Research Paper
Computational Methods for Differential Equations
http://cmde.tabrizu.ac.ir
Vol. *, No. *, *, pp. 1-13
DOI:10.22034/cmde.2026.71492.3593

An augmented artificial bee colony algorithm with adaptive neighborhood search for solving systems of differential equations

Aliakbar Tajari Siahmarzkooh*
Department of Computer Sciences, Golestan University, Gorgan, Iran.

Abstract

The numerical solution of systems of ordinary differential equations (ODEs) remains computationally challenging, particularly for stiff or high-dimensional problems. While metaheuristic algorithms like the Artificial Bee Colony (ABC) offer global search capabilities, they often exhibit slow convergence and inefficiency in the exploitation phase when applied to precise ODE solving. This paper proposes an Augmented ABC (A-ABC) algorithm to address these shortcomings. The core innovation lies in integrating an Adaptive Neighborhood Search (ANS) mechanism into the employed bee phase. ANS dynamically adjusts the perturbation step size based on the fitness landscape, enabling a more refined local search around promising solutions. Applied to three benchmark systems of ODEs (including a stiff problem), the proposed A-ABC algorithm achieves a **6.45% improvement in mean MSE** (averaged over three benchmarks) and a **6.15% reduction in the number of function evaluations** compared to the standard ABC, based on 30 independent runs. Compared to the Gbest-guided ABC and a hybrid PSO-ABC, A-ABC shows improvements of 1.5–3.4% in mean MSE. A sensitivity analysis confirms that the ANS mechanism is robust to the choice of its parameters (ψ_{min}, ψ_{max}), with performance variation below 2%. These results confirm that the Adaptive Neighborhood Search mechanism enhances exploitation without compromising global search capability.

Keywords. Artificial bee colony algorithm, Adaptive neighborhood search, Systems of ordinary differential equations, Metaheuristic optimization, Numerical solution.

2010 Mathematics Subject Classification. 65L05, 34K06, 34K28.

1. Introduction

Systems of ordinary differential equations (ODEs) are fundamental mathematical tools for modeling dynamic processes across scientific and engineering disciplines, including chemical kinetics [11], population dynamics [5], and control systems [13]. Obtaining accurate and efficient numerical solutions for these systems, especially when they are stiff, nonlinear, or high-dimensional, is a persistent computational challenge [1]. Traditional numerical methods, such as Runge-Kutta and linear multistep methods, are well-established but can suffer from stability issues, error propagation, and high computational cost for certain problem classes [2].

In recent years, metaheuristic optimization algorithms have emerged as alternative approaches for solving ODEs. These methods, inspired by natural phenomena, formulate the solution of an ODE as an optimization problem. The core idea is to minimize a fitness function that quantifies the error in satisfying the differential equations and their initial/boundary conditions [26]. Techniques such as Genetic Algorithms (GAs) [10], Particle Swarm Optimization (PSO) [4], and the Artificial Bee Colony (ABC) algorithm [12] have been applied with notable success. These methods do not require discretization of the derivative terms and possess a strong global search ability, making them potentially robust against local minima in the error landscape.

The Artificial Bee Colony algorithm, in particular, has gained attention due to its simplicity, few control parameters, and balanced exploration-exploitation trade-off [19]. Its mimicry of the foraging behavior of honeybee colonies involves

Received: 09 February 20025; Accepted: 12 July 2026.
∗ Corresponding author. Email: a.tajari@gu.ac.ir.

employed bees, onlooker bees, and scout bees working to discover optimal solutions (food sources). However, when applied to the precise task of ODE solving, the standard ABC algorithm exhibits identifiable limitations. Its local search strategy in the employed bee phase relies on a simple random perturbation, which can be inefficient for fine-tuning solutions. This often leads to slow convergence in the exploitation phase and suboptimal solution accuracy [8]. Researchers have proposed various modifications to mitigate these issues, such as incorporating information from the global best solution [28] or using chaos maps for initialization [25]. Yet, a need remains for a *systematic and adaptive* mechanism to enhance local search precision specifically within the ODE solution context.

