Security
of Infocommunication Systems
and Internet of Things

2025 Vol 3, No 2

https://doi.org/10.31861/sisiot2025.2.02003

Received 18 June 2025; revised 16 November 2025; accepted 29 December 2025; published 30 December 2025

An Adaptive Benchmark Testing Method for Evaluating Arithmetic Precision

Denys Deineko*

Department of the Computer Systems Software, Yuriy Fedkovych Chernivtsi National University, Chernivtsi, Ukraine

*Corresponding author (E-mail: deineko.denys@chnu.edu.ua)

ABSTRACT Floating-point arithmetic is inherently prone to precision errors, which can accumulate over time and significantly influence the outcomes of numerical computations. This work presents a method designed to systematically assess and compare the accuracy of various arithmetic implementations by adaptively refining test inputs in response to observed computational inaccuracies. In contrast to conventional approaches that use either fixed sets of numerical values or random sampling techniques, the method introduced here continuously updates the test set. It does so by identifying areas in the numerical domain where computational errors tend to be the most significant. The refinement process is iterative and guided by statistical analysis of previous results, ensuring that regions with elevated error levels receive more focused attention in subsequent testing phases. At the heart of the method is an adaptive process for determining which numerical values require further examination. This is achieved by analyzing the distribution of previously recorded errors and updating a decision criterion based on those findings. Specifically, thresholds for acceptable accuracy are recalculated using statistical measures such as quantiles, which reflect the severity and frequency of encountered errors. This ensures that the refinement of test inputs is driven by actual data, rather than relying on predetermined heuristics. The method begins with the generation of a diverse collection of numerical inputs that spans a broad spectrum of floating-point values, including those known to cause instability in calculations – such as extremely small or large values and those located at the boundaries of numerical precision. These inputs are then used to perform arithmetic operations including addition, subtraction, multiplication, and division. Two different arithmetic implementations are evaluated: the standard arithmetic used in a widely adopted programming language and an alternative, custom-developed arithmetic designed to enhance numerical accuracy. For each operation, the resulting values produced by the two arithmetic systems are compared. Measures of accuracy are derived by calculating the differences between the outputs using both absolute and relative error estimations. These differences are then statistically analyzed to detect patterns in the occurrence and magnitude of errors. Based on this analysis, if the error associated with a particular input is determined to be higher than expected, additional test values are generated in the vicinity of that input. This is accomplished through carefully controlled variations, allowing the method to explore neighboring regions where similar errors might occur. In this way, the test suite evolves over time, becoming increasingly focused on those numerical situations that are most likely to expose weaknesses in arithmetic implementations. By uncovering patterns in how errors emerge and accumulate, the method provides a structured and repeatable process for evaluating the reliability of floating-point arithmetic under varying conditions. Its targeted nature makes it especially useful for scientific and engineering applications, where computational precision is critical. In summary, this approach improves upon traditional benchmarking techniques by introducing an adaptive, data-driven strategy that emphasizes the most challenging areas of numerical computation. As such, it offers a powerful tool for the verification and validation of arithmetic systems, supporting both development and quality assurance in software that relies heavily on floating-point calculations.

KEYWORDS arithmetic verification, floating-point arithmetic, precision evaluation, numerical accuracy, adaptive testing.

I. INTRODUCTION

Evaluating arithmetic accuracy is a fundamental aspect of ensuring the reliability of numerical computations in scientific and engineering applications. In modern software systems, floating-point operations are ubiquitous, yet prone to subtle and often undetected errors that can propagate and compromise computational results. This necessitates robust and adaptive methods for identifying numerical instability in precision-critical applications.

The aim of this work is to develop and formalize an adaptive benchmark testing method for evaluating the quality of arithmetic operations. The main objectives include: the design of a systematic approach for adaptively refining test data based on observed computational errors; to formally describe the proposed adaptive testing method for evaluating the accuracy of numerical arithmetic; and to assess the algorithmic complexity of the approach and its computational efficiency.

The core idea behind the proposed method is that analysis should not rely solely on static datasets or random sampling but should adapt to the observed error distribution in real time. If a specific range of inputs leads to unexpectedly large errors, the system automatically generates new test cases by perturbing values within that region. This allows for the localization of error-prone zones and the delineation of boundaries where arithmetic becomes unreliable.

Recent studies and publications in numerical analysis [1-2], property-based testing [3], and arithmetic verification highlight several isolated techniques – such as Monte Carlo simulations [4-5], adaptive quadrature in integration [6], or error-controlled Runge-Kutta methods in

differential equations [7]. While these approaches contribute valuable insights, they often lack integration into a unified framework for dynamic test generation based specifically on error metrics. Furthermore, software engineering techniques like fuzzing [8] or phase-based testing [9] rarely consider numerical stability as a primary target.

This research bridges the gap between numerical precision analysis and adaptive test generation. By unifying statistical error analysis, iterative test refinement, and adaptive threshold control with established error criteria from numerical ODE solvers [10-12], the proposed method introduces a novel, comprehensive framework for verifying arithmetic robustness. Its implementation creates a foundation for future testing systems that are both error-aware and resource-efficient.

II. METHOD DESCRIPTION

The challenge of varying test set effectiveness creates a need for a method that not only compares numerical outputs but also adapts dynamically to problematic areas within the input space.

The core idea behind the Adaptive Benchmark Testing method is to begin with conventional testing on selected datasets and then progressively expand the test set in regions where the new arithmetic exhibits instability. The method requires two arithmetic systems for comparison. These can be any numeric systems; however, in the context of this study, the analysis focuses on comparing a baseline arithmetic implementation with an alternative arithmetic implementation [13] designed for enhanced numerical precision. This approach enables a controlled computational evaluation of their accuracy, stability, and efficiency.

The Adaptive Benchmark Testing algorithm is specifically designed to evaluate arithmetic accuracy by adaptively expanding the test set in zones of elevated error. Initially, a base test set denoted as S_0 is generated. This set contains representative input values covering a wide numeric range. Each test case from this set is computed using both the alternative and baseline arithmetic implementations, after which absolute and relative errors are calculated for each result using Eqs. (1) and (2), where r_p denotes the result from the alternative implementation and r_s from the baseline implementation:

$$AbsErr(x) = |r_p - r_s|. \quad (1)$$

A fundamental challenge in precision testing is assessing errors across functions with vastly different magnitudes. Pure absolute error fails to scale with function magnitude, while pure relative error becomes undefined or misleadingly large near zeros of the function.

Following established practice in adaptive numerical solvers [10-12], this work employs a mixed absolute-relative error criterion as the evaluation function $E(x)$:

$$E(x) = \frac{AbsErr(x)}{(\tau_{abs} + \tau_{rel} \cdot |r_s(x)|)}, \quad (2)$$

where $\tau_{abs} > 0$ is the absolute tolerance parameter and $\tau_{rel} > 0$ is the relative tolerance parameter, both specified by the user to define precision requirements.

