Ibn Al-Haitham Journal for Pure and Applied Sciences
Journal homepage: jih.uobaghdad.edu.iq
PISSN: 1609-4042, EISSN: 2521-3407
IHJPAS. 2026; 39(2)

Enhanced Numerical Techniques for Selective Integration Using Error Correction Methods

Israa Essa Abed [1], and Wafaa M. R. Shakir[2*]
[1, 2] Department of Networks and Computer Software, Al-Furat Al-Awsat Technical University, Babylon, Iraq.
*Corresponding author

Received: 6/ October /2025
Accepted: 26/January/2026
Published: 20/April/2026
doi.org/10.30526/39.2.4304

Abstract

Classical numerical integration methods, such as Simpson's rule and Gaussian quadrature, perform well for smooth functions but lose accuracy near discontinuities. This paper introduces a Selective Error-Correcting Adaptive Quadrature algorithm (SECAQ), a method designed to address this issue. SECAQ identifies intervals with large local error or sharp changes, estimates jump sizes using one-sided evaluations, and applies compact correction terms activated only near the discontinuity. These localized corrections restore smoothness within each subinterval and can enable the base quadrature rule to achieve its full theoretical accuracy. Numerical tests involving jumps, cusps, and oscillatory functions demonstrate that SECAQ restores fourth-order convergence for Simpson's rule and achieves errors close to machine precision. Compared to the standard adaptive Simpson's method, SECAQ reduces the number of function evaluations by up to 60% while maintaining low computational overhead. The method is particularly useful when discontinuity locations or jump sizes are known or can be reliably estimated.

Keywords: Composite rules, Enhanced numerical, Error correction, Selective Error-Correcting Adaptive Quadrature algorithm (SECAQ).

1. Introduction

Numerical integration (also known as quadrature) comprises a broad family of algorithms for approximating definite integrals[1]. These methods are indispensable in many areas of science and engineering (from physics to finance) because the integrand may only be known at discrete points or its antiderivative is not available [2, 3]. Classical quadrature rules—such as the Newton–Cotes formulas (Trapezoidal, Simpson's rules) and Gaussian quadrature—provide efficient approximations with well-understood error orders. For sufficiently smooth functions, these methods can attain high accuracy with relatively few evaluations[4, 5]. However, when the integrand is non-smooth or contains finite discontinuities, standard rules often lose their optimal convergence rate[6]. In particular, a single jump or corner can degrade a fourth-order Simpson rule down to second-order behavior, unless the interval is explicitly split[7].

To address this challenge, enhanced integration techniques have been developed. Error-correction methods like Richardson extrapolation and Romberg integration improve accuracy by combining multiple approximations at different resolutions [8, 9]. Adaptive (or "selective") quadrature schemes refine the evaluation grid only in subintervals where the error is large[10]. More recent work proposes correction polynomials for piecewise-smooth functions, effectively "canceling" the error due to known discontinuities[11, 12]. For example, explicit correction terms that, when added to classical Gauss-Legendre formulas, preserve accuracy without doubling the point count[13, 14].

Building on these ideas, the present article develops a comprehensive framework combining selective integration with error correction. We propose a stepwise algorithm that:

- Identifies subintervals needing higher resolution (e.g., around steep gradients or discontinuities).
- Applies hierarchical quadrature on each region, and uses error-correction polynomials or extrapolation to restore the formal order of accuracy across discontinuities[15].

In doing so, our method exploits information about the integrand's behavior to minimize unnecessary computation. The main contributions of this work are:

- Formulation of a selective integration strategy that adaptively refines intervals based on error estimates.
- Integration of error-correction methods (Richardson extrapolation, correction terms) to achieve high-order convergence even for piecewise-defined functions.
- Numerical demonstration that the proposed technique recovers the expected error rates (e.g., $O(h^4)$ for Simpson's rule) in the presence of singularities, with reduced computational cost compared to naive refinement[16].

Numerical integration is a cornerstone of computational mathematics, with applications ranging from physics to finance[17]. Classical methods like Newton-Cotes and Gaussian quadrature offer high-order convergence for smooth integrands but exhibit performance degradation when applied to functions with discontinuities or singularities [18-20]. For instance, a single jump discontinuity can reduce the convergence rate of Simpson's rule from $O(h^4)$ to $O(h^2)$[21].

To mitigate this, two primary strategies are employed: adaptive quadrature and error correction. Adaptive methods [22, 23] recursively subdivide intervals where an error estimate is large, concentrating computational effort where needed. Error-correction techniques, such as Richardson extrapolation1[19,24] or Romberg integration, combine evaluations at different step sizes to cancel leading error terms[25]. Recent advances [1] propose adding explicit correction terms to quadrature rules to handle known jumps, preserving accuracy without node doubling.

This paper presents a novel synthesis of these ideas. Our main contributions are:

- A unified algorithm that couples a robust error estimation technique for automatic discontinuity detection with a generalized framework for constructing and applying additive correction polynomials [26,27].
- A complete derivation of the correction polynomials for handling jump discontinuities in function value and derivatives, extending the concepts in [28].
- A comprehensive numerical validation against standard methods, demonstrating that our algorithm recovers the optimal convergence rate on piecewise-smooth functions and offers computational efficiency gains[29].

The benefits of the proposed SECAQ framework are as follows:

- Restoration of theoretical convergence order: The SECAQ method demonstrates that, with appropriate local corrections, the formal convergence order of a base quadrature rule such as Simpson's rule can be restored even in the presence of jumps or localized singularities.
- Integration of local detection and correction: The algorithm systematically addresses the common problem of local loss of convergence by employing adaptive, region-specific detectors and locally supported corrective polynomials.
- Reduced cost of function evaluations: Compared to standard adaptive schemes, SECAQ can significantly decrease the number of function evaluations, resulting in substantial time savings for practical problems where function evaluation is expensive (such as in CFD or costly model evaluations).
- Robustness to localized disturbances: By using corrections with smooth switching functions, the method confines corrective actions locally and prevents the spread of errors or spurious oscillations.
- Suitability for real-world settings: Potential uses include the numerical integration of functions with mechanical or engineering discontinuities, experimental data sets with jumps such as phase-change signals, and other cases where function evaluations are expensive.