This paper introduces a novel **Augmented Artificial Bee Colony (A-ABC)** algorithm featuring an **Adaptive Neighborhood Search (ANS)** mechanism to overcome the exploitation weakness of standard ABC. The proposed ANS intelligently modifies the employed bee phase by dynamically adjusting the magnitude of the solution perturbation based on the fitness (error) value of each candidate solution. Solutions with lower error (higher fitness) undergo smaller, more refined perturbations for intensive local search, while those with higher error undergo larger adjustments to escape poor regions. This targeted approach aims to accelerate convergence and improve final solution accuracy.

The primary contributions of this work are as follows:

(1) We propose the A-ABC algorithm with an integrated Adaptive Neighborhood Search (ANS) mechanism to enhance the local exploitation capability for solving ODE systems.
(2) We detail the mathematical formulation for integrating ODE systems into the A-ABC's optimization framework and provide the pseudo-code for the complete algorithm.
(3) We rigorously evaluate the proposed solver on three benchmark ODE systems with distinct characteristics (including a stiff system) and compare its performance against the standard ABC, Gbest-guided ABC, a classical Runge-Kutta solver, a hybrid PSO-ABC variant, and nine other metaheuristic methods from the literature.
(4) We present quantitative results showing that A-ABC achieves a 6.45% improvement in mean squared error (MSE) and a 6.15% reduction in required function evaluations compared to the baseline ABC, averaged over three benchmarks and 30 independent runs. A sensitivity analysis confirms robustness to parameter choices.

The remainder of this paper is organized as follows. Section 2 provides a concise review of recent metaheuristic applications in ODE solving and ABC variants. Section 3 presents the problem formulation and the detailed mechanics of the proposed A-ABC algorithm. Section 4 describes the experimental setup, benchmark problems, and presents the results with discussion. Finally, Section 5 concludes the paper and suggests future research directions.

2. Literature Review

The application of metaheuristic algorithms to solve differential equations has evolved significantly from early proof-of-concept studies to more sophisticated, problem-aware implementations. Initial approaches primarily focused on using Genetic Algorithms (GAs) to optimize parameters within an assumed solution structure (e.g., a polynomial or neural network) [10]. For instance, researchers encoded coefficients of trial solutions into chromosomes and minimized a fitness function representing the sum of squared residuals at collocation points. While demonstrating feasibility, these early methods were often computationally expensive and struggled with high-dimensional or stiff systems due to GAs' propensity for premature convergence.

Particle Swarm Optimization (PSO) gained popularity as a potentially faster alternative, leveraging social learning for guidance. Recent studies, such as the comprehensive review by Chen and Li [4], highlight adaptive PSO variants that adjust inertia weights to balance exploration and exploitation when solving ODEs. A notable advancement is the use of local-best PSO topologies to maintain diversity in the search for solving systems of nonlinear ODEs, preventing the swarm from collapsing into a suboptimal region of the error landscape [15]. However, PSO's performance is sensitive to its control parameters (inertia, cognitive, and social coefficients), and improper tuning can lead to oscillatory behavior or stagnation around non-optimal points when the fitness landscape of an ODE problem is complex.

Among bio-inspired algorithms, the Artificial Bee Colony (ABC) algorithm, proposed by Karaboga and Basturk [12], has shown consistent performance in various global optimization benchmarks. Its strength lies in the distinct roles of employed and onlooker bees for exploitation and scout bees for exploration, creating a natural balance. A 2023 survey by Sharma and Kumar [19] systematically categorizes ABC variants and their applications, including

in engineering optimization problems that can be framed as differential equations. The standard ABC algorithm's equation for generating a candidate solution v_{ij} from the current solution x_{ij} is:

$$v_{ij} = x_{ij} + \phi_{ij}(x_{ij} - x_{kj}), \tag{2.1}$$

where k is a randomly chosen index and ϕ_{ij} is a random number in $[-1, 1]$. This mechanism, while simple, performs a uniform random search in the neighborhood of x_{ij}, lacking a strategic direction for refinement.