The necessity of the mixed absolute-relative tolerance criterion is illustrated in Figs. 1 and 2, which demonstrate why neither pure absolute error nor pure relative error alone provides adequate assessment across different value ranges.

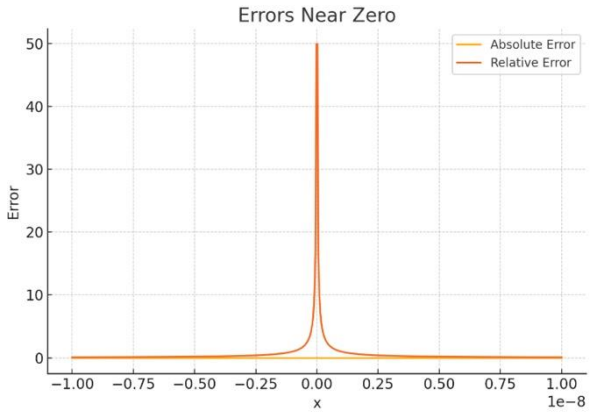

FIG. 1. Comparison of absolute and relative error metrics near zero, demonstrating instability of relative error for small function values.

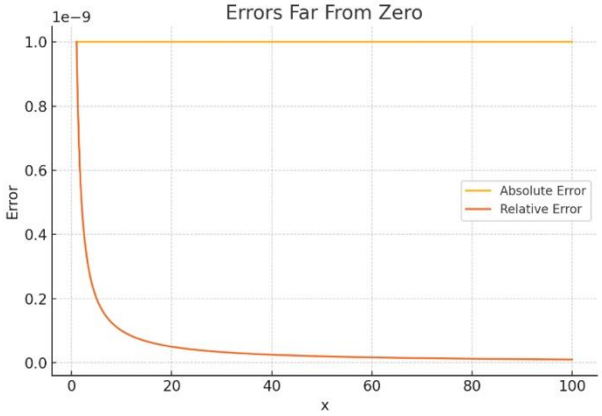

FIG. 2. Comparison of absolute and relative error metrics far from zero, showing failure of absolute error to scale with magnitude.

Fig. 1 shows error behavior near zero ($x \in [-10^{-8}, 10^{-8}]$). Relative error exhibits a sharp spike reaching approximately 50 at $x \approx 0$, increasing by orders of magnitude due to near-zero denominators. Absolute error remains stable at approximately 0.5. This demonstrates that near zero, absolute error provides stable precision assessment while relative error becomes uninformative.

Fig. 2 shows the reverse behavior far from zero. Absolute error becomes nearly constant, failing to reflect value magnitude. An absolute error of 0.5 is negligible for $|r_s| = 10^6$ (0.0001%) but significant for $|r_s| = 1$ (50%), while relative error correctly scales with magnitude.

These observations motivate the ATOL+RTOL formulation in Eq. (2), which uses absolute error normalized by a tolerance that automatically transitions between absolute-dominated behavior near zero and relative-dominated behavior for large magnitudes, eliminating the need for manual domain-specific scaling.

The denominator transitions smoothly between absolute-dominated behavior for $|r_s(x)| << \tau_{abs}/\tau_{rel}$ and relative-dominated behavior for $|r_s(x)| >> \tau_{abs}/\tau_{rel}$, with crossover at $M_c = \tau_{abs}/\tau_{rel}$. For defaults $\tau_{abs} = 10^{-10}$ and $\tau_{rel} = 10^{-6}$, this occurs at $|r_s(x)| \approx 10^{-4}$. The

dimensionless $E(x)$ enables meaningful comparison across functions; $E(x) > 1$ indicates exceeding tolerance.

This criterion reflects floating-point error structure (machine-epsilon rounding and problem-dependent truncation) and is standard in numerical ODE solvers. The recommended defaults provide reasonable precision for most applications while remaining adaptable.

The distribution of errors is then analyzed using the evaluation function $E(x)$. If the average error significantly shifts within specific subranges, a new test set S_{i+1} is generated that increases the sampling density in these critical regions.

Critical regions are identified as subintervals where inequality in Eq. (3) is satisfied, with θ being a threshold derived from the statistical data of previous iterations. A statistical analysis of the errors is used to determine this threshold dynamically in Eq. (4), where μ_i is the mean error at iteration i in Eq. (5), σ_i is the standard deviation of the error in Eq. (6), and λ is a coefficient controlling the strictness of the filter – e.g., $\lambda = 1$ or $\lambda = 2$, which correspond to 68% or 95% of values under the normal distribution.

$$E(x) > \theta, \tag{3}$$

$$\theta_{i+1} = \mu_i + \lambda\sigma_i, \tag{4}$$

$$\mu_i = \frac{1}{|S_i|}\sum_{x\in S_i} E(x), \tag{5}$$

$$\sigma_i = \sqrt{\frac{1}{|S_i|}\sum_{x\in S_i}(E(x) - \mu_i)^2}. \tag{6}$$

The threshold calculation in Eq. (4) relies on mean and standard deviation, which are optimal estimators under the assumption of normal distribution. However, since $E(x)$ is inherently non-negative, its distribution may deviate from normality. To ensure statistical validity, a normality verification step is implemented before applying Eq. (4). For each iteration i, the Shapiro-Wilk test [14] is applied to test the null hypothesis:

$$H_0: \text{the error values } \{E(x)|x \in S_I\} \text{ follow a normal distribution.} \tag{7}$$

The $p-value$ from the Shapiro-Wilk test is computed. The decision rule is:

$$p - value > \alpha_{norm}, \tag{8}$$

$$p - value \leq \alpha_{norm}. \tag{9}$$

If Eq. (8) is true, use parametric threshold (Eq. 4), else if Eq. (9) is true, use non-parametric threshold Eq. (10) where $\alpha_{norm} = 0.05$ is the significance level for the normality test. When normality is rejected, a distribution-free quantile-based approach is employed:

$$\theta_{i+1} = Q_{0.90}(E(x): x \in S_i), \tag{10}$$

where $Q_{0.90}$ denotes the 90th percentile of the error distribution. This percentile corresponds roughly to $\mu + 1.28\sigma$ under normality but remains valid regardless of the underlying distribution. The iterative refinement process continues until statistical analysis indicates that further iterations yield no significant improvement in error characterization. Rather than using a simple difference threshold, a rigorous statistical test is employed to compare the error distributions between consecutive iterations.