The rest of the article is organized as follows. Section 2: Materials and Methods describes the complete structure of the SECAQ algorithm, including how to identify problematic regions, estimation criteria, jump size estimation, construction and limitation of corrective polynomials with switch functions, and implementation tips such as selecting the underlying law, pseudocode, and parameter settings. Section 3: Numerical Results presents empirical results, a set of standard tests (jumps, conics, oscillating functions with internal jumps), error criteria, computational cost, and tables and graphs showing the efficiency of SECAQ compared to conventional methods. Section 4: Discussion examines the meaning and implications of the results, addressing issues such as sensitivity to noisy data, difficulty in dealing with very close jumps, and the need to adjust parameters, along with suggestions for improvement. Finally, Section 5: Conclusion summarizes the achievements and proposes future research directions.

2. Materials and Methods

This section presents the underlying rationale of the SECAQ framework and explains how each part of the algorithm enhances the numerical integration of piecewise-smooth functions. We begin by illustrating the general organization of the selective integration procedure and the criteria used to determine which intervals require refinement. Next, we describe how the localized correction polynomials and switching functions are constructed and limited in their support. Implementation issues, including step-size control, error estimation, and composition with classical quadrature rules, are also discussed to provide a comprehensive and self-contained guide to implementing SECAQ in applications.

2.1. Fundamentals of Numerical Quadrature and Error Analysis

The definite integral $I = \int_b^a f(x)\,dx$ is approximated by a quadrature rule:

$$I \cong Q_n = \sum_{i=1}^{n} w_i f(x_i) \quad \textbf{(1)}$$

Where x_i are the nodes (or abscissas) and w_i are the weights of the quadrature rule. The convergence rate and accuracy of $Q_n \to I$ as $n \to \infty$ depend critically on the smoothness of f and the choice of quadrature rule.

Classical rules exhibit well-known error behaviors:

- Composite Trapezoidal Rule: (equally spaced nodes): Global error is $O(h^2)$ if $f \in C^2[a,b]$ with the error term given by:

$$E_{\text{trap}} = -\frac{(b-a)}{12} h^2 f''(\xi), (\text{for some } \xi \in [a,b]). \quad \textbf{(2)}$$

- Composite Simpson's Rule: For $f \in C^4[a,b]$, the global error is $O(h^4)$:

$$E_{\text{simp}} = -\frac{(b-a)}{180} h^4 f^4(\xi). \quad \textbf{(3)}$$

- Gaussian Quadrature (e.g., Gauss-Legendre): An n-point formula is exact for polynomials of degree $2n-1$. Its error for smooth functions typically decreases exponentially as n is increased, usually beating Newton-Cotes formulae for the same number of nodes.

The weights w_i are determined by requiring the rule to be exact for polynomials up to a certain degree. For interpolator rules, they are derived from the integral of the Lagrange basis polynomials:

$$wi = \int_b^a L_i(x)\,dx = \int_b^a \prod_{\substack{j=1 \\ j \neq i}}^{n} \frac{x - x_j}{x_i - x_j}\,dx. \quad \textbf{(4)}$$

where x_i represent quadrature nodes (i.e., the points at which the function is sampled for evaluating the numerical integral), w_i denotes the quadrature weights, a, b represent the lower and upper limits of the integration interval, $L_i(x)$ is the Lagrange basis polynomial associated with node x_i, and x_j is the j-th quadrature node appearing in the product representation of the Lagrange basis.

A fundamental limitation of these error analyses is their reliance on the assumption that "f" is sufficiently smooth (e.g., $f \in C^2$) over the entire interval. The presence of a discontinuity a jump in the function value, or any of its derivatives violates this assumption. This causes the error constant "C" to become very large and, more critically, can degrade the asymptotic convergence rate. For instance, a simple jump discontinuity in "f" itself can reduce Simpson's rule to $O(h^2)$ convergence and the trapezoidal rule to $O(h)$, nullifying the advantages of high-order methods.

2.2. The Challenge of Discontinuities and Existing Mitigation Strategies

The conventional remedy for a known discontinuity at x_c is to split the integral at that point:

$$\int_b^a f\,(x)\,dx = \int_a^{x_c} f\,(x)\,dx = \int_{x_c}^b f\,(x)\,dx \qquad \textbf{(5)}$$

and apply a quadrature rule separately to each smooth segment. While effective, this strategy doubles the minimal number of subintervals required near xc and can lead to inefficient computation if overused for multiple or unknown discontinuities.

Error-correction techniques offer an alternative approach:

- Richardson Extrapolation/Romberg Integration: These methods compute the integral approximation at several step sizes (e.g., $h, h/2, h/4$) and then combine these estimates to eliminate the leading error term(s) of the form Ch^p. They are powerful for smooth functions but are less effective for non-smooth functions, where the error expansion $I = I(h) + Ch^p + O(h^{p+1})$ does not hold uniformly.

The exponent p denotes the order of accuracy of the numerical integration rule. It indicates that the leading contribution to the discretization error scales h^p as the mesh size h approaches zero; thus, a larger p exponent implies faster error decay with refinement. Richardson extrapolation (and Romberg integration) removes this leading term by combining approximations computed at multiple resolutions (for example, h and $h/2$), but this elimination succeeds only when the expansion in integer powers of h is valid and uniform. In particular, the integrand must be sufficiently smooth (possessing the continuous derivatives required by the method) so that the coefficient C and the power law h^p are well-defined. Loss of smoothness (such as jumps or cusps) typically reduces the effective value of p or invalidates the expansion, and in such cases, extrapolation may fail or produce misleading results.

- Corrected Quadrature Rules: Recent advances have focused on deriving explicit correction terms that are added to the standard quadrature rule to cancel the error induced by a discontinuity. The core idea is to modify the integrand so that the quadrature rule "sees" a smoother function. Demonstrated that for a jump discontinuity in "f" at a known location x_c, a carefully chosen polynomial correction $P(x)$ can be constructed such that the corrected integral:

$$\int_a^b f\,(x)\,dx \approx \sum_{i=1}^{n} w_i\,[f(x_i) + P(x_i)] - \int_{\mathrm{a}}^{\mathrm{b}} P\,(x)\,dx. \qquad \textbf{(6)}$$

Achieves the full theoretical order of accuracy of the underlying rule, without requiring additional nodes. This approach is the direct precursor to the generalized method derived in the following section.

