https://doi.org/10.54361/ajmas.2584132

Original article

Error Estimation and Floating-Point Effects in Numerical Integration: A Comparative Study

Naima Shamsi

Department of Mathematics, Faculty of Science, Sabha University, Sebha
Corresponding Email. nai.saadshamsi@sebhau.edu.ly

Abstract
This study investigates how truncation (discretization) error and floating-point round-off jointly influence the practical accuracy of three classical composite quadrature rules: the composite rectangle rule, the composite trapezoidal rule, and the composite Simpson rule. After summarizing the standard theoretical error orders for sufficiently smooth integrands, the methods are implemented in MATLAB using explicit IEEE-754 single-precision (32-bit) and double-precision (64-bit) arithmetic to isolate precision effects. Numerical experiments are performed on benchmark test integrals with known analytical values for four representative integrands, namely sin(x), cos(x),e^x and $log(x)$, and absolute errors are reported for each method. Across the tested cases, double precision substantially suppresses round-off effects and enables markedly smaller errors, with several results approaching machine-level magnitudes. Simpson's rule generally provides the highest accuracy in double precision, attaining absolute errors on the order of 10^{-13}–10^{-16} for multiple test functions, while single precision typically yields errors in the 10^{-8}–10^{-6} range depending on the integrand and method. The exponential function exhibits larger errors in single precision, consistent with its rapid growth and increased sensitivity to accumulated rounding during composite summation. Overall, the results demonstrate that achieving reliable high-accuracy numerical integration requires not only an appropriate quadrature order but also sufficient arithmetic precision, and that empirical evaluation across multiple test functions remains essential when floating-point effects are non-negligible.
Keywords: Numerical Integration, Error Estimation, Floating-Point Precision, Rectangle Rule, Trapezoidal Rule, Simpson's Rule.

Introduction

Numerical integration is a cornerstone of numerical analysis, providing approximate solutions for integrals that are difficult or impossible to evaluate analytically. Different numerical integration methods vary in terms of computational accuracy, efficiency, and convergence speed, making error estimation a critical factor in their evaluation. In practical computations, the representation of numbers using floating-point arithmetic introduces additional numerical errors, and recent studies on mixed-precision computation have demonstrated that rounding and precision effects can significantly influence the reliability of numerical algorithms [1]. These effects may degrade the accuracy of integration results, particularly when a large number of subintervals is used or when extensive summations are performed [2].

This study aims to conduct a comprehensive comparative analysis of several numerical integration techniques by evaluating their estimation accuracy, quantifying the associated error magnitudes, and examining their computational performance in practical applications. The analysis is supported by simulation experiments implemented in MATLAB, which illustrate the performance of different numerical schemes in solving test problems [3]. The importance of numerical integration arises from its wide range of applications in many scientific and engineering fields, including the numerical solution of differential equations, scientific simulations, and data analysis. Although the theoretical foundations of many classical methods are well established, the selection of an optimal technique strongly depends on the characteristics of the integrand, the sensitivity of the results to numerical errors, and the computational performance of the method under practical constraints [4]. Consequently, the combined study of error estimation, floating-point effects, and performance evaluation plays a central role in identifying the most effective numerical strategies that achieve a suitable balance between computational accuracy and execution efficiency, which constitutes the primary objective of this work.

Methods

Numerical integration (Rectangle Rule)

The Rectangle Rule is one of the simplest numerical integration methods. Also known as the midpoint rule, it approximates the area under the curve using rectangles. The method divides the interval $[a, b]$into $n$equal subintervals and uses Riemann sums to estimate the integral. Specifically, the integral is approximated as the sum of the areas of these rectangles, each having a width corresponding to the subinterval length [5].

***Received**: 27-10-2025 - **Accepted**: 21-12-2025 - **Published**: 30-12-2025*

$$\int_a^b f(x)\,dx \approx \sum_{i=0}^{n-1} f(x_i)\,\Delta x$$

Where:

- $\Delta x = \frac{b-a}{n}$is the width of each subinterval,
- $x_i$is a point in each subinterval (can be a left endpoint, right endpoint, or midpoint).

1. Types of Rectangle Rule
   - Left Rectangle Rule: $x_i = a + i\Delta x$
   - Right Rectangle Rule: $x_i = a + (i+1)\Delta x$
   - Midpoint Rule: $x_i = a + \left(i + \frac{1}{2}\right)\Delta x$– usually more accurate than left or right endpoints.
2. Error Estimation

The approximation error depends on the derivatives of the function as follows:
For the left/right endpoints:

$$E \approx \frac{(b-a)^2}{2n} \max_{x\in[a,b]} \mid f'(x) \mid$$

For the midpoint rule (more accurate, like Simpson's approximation):

$$E \approx \frac{(b-a)^3}{24n^2} \max_{x\in[a,b]} \mid f''(x) \mid$$

3. Error Derivation (Mathematical Proof)

Using the Mean Value Theorem for Integrals, for each subinterval $[x_i, x_{i+1}]$there exists $\xi_i \in [x_i, x_{i+1}]$such that:

$$\int_{x_i}^{x_{i+1}} f(x)\,dx = f(\xi_i)\Delta x$$

The approximation uses $f(x_i)$, so the local error on the subinterval is:

$$E_i = \int_{x_i}^{x_{i+1}} f(x)\,dx - f(x_i)\Delta x = (f(\xi_i) - f(x_i))\Delta x$$