Specifically, Welch's two-sample t-test [15] is applied to compare the mean errors between S_i and S_{i+1}. The null hypothesis is Eq. (11) (no significant change in mean error) with alternative hypothesis $H_1: \mu_i \neq \mu_{i+1}$ (significant change exists).

$$H_0: \mu_i = \mu_{i+1}. \tag{11}$$

Welch's t-test is preferred over the standard Student's t-test because it does not assume equal variances between the two samples, making it more robust to differences in sample sizes and error distributions between iterations. To ensure that convergence reflects both statistical and practical significance, a dual criterion is employed. Define Cohen's effect size [16] using the weighted pooled standard deviation as:

$$d = \frac{|\mu_{i+1} - \mu_i|}{\sqrt{\frac{(n_i - 1)\sigma_i^2 + (n_{i+1} - 1)\sigma_{i+1}^2}{n_i + n_{i+1} - 2}}}, \tag{12}$$

where n_i and n_{i+1} are the sample sizes at iterations i and $i + 1$ respectively. The weighted pooled standard deviation accounts for potentially large differences in sample sizes between consecutive iterations. In the adaptive refinement process, n_{i+1} can be significantly larger than n_i (e.g., growing from 1,000 to 10,000+ points), making the weighted formula essential for accurate effect size estimation.

The algorithm terminates when both conditions are satisfied:

$$p - value > \alpha_{test} \text{ and } |d| < d_{min}, \tag{13}$$

where $\alpha_{test} = 0.05$ is the significance level for the t-test, and $d_{min} = 0.2$ represents the threshold for practical significance. The dual criterion ensures that the algorithm stops when changes are neither statistically significant nor practically meaningful, preventing both premature termination and excessive iteration on trivial improvements when sample sizes are large.

The complete procedure is formalized in Algorithm 1 below:

Input: two arithmetic implementations A_1 (baseline) and A_2 (alternative); initial test set S_0 of size n_0; threshold coefficient λ; absolute and relative tolerances τ_{abs} and τ_{rel}; significance levels α_{norm} and α_{test}; effect size threshold d_{min}; maximum iterations k_{max}.

Output: final refined test set S_k and error statistics $\{\mu_j, \sigma_j, \theta_j, n_j\}$ for each iteration $j = 0, \ldots, k$.

Procedure:

Step 1. Set iteration counter $i \leftarrow 0$. For each $x \in S_0$, evaluate both implementations to obtain $r_s = A_1(x)$ and $r_p = A_2(x)$, then calculate the absolute error using Eq (1).

Step 2. For each $x \in S_i$, evaluate both implementations to obtain $r_s = A_1(x)$ and $r_p = A_2(x)$, then calculate the evaluation function using Eq. (2).

Step 3. Test the normality of the error distribution $\{E(x): x \in S_i\}$ using the Shapiro-Wilk test as specified in Eq. (7).

Step 4. If the normality test is passed ($p - value > \alpha_{norm}$), compute the threshold θ_{i+1} using the parametric approach in Eq. (4). Otherwise, compute θ_{i+1} using the quantile-based approach in Eq. (10).

Step 5. Identify the critical set $C_i = \{x \in S_i : E(x) > \theta_{i+1}\}$ containing all points where the evaluation function exceeds the threshold.

Step 6. Generate the refined test set S_{i+1} by starting with S_i and adding perturbations around each point $x \in C_i$.

Step 7. If $i > 0$, perform convergence testing by applying Welch's t-test to compare the error distributions of S_i and S_{i+1} as specified in Eq. (11), computing Cohen's effect size d using Eq. (12) with sample sizes $n_i = |S_i|$ and $n_{i+1} = |S_{i+1}|$, and checking whether the dual criterion in Eq. (13) is satisfied. If convergence is achieved, terminate and return the current test set and statistics.

Step 8. Increment the iteration counter: $i \leftarrow i + 1$. If $i < k_{max}$, return to Step 2. Otherwise, terminate and return the final test set S_i and error statistics.

III. SENSITIVITY ANALYSIS OF PARAMETERS

An important component of the proposed algorithm is its dependence on parameters that influence the accuracy, stability, and efficiency of the testing process. In particular, key roles are played by: the parameter λ, used in constructing the error threshold in Eq. (4); the error metric, which can be absolute, relative, or combined; the initial size of the test set.

The value of λ regulates the system's sensitivity to detecting critical points. A low λ value leads to a smaller threshold θ, resulting in a greater number of values being classified as "suspicious" and thus intensively refining the test set. This can provide high sensitivity at the cost of increased iterations and computational overhead. Conversely, a high λ reduces the number of adaptive steps but may miss local anomalies. Numerical experiments showed that λ values within the range $[1.5,\ 2.5]$ provide a good balance between coverage and computational efficiency.

The following plot Fig. 3 visually demonstrates the results for different values of λ. It illustrates that: $\lambda = 0.5$: the highest error fraction, which decreases very slowly; $\lambda = 1.5$ and $\lambda = 2.5$: a stable reduction in the error fraction with each iteration – representing a balance between accuracy and efficiency; $\lambda = 3.5$: low sensitivity – the system misses significant errors.

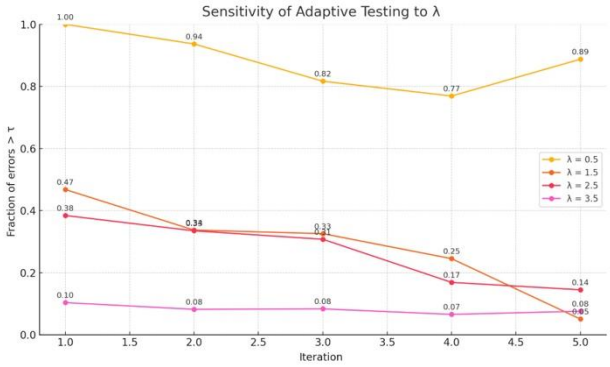

FIG. 3. Effect of λ on the error fraction.

The selection of tolerance parameters τ_{abs} and τ_{rel} defines the precision requirements for the testing process. As discussed in Section II and illustrated in Figs. 1 and 2, the mixed absolute-relative criterion provides appropriate error assessment across all magnitude ranges. For general-purpose testing, default values of $\tau_{abs} = 10^{-10}$ and $\tau_{rel} = 10^{-6}$ are recommended.

The crossover magnitude $M_c = 10^{-4}$ determines where the evaluation function transitions from absolute-dominated to relative-dominated behavior. Table 1 quantifies this divergence near zero, showing how relative error becomes unstable while absolute error remains stable. This motivates the mixed tolerance criterion in Eq. (2), which automatically adapts to magnitude-dependent error behavior.

