 but less than 1.0, the fractional error between $p(u)$ and $q(u)$ approaches 100%. The fractional error is calculated using the formula:

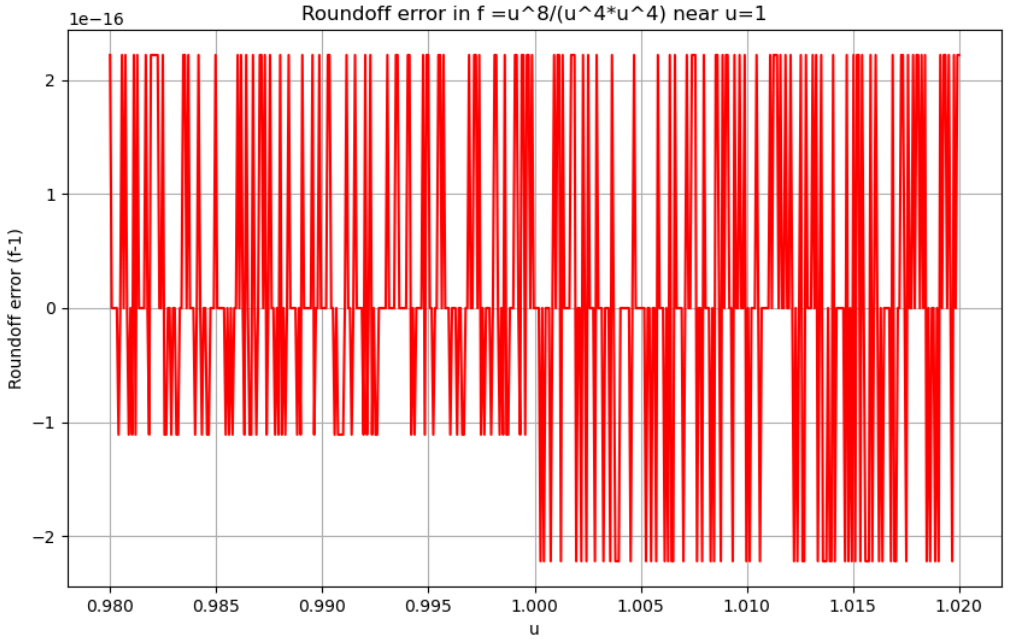
$$\text{Fractional error} = \frac{|p(u) - q(u)|}{|p(u)|}$$



From the graph, we observe that as u approaches 1, the fractional error fluctuates significantly, eventually surpassing 1, which corresponds to a 100% error. The fractional error becomes particularly pronounced at around $u = 0.9819$, where the error exceeds 100%, which we calculated by using the fractional error equation.
The plot confirms that the fractional error increases rapidly as u nears 1. This behaviour is expected because roundoff errors dominate the calculation in this region. The expanded form $q(u)$ suffers from rounding issues. These errors are reflected in the significant deviations between $p(u)$ and $q(u)$. While the error is somewhat noisy and fluctuates as u increases, the overall trend clearly shows that the fractional error grows more pronounced the closer u is to 1. This is consistent with the prediction that roundoff errors cause the discrepancy between $p(u)$ and $q(u)$ to become more severe near $u = 1$.

d)

In this part, we calculate the roundoff error in the expression $f = u^8/(u^4 \cdot u^4)$, which theoretically should equal 1 for all values of u. However, due to roundoff error in floating-point arithmetic, the result will deviate slightly from 1, especially for values of u near 1.



The plot of $f - 1$ versus u illustrates how the roundoff error fluctuates for values of u close to 1. From the graph, we can observe that the roundoff error oscillates between positive and negative values, with an amplitude reaching approximately $\pm 2 \times 10^{-16}$. This fluctuation is consistent with the expected precision limitations of floating-point arithmetic, which is around 1×10^{-16}.
The nature of the error occurs because floating-point operations cannot perfectly represent the result of dividing two large numbers that are mathematically equivalent, particularly as u approaches 1. The result is that small errors accumulate, leading to deviations from the expected value of 1.
When comparing the observed roundoff error to the estimated error predicted by textbook equation (4.5), $\text{estimated error} = C \cdot f$, where $C \approx 10^{-16}$, we find that the calculated mean roundoff error is approximately 1.5×10^{-17}, which is smaller than the mean estimated error of 1×10^{-16}. This is consistent with the expected behaviour, as the estimated error provides an upper bound, while the actual roundoff error is typically smaller but fluctuates around this upper limit.