Expand using the Taylor series around x_i:$f(\xi_i)$using Taylor series around x_i:

$$f(\xi_i) = f(x_i) + f'(\eta_i)(\xi_i - x_i), \eta_i \in [x_i, \xi_i]$$

Then:

$$E_i = f'(\eta_i)(\xi_i - x_i)\Delta x$$

Since $0 \le \xi_i - x_i \le \Delta x$:

$$\mid E_i \mid \le \Delta x^2 \max_{x\in[x_i, x_{i+1}]} \mid f'(x) \mid$$

Total Error over ￼: Summing over all subintervals.$[a, b]$:Summing over all subintervals.

$$\mid E \mid = \mid \int_a^b f(x)dx - \sum_{i=0}^{n-1} f(x_i)\Delta x \mid \le \sum_{i=0}^{n-1} \mid E_i \mid \le n\Delta x^2 \max_{x\in[a,b]} \mid f'(x) \mid$$

Substitute $n\Delta x = b - a$:

$$E = \mid E_{\text{total}} \mid \le (b-a)\Delta x \max_{x\in[a,b]} \mid f'(x) \mid = \frac{(b-a)^2}{n} \max_{x\in[a,b]} \mid f'(x) \mid$$

- The error is proportional to the first derivative of the function.
- The error decreases linearly with the number of subintervals n.
- Midpoint Rule improves the error order by$O(\Delta x^2)$.

Trapezoidal Rule

The Trapezoidal Rule is a numerical integration method that approximates the area under a curve using trapezoids instead of rectangles. The interval $[a, b]$is divided into $n$equal subintervals of width [6]:

$$\Delta x = \frac{b-a}{n}, x_i = a + i\Delta x, i = 0,1,\dots,n$$

1. The integral is approximated as:

$$\int_a^b f(x)\,dx \approx \frac{\Delta x}{2}[f(x_0) + 2\sum_{i=1}^{n-1} f(x_i) + f(x_n)]$$

Here:

- The first and last terms $f(x_0)$and $f(x_n)$are counted once.$f(x_0)$and $f(x_n)$are counted once.
- The interior points $f(x_1), \dots, f(x_{n-1})$are multiplied by 2 because they are shared between two trapezoids.

2. Error Estimation

1. If $f''(x)$is continuous on $[a, b]$, the trapezoidal error is given by:

$$E_T = -\frac{(b-a)^3}{12n^2} f''(\xi), \text{for some } \xi \in [a, b]$$

- The error decreases proportionally to $1/n^2$, making it more accurate than the rectangle rule for smooth functions.

Received: *27-10-2025* - ***Accepted***: *21-12-2025* - ***Published***: *30-12-2025*

- The negative sign indicates that the approximation may overestimate or underestimate, depending on the concavity of $f(x)$.

3. Error Derivation (Mathematical Proof)

If $f''(x)$is continuous on $[a,b]$, the local error on a single subinterval $[x_i,x_{i+1}]$is derived from the Taylor expansion of $f(x)$around x_i:

$$f(x)=f(x_i)+f'(x_i)(x-x_i)+\frac{f''(\xi_i)}{2}(x-x_i)^2, \xi_i\in[x_i,x_{i+1}]$$

Integrating over $[x_i,x_{i+1}]$:

$$\int_{x_i}^{x_{i+1}} f(x)\,dx=f(x_i)\Delta x+\frac{f'(x_i)}{2}(\Delta x)^2+\frac{f''(\xi_i)}{6}(\Delta x)^3$$

The Trapezoidal approximation for the same subinterval is:

$$T_i=\frac{\Delta x}{2}[f(x_i)+f(x_{i+1})]$$

Expanding $f(x_{i+1})$by Taylor series:

$$f(x_{i+1})=f(x_i)+f'(x_i)\Delta x+\frac{f''(\eta_i)}{2}(\Delta x)^2, \eta_i\in[x_i,x_{i+1}]$$

Then:

$$T_i=\frac{\Delta x}{2}[f(x_i)+f(x_i)+f'(x_i)\Delta x+\frac{f''(\eta_i)}{2}(\Delta x)^2]=f(x_i)\Delta x+\frac{f'(x_i)}{2}(\Delta x)^2+\frac{f''(\eta_i)}{4}(\Delta x)^3$$

Local Error on the subinterval:

$$E_i=\int_{x_i}^{x_{i+1}} f(x)\,dx-T_i=\frac{f''(\xi_i)}{6}(\Delta x)^3-\frac{f''(\eta_i)}{4}(\Delta x)^3\approx-\frac{f''(\zeta_i)}{12}(\Delta x)^3, \zeta_i\in[x_i,x_{i+1}]$$

Total Error over$[a,b]$

Summing over all subintervals:

$$E_T=\sum_{i=0}^{n-1}E_i\approx-\frac{(\Delta x)^3}{12}\sum_{i=0}^{n-1}f''(\zeta_i)\approx-\frac{(b-a)(\Delta x)^2}{12}f''(\xi), \xi\in[a,b]$$

Substituting $\Delta x=\frac{b-a}{n}$:

$$E_T=-\frac{(b-a)^3}{12n^2}f''(\xi), \xi\in[a,b]$$

This is the standard error formula for the Trapezoidal Rule.

The error decreases quadratically with n, making the trapezoidal method more accurate than the rectangle method.

The error depends on the second derivative of the function: larger curvature → larger error

Simpson's Rule

Simpson's rule is a numerical analysis method for approximating definite integrals. It works by approximating the integrand as a quadratic polynomial over intervals and taking a weighted average. - The error in Simpson's rule is proportional to $(b-a)^5$, providing an exact result for polynomials up to degree [6]

1. The Simpson's Rule approximation is:

For a continuous function $f(x)$on the interval $[a,b]$, divided into an even number n of subintervals of width:

$$\Delta x=\frac{b-a}{n},\quad x_i=a+i\Delta x,\quad i=0,1,\dots,n$$

$$\int_a^b f(x)\,dx\approx\frac{\Delta x}{3}[f(x_0)+4\sum_{\text{odd } i}f(x_i)+2\sum_{\text{even } i, i\neq 0,n}f(x_i)+f(x_n)]$$

The first and last terms are counted once.