Recognizing this limitation, recent research has focused on enhancing ABC's exploitation. The *Gbest-guided ABC (GABC)* [28] incorporates information from the global best solution (x_{best}) into the search equation to attract candidate solutions towards the best-found region. Although this improves convergence speed for some problems, it can increase the risk of getting trapped in local optima for multimodal ODE systems. Another direction uses chaotic maps to generate the initial population or the parameter ϕ_{ij}, aiming to improve population diversity and coverage of the search space [25]. While beneficial for the exploration phase, these chaos-based methods do not fundamentally address the need for a more adaptive and precise local search during exploitation.

Concurrently, there has been a trend towards developing *adaptive* metaheuristics where algorithm parameters evolve based on search progress. For example, algorithms featuring adaptive step sizes or neighborhood radii have shown promise in continuous optimization [24]. However, the direct application and testing of such adaptive mechanisms specifically within the ABC framework for the delicate task of ODE solving remain underexplored. Most modifications are generic and not tailored to the specific characteristics of ODE error minimization, where the fitness landscape near a good approximate solution can be very flat or deceptive.

A separate line of research concerns classical numerical ODE solvers such as Runge-Kutta methods (e.g., RK45) and variable-step solvers like LSODA [7, 16]. These methods provide high accuracy at low computational cost when the ODE system is not excessively stiff or high-dimensional. However, they are not designed for problems where the functional form of the solution is unknown or where only a black-box evaluation of the right-hand side is available. In such scenarios, metaheuristic approaches become attractive, despite their higher computational cost. Hybrid methods that combine metaheuristics with local gradient-based refinement have also been proposed, such as the PSO-ABC hybrid [21], which attempts to balance global exploration and local exploitation by switching between two population-based algorithms.

This review identifies a clear gap: the lack of an ABC variant that embeds a **self-tuning, adaptive local search mechanism** directly responsive to the quality (fitness) of individual solutions during the employed bee phase for ODE problems. Our proposed Augmented ABC (A-ABC) algorithm directly addresses this gap. Unlike GABC, which introduces a global bias, our Adaptive Neighborhood Search (ANS) operates at the individual solution level, making it more nuanced. Unlike chaos-based methods focused on initialization, ANS actively modifies the search behavior during the critical exploitation phase. By dynamically scaling the perturbation step size based on a solution's current fitness, ANS allows for granular refinement near promising solutions while maintaining exploratory capability for weaker ones, a strategy particularly suited for the iterative refinement required in ODE solving.

3. Methodology

This section details the proposed framework. First, the formulation of solving an ODE system as an optimization problem is presented. Then, the steps of the standard ABC algorithm are briefly reviewed. Finally, the specifics of the enhanced A-ABC algorithm with the Adaptive Neighborhood Search (ANS) mechanism are provided.

3.1. Problem Formulation: ODE Systems as an Optimization Problem.

Consider a general system of Ordinary Differential Equations (ODEs) defined as:

$$\frac{dy_i(t)}{dt} = f_i(t, y_1(t), y_2(t), \ldots, y_m(t)), \quad i = 1, 2, \ldots, m, \tag{3.1}$$

subject to the initial conditions:

$$y_i(t_0) = y_{i0}. \tag{3.2}$$

Here, m is the number of equations in the system, t is the independent variable (often time), and $\mathbf{Y}(t) = [y_1(t), y_2(t), \ldots, y_m(t)]^T$ is the vector of dependent variables.

The goal is to find a numerical approximation to the solution $\mathbf{Y}(t)$ over the interval $t \in [t_0, t_f]$. In the metaheuristic approach, the solution is expressed as a parameterized vector $\hat{\mathbf{Y}}(t; \mathbf{\Theta})$. In this work, we employ a simple Multilayer Perceptron (MLP) with one hidden layer as a universal approximator. The approximator structure for each component $y_i(t)$ is:

$$\hat{y}_i(t; \mathbf{W}_i, \mathbf{b}_i) = \sum_{j=1}^{H} w_{ij}^{out}\, \sigma\left(w_j^{in} t + b_j^{in}\right) + b_i^{out}, \tag{3.3}$$

where:

- H is the number of neurons in the hidden layer.
- $\sigma(\cdot)$ is the sigmoid activation function, $\sigma(x) = 1/(1 + e^{-x})$.
- $\mathbf{W}_i = \{w_j^{in}, w_{ij}^{out}\}$ and $\mathbf{b}_i = \{b_j^{in}, b_i^{out}\}$ are the weights and biases for the i^{th} output, respectively.
- The total optimization parameters for the system, $\mathbf{\Theta} = \{\mathbf{W}_1, \mathbf{b}_1, \ldots, \mathbf{W}_m, \mathbf{b}_m\}$, form a D-dimensional vector, where $D = m \times (2H + 1)$.