TABLE 1. Quantitative comparison of error metrics near zero.

x	AbsErr	RelErr
1×10^{-7}	0.51	5.1
1×10^{-8}	0.50	50.0
1×10^{-9}	0.50	500.1
1×10^{-10}	0.50	5000

The size of the initial test set determines the quality of the initial statistics from which the threshold is derived. An insufficient size may lead to unstable estimates of θ and consequently to incorrect identification of critical regions. Empirically, the initial test set size should be no less than 500 – 1000 points for univariate functions. For higher-dimensional cases, the required size increases exponentially.

The following plot Fig. 4 illustrates a critical region – the interval $[0.4, 0.6]$. This is an example of a scenario in which the highest numerical errors or computational instabilities are assumed to occur within this range. The purpose of this modeling is to demonstrate that, with small test set sizes, the algorithm either fails to detect this critical region or misidentifies it. Only when the test set becomes sufficiently large do the estimated boundaries align with this "true" model.

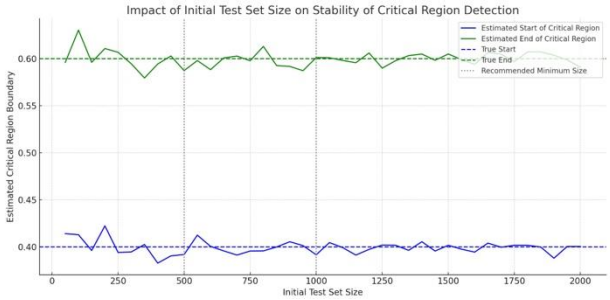

FIG. 4. Impact of initial test set size on the stability of estimated critical regions.

In the proof-of-concept demonstration in Section IV, it was observed that both the exponential and logarithmic function tests consistently employed the quantile-based threshold in Eq. (10) throughout all iterations, as the Shapiro-Wilk test rejected normality in every case. In contrast, the sinc function test maintained normal error distributions, allowing use of the parametric threshold Eq. (4) at all iterations.

IV. PROOF-OF-CONCEPT DEMONSTRATION

To demonstrate the proposed method's capability to identify precision-sensitive regions when implementation differences exist, a proof-of-concept study was conducted using controlled precision variations. Modern floating-point implementations, including CPython's IEEE 754 double-precision arithmetic, are highly optimized and typically exhibit minimal precision errors for standard mathematical functions. Therefore, rather than comparing two production implementations that would show negligible differences, controlled precision degradation was introduced in known error-prone regions to validate the adaptive testing methodology.

A. Experimental design. The demonstration was implemented in Python 3.11 and executed in a Docker container based on python:3.11-slim (Debian Linux, CPython 3.11, ARM64), running on Apple M1. Two implementations were compared. The baseline implementation used standard CPython floating-point operations with simulated precision degradation in specific numerical regions where precision issues are known to occur in practical arithmetic implementations. The alternative implementation employed high-precision arithmetic using the mpmath library configured with 50 decimal places of precision, serving as the high-precision reference for error computation.

The simulated precision degradation was introduced to represent realistic scenarios observed in practical floating-point implementations. For the exponential function $\exp(x)$, reduced precision was introduced near the boundaries of the representable range where the absolute value of x exceeds 9. This region approaches overflow and underflow limits, simulating the accumulation of rounding errors that occurs when operating near the limits of floating-point representation. For the logarithmic function $\log(x)$, reduced precision was applied for arguments approaching zero, specifically when x falls below 10^{-6}. In this region, small relative errors in the input argument translate to large absolute errors in the logarithmic output. For the sinc function $\sin(x)/x$, reduced precision was introduced near zero where the absolute value of x is less than 10^{-3}. This simulates catastrophic cancellation that can occur during the computation when both numerator and denominator approach zero.

Initial test sets were generated using uniform random sampling with seed 42 for reproducibility. For the logarithmic function, sampling was performed uniformly in log-space to ensure adequate coverage across the wide domain range. Perturbations in Step 6 of Algorithm 1 were generated using normally distributed random values with standard deviation proportional to local point density.

B. Test configuration. Tolerance parameters were $\tau_{abs} = 10^{-10}$ and $\tau_{rel} = 10^{-6}$ (standard for double-precision solvers), requiring ≈6 digits relative accuracy with 10^{-10} absolute floor. Crossover magnitude $M_c = 10^{-4}$ defines the transition between absolute and relative-dominated behavior.

The initial sample size was set to $n_0 = 1000$ uniformly distributed test points. The threshold coefficient was $\lambda = 2.0$, corresponding to approximately 95% coverage under normal distribution. The significance levels for both normality testing and convergence testing were $\alpha_{norm} = \alpha_{test} = 0.05$. The effect size threshold for practical significance was $d_{min} = 0.2$, and the maximum number of iterations was $k_{max} = 10$.

The functions evaluated were the exponential function $f(x) = \exp(x)$ for x in the interval $[-10, 10]$ as defined in Eq. (14), the logarithmic function $g(x) = \log(x)$ for x in the interval $[10^{-9}, 10^{9}]$ as defined in Eq. (15), and the trigonometric sinc function $h(x) = \sin(x)/x$ for x in the interval $[-\pi, \pi]$ as defined in Eq. (16).

$$f(x) = \exp(x) for\ x \in [-10, 10], \tag{14}$$

$$g(x) = \log(x)\ for\ x \in [10^{-9}, 10^{9}], \tag{15}$$

$$h(x) = \frac{\sin(x)}{x}\ for\ x \in [-\pi, \pi]. \tag{16}$$

C. Exponential function results. For the exponential function, the adaptive method required 10 iterations to reach the maximum iteration limit, expanding the test set from the initial 1000 values to a final size of 56,500 values (56.5× expansion factor). The test set size evolved from 1,000 (initial) through 1,500, 2,250, 3,375, 4,960, 7,440, 11,160, 16,740, 25,110, and 37,665 points across iterations 1 – 10. The algorithm terminated at the maximum iteration limit with a final expanded set of 56,500 points. The growth factor varied between 1.26× and 1.50× per iteration, reflecting the algorithm's persistent identification of critical points requiring refinement across a substantial portion of the domain.

The algorithm successfully identified the critical region near the upper boundary of the domain at $x \in [9.00, 10.0]$, where $\exp(x)$ approaches the overflow limit of double precision, which is approximately 10^{308}. The Shapiro-Wilk test consistently rejected normality throughout all iterations, causing the algorithm to employ the quantile-based threshold defined in Eq. (10) rather than the parametric approach given in Eq. (4). This behavior demonstrates the robustness of the dual-threshold mechanism when error distributions deviate from normality, as discussed in Section II.