Odd-indexed interior points are multiplied by 4, even-indexed (except first and last) by 2.

Requires n to be even.

2. Error Estimation (Mathematical Proof)

2. If $f^{(4)}(x)$(fourth derivative) is continuous on $[a,b]$, the error of Simpson's Rule is:

$$E_S=-\frac{(b-a)^5}{180n^4}f^{(4)}(\xi),\text{for some } \xi\in[a,b]$$

3. Derivation Idea:

a) Simpson's Rule is derived using quadratic interpolation on each pair of subintervals $[x_{2i},x_{2i+2}]$.

b) Using Taylor expansion, the local error on two subintervals is proportional to $(\Delta x)^5f^{(4)}(\zeta_i)$for some $\zeta_i\in[x_{2i},x_{2i+2}]$.

c) Summing over all $n/2$pairs of subintervals gives the total error formula above.

4. Error Derivation of Simpson's Rule

Received: *27-10-2025* - ***Accepted***: *21-12-2025* - ***Published***: *30-12-2025*

Let $f(x)$be a function whose fourth derivative $f^{(4)}(x)$is continuous on the interval $[a, b]$. Divide the interval into even number $n$of subintervals of width:

$$\Delta x = \frac{b-a}{n}, x_i = a + i\Delta x, i = 0,1, \dots, n$$

Simpson's Rule approximates the integral as:

$$\int_a^b f(x)\,dx \approx \frac{\Delta x}{3}[f(x_0) + 4\sum_{\text{odd } i} f(x_i) + 2\sum_{\text{even } i, i\neq 0,n} f(x_i) + f(x_n)$$

Quadratic Interpolation on Pairs of Subintervals

Simpson's Rule is derived by approximating $f(x)$with a quadratic polynomial over each pair of subintervals $[x_{2i}, x_{2i+2}]$[7].

Let $P_2(x)$be the interpolating quadratic polynomial passing through $(x_{2i}, f(x_{2i})), (x_{2i+1}, f(x_{2i+1})), (x_{2i+2}, f(x_{2i+2}))$.

The local approximation on these two subintervals is:

$$\int_{x_{2i}}^{x_{2i+2}} f(x)\,dx \approx \int_{x_{2i}}^{x_{2i+2}} P_2(x)\,dx$$

Taylor Expansion for Local Error: Using Taylor expansion around x_{2i}:

$$f(x) = P_2(x) + \frac{f^{(4)}(\zeta_i)}{4!}(x - x_{2i})(x - x_{2i+1})(x - x_{2i+2})(x - \cdot), \zeta_i \in [x_{2i}, x_{2i+2}]$$

The local error over two subintervals is proportional to:

$$E_{2i} = -\frac{(\Delta x)^5}{90} f^{(4)}(\zeta_i)$$

- This uses the fact that Simpson's Rule is exact for polynomials of degree ≤ 3.

Total Error Over $[a, b]$

There are $n/2$pairs of subintervals. Summing the local errors gives the total error:

$$E_S = \sum_{i=0}^{n/2-1} E_{2i} = -\frac{(\Delta x)^5}{90}\sum_{i=0}^{n/2-1} f^{(4)}(\zeta_i)$$

Using the Mean Value Theorem for Integrals, there exists $\xi \in [a, b]$such that:

$$\sum_{i=0}^{n/2-1} f^{(4)}(\zeta_i) = \frac{n}{2} f^{(4)}(\xi)$$

Substitute $\Delta x = \frac{b-a}{n}$to get:

$$E_S = -\frac{(\Delta x)^5}{90}\cdot\frac{n}{2} f^{(4)}(\xi) = -\frac{(b-a)^5}{180n^4} f^{(4)}(\xi), \xi \in [a, b]$$

Error Definition and Convergence Study

The absolute error is defined as

$$E_n =| I_{\text{exact}} - I_n |,$$

where I_{exact}denotes the analytical value of the integral and $I_n$is the numerical approximation obtained by the chosen quadrature method.

To investigate the interaction between truncation error and floating-point round-off, the error $E_n$is evaluated for the sequence

$$n = \{10,20,40,80,160,320,640\}.$$

The error is plotted against the step size $h$on log–log axes in order to identify the convergence regime and the saturation region where floating-point effects dominate.

Benchmark Function (Example)

For reproducibility, a benchmark function should be defined, e.g.:

$$f_1(x) = \sin(x), [a, b] = [0, \pi]$$
$$f_2(x) = cos\ (x), [a, b] = [0, \pi]$$
$$f_3(x) = log\ (x + 1), [a, b] = [0, \pi]$$
$$f_4(x) = e^x, [a, b] = [0, \pi]$$

For these experiments, each interval $[a, b]$is divided into $n = 2000$ equal subintervals, ensuring a consistent discretization for numerical integration. These benchmark functions can be replaced or extended with other smooth, oscillatory, or sharply varying functions to verify the generality and robustness of the numerical integration methods.

Received: *27-10-2025* - ***Accepted***: *21-12-2025* - ***Published***: *30-12-2025*

Floating numbers

Definition

Floating numbers, or floating-point numbers, are a way to represent real numbers in computers that allows handling very large or very small values, both positive and negative, while maintaining a certain level of precision. The term "floating" comes from the fact that the decimal point can "float" or move Depending on the number to fit its binary representation in memory [8].