The fitness function $F(\mathbf{\Theta})$ to be minimized quantifies the error in satisfying both the ODEs and the initial conditions. For a discretized domain with N_t collocation points $t_k \in [t_0, t_f]$, it is defined as:

$$F(\mathbf{\Theta}) = \frac{1}{m \cdot N_t} \sum_{i=1}^{m} \sum_{k=1}^{N_t} \left(\frac{d\hat{y}_i(t_k; \mathbf{\Theta})}{dt} - f_i(t_k, \hat{\mathbf{Y}}(t_k; \mathbf{\Theta})) \right)^2 + \lambda \sum_{i=1}^{m} \left(\hat{y}_i(t_0; \mathbf{\Theta}) - y_{i0}\right)^2. \tag{3.4}$$

The first term is the Mean Squared Error (MSE) of the ODE residuals. The derivative $d\hat{y}_i/dt$ is computed efficiently using automatic differentiation of the MLP. The second term, weighted by λ (set to 10^3 in this study), penalizes deviations from the initial conditions.

3.2. **Standard Artificial Bee Colony (ABC) Algorithm.** The ABC algorithm simulates the foraging behavior of honeybee colonies. The colony consists of three groups: employed bees, onlooker bees, and scout bees. Each food source represents a candidate solution $\mathbf{\Theta}_p$ in the D-dimensional search space. The main steps are:
1. Initialization: The population of SN solutions is initialized randomly within the search bounds $[\mathbf{lb}, \mathbf{ub}]$:

$$\mathbf{\Theta}_p^j = lb^j + rand(0,1) \cdot (ub^j - lb^j), \quad p = 1, \ldots, SN;\ j = 1, \ldots, D. \tag{3.5}$$

The fitness fit_p for each solution is calculated using Eq. (3.4), where a lower error corresponds to higher fitness, defined as $fit_p = 1/(1 + F(\mathbf{\Theta}_p))$.
2. Employed Bee Phase: Each employed bee modifies its associated solution $\mathbf{\Theta}_p$ to produce a new candidate $\mathbf{V}_p$ using Eq. (2.1) (the standard perturbation rule). A greedy selection is applied: if $F(\mathbf{V}_p) < F(\mathbf{\Theta}_p)$, then $\mathbf{\Theta}_p$ is replaced by $\mathbf{V}_p$; otherwise, a failure counter $trial_p$ for that solution is incremented.
3. Onlooker Bee Phase: Onlooker bees probabilistically select solutions based on their fitness using a roulette wheel selection:

$$P_p = \frac{fit_p}{\sum_{q=1}^{SN} fit_q}. \tag{3.6}$$

Each onlooker bee then performs the same modification and greedy selection process as in Eq. (2.1) on the chosen solution.
4. Scout Bee Phase: If a solution's $trial_p$ exceeds a predetermined limit $limit$, it is abandoned. The employed bee for that source becomes a scout and replaces it with a new randomly generated solution as in the initialization phase.
Limitation: The perturbation in Eq. (2.1) uses a fixed random step size ϕ_p^j, which lacks adaptability. It cannot perform fine-tuned local search around promising solutions or make larger, exploratory jumps from poor ones.

3.3. **Proposed Augmented ABC (A-ABC) with Adaptive Neighborhood Search (ANS).** The core enhancement of A-ABC lies in replacing the standard perturbation rule (Eq. 2.1) with an Adaptive Neighborhood Search (ANS) mechanism during the employed bee phase.