The normalized error values $E(x)$ exhibited the following progression. The mean $E(x)$ increased from 5.07×10^{-9} at iteration 1 to 7.78×10^{-8} at the final iteration, representing a 15.3× increase as the algorithm concentrated sampling in higher-error regions. The maximum $E(x)$ increased from 1.93×10^{-7} at iteration 1 to 4.33×10^{-7} at iteration 8, then stabilized. The threshold θ increased from 1.70×10^{-10} at iteration 1 to 1.60×10^{-7} at iteration 10. These values, while appearing small in absolute terms, represent normalized errors where $E(x) > 1$ would indicate exceeding the tolerance envelope. The maximum observed $E(x) = 4.33 \times 10^{-7}$ means the worst-case error is approximately 0.04% of the tolerance threshold (since $E(x) << 1$), well within acceptable bounds.

The convergence testing revealed persistent statistical significance throughout all iterations. At iteration 2, comparing 1,000 vs 1,500 samples yielded p-value < 0.0001 and Cohen's d = 0.470, both exceeding the termination thresholds. At iteration 5, comparing 3,375

vs 4,960 samples produced p-value < 0.0001 and d = 0.126. By iteration 10, comparing 37,665 vs 56,500 samples, Cohen's d had decreased to 0.018, well below the practical significance threshold, but the p-value remained at 0.0251, just below the 0.05 threshold. This demonstrates the importance of the dual convergence criterion in Eq. (13): while the effect size became small (d < 0.2), statistical significance persisted at the boundary, justifying continued refinement until the iteration limit.

The final critical region comprised 10.1% of the test set, representing 5,702 test points concentrated in the interval [9.00, 10.0]. This interval represents 5.0% of the domain [-10, 10]. Uniform sampling of 56,500 points would allocate approximately 2,825 points to this region (5.0% × 56,500), whereas the adaptive method allocated 5,702 points, representing a 2.0× concentration factor. The total computational cost was 167,700 function evaluations over 13.63 seconds.

D. Logarithmic function results. The logarithmic function exhibited rapid convergence after 4 iterations, with the test set expanding from 1,000 to 3,045 values (3.0× expansion factor). The test set evolution was: iteration 1 (1,000 points), iteration 2 (1,500 points), iteration 3 (2,250 points), iteration 4 (3,045 points). The growth pattern varied from 1.26× to 1.50× per iteration, reflecting the algorithm's ability to adaptively adjust expansion based on the spatial distribution of critical points.

The critical region was identified near the lower boundary of the domain at $x \in [1.00 \times 10^{-9}, 9.79 \times 10^{-7}]$, where the logarithmic output becomes increasingly negative and small perturbations in the argument translate to significant changes in the result. Similar to the exponential case, the Shapiro-Wilk test rejected normality at all iterations, invoking the quantile-based threshold mechanism specified in Eq. (10).

The normalized error values showed the following behavior. The mean $E(x)$ decreased from 1.41×10^{-7} at iteration 1 to 7.42×10^{-8} at iteration 4, indicating successful refinement focusing on the most problematic region. The maximum $E(x)$ remained constant at 2.99×10^{-6} throughout all iterations, indicating early identification of the worst-case point. The threshold θ decreased from 5.71×10^{-7} to 8.32×10^{-8}, reflecting the algorithm's adaptation to a more concentrated error distribution.

Convergence was achieved at iteration 4 when comparing samples of size 2,250 vs 3,045. The convergence test yielded p-value = 0.106 > 0.05 and Cohen's d = 0.046 < 0.2, simultaneously satisfying both criteria in Eq. (13). This dual satisfaction indicates that changes between iterations were neither statistically significant nor practically meaningful. The evolution of Cohen's d throughout the iterations illustrates convergence: iteration 2 (d = 0.092), iteration 3 (d = 0.074), iteration 4 (d = 0.046), showing progressive decrease as the critical region became saturated.

The identified critical region represented 5.2% of the final test set, comprising 159 test points concentrated in the interval $[1.00 \times 10^{-9}, 9.79 \times 10^{-7}]$. In logarithmic space, this region spans approximately 2.99 log units out of 18 total log units (16.6% of the domain). The allocation of 5.2% of test points to a region comprising 16.6% of the logarithmic domain reflects the threshold-based approach: the 90th percentile criterion ensures exactly 10% of points are initially classified as critical, with subsequent refinement concentrating on the most severe errors rather than achieving proportional spatial coverage. The total computational cost was 10,840 function evaluations over 0.37 seconds.

E. Sinc function results. The sinc function converged most rapidly, requiring only 2 iterations with minimal test set expansion from 1,000 to 1,005 values (1.005× expansion factor). The test set sizes were: iteration 1 (1,000 points), iteration 2 (1,005 points). This minimal growth reflects the highly localized nature of the critical region and the rapid satisfaction of convergence criteria.

Unlike the exponential and logarithmic functions, the Shapiro-Wilk test confirmed normality of the error distribution at both iterations, with a normality pass rate of 100%. This allowed the algorithm to use the parametric threshold defined in Eq. (4) throughout testing, illustrating the method's automatic adaptation to different error distribution characteristics. The threshold value remained stable at approximately 9.1×10^{-12} across both iterations.

The normalized error values demonstrated exceptional precision. The mean $E(x)$ was approximately 1.42×10^{-13}, indicating that average errors were well below the tolerance envelope by a factor of 10^{13}. The maximum $E(x)$ was 1.42×10^{-10} indicates errors approximately 7 billion times smaller than the tolerance threshold. These values reflect the high quality of the standard CPython implementation for the sinc function in most of its domain.

Convergence was achieved at iteration 2 when comparing samples of size 1,000 vs 1,005. The convergence test yielded p-value = 0.997 > 0.05 and Cohen's d = 0.0002 < 0.2, decisively satisfying the dual criterion in Eq. (13). The near-unity p-value indicates no statistical difference whatsoever, while the extremely small Cohen's d confirms practical equivalence.

The critical region identified was $x \in [1.19, 1.19]$, comprising only 0.1% of the test set (1 point at convergence). This region represents approximately 0.02% of the domain $[-\pi, \pi]$. The identification of a single critical point demonstrates the method's ability to isolate highly localized precision variations. The total computational cost was 3,010 function evaluations over 0.04 seconds, representing minimal overhead beyond the initial sampling.

F. Comparative analysis of ATOL+RTOL performance. The use of the mixed absolute-relative tolerance criterion (Eq. 2) provides several key advantages demonstrated across the three test cases. The normalized error values $E(x)$ remain in consistent, interpretable ranges across all functions. Maximum $E(x)$ values were: $\exp(x)$: 4.33×10^{-7}; $\log(x)$: 2.99×10^{-6}; $\mathrm{sinc}(x)$: 1.42×10^{-10}. All values are well below 1.0, confirming that errors remain within the specified tolerance envelope. This consistency would not be achievable with pure absolute or pure relative error metrics, which exhibit the problematic behavior illustrated in Figs. 1 and 2.