3. Trapezoidal and Simpson's rules for integration

Numerical integration techniques, such as the Trapezoidal Rule and Simpson's Rule, are commonly used to approximate the definite integrals of functions when an exact analytical solution is difficult to obtain. Both methods work by partitioning the interval of integration into sub-intervals, and their accuracy improves as the number of slices increases. In this question, we compare these two methods by approximating the integral:

$$I = \int_0^1 \frac{4}{1+x^2}\, dx$$

We start by defining the function to be integrated and set up the exact value of I, and then Implement the Trapezoidal Rule and Simpson's Rule. After that, calculate the errors for $N = 4$ and output the results. We then find the number of slices N for each method to achieve an error of 1.0×10^{-9}. Then estimate the error for the Trapezoidal Rule based on different slice counts and finally compare the efficiency of the two methods by assessing computation time and accuracy.

We begin by defining the function to be integrated, setting the exact value of the integral $I = \pi$, and implementing both the Trapezoidal Rule and Simpson's Rule. Afterward, we calculate the integral using $N = 4$ slices for each method and evaluate the respective errors. Next, we determine the number of slices N required for both methods to achieve an error threshold of 1.0×10^{-9}. Following this, we estimate the error for the Trapezoidal Rule by analyzing results from different slice counts, such as $N_1 = 16$ and $N_2 = 32$. Finally, we compare the efficiency of both methods in terms of computation time and accuracy, revealing that Simpson's Rule requires fewer slices and performs significantly better than the Trapezoidal Rule, making it the more efficient and accurate choice for this integration task.

a)

The exact value of the integral I is π, approximately 3.1415926535897936 from computation on the computer. This value will serve as the benchmark against which we compare the results obtained using the Trapezoidal Rule and Simpson's Rule.

b)

We computed the integral using both the Trapezoidal Rule and Simpson's Rule with $N = 4$ slices. For the Trapezoidal Rule, we obtained a relative error of about 0.0104, whereas for Simpson's Rule, the error was much smaller at approximately 2.4×10^{-5}.

The results clearly show that the Trapezoidal Rule produces a significantly larger error when compared to the exact value of π, while Simpson's Rule is far more accurate with the same number of slices. This demonstrates that Simpson's Rule provides better precision with fewer slices compared to the Trapezoidal Rule.

c)

To achieve an error on the order of 10^{-9}, the number of slices N needed for both methods was calculated. For the Trapezoidal Rule, $N = 16382$ slices were required to yield an error of approximately 6.21×10^{-10}, with a computation time of 0.00838 seconds. In contrast, Simpson's Rule required only $N = 32$ slices to achieve an error of 3.69×10^{-11}, with a much shorter computation time of 6.1×10^{-5} seconds.
These results demonstrate that while the Trapezoidal Rule needs a much larger number of slices to achieve an error on the order of 10^{-9}, Simpson's Rule can reach this level of accuracy with far fewer slices. Additionally, Simpson's Rule is not only more accurate but also computationally more efficient, taking significantly less time to compute the integral.

d)

Using the practical estimation of the error formula from the textbook equation 5.28 where, $\epsilon_2 = (N_2 - N_1)\frac{1}{3}$ for the Trapezoidal Rule, we calculated the error for $N_2 = 32$ slices, using $N_1 = 16$ as a reference. The practical estimated error for the Trapezoidal Rule with $N_2 = 32$ slices is approximately 0.0001628. This estimation indicates that even with an increased number of slices, the Trapezoidal Rule still has a noticeable error compared to the exact value of the integral. Although this represents an improvement over the error obtained with $N = 4$, the error remains significantly larger than that obtained using Simpson's Rule, even when both methods use the same number of slices.

e)

In some cases, the practical error estimation method used for the Trapezoidal Rule cannot be directly applied to Simpson's Rule. This is because Simpson's Rule uses a quadratic approximation for integration, whereas the Trapezoidal Rule uses a linear approximation. The error scaling for Simpson's Rule behaves differently due to its higher-order polynomial approximation, which follows an $O(h^4)$ error behaviour.

To adapt the practical error estimation method for Simpson's Rule, we would need to account for this higher-order behaviour. The error estimate for Simpson's Rule should include terms that reflect its $O(h^4)$ scaling. Adjusting the step size accordingly would ensure that the error estimate decreases with finer approximations. Thus, Simpson's Rule requires a different approach to error estimation to take full advantage of its higher accuracy compared to the Trapezoidal Rule for a given number of slices.