Structure

A floating-point number is generally represented using three main components:

1. Sign: Determines whether the number is positive or negative.
2. Exponent: Determines the position of the decimal point (or the power of 2 in binary).
3. Mantissa or Significand: Represents the actual digits of the number after the decimal point.

The number can be expressed approximately as:

$$\text{Number} = (-1)^{\text{Sign}} \times 1.\text{Mantissa} \times 2^{\text{Exponent}}$$

Precision Types

Single Precision (32-bit)

- 1 bit for sign, 8 bits for exponent, 23 bits for mantissa.
- Approximate precision: 7 decimal digits.
- Uses less memory but is more prone to rounding errors.

Double Precision (64-bit)

- 1 bit for sign, 11 bits for exponent, 52 bits for mantissa.
- Approximate precision: 15–16 decimal digits.
- Less prone to errors and is widely used in scientific and engineering calculations.

Floating-Point Precision and Implementation

All numerical methods are implemented in MATLAB using IEEE-754 arithmetic.
Double-precision experiments employ MATLAB's default double data type.
Single-precision experiments enforce single arithmetic by explicitly casting the grid and all function evaluations to the single type, and by accumulating the quadrature sum in a variable of type single. This strategy isolates the influence of floating-point round-off errors arising from both function evaluation and repeated summation.

Limitations

This study uses a limited set of benchmark integrals and a specific implementation strategy in MATLAB. Observed method rankings can depend on the smoothness of f, the degree of cancellation in the summation, the grid sequence, and implementation choices such as accumulation precision and summation order. Future work should expand the benchmark set (e.g., smooth polynomial-like functions, oscillatory functions, and sharply varying functions) and include numerically robust accumulation strategies (e.g., pairwise or compensated summation) to quantify and reduce floating-point accumulation effects.

Results

This work uses MATLAB to examine numerical integration methods and explore the effects of single and double-precision floating-point numbers on computational error. The numerical experiments were implemented in MATLAB to quantify the absolute errors obtained from the composite rectangle, trapezoidal, and Simpson quadrature rules under single-precision and double-precision floating-point arithmetic. The overall distribution of absolute error magnitudes across the tested methods and precision settings is summarized in the absolute error bar chart (Figure 1). For the sin(x) test case, the single-precision absolute errors were 2.3842e−07 (Rectangle), 8.3447e−07 (Trapezoidal), and 2.3842e−07 (Simpson), whereas the corresponding double-precision absolute errors were 2.0562e−07, 4.1123e−07, and 6.839e−14, respectively (Table 1); the associated single- versus double-precision absolute error curves are presented in Figure 2. Subsequently, for the cos(x) test case, the single-precision absolute errors were 6.7505e−08 (Rectangle), 1.0018e−08 (Trapezoidal), and 1.1735e−08 (Simpson), while the double-precision absolute errors were 3.739e−16, 1.3951e−18, and 1.8335e−16, respectively (Table 2); the corresponding absolute error curves are reported in Figure 3. In addition, for the e^x test case, the single-precision absolute errors were 6.5976e−06 (Rectangle), 4.8465e−06 (Trapezoidal), and 8.7558e−07 (Simpson), and the double-precision absolute errors were 2.2762e−06, 4.5525e−06, and 7.5318e−13, respectively (Table 3); the associated absolute error curves are provided in Figure 4. Finally, for the log(x) test case, the single-precision absolute errors were 1.4705e−07 (Rectangle), 6.2389e−07 (Trapezoidal), and 3.8547e−07 (Simpson), while the double-precision absolute errors were 7.7985e−08, 1.5597e−07, and 6.7502e−14, respectively (Table 4); the corresponding absolute error curves are shown in Figure 5.

Table 1: Absolute Errors for $\sin(x)$

	Rectangle	Trapezoidal	Simpson
Single Precision	2.3842e-07	8.3447e-07	2.3842e-07
Double Precision	2.0562e-07	4.1123e-07	6.839e-14

Table 2: Absolute Errors for $\cos(x)$

	Rectangle	Trapezoidal	Simpson
Single Precision	6.7505e-08	1.0018e-08	1.1735e-08
Double Precision	3.739e-16	1.3951e-18	1.8335e-16

Table 3: Absolute Errors for $\exp(x)$

	Rectangle	Trapezoidal	Simpson
Single Precision	6.5976e-06	4.8465e-06	8.7558e-07
Double Precision	2.2762e-06	4.5525e-06	7.5318e-13

Table 4: Absolute Errors for $\log(x+1)$

	Rectangle	Trapezoidal	Simpson
Single Precision	1.4705e-07	6.2389e-07	3.8547e-07
Double Precision	7.7985e-08	1.5597e-07	6.7502e-14

Based on the analysis of the previous tables, the results show that Simpson's method consistently outperforms other numerical integration techniques, particularly when using double precision, which substantially reduces errors compared to single precision and frequently approaches machine epsilon (~10^-16). Single precision is generally sufficient for most practical applications; however, certain functions, such as the exponential function e^x, display relatively larger errors due to their rapid growth. The Rectangle method proves adequate but is less accurate than Simpson's method, especially for functions with significant curvature. The Trapezoidal method performs well for functions that are approximately linear over small subintervals, yet it does not reach the same level of accuracy as Simpson's method in scenarios demanding high precision.