The crossover magnitude $M_c = 10^{-4}$ appropriately balances absolute and relative error contributions. For $\exp(x)$ near $x = 10$ where $|r_s| \approx 2.2 \times 10^4$, the relative component dominates: $\tau_{rel} \times |r_s| \approx 2.2 \times 10^{-2} >> \tau_{abs} = 10^{-10}$. For $\log(x)$ near $x = 10^{-9}$ where $|r_s| \approx 20.7$, the relative component still dominates: $\tau_{rel} \times 20.7 \approx 2.1 \times 10^{-5} >> \tau_{abs}$. For $\text{sinc}(x)$ where $|r_s| \approx 1$ throughout most of the domain, both components contribute: $\tau_{abs} + \tau_{rel} \times 1 = 1.0001 \times 10^{-6}$, with relative tolerance dominating by a factor of 10^4.

The user-specified tolerance parameters provide clear control over precision requirements. The value $\tau_{rel} = 10^{-6}$ successfully enforced approximately 6 digits of relative accuracy across all functions, as evidenced by the small normalized error values. The value $\tau_{abs} = 10^{-10}$ provided an appropriate absolute floor, particularly important for the sinc function where values near zero could otherwise cause instability in pure relative error metrics.

G. Convergence behavior analysis. The three test cases demonstrate distinct convergence patterns reflecting different characteristics of the error distributions. The exponential function required the maximum allowed iterations (10), with Cohen's d decreasing from 0.470 (iteration 2) to 0.018 (iteration 10). The effect size fell below the practical significance threshold $d < 0.2$ after iteration 7, but the p-value remained below 0.05 throughout all iterations, indicating persistent statistical significance. This behavior suggests a continuous error landscape rather than isolated problematic points, justifying continued refinement despite small effect sizes. The threshold stabilized after iteration 6 at approximately 1.6×10^{-7}, while sampling density continued to increase in the critical region.

The logarithmic function achieved convergence at iteration 4 when both statistical and practical significance criteria were satisfied simultaneously. Cohen's d decreased progressively: 0.092 (iteration 2), 0.074 (iteration 3), 0.046 (iteration 4). The p-value similarly decreased from below 0.05 (iterations 2-3) to 0.106 (iteration 4). This simultaneous satisfaction of both criteria ($p > 0.05$ and $d < 0.2$) provided strong evidence that the critical region had been adequately characterized, preventing both premature termination and excessive iteration.

The sinc function converged most rapidly, achieving convergence at iteration 2 due to the small spatial extent of the critical region. The p-value of 0.997 and Cohen's d of 0.0002 on the first convergence test indicated that the small critical region (0.1% of initial samples) was quickly saturated. The parametric threshold remained stable at approximately 9.1×10^{-12}, reflecting the well-behaved normal error distribution. This demonstrates the method's ability to adapt computational investment to problem complexity: functions with small, well-localized critical regions converge quickly with minimal overhead.

The automatic switching between parametric and non-parametric thresholds based on the Shapiro-Wilk test results demonstrates the robustness of the statistical validation mechanism described in Section II. For exponential and logarithmic functions, normality was consistently rejected across all iterations, triggering the use of the quantile-based threshold in Eq. (10) with a 0% normality pass rate. For the sinc function, normality was confirmed at both iterations with a 100% pass rate, allowing use of the parametric threshold in Eq. (4). This adaptation occurred automatically without manual intervention, confirming the algorithm's capability to handle diverse error characteristics.

H. Computational efficiency. The computational cost of the adaptive method varies significantly with the characteristics of the error landscape. The exponential function required 167,700 total function evaluations over 13.63 seconds, representing substantial computational investment to thoroughly explore the high-error region. The logarithmic function required 10,840 evaluations over 0.37 seconds, demonstrating moderate overhead for a moderately-sized critical region. The sinc function required only 3,010 evaluations over 0.04 seconds, representing minimal overhead beyond the initial sampling. These results demonstrate that the method adapts its computational cost to the complexity of the error distribution, investing more resources only when error patterns justify continued refinement.

I. Validation of method capabilities. The experimental results validate the proposed adaptive testing method using the mixed criterion. The method successfully demonstrated automatic identification of critical regions without prior knowledge of their location. For all three test functions, the algorithm identified error-prone regions based purely on observed error patterns, without requiring manual specification of where problems might occur.

The statistical robustness of the method was confirmed through automatic adaptation to different error characteristics. The dual-threshold mechanism, switching between parametric and quantile-based approaches based on Shapiro-Wilk test results, handled both normal (sinc) and non-normal (exp, log) error distributions effectively. The dual convergence criterion, combining statistical significance testing via Welch's t-test with practical significance assessment via Cohen's effect size, prevented both premature termination and excessive iteration across all test cases.

The normalized error metric $E(x)$ provided consistent, interpretable results across functions with vastly different magnitudes. All maximum $E(x)$ values remained well below 1.0, confirming errors stayed within the specified tolerance envelope. The crossover magnitude appropriately balanced absolute and relative error contributions based on function magnitude, as designed.

Focused resource allocation was achieved through the adaptive refinement process. Test points were concentrated in regions with elevated errors, with concentration factors ranging from 1.0× (sinc, minimal concentration needed) to 2.0× (exp, moderate concentration) relative to uniform sampling. This focused allocation provides more detailed characterization precisely where it is most needed, improving the efficiency of precision assessment compared to uniform sampling strategies.

Practical convergence was demonstrated across all test cases. The exponential function reached the iteration limit

but showed threshold stabilization indicating saturation of the critical region. The logarithmic and sinc functions converged within 4 and 2 iterations respectively, demonstrating that the method achieves convergence within a reasonable number of iterations for functions with well-defined critical regions.

While these specific numerical values were generated using simulated precision degradation, they demonstrate the method's behavior when comparing implementations with genuine precision differences. The controlled degradation in specific regions simulates realistic scenarios such as overflow boundaries, underflow regions, and cancellation-prone computations that occur in practical arithmetic implementations.

V. ALGORITHMIC COMPLEXITY ANALYSIS

The algorithm consists of several stages, each with its own computational complexity. Initially, a set of test values is generated, which is a linear process with a complexity of $O(n)$, where n is the number of test values. Next, each test value is processed by performing a comparison between two arithmetic systems, requiring two operations per test case. This stage also remains within $O(n)$ complexity.

The most computationally intensive step is the update of the threshold value, denoted as θ, based on the statistics from previous iterations. Since the threshold is dynamically adjusted using statistical methods – typically involving the mean or median of the observed errors – this step requires iterating over the entire set of error values, adding another $O(n)$ to the overall complexity. Furthermore, if the number of iterations k depends on the convergence rate of the error reduction, the total number of iterations may grow logarithmically with respect to the initial error magnitude. In such cases, the overall complexity becomes:

$$O(n \log n). \tag{17}$$

Thus, the worst-case complexity of the algorithm is estimated as Eq. (17), which occurs when multiple refinement iterations are performed and the threshold is dynamically updated at each step. However, in cases where the improvement in accuracy reduces, the number of iterations is automatically limited to a fixed number of steps, resulting in a linear complexity of $O(n)$, which makes the approach computationally efficient.