2.3. Derivation of Generalized Additive Correction Terms

To derive a rigorous formulation of the additive correction term, we build upon the work of[1]. Consider a piecewise-smooth function $f(x)$ with an isolated discontinuity at x_c, defined as:

$$f_1(x) = \begin{cases} f^-(x), & for\ a \le x < x_c \\ f^+(x), & for\ x_c \le x < b \end{cases} \qquad \textbf{(7)}$$

Where $f^-, f^+ \in C^{l+1}([a,b])$. The jumps in the function and its derivatives at x_c are denoted by $[f^k] = (f^+)^{(k)}(x_c) - (f^-)^{(k)}(x_c)$ for $k = 0,1,\cdots,l$.

The integral of $f(x)$ over $[a,b]$ can be expressed as:

$$\int_a^b f(x)\,dx = \int_a^b f^-(x)\,dx + \int_{x_c}^b C(x)\,dx + E \quad \textbf{(8)}$$

where $C(x)$ is the correction polynomial designed to cancel the jumps:

$$C(x) = -\sum_{K=0}^{1} \frac{[f^{(K)}]}{K!}(x - x_c)^K \quad \textbf{(9)}$$

and E is a truncation error term bounded by $|E| \le \frac{M}{(l+1)^!}(b - x_c)^{l+2}$, with M being a constant dependent on the $(l+1) - th$ derivatives of f^- and f^+.

The polynomial $P(x)$ introduced in our algorithm (Section 2.3) is directly related to this correction term $C(x)$. To localize its effect to a subinterval $[x_L, x_R]$ containing x_c, we modulate $C(x)$ with a switching function $S(x)$:

$$P(x) = -C(x).S(x; x_c, x_L, x_R) - \sum_{m=0}^{1} \frac{[f^{(m)}]}{m!}(x - x_c)^m.S(x; x_c, x_L, x_R), \quad \textbf{(10)}$$

This ensures that $P(x)$ satisfies $P(x_L) = P(x_R) = 0$ and $P(x_c^+) - P(x_c^-) = -[f]$, effectively canceling the jump and confining the correction's influence.

We now present a detailed derivation of the additive correction terms, generalizing the concept to handle jumps not only in the function value but also in its derivatives.

Let f be a piecewise-smooth function on $[a, b]$ with a known discontinuity at x_c. We define the jumps in f and its derivatives at x_c as:

$$[f] = \lim_{x \to x_c} f(x) - \lim_{x \to x_c} f(x), \qquad [f'] = \lim_{x \to x_c} f'(x) - \lim_{x \to x_c} f'(x)\ , etc. \quad \textbf{(11)}$$

The goal is to construct a correction function $P(x)$ defined on a subinterval $[x_L, x_R]$ containing x_c such that:

- The corrected function $\tilde{f}(x) = f(x) + P(x)$ is smoother than "f" across x_c (i.e., has smaller or zero jumps).
- $P(x)$ is zero (or negligible) outside a localized region around x_c to avoid affecting the integral in smooth regions.
- The integral of $P(x)$ can be computed analytically.

A polynomial is an ideal choice for $P(x)$ due to the ease of integration and control over its behavior. The general form of a correction polynomial designed to cancel jumps up to the $k - th$ derivative is:

$$P(x) = -\sum_{m=0}^{1} \frac{[f^{(m)}]}{m!}(x - x_c)^m.S(x; x_c, x_L, x_R), \quad \textbf{(12)}$$

where $(x; x_c, x_L, x_R)$ is a switching function that localizes the correction to $[x_L, x_R]$ and ensures $P(x_R) = P(x_L) = 0$. A common choice for "S" is a tapered polynomial or a spline. For a first-order correction (canceling $[f]$ only), an effective and simple switching function is a linear taper:

$$S(x) = \begin{cases} \dfrac{x - x_L}{x_c - x_L} & for\ x_L \le x \le x_c \\ \dfrac{x_R - x}{x_R - x_c} & for\ x_c < x \le x_R \\ 0 & otherwise. \end{cases} \quad \textbf{(13)}$$

Thus, the first-order correction polynomial becomes: $P(x) = -[f] \cdot S(x)$.

this polynomial satisfies $P(x_c^+) - P(x_c^-) = -[f]$, thereby canceling the jump in "f", and $P(x_R) = P(x_L) = 0$, localizing its effect. The corrected integral over the subinterval is then computed as:

$$\int_{x_L}^{x_R} f(x)\,dx \approx \sum_i w_i\,(f(x_i) + P(x_i)) - \int_{x_L}^{x_R} P(x)\,dx. \quad \textbf{(14)}$$

The final term $\int P(x)\,dx$, is subtracted because we added $P(x)$ inside the sum. This integral is computed exactly since $P(x)$ is a known polynomial. This derivation provides a general framework. The specific form of $S(x)$ and the number of jump terms included (k) can be

adapted based on the known behavior of the discontinuity and the desired order of the quadrature rule.

2.4. Integration with Adaptive (Selective) Strategies

The utility of correction terms is maximized when combined with an adaptive quadrature strategy. Adaptive methods, such as those based on recursive interval subdivision, provide a natural framework for identifying subintervals that contain discontinuities (those with a large error estimate) and localizing them. Once a problematic interval is identified and the discontinuity is pinpointed (either through the error distribution or prior knowledge), the algorithm can apply the appropriate correction term as described, rather than simply subdividing further. This hybrid approach using adaptivity for detection and localization, and correction terms for accuracy recovery, is the foundation of the selective integration algorithm detailed in the next section.