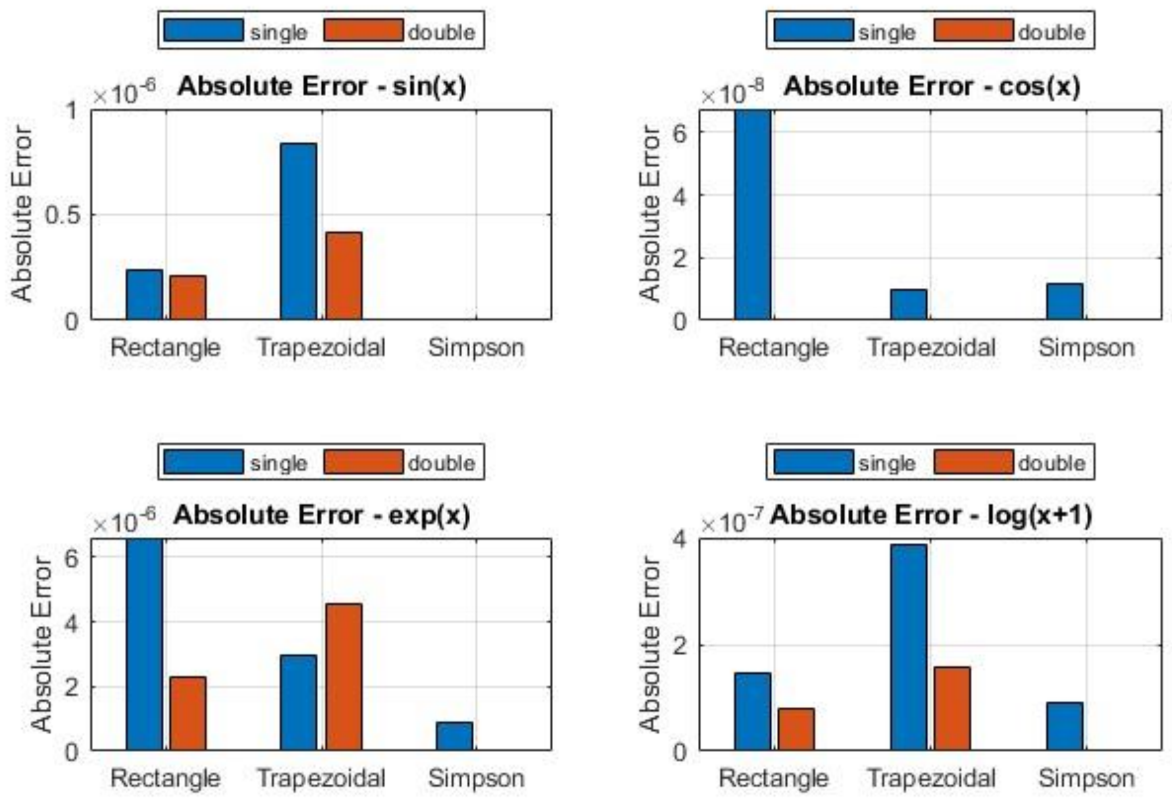

Figure 1: Absolute Error Bar Chart

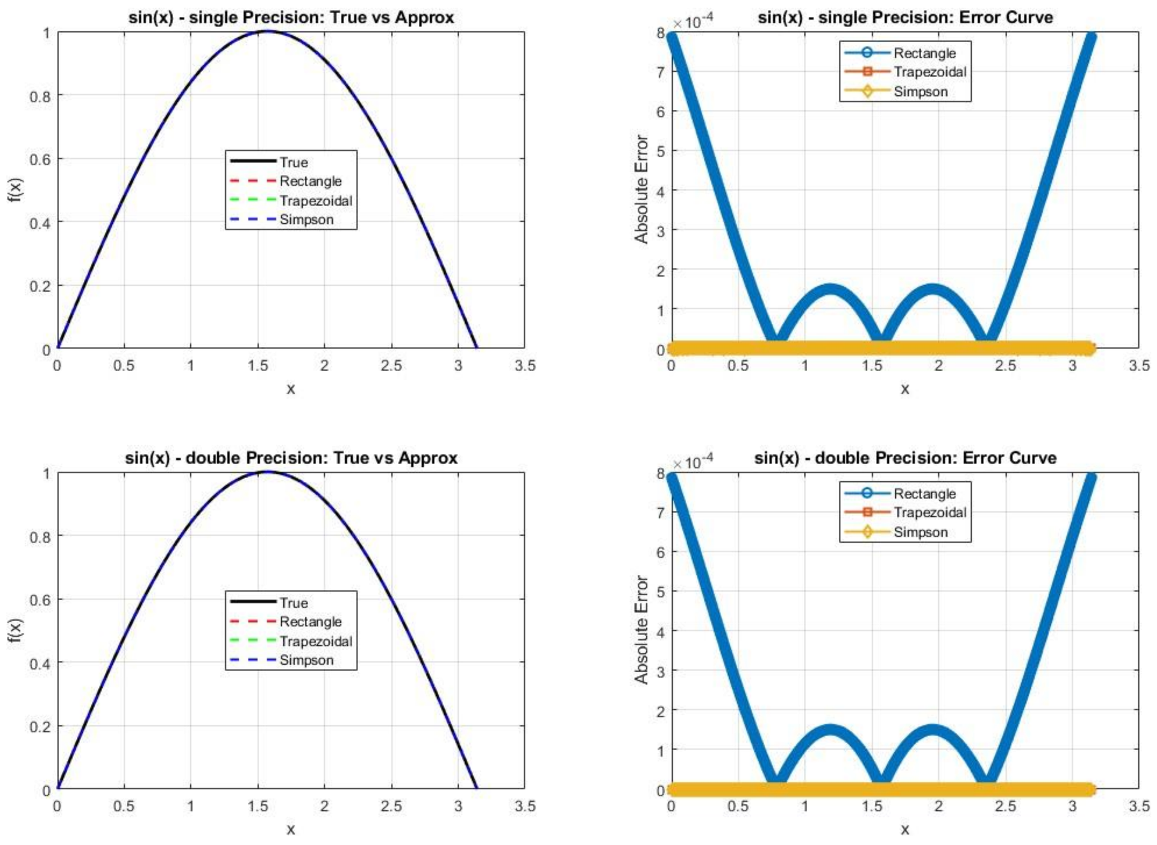

Figure 2: Comparison of Numerical Integration Methods for sin(x): Single vs Double Precision with Absolute Error Curves