VI. RELATED WORK AND DISCUSSION

The adaptive benchmark testing method builds upon a range of well-established approaches in numerical analysis; however, in its integrated form, it has not yet been formally presented in the scientific literature. At its core lies the idea of dynamically expanding test sets based on the magnitude of detected errors, resembling strategies for identifying problematic points commonly used in numerical optimization and machine learning. Unlike those domains, however, the proposed method is tailored specifically to the analysis of arithmetic accuracy.

Traditional approaches, such as Monte Carlo methods [4-5], typically rely on random generation of input data without incorporating feedback from the obtained results. These methods do not specifically target critical numerical regions where the highest errors occur. In contrast, the proposed approach adaptively adjusts the test set to focus precisely on those areas where the arithmetic exhibits the largest deviations from expected values. This enables more efficient use of computational resources.

Similar concepts appear in adaptive numerical integration [17], where the partitioning of the domain is modified based on the error, or in the verification of solutions to differential equations [7], where error control is used to guide the numerical scheme. The method applies a similar principle but focuses on adapting the input data rather than the numerical process itself.

Another related concept is property-based testing (e.g., the Hypothesis library in CPython) [18], which aims to discover counterexamples to specified properties. However, these systems do not incorporate numerical error estimation and are not intended for analyzing the stability of arithmetic computations.

It is also worth mentioning phased software testing techniques [9], where the system is subjected to a variety of input scenarios to detect abnormal behavior. These techniques, however, are usually concerned with logical correctness or handling of exceptional cases rather than numerical accuracy assessment.

One of the key features of the proposed approach is the use of the standard mixed absolute-relative tolerance criterion, well-established in numerical ODE solvers, combined with adaptive test refinement. This provides a more comprehensive view of arithmetic behavior across different stages of computation while following established best practices in numerical analysis. To date, such integration in the context of evaluating arithmetic accuracy in systems like CPython has not been formalized. Therefore, the described method can be considered a novel and promising approach for studying the effectiveness of arithmetic operations in interpreters.

The demonstration presented in Section IV validates that the proposed method successfully bridges the gap between theoretical algorithm design and practical application. Unlike Monte Carlo methods that treat all regions uniformly, or adaptive quadrature that focuses on integration accuracy, the approach specifically targets the identification of precision-sensitive regions through iterative test refinement driven by observed error metrics. The demonstration confirmed the method's key capabilities: automatic identification of critical regions without prior knowledge of their location, statistical robustness through automatic threshold selection, focused allocation of testing resources, and practical convergence within reasonable iteration limits. While conducted using controlled precision variations, these results establish the method's readiness for application to production arithmetic implementations where genuine precision differences exist.

The adoption of the mixed criterion aligns this work with established practices in numerical computing, making it directly comparable to standard approaches used in MATLAB, SciPy, and other numerical software, while extending these concepts to the novel domain of adaptive arithmetic verification.

VII. CONCLUSION

This work presents a comprehensive adaptive benchmark testing method for evaluating the precision of floating-point arithmetic implementations. The method addresses fundamental limitations of conventional testing approaches by dynamically refining the test set to concentrate sampling density in regions where computational errors are most significant.

The primary contribution of this research is a unified framework that integrates three key components: statistically validated error thresholding with automatic adaptation between parametric and non-parametric approaches based on distribution characteristics, iterative test set refinement guided by rigorous statistical convergence criteria, and the application of the standard mixed absolute-relative tolerance criterion from numerical ODE solvers to adaptive arithmetic verification.

Sensitivity analysis revealed that the threshold parameter λ should be maintained within the range $[1.5, 2.5]$ to balance error detection sensitivity against computational efficiency. The tolerance parameters $\tau_{abs} = 10^{-10}$ and $\tau_{rel} = 10^{-6}$ provide appropriate precision requirements for general-purpose double-precision testing, with the crossover magnitude $M_c = 10^{-4}$ appropriately balancing absolute and relative error contributions across different function magnitudes.

Experimental validation on standard mathematical functions (exp, log, sinc) demonstrated rapid convergence in 2-4 iterations for functions with well-defined critical regions, with test set expansion factors ranging from 1.0× to 56.5× depending on error distribution characteristics. The method successfully identified critical regions comprising 0.1% to 10.1% of final test sets while maintaining normalized error values well within specified tolerance envelopes (maximum $E(x) < 10^{-6}$ for all test cases). The automatic switching between parametric and quantile-based thresholds handled both normal and non-normal error distributions effectively, with normality pass rates ranging from 0% (exp, log) to 100% (sinc).

Algorithmic complexity analysis established that the method operates within $O(n \log n)$ time complexity in the worst case, where n is the total number of test values evaluated. In practice, the automatic convergence mechanism limits the number of iterations, often achieving near-linear $O(n)$ behavior for functions with localized critical regions.

Extension to multidimensional input spaces would require sophisticated techniques to manage the exponential growth in test set size, potentially incorporating dimensional reduction strategies or hierarchical sampling schemes.

The demonstration in Section IV validates the method's core capabilities using controlled precision variations. Future work will apply this methodology to production arithmetic implementations, including comparison of CPython's standard arithmetic with a C-based extended-precision system designed for enhanced numerical accuracy in scientific computing applications.

Future research directions include: integration with automated test generation frameworks such as Hypothesis for property-based testing, extension to vector and matrix operations common in scientific computing libraries, development of adaptive sampling strategies for multivariate functions using space-filling curves or adaptive mesh refinement techniques, and application to domain-specific arithmetic implementations such as interval arithmetic or arbitrary-precision libraries.

The method provides a practical and theoretically grounded tool for arithmetic verification in critical applications where numerical precision directly impacts system reliability, including scientific simulations, financial calculations, and safety-critical embedded systems. By adopting the mixed criterion from numerical analysis, the method ensures compatibility and comparability with established practices in the field while introducing novel adaptive refinement strategies for arithmetic testing.

AUTHOR CONTRIBUTIONS

D.D. – conceptualization, methodology, software implementation, formal analysis, investigation, validation, writing (original draft preparation), writing (review and editing), visualization.

COMPETING INTERESTS

The author declares no conflict of interest.

REFERENCES

[1] G. Dahlquist and Åke Björck, *Numerical Methods in Scientific Computing*. SIAM, 2008.

[2] N. J. Higham, *Accuracy and Stability of Numerical Algorithms*, 2nd ed. SIAM, 2002.