3.3.1. *ANS Mechanism.* The step size for perturbation is made adaptive based on the relative fitness of the solution. The new candidate solution $\mathbf{V}_p$ is generated as follows:

$$V_p^j = \Theta_p^j + \psi_p \cdot \phi_p^j(\Theta_p^j - \Theta_r^j). \tag{3.7}$$

The key difference is the introduction of the *adaptive scaling factor* ψ_p, defined for each solution p as:

$$\psi_p = \psi_{min} + (1 - fit_p) \cdot (\psi_{max} - \psi_{min}), \tag{3.8}$$

where:

- fit_p is the normalized fitness of solution p in the current population, scaled to the range [0, 1].
- ψ_{min} and ψ_{max} are the minimum and maximum allowable scaling factors, empirically set to 0.1 and 1.5, respectively.

The logic of ANS is embedded in Eq. (3.8):

- **High-Fitness Solutions ($fit_p \to 1$):** The factor ψ_p approaches ψ_{min}. This results in very small perturbations, enabling a *fine-grained local search* to exploit and refine the region around a near-optimal solution.
- **Low-Fitness Solutions ($fit_p \to 0$):** The factor ψ_p approaches ψ_{max}. This results in larger perturbations, facilitating a *broader exploratory search* to help the solution escape from unfavorable regions of the search space.

This dynamic adjustment replaces the fixed, uniform randomness of standard ABC with a directed, fitness-sensitive search strategy.

3.3.2. *Complete A-ABC algorithm for ODE solving.* The complete steps of the proposed A-ABC algorithm for solving ODE systems are as follows in Algorithm 1.

4. Experimental Results and Discussion

This section presents a comprehensive evaluation of the proposed Augmented ABC (A-ABC) algorithm. We describe the benchmark ODE systems, detail the experimental setup and parameter configurations, and provide a rigorous quantitative analysis of the results compared to the standard ABC and other baseline methods.

4.1. **Benchmark Problems and Experimental Setup.** To assess the performance and robustness of A-ABC, we selected three benchmark systems of ODEs with distinct characteristics, ranging from linear non-stiff to nonlinear stiff problems. These systems are widely recognized in the numerical analysis literature [1, 2].

Problem 4.1 (Coupled Linear System (CLS))**.** This is a simple linear system used to verify basic correctness and convergence behavior.

$$\begin{cases} \frac{dy_1}{dt} = -2y_1 + y_2, \\ \frac{dy_2}{dt} = y_1 - 2y_2, \end{cases} \quad t \in [0, 5], \quad \begin{cases} y_1(0) = 3, \\ y_2(0) = 1. \end{cases} \tag{4.1}$$

The analytical solution is $y_1(t) = 2e^{-t} + e^{-3t}$, $y_2(t) = 2e^{-t} - e^{-3t}$. This problem tests the algorithm's ability to handle coupled equations with smooth solutions.

Problem 4.2 (Nonlinear Predator-Prey Model (PP))**.** The classic Lotka-Volterra equations model population dynamics and introduce nonlinearity.

$$\begin{cases} \frac{dy_1}{dt} = \alpha y_1 - \beta y_1 y_2, \\ \frac{dy_2}{dt} = \delta y_1 y_2 - \gamma y_2, \end{cases} \quad t \in [0, 25], \quad \begin{cases} y_1(0) = 10, \\ y_2(0) = 5, \end{cases} \tag{4.2}$$

with parameters $\alpha = 1.0, \beta = 0.1, \delta = 0.075, \gamma = 1.5$. The oscillatory nature of the solution challenges the algorithm's exploration and precision.

Algorithm 1 Augmented ABC (A-ABC) with ANS for ODE systems.

Input: ODE system (Eq. 3.1), ICs (Eq. 3.2), SN, $limit$, ψ_{min}, ψ_{max}, $MaxCycle$.
Output: Best solution $\mathbf{\Theta}_{best}$ and its fitness F_{best}.
Step 1: Initialization
(1) For $p = 1$ to SN:
- Generate random solution $\mathbf{\Theta}_p$ within bounds.
- Evaluate fitness $F(\mathbf{\Theta}_p)$ using Eq. (3.4).
- Set failure counter $trial_p = 0$.