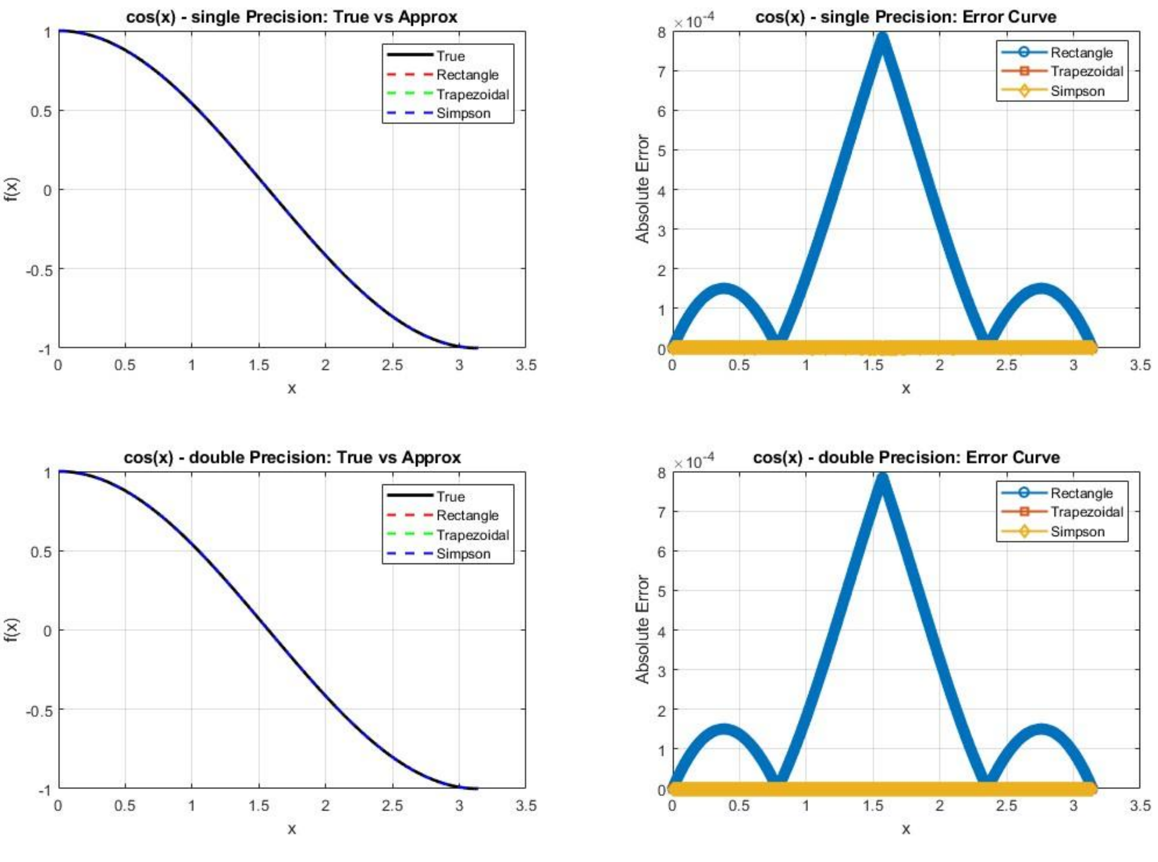

Figure 3: Comparison of Numerical Integration Methods for cos(x): Single vs Double Precision with Absolute Error Curves