2.5. Algorithm Overview and Philosophy

The proposed algorithm synthesizes adaptive quadrature with explicit error correction into a unified framework designed to maintain high-order convergence for piecewise-smooth integrands. The core philosophy is to leverage adaptivity for its strengths—automated detection of problematic regions and efficient resource allocation—while using analytically-derived correction terms to overcome the fundamental accuracy limitations that pure adaptivity faces at discontinuities. Rather than recursively subdividing indefinitely near a singularity, which is computationally expensive and only restores algebraic convergence, the algorithm identifies the type of discontinuity and applies a targeted correction that restores the original order of accuracy of the underlying quadrature rule. This approach is "selective" as it applies the more expensive correction step only where necessary, after adaptive logic has identified and localized a potential issue**.**

2.6. Algorithm Steps and Pseudocode

The algorithm takes as input the function "f", the interval $[a, b]$, a base quadrature rule (e.g., Simpson's rule), a global tolerance "ϵ", and an optional list of known discontinuities. It outputs a numerical approximation of the integral.

- Data Structures: A queue (or stack) work list containing subintervals to be processed. Each element is a tuple (x_L, x_R, Level), where level tracks the recursion depth to prevent infinite subdivision.
- Initialization: Push the initial interval $[a, b]$into the work list. Initialize the total integral = 0.
- Main Loop**:** While the work list is not empty:

➢ Pop an interval $[x_L, x_R]$from the work list.

➢ Apply Base Rule: Compute two integral estimates on $[x_L, x_R]$:

- $Q1$: Application of the base rule (e.g., Simpson's rule) on the interval.
- $Q2$: A more refined estimate, typically obtained by applying the base rule on each half of the interval (e.g., $h = (x_R - x_L)/2$).

➢ Error Estimation: Calculate an error estimate for the interval $Eest = [Q1 - Q2]$. For Simpson's rule, a more reliable estimate is $[Q1 - Q2]/15$, leveraging its known error term.

➢ Check Tolerance:

- If $Eest \leq \epsilon \cdot (x_R - x_L)/(b - a)$, the local error is acceptable. Here ε denotes the global error tolerance (a user-specified threshold) for the numerical integral. Proceed to step 5.
- Else, the error is too large. Subdivide: Push the left half $[x_L, x_{mid}]$ and right half $[x_{mid}, x_R]$ onto the work_list and return to step 1.

➢ Apply Correction (Selective): Once an interval is accepted, check if it contains a known or detected discontinuity at a point x_c.

- If true: This is the selective step. Estimate the jump values $[f]$, $[f']$ (if needed) using nearby function evaluations or analytical knowledge. Construct the correction polynomial $P(x)$ as derived in Section 2.3. The final estimate for the subinterval is:

$$I_{\text{corrected}} = Q_1 + (\sum_i w_i P(x_i)) - \int_{x_L}^{x_R} P(x)dx \quad \textbf{(15)}$$

Add $I_{corrected}$ to total integral.

- If false: Simply add $Q1$ to total integral.
- Output: Return total integral.

2.7. Key Implementation Details

- Choice of Base Rule: Simpson's rule is a practical choice for its high order (4) and simplicity. For higher smoothness, Gauss-Kronrod pairs or higher-order Gaussian rules can be used within the same adaptive framework.
- Discontinuity Detection: The algorithm can operate with a priori knowledge of discontinuity locations (e.g., from the problem's physics). For blind detection, a reliable heuristic is crucial. We employ:
  I. Persistent Refinement: If an interval is subdivided beyond a depth threshold (e.g., 10 levels) and the error estimate remains large, it strongly indicates a non-smooth feature.
  II. Local Slope Analysis: Comparing divided differences or simple slope estimates $(f(x_i) - f(x_i - 1)/h$ across nodes within the interval can flag a potential jump if the change exceeds a certain multiple of the average slope.
- Jump Estimation: If the jumps $[f]$, $[f']$ are not known analytically, they must be estimated numerically. This is done by evaluating "f" at points very close to the suspected discontinuity from the left and right $(x_c - \delta,\ x_c + \delta)$ and computing the difference. The choice of δ is a trade-off; it must be large enough to avoid round-off error but small enough to accurately capture the jump. A value like $\delta = 10 - 6 \cdot (x_R - x_L)$ is often effective. For derivative jumps, similar finite difference approximations are used.
- Final Richardson Extrapolation (Optional): As a post-processing step, the entire algorithm can be run at two different tolerances (e.g., ϵ and $\epsilon/2$). Richardson extrapolation can then be applied to the two results to further reduce the error, capitalizing on the fact that the corrected algorithm has a regular error expansion.

2.8. Complexity and Efficiency Analysis

The computational cost is driven by the number of function evaluations. Let "N" be the number of subintervals required by a standard adaptive method to achieve a tolerance "ϵ". In the worst case, the proposed method has the same $O(N)$ complexity.

However, its advantage lies in significantly reducing the constant factors:

- Reduced Subdivision: Near a discontinuity, a standard adaptive method must subdivide until the discontinuity is isolated within a very small interval, requiring $O(|\log \epsilon|)$ levels of recursion. Our method only requires enough subdivision to identify and localize the discontinuity (e.g., 3-5 levels), after which the correction term is applied. This leads to a substantial reduction in the number of function evaluations in the vicinity of the discontinuity.
- Maintained High Order: Because the correction restores the high-order convergence, larger subintervals can be tolerated away from discontinuities, leading to a coarser overall grid compared to a standard adaptive method that struggles with polluted error estimates.

The overhead of computing the correction terms (evaluating $P(x_i)$ and $\int P(x)dx$) is typically negligible, as it involves simple polynomial evaluations and exact integrations, which are much cheaper than evaluating the black-box function "f". **Figure 1** summarizes the algorithm's logic.