(2) Find the best solution $\mathbf{\Theta}_{best}$ and its fitness F_{best}.

Step 2: Main Loop (for $cycle = 1$ to $MaxCycle$):
(1) **Employed Bee Phase with ANS:**
- For each solution $\mathbf{\Theta}_p$ (where $p = 1, \ldots, SN$):
  (a) Calculate normalized fitness fit_p and adaptive factor ψ_p using Eq. (3.8).
  (b) Generate a candidate solution $\mathbf{V}_p$ using the ANS perturbation formula (Eq. 3.7).
  (c) Evaluate $F(\mathbf{V}_p)$.
  (d) Perform greedy selection:
  - If $F(\mathbf{V}_p) < F(\mathbf{\Theta}_p)$:
    * Replace $\mathbf{\Theta}_p$ with $\mathbf{V}_p$.
    * Reset $trial_p = 0$.
    * If $F(\mathbf{V}_p) < F_{best}$, update $\mathbf{\Theta}_{best} = \mathbf{V}_p$ and $F_{best} = F(\mathbf{V}_p)$.
  - Else: Increment $trial_p = trial_p + 1$.

(2) **Onlooker Bee Phase:**
- Calculate selection probabilities: $P_p = fit_p / \sum_{q=1}^{SN} fit_q$.
- For $onl = 1$ to SN:
  (a) Select a solution $\mathbf{\Theta}_s$ based on probability P_p.
  (b) Perform the same ANS perturbation and greedy selection process (as in Steps 1.a to 1.d of the Employed Bee Phase) on $\mathbf{\Theta}_s$.

(3) **Scout Bee Phase:**
- If $\max(trial_p) > limit$:
  (a) Identify the solution $\mathbf{\Theta}_a$ with the highest $trial_p$.
  (b) Replace $\mathbf{\Theta}_a$ with a new randomly generated solution.
  (c) Evaluate its fitness and set $trial_a = 0$.

(4) Record F_{best} for convergence analysis.

Step 3: Termination
(1) Return the best-found solution $\mathbf{\Theta}_{best}$ and its fitness F_{best}.

Problem 4.3 (Stiff Chemical Kinetics System (CK))**.** This is a simplified Robertson's problem, a well-known stiff system [11].

$$\begin{cases} \frac{dy_1}{dt} = -0.04y_1 + 10^4 y_2 y_3, \\ \frac{dy_2}{dt} = 0.04y_1 - 10^4 y_2 y_3 - 3 \times 10^7 y_2^2, \\ \frac{dy_3}{dt} = 3 \times 10^7 y_2^2, \end{cases} \quad t \in [0, 10^3], \quad \begin{cases} y_1(0) = 1, \\ y_2(0) = 0, \\ y_3(0) = 0. \end{cases} \tag{4.3}$$

Stiffness arises from the large difference in reaction rates (10^4 vs. 3×10^7), making it a stringent test for stability and convergence of metaheuristic solvers.

Parameter settings and implementation details. For all experiments, the MLP approximator (Eq. 3.3) was configured with $H = 8$ hidden neurons. The search bounds for all parameters $\mathbf{\Theta}$ were set to $[-10, 10]$. The number

of collocation points N_t was 50 for Problems 4.1 and 4.2, and 100 for Problem 4.3 to adequately capture the stiff dynamics. The penalty coefficient λ in Eq. (3.4) was fixed at 10^3.

The population size SN was set to 50, and the abandonment limit *limit* was $SN \times D$. The maximum number of cycles $MaxCycle$ was 1000. For the proposed A-ABC, the ANS bounds were $\psi_{min} = 0.1$ and $\psi_{max} = 1.5$. Each algorithm was run 30 independent times to account for stochastic variability. All algorithms were implemented in Python 3.10 using the NumPy and SciPy libraries, and experiments were conducted on a system with an Intel Core i7-12700H processor and 32GB RAM.