Received: 27-10-2025 - **Accepted**: 21-12-2025 - **Published**: 30-12-2025

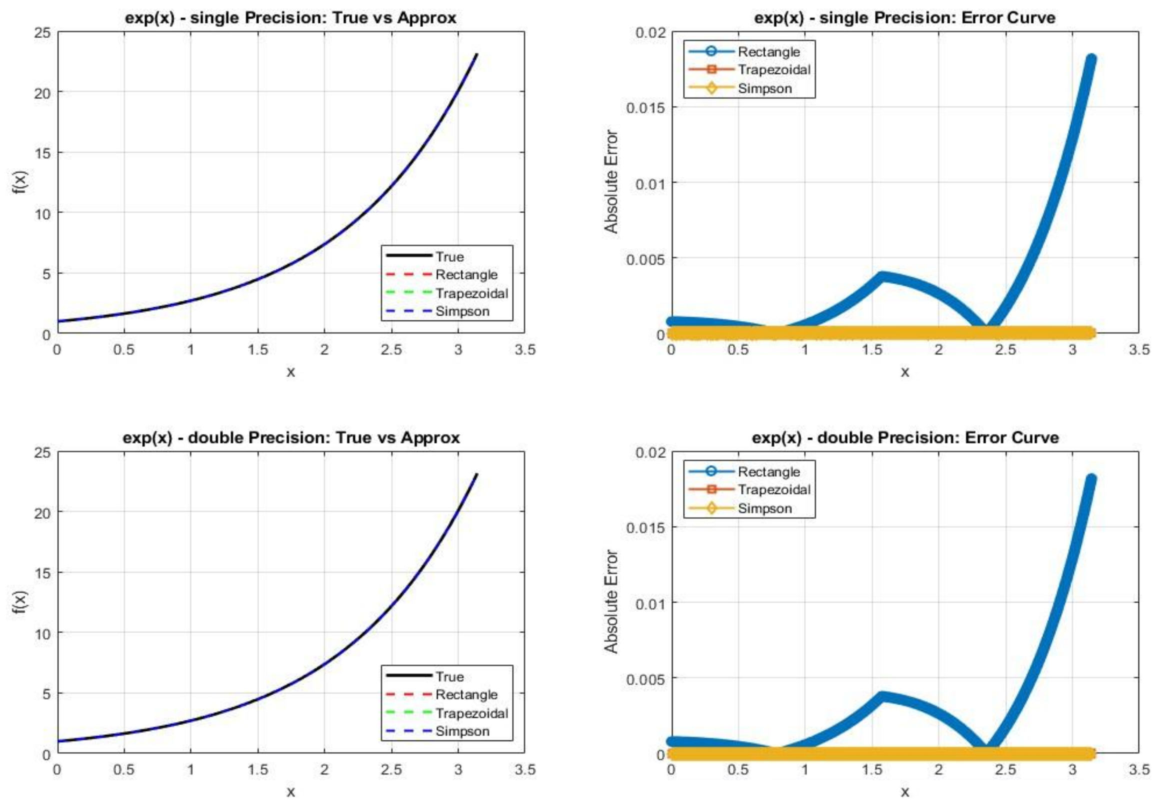

Figure 4: Comparison of Numerical Integration Methods for e^x: Single vs Double Precision with Absolute Error Curves

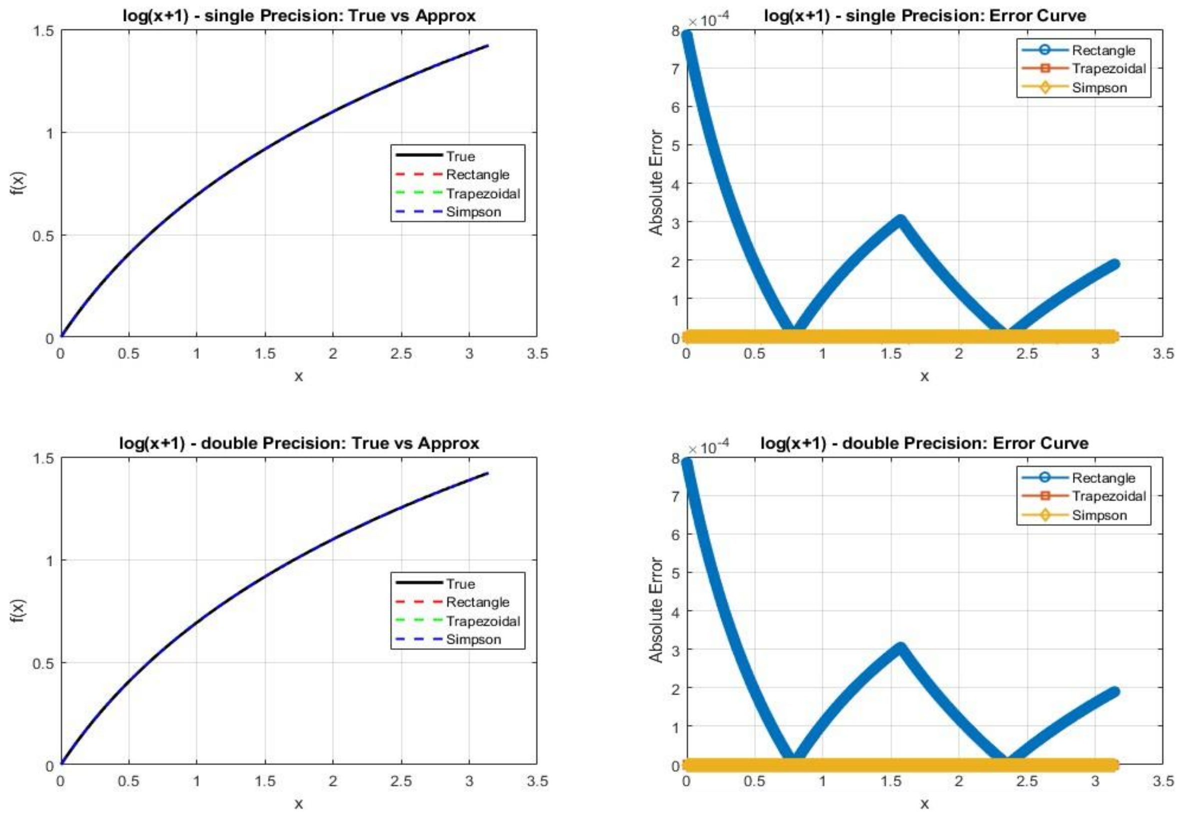

Figure 5: Comparison of Numerical Integration Methods for lox(x): Single vs Double Precision with Absolute Error Curves

Discussion

The results corroborate that the observed numerical integration error arises from the combined action of truncation (discretization) error—governed primarily by the formal order of the quadrature rule—and floating-point round-off error induced by finite-precision arithmetic. In the pre-asymptotic range of grid refinement, the discretization component is expected to dominate; accordingly, higher-order schemes should, in principle, exhibit more rapid error reduction for sufficiently smooth integrands. This trend is clearly reflected in the double-precision outcomes reported in Tables 1–4 and Figures 2–5, where Simpson's rule attains markedly smaller absolute errors for several test functions (notably sin(x), $\boldsymbol{e^x}$, and $log(x)$,), reaching near-machine-level magnitudes compared with the lower-order rectangle and trapezoidal rules. However, as n increases, the truncation term contracts while the relative contribution of rounding grows, such that further refinement may yield diminishing returns, error stagnation, or even irregular behavior—an effect that is particularly evident in single precision, where accumulated summation error can become comparable to (or larger than) the remaining discretization error [10,12]. This is consistent with the structure of composite quadrature formulas, which require repeated addition of many function evaluations; consequently, the rounding incurred at each floating-point addition may accumulate and, depending on the integrand and the step size, can obscure the expected asymptotic ordering of methods [12,13].