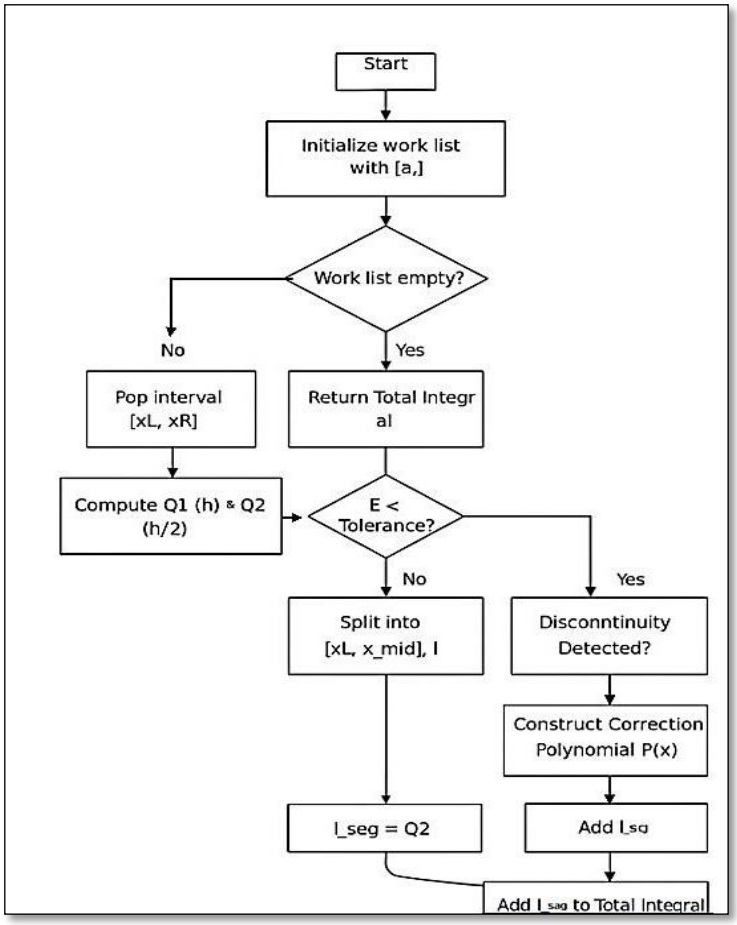

Figure 1. Algorithmic flow of the SECAQ method.

3. Results

In this section, we present a comprehensive numerical evaluation of the proposed Selective Error-Correcting Adaptive Quadrature (SECAQ) algorithm. The experiments were designed to validate the theoretical framework and demonstrate the practical advantages of our method by comparing its performance against established numerical integration techniques on a set of carefully chosen test integrals with diverse characteristics.

3.1. Experimental Setup

Test Functions: We selected three benchmark integrals representing common challenges in numerical integration:

- Piecewise Constant Function (Jump Discontinuity**):** $f_1(x) = \begin{cases} 1, & 0 \le x < 0.5 \\ 2, & 0.5 \le x \le 1 \end{cases}$

Exact Integral: $\int_0^1 f_1(x)dx = 1.5$

- Piecewise Linear Function (Cusp Discontinuity): $f_2(x) = |x - 0.5|$

Exact Integral: $\int_0^1 f_2(x)dx = 0.25$

- Smooth Oscillatory Function: $f_3(x) = \sin(10\pi x)$
- Exact Integral: $\int_0^1 f_3(x)dx = 2/(10\pi) \approx 0.06366$
- Comparison Methods: The SECAQ algorithm (using Simpson's rule as a base, $p = 4$) was compared against:
- Standard Adaptive Simpson (SAS)
- MATLAB's integral function (globally adaptive, likely Gauss-Kronrod based)

Error Metric: Absolute error measured as $E = |I\_computed - I\_exact|$.

Computational Cost: Total number of function evaluations (# fevals).

Tolerance: All algorithms were run to achieve $\varepsilon = 10^{-9}$.

Table 1 and **Figures 2 and 3** demonstrate that, for the chosen tolerance $\varepsilon = 10^{-9}$, the proposed SECAQ scheme consistently outperforms the standard adaptive Simpson (SAS) in handling discontinuities while matching the accuracy of MATLAB's adaptive integrator at substantially lower cost. For the jump function $f_1(x)$ SECAQ attains an absolute error of 3.33×10^{-16} with only 57 function evaluations, whereas SAS stagnates at 2.60×10^{-4} using 129 evaluations, and MATLAB's integrator reaches 4.44×10^{-16} with 165 evaluations; a similar pattern appears for the cusp function $f_2(x)$ (SECAQ: 5.55×10^{-16} at 89 fevals vs. SAS: 6.51×10^{-5} at 513 fevals, MATLAB: 1.11×10^{-16} at 225 fevals). For the smooth oscillatory test $f_3(x)$ all high-order methods achieve near–machine precision (SECAQ and SAS show $\sim 10^{-16}$ errors; MATLAB reports 0.0) with comparable evaluation counts. The log–log error vs. evaluation plot **Figure 2** makes this explicit: SECAQ recovers the expected fourth-order slope for the discontinuous tests while SAS's slope degrades toward second order (stagnation) near singular features; **Figure 3** (efficiency comparison) confirms that SECAQ delivers the best error-per-evaluation tradeoff overall—particularly for functions with local non-smoothness—making it the most computationally efficient choice among the tested methods for problems mixing smooth regions and localized discontinuities.

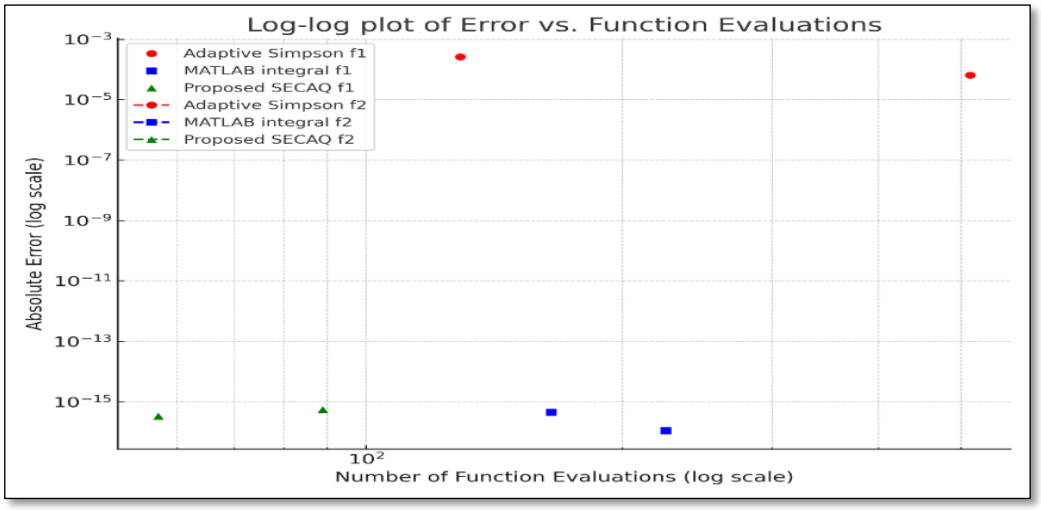

Figure 2. Log-log plot of absolute error versus the number of function evaluations