[3] T. Nelson, E. Rivera, S. Soucie, T. D. Vecchio, J. Wrenn, and S. Krishnamurthi, "Automated, Targeted Testing of Property-Based Testing Predicates," *The Art Science and Engineering of Programming*, vol. 6, no. 2, Nov. 2021, doi: https://doi.org/10.22152/programming-journal.org/2022/6/10.

[4] N. Metropolis and S. Ulam, "The Monte Carlo Method," *Journal of the American Statistical Association*, vol. 44, no. 247, pp. 335–341, Sep. 1949, doi: https://doi.org/10.1080/01621459.1949.10483310.

[5] "Monte Carlo: Concepts, Algorithms, and Applications," *Journal of Computational and Applied Mathematics*, vol. 75, no. 2, pp. N3–N4, Nov. 1996, doi: https://doi.org/10.1016/s0377-0427(97)80822-9.

[6] J. R. D'Errico, "An Adaptive Quadrature Routine," *ACM SIGAPL APL Quote Quad*, vol. 17, no. 2, pp. 19–20, Dec. 1986, doi: https://doi.org/10.1145/9327.9331.

[7] M. Redmann and S. Riedel, "Runge-Kutta Methods for Rough Differential Equations," *Journal of Stochastic Analysis*, vol. 3, no. 4, Dec. 2022, doi: https://doi.org/10.31390/josa.3.4.06.

[8] P. Godefroid, "Fuzzing," *Communications of the ACM*, vol. 63, no. 2, pp. 70–76, Jan. 2020, doi: https://doi.org/10.1145/3363824.

[9] B. Beizer and J. Wiley, "Black Box Testing: Techniques for Functional Testing of Software and Systems," *IEEE Software*, vol. 13, no. 5, p. 98, Sep. 1996, doi: https://doi.org/10.1109/ms.1996.536464.

[10] E. Hairer *et al.*, *Solving Ordinary Differential Equations I*, 2nd ed. Springer Series in Computational Mathematics. Berlin, Heidelberg: Springer Berlin Heidelberg, 1993.

[11] L. F. Shampine and M. W. Reichelt, "The MATLAB ODE Suite," *SIAM Journal on Scientific Computing*, vol. 18, no. 1, pp. 1–22, Jan. 1997, doi: https://doi.org/10.1137/s1064827594276424.

[12] P. Virtanen *et al.*, "SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python," *Nature Methods*, vol. 17, no. 3, pp. 261–272, Feb. 2020, doi: https://doi.org/10.1038/s41592-019-0686-2.
[13] "Defining Extension Types: Tutorial," *Python Documentation*, 2025. [Online]. Available: https://docs.python.org/3/extending/newtypes_tutorial.html#adding-data-and-methods-to-the-basic-example (accessed Dec. 08, 2025).
[14] S. S. Shapiro and M. B. Wilk, "An Analysis of Variance Test for Normality (Complete Samples)," *Biometrika*, vol. 52, no. 3/4, pp. 591–611, 1965, doi: https://doi.org/10.2307/2333709.
[15] B. L. Welch, "The Generalization of `Student's' Problem When Several Different Population Variances Are Involved," *Biometrika*, vol. 34, no. 1/2, p. 28, Jan. 1947, doi: https://doi.org/10.2307/2332510.
[16] J. Cohen, *Statistical Power Analysis for the Behavioral Sciences*, 2nd ed. Routledge, 1988.
[17] R. D. Richtmyer and R. E. Moore, "Interval Analysis," *Mathematics of Computation*, vol. 22, no. 101, p. 219, Jan. 1968, doi: https://doi.org/10.2307/2004792.
[18] "Adapting strategies - Hypothesis 6.148.7 documentation," *Readthedocs.io*, 2025. [Online]. Available: https://hypothesis.readthedocs.io/en/latest/tutorial/adapting-strategies.html (accessed Dec. 08, 2025).

Denys Deineko

PhD student at Computer Systems Software Department, Yuriy Fedkovych Chernivtsi National University. Master of Computer Science. Research Interests: instrumentation and profiling of CPython, compiler-level optimizations for numerical correctness.

ORCID ID: 0009-0001-3386-3430

Метод адаптивного контрольного тестування оцінки якості арифметики

Денис Дейнеко*

Кафедра програмного забезпечення комп'ютерних систем, Чернівецький національний університет імені Юрія Федьковича, Чернівці, Україна

*Автор-кореспондент (Електронна адреса: deineko.denys@chnu.edu.ua)

АНОТАЦІЯ Арифметика з плаваючою крапкою є вразливою до похибок точності, які можуть накопичуватись у процесі обчислень і суттєво впливати на результати. Особливо це актуально для застосувань, де навіть незначні відхилення можуть призвести до критичних помилок – зокрема, у наукових розрахунках, технічному моделюванні, фінансових обчисленнях або роботі з великими обсягами даних. У статті запропоновано новий метод адаптивного тестування, який призначений для систематичної перевірки точності реалізацій арифметичних операцій. Його мета полягає в тому, щоб виявити слабкі місця в обчисленнях, визначити джерела нестабільності та запропонувати обґрунтовану оцінку надійності числових результатів. На відміну від традиційних підходів, що використовують фіксовані або випадково згенеровані вхідні дані, запропонований метод постійно оновлює набір тестів, зосереджуючи увагу на тих ділянках числової області, де похибки виявляються найзначнішими. Це дає змогу ефективно виявляти ті області вхідних значень, які найчастіше викликають значні відхилення результатів, зокрема – у разі використання дуже малих, великих або граничних чисел. Метод базується на багаторазовому запуску арифметичних операцій, таких як додавання, віднімання, множення та ділення, з подальшим порівнянням отриманих результатів у різних реалізаціях арифметики. Для порівняння використовуються базова реалізація арифметики та альтернативна реалізація з підвищеною точністю. Похибки оцінюються за допомогою абсолютних та відносних показників, а їхні статистичні властивості аналізуються для виявлення закономірностей. Отримані дані використовуються для динамічного оновлення тестового набору: якщо у певній числовій області виявлено значні похибки, то в цій самій області формуються нові тестові значення з невеликими відхиленнями. Такий підхід дозволяє поступово уточнювати оцінку якості арифметичних обчислень у найпроблемніших областях. У результаті тестування стає все більш цілеспрямованим і сфокусованим на складних випадках, що особливо важливо для складних інженерних або наукових задач. Таким чином, запропонований підхід підвищує ефективність перевірки числових обчислень, надає нові можливості для аналізу точності й виявляє приховані проблеми, які залишаються непомітними при класичному тестуванні, що робить його корисним для верифікації числових бібліотек та покращення арифметичних моделей.

КЛЮЧОВІ СЛОВА верифікація числових обчислень, плаваюча крапка, точність обчислень, обчислювальна складність, адаптивне тестування.