In this setting, an apparent reversal of the nominal accuracy hierarchy can occur at specific n values and for specific benchmarks—e.g., instances where a lower-order method matches or outperforms a higher-order rule in single precision—underscoring the need to assess performance across multiple grid sizes and multiple integrands before drawing general conclusions [10,13]. From a practical standpoint, these findings support the standard recommendation that, when high accuracy is required, the most robust strategy is to pair a stable higher-order quadrature method with double precision, while also considering numerically improved accumulation techniques (e.g., compensated summation) when very fine discretizations are used [11,14].

Conclusion

In conclusion, the reported experiments confirm that the attainable accuracy of composite numerical integration is governed not only by the formal order of the quadrature rule (truncation error) but also by the arithmetic precision used to accumulate the discrete sums (round-off error). Across the four benchmark integrands considered—sin(x), cos(x), $e^{\,x}$, and $log(x)$, —double-precision arithmetic consistently produced substantially smaller absolute errors than single precision, in several cases reducing the error to near machine-level magnitudes. Within this setting, Simpson's rule achieved the highest accuracy overall, particularly in double precision, where it yielded extremely small absolute errors for multiple test functions relative to the composite rectangle and trapezoidal rules. In contrast, the rectangle and trapezoidal rules generally exhibited larger errors, with the separation between methods being most pronounced in single precision. Notably, method ranking was not uniform across all cases and precision settings: for sin(x) and $log(x)$, in single precision, the rectangle rule produced smaller absolute errors than the trapezoidal rule, whereas for cos(x) and $e^{\,x}$ the trapezoidal rule outperformed the rectangle rule. This variability is consistent with the fact that, under finite precision, repeated summation can alter the practical error profile and may partially mask the theoretical convergence hierarchy for integrands. Accordingly, reliable computational practice requires selecting the quadrature rule and arithmetic precision as a coupled design choice. For applications demanding high accuracy, double-precision arithmetic is strongly recommended, and higher-order quadrature (e.g., Simpson's rule) should be preferred, provided that the implementation remains numerically stable and performance is verified empirically across representative integrands and discretization's.

Conflict of interest. Nil

References

4. Haidar A, Bayraktar H, Tomov S, Dongarra J, Higham NJ. Mixed-precision iterative refinement using tensor cores on GPUs to accelerate solution of linear systems. Proc Math Phys Eng Sci. 2020 Nov;476(2243):20200110. doi:10.1098/rspa.2020.0110.
5. Panchekha P, Tatlock Z, Sanchez-Stern A, Wilcox JR. Automatically improving accuracy for floating point expressions. SIGPLAN Not. 2015 Jun;50(6):1-11. doi:10.1145/2813885.2737959.
6. Shah K, Sarwar M, Naz H, Abdeljawad T. On spectral numerical method for variable-order partial differential equations. AIMS Math. 2022;7(6):10422-38. doi:10.3934/math.2022581.
7. Chong L, Iyemperumal S, Thorson JT, Maunder MN, Brodziak J, Dietze MC. Performance evaluation of data-limited, length-based stock assessment methods. ICES J Mar Sci. 2020 Jan-Feb;77(1):97-108. doi:10.1093/icesjms/fsz212.
8. Masud AB, Shimi FN, Gope RC. Numerical integration techniques: a comprehensive review. Int J Innov Sci Res Technol. 2024;9(9). doi:10.38124/ijisrt/IJISRT24SEP1327.
9. Budak H, Hezenci F, Kara H, Sarikaya MZ. Bounds for the error in approximating a fractional integral by Simpson's rule. Mathematics. 2023;11(10):2282. doi:10.3390/math11102282.

10. Khanjar H. A comparison of Simpson's rule generalization through Lagrange and Hermite interpolating polynomials. Int J Math Sci Comput. 2024;10(3):37-50. doi:10.5815/ijmsc.2024.03.04.
11. Overton ML. Numerical computing with IEEE floating point arithmetic. 2nd ed. Philadelphia (PA): SIAM; 2025.
12. Joldes M, Tucker W, Popescu V, Muller JM. CAMPARY: Cuda multiple precision arithmetic library and applications. In: 5th International Congress on Mathematical Software (ICMS 2016). 2016. p.232-40. doi:10.1007/978-3-319-42432-3_29.
13. Alspihe H. An analytical comparison of definite numerical integration methods: a study on efficiency and accuracy. Libyan J Contemp Acad Stud. 2025;3(2):114-22. doi:10.65417/ljcas.v3i2.170.
14. Matos JAO, Vasconcelos PB. Effectiveness of floating point precision on the numerical approximation by spectral methods. Math Comput Appl. 2021;26(2):42. doi:10.3390/mca26020042.
15. Higham NJ. The accuracy of floating point summation. SIAM J Sci Comput. 1993 Jul;14(4):783-99. doi:10.1137/0914050.
16. Goldberg D. What every computer scientist should know about floating-point arithmetic. ACM Comput Surv. 1991 Mar;23(1):5-48. doi:10.1145/103162.103163.
17. Ogita T, Rump SM, Oishi S. Accurate sum and dot product. SIAM J Sci Comput. 2005;26(6):1955-88. doi:10.1137/030601818.

Received*: 27-10-2025 -* ***Accepted****: 21-12-2025 -* ***Published****: 30-12-2025*