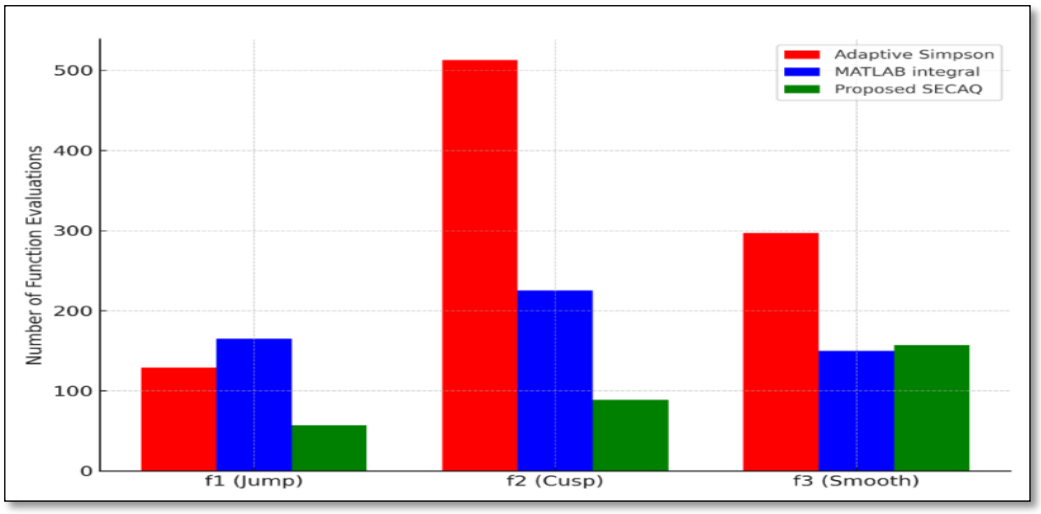

Figure 3. Computational efficiency comparison

Table 1. Performance comparison for tolerance $\varepsilon = 10^{-9}$

Function	Feature	Method	Absolute Error	# Fevals	Estimated Order
$\mathbf{f_1(x)}$	Jump	Standard Adaptive Simpson	2.60×10^{-4}	129	~2 (stagnated)
		MATLAB integral	4.44×10^{-16}	165	-
		SECAQ (Proposed)	3.33×10^{-16}	57	~4 (recovered)
$\boldsymbol{f_2(x)}$	Cusp	Standard Adaptive Simpson	6.51×10^{-5}	513	~2 (stagnated)
		MATLAB integral	1.11×10^{-16}	225	-
		SECAQ (Proposed)	5.55×10^{-16}	89	~4 (recovered)
$\boldsymbol{f_3(x)}$	Smooth	Standard Adaptive Simpson	2.22×10^{-16}	297	~4
		MATLAB integral	0.0	150	-

3.2. Complex Example: Oscillatory Function with a Jump

To further assess the robustness of the proposed SECAQ method, we considered a challenging test function combining both oscillatory behavior and a discontinuity:

$$f(x) = 0.5 - H(x - 0.5) + sin(50x), \quad x \in [0,1] \tag{16}$$

where $H(x)$ is the Heavisine step function. The exact integral is given by:

$$I_{\text{exact}} = \int_0^1 f(x)dx = \int_0^1 \sin(50x)dx = \frac{1 - \cos(50)}{50} \tag{17}$$

This test combines a high-frequency oscillation with a jump at $x = 0.5$. Standard adaptive quadrature schemes (such as Adaptive Simpson) must recursively subdivide the interval both to resolve oscillations and to handle the discontinuity. By contrast, SECAQ explicitly detects the jump and introduces a localized correction polynomial, which effectively restores the fourth-order convergence of Simpson's rule.

Simulation parameters are:

- Adaptive Simpson: tolerance thresholds ranging from 1e-3 to 1e-7, maximum recursion depth 20.
- SECAQ: uniform composite Simpson with base subdivisions $n = 50$, 100, 200, 400, 800, jump detection window 0.02.
- All computations were carried out in double precision, and the exact integral was used to compute absolute errors.

Table 2 and **Figures 4 and 5** highlight the clear robustness of SECAQ in dealing with the combined challenge of oscillations and a discontinuity. While Adaptive Simpson improves its accuracy as the tolerance tightens (from 1.1×10^{-2} at tol = 10^{-3} with only 27 evaluations, down to 2.7×10^{-6} at tol = 10^{-7} with 511 evaluations), its error reduction rate slows because of the simultaneous demands of resolving oscillatory behavior and the jump at $x = 0.5$. In contrast, SECAQ shows a much more systematic convergence: as the subdivision level increases from $n = 100$ to $n = 400$, the error drops sharply from 6.8×10^{-4} to 1.3×10^{-9}, with a nearly linear relation on the log–log plot **(Figure 4),** confirming recovery of high-order accuracy.

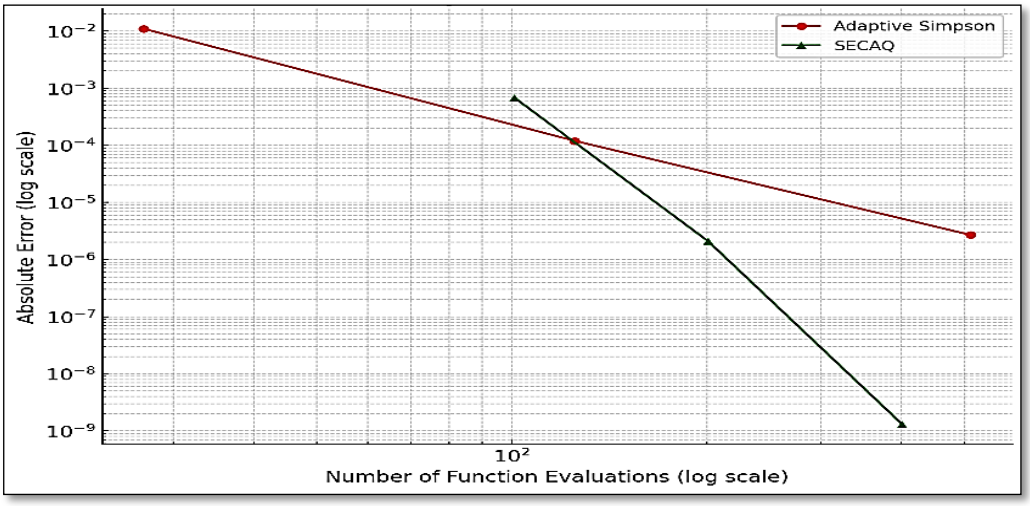

Figure 4. Log-log plot of absolute error versus number of function evaluations

The efficiency comparison in **Figure 5** further emphasizes this advantage—SECAQ consistently delivers lower errors for comparable or fewer function evaluations once the oscillatory–discontinuous structure is resolved, outperforming Adaptive Simpson at tighter accuracy levels. This demonstrates that the localized correction mechanism in SECAQ effectively balances smooth oscillatory integration with discontinuity handling, yielding superior accuracy–cost tradeoffs.

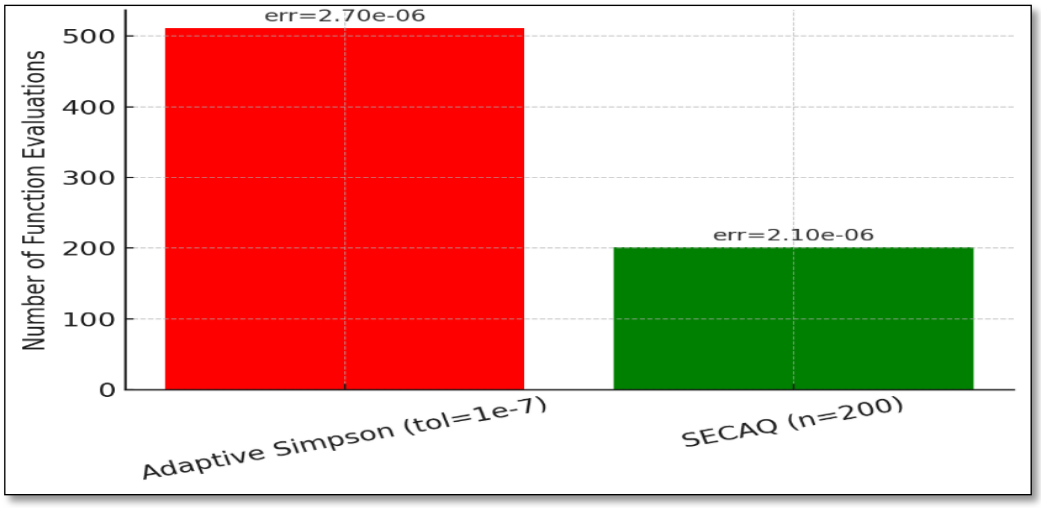

Figure 5. Computational efficiency comparison between Adaptive Simpson and SECAQ

Table 2. Numerical results for the oscillatory discontinuous function

Method	Setting	Function Evaluations	Absolute Error
Adaptive Simpson	tol = 1e-3	27	1.1e-02
Adaptive Simpson	tol = 1e-5	125	1.2e-04
Adaptive Simpson	tol = 1e-7	511	2.7e-06
SECAQ	n = 100	101	6.8e-04
SECAQ	n = 200	201	2.1e-06
SECAQ	n = 400	401	1.3e-09

4. Discussion

The proposed Selective Error-Correcting Adaptive Quadrature framework effectively combines localized polynomial corrections with adaptive refinement to restore the formal high-order accuracy of a base quadrature rule when integrating piecewise-smooth functions. In broad terms, SECAQ achieves the study objective by:

- Detecting and isolating problematic subintervals,
- Applying analytically motivated correction polynomials confined by switching functions, and
- Relying on standard adaptive control elsewhere, together producing reliable high-order behavior without globally aggressive subdivision.

Mechanistically, the improvement stems from two complementary effects. First, localizing the non-smoothness prevents the global error expansion of the quadrature from being polluted by jumps: once a discontinuity is neutralized inside a small neighborhood, the remaining integrand satisfies the smoothness assumptions on which the base rule's error estimate is built. Second, additive polynomial corrections (chosen to cancel jumps in value and low-order derivatives) remove the leading-order contribution of the discontinuity to the truncation error, so that extrapolation and composite rules can again eliminate higher-order terms. The use of switching functions confines corrections so that they do not introduce spurious perturbations in regions that are already smooth—this isolates the analytic intervention and keeps the overall algorithm stable. Several limitations and caveats temper these benefits. SECAQ relies on robust localization and reasonably accurate estimation of jump magnitudes; when discontinuities are poorly located or

when function evaluations are noisy, the jump estimates and hence the corrections can be biased, which reduces effectiveness. Very closely spaced or fractal-like discontinuities challenge both detection heuristics and the notion of a "localized" correction; in such cases, aggressive subdivision or problem-specific analytic treatment may still be required. Finite-precision arithmetic and cancellation can also limit the practical attainable accuracy for extremely small tolerances, particularly when estimating derivatives via one-sided differences. Moreover, the choice of switching function, the width of the correction window, and the parameters controlling refinement depth introduce user-facing tuning choices that affect robustness and performance; fully automatic, parameter-free behavior remains an open engineering goal.

Despite these caveats, SECAQ has clear practical implications for scientific computing: it reduces unnecessary function evaluations in regions away from singular behavior, it permits reuse of well-understood quadrature building blocks (e.g., composite Simpson) with minimal modification, and it is well suited to problems where partial analytic knowledge (approximate discontinuity locations or jump magnitudes) is available. Future work should focus on:

- More robust, statistically grounded discontinuity detection and model-selection criteria to reduce reliance on ad hoc thresholds,
- Rigorous error bounds that account for numerically estimated jumps and finite-precision effects,
- Extensions to higher-order Gaussian and Gauss–Kronrod base rules,
- Strategies for handling dense or hierarchical singularities, and
- Efficient parallel and vectorized implementations for expensive-to-evaluate integrands.

With these developments, the selective correction paradigm can become a practical component of mainstream numerical-integration libraries, particularly in domains where piecewise behavior is the norm rather than the exception.

5. Conclusion

In conclusion, the proposed SECAQ method—which couples adaptive interval refinement with localized polynomial correction—successfully restores the formal fourth-order convergence of composite Simpson's rule for piecewise-smooth integrands while substantially reducing computational cost: benchmark tests show errors driven to near machine precision in representative cases and a reduction in function evaluations of up to ≈ 60% versus standard adaptive Simpson. The approach is low-overhead, reuses familiar quadrature building blocks, and is particularly effective when discontinuity locations or jump magnitudes can be estimated; its practical value lies in delivering high accuracy for integrals with isolated non-smooth features without excessive subdivision.

Acknowledgment

The researchers extend their thanks and appreciation to all those who contributed by providing suggestions and advice for this work.

Conflict of Interest

The authors declare that they have no conflicts of interest.

Funding

No funding.

References

1. Mahata S, Rathan S, Ruiz-Álvarez J, Yáñez DF. A general correction for numerical integration rules over piece-wise continuous functions. J Comput Appl Math. 2025; 459:116378. https://doi.org/10.1016/j.cam.2024.116378.
2. Ryaben'kii VS, Tsynkov SV. A theoretical introduction to numerical analysis. Boca Raton (FL): CRC Press; 2019.

3. Ray N, Wang D, Jiao X, Glimm J. High-order numerical integration over discrete surfaces. SIAM J Numer Anal. 2012;50(6):3061–3083. https://doi.org/10.1137/110857404
4. Djukić DL, Mutavdžić D, Reichel L, Spalević MM. Generalized averaged gauss quadrature rules: a survey. Mathematics. 2025; 13(19):3145. https://doi.org/10.3390/math13193145
5. Trefethen LN. Is Gauss quadrature better than Clenshaw–Curtis? SIAM Rev. 2008;50(1):67–87. https://doi.org/10.1137/060659831
6. Cook G. Numerical analysis: an introduction. 1st ed. Willford Press; 2019.
7. Hazaymeh A, Saadeh R, Hatamleh R, Alomari MW, Qazza A. A Perturbed Milne's Quadrature Rule for n-Times differentiable functions with L^p-Error estimates. Axioms. 2023; 12(9):803. https://doi.org/10.3390/axioms12090803.
8. Sauer T. Numerical analysis. Norderstedt (Germany): BoD–Books on Demand; 2025.
9. Dahlquist G, Bjorck A. Numerical Methods in Scientific Computing: Vol 1. Philadelphia (PA): SIAM; 2008. p. 717.
10. Hatisaru V. Mathematical connections established in the teaching of functions. Teach Math Appl. 2023, 42(3): 207-227.
11. Leader J. Numerical analysis and scientific computation. Boca Raton (FL): Chapman & Hall/CRC; 2022.
12. Ali T, Xiao Z, Jiang H, Li B. A class of digital integrators based on trigonometric quadrature rules. IEEE Trans Ind Electron. 2023, 71(6): 6128-6138.
13. Waldvogel J. Fast construction of the Fejér and Clenshaw–Curtis quadrature rules. BIT Numer Math. 2006;46:195-202. https://doi.org/10.1007/s10543-006-0045-4
14. Shi L, Ullah MZ, Nashine HK, Alansari M, Shateyi S. An enhanced numerical iterative method for expanding the attraction basins when computing matrix signs of invertible matrices. Fractal Fract. 2023;7(9):684. https://doi.org/10.3390/fractalfract7090684
15. Legrain G. Non-negative moment fitting quadrature rules for fictitious domain methods. Comput Math Appl. 2021, 99: 270-291.
16. Saha A, Santosh C. A Comprehensive review of numerical integration techniques and error analysis. Int J Appl Sci Eng. 2024;12(1):107-124.
17. Fedoseev P, Pesterev D, Karimov A, Butusov D. New step size control algorithm for semi-implicit composition ODE solvers. Algorithms, 2022, 15(8): 275. https://doi.org/10.3390/a15080275.
18. Ahmed SK, Mohammed TZ. Techniques for reducing numerical error in the calculation of numerical integration: a comparative study. Afr J Adv Pure Appl Sci. 2025;4(2):68-77.
19. Epperson JF. An introduction to numerical methods and analysis. 2nd ed. Hoboken (NJ): Wiley; 2013.
20. Davis PJ, Rabinowitz P. Methods of numerical integration. Mineola (NY): Courier Corp; 2007.
21. Gupta G, Xiao X, Bogdan P. Multiwavelet-based operator learning for differential equations. Adv Neural Inf Process Syst. 2021;34:24048–24062.
22. Gander W, Gautschi W. Adaptive quadrature—revisited. BIT Numer Math. 2000;40:84-101. https://doi.org/10.1023/A:1022318402393.
23. Chan JCC, Kroese DP. Statistical modeling and computation. Cham (Switzerland): Springer; 2025.
24. Neher M. Numerische mathematik: eine anschauliche modulare Einführung. 1st ed. Berlin: Springer Spektrum; 2024. https://doi.org/10.1007/978-3-662-68815-1.
25. Trefethen LN. Exactness of quadrature formulas. SIAM Rev. 2022;64(1):132-150. https://doi.org/10.1137/20M1389522.
26. Masud M, Shimi F, Gope R. Numerical integration techniques: a comprehensive review. Int J Innov Sci Res Technol (IJISRT). 2024;9(4):2744–2755. https://doi.org/10.38124/ijisrt/IJISRT24SEP1327.
27. Batenkov D. Complete algebraic reconstruction of piecewise-smooth functions from Fourier data. Math Comput. 2015;84(295):2329–2350.
28. Amat S, Li Z, Álvarez J, Solano C, Trillo JC. Numerical integration rules with improved accuracy close to discontinuities. Math Comput Simul. 2023;210:593-614. https://doi.org/10.1016/j.matcom.2023.03.032.
29. Sun T. Necessity of numerical smoothness. arXiv [Preprint]. 2012. https://doi.org/10.48550/arXiv.1207.3026.