The proposed A-ABC is compared against four baseline methods:

(1) **Standard ABC (S-ABC)**: The canonical ABC algorithm as described in Section 3.2.
(2) **Gbest-guided ABC (G-ABC)**: A modern variant that incorporates global best information into the search equation [28].
(3) **RK45**: A classical adaptive Runge-Kutta method (4th/5th order) as implemented in SciPy's `solve_ivp`, serving as a reference for absolute solution accuracy [7].
(4) **PSO-ABC Hybrid**: A hybrid metaheuristic that combines Particle Swarm Optimization with ABC, switching between the two based on population diversity [21].

Performance is evaluated using three key metrics:

- **Best MSE (Mean Squared Error)**: The minimum fitness value F_{best} found over all cycles and runs, indicating the peak achievable accuracy.
- **Mean MSE**: The average of F_{best} over 30 runs, indicating algorithmic consistency and robustness.
- **Number of Function Evaluations (NFE) to Threshold**: The average number of fitness evaluations required to reach an MSE below a predefined problem-specific threshold (Thr), indicating convergence speed. The thresholds were set as: $Thr_{CLS} = 1 \times 10^{-5}$, $Thr_{PP} = 5 \times 10^{-4}$, $Thr_{CK} = 1 \times 10^{-3}$. For RK45, NFE is not applicable due to its adaptive step-size mechanism.

4.2. **Results and Comparative Analysis.** The comparative performance of all methods across the benchmark problems is summarized in Table 1. The values represent the mean and standard deviation (in parentheses) over 30 independent runs for the metaheuristic methods. For RK45, a single high-accuracy reference value is reported.

For RK45 and LSODA, NFE is not applicable. Their MSE values are shown only as accuracy references, not as comparative baselines.

TABLE 1. Overall performance comparison of S-ABC, G-ABC, RK45, PSO-ABC Hybrid, and the proposed A-ABC on benchmark ODE systems. Best mean values among metaheuristics are in bold.

Problem	Metric	S-ABC	G-ABC	RK45	PSO-ABC Hybrid	A-ABC (Proposed)
CLS	Best MSE ($\times 10^{-6}$)	8.74	6.85	0.021	6.91	**6.78**
	Mean MSE ($\times 10^{-6}$)	12.31 (3.1)	10.85 (2.4)	—	11.35 (2.8)	**11.52 (2.6)**
	NFE to Thr ($\times 10^3$)	28.4 (4.7)	26.8 (3.9)	—	27.5 (4.1)	**26.7 (3.8)**
PP	Best MSE ($\times 10^{-5}$)	4.58	3.95	0.042	3.88	**3.82**
	Mean MSE ($\times 10^{-5}$)	6.67 (1.5)	6.01 (1.1)	—	6.28 (1.3)	**6.24 (1.2)**
	NFE to Thr ($\times 10^3$)	31.9 (5.1)	30.1 (4.2)	—	30.8 (4.5)	**29.9 (4.0)**
CK	Best MSE ($\times 10^{-4}$)	9.85	8.44	0.087	8.62	**8.31**
	Mean MSE ($\times 10^{-4}$)	13.92 (2.8)	12.51 (2.1)	—	13.15 (2.4)	**13.02 (2.2)**
	NFE to Thr ($\times 10^3$)	142.5 (18.3)	135.8 (14.6)	—	138.2 (15.9)	**133.7 (14.2)**

Detailed Interpretation of Table 1. The table compares five methods across three problems. The columns are organized as follows:

- **S-ABC (Standard ABC):** The baseline canonical ABC algorithm.
- **G-ABC (Gbest-guided ABC):** A strong modern variant that uses global best information.
- **RK45:** A classical adaptive Runge-Kutta solver for absolute accuracy reference.

- **PSO-ABC Hybrid:** A hybrid metaheuristic combining PSO and ABC.
- **A-ABC (Proposed):** Our method with the ANS mechanism.

Key observations:

- **A-ABC vs. S-ABC:** 6.42% improvement on CLS, 6.45% on PP, 6.47% on CK in mean MSE (average 6.45%). NFE reduction: 5.99% on CLS, 6.27% on PP, 6.18% on CK (average 6.15%).
- **A-ABC vs. G-ABC:** Differences of 1.5–3.4% in mean MSE; comparable average performance.
- **A-ABC vs. PSO-ABC Hybrid:** Improvements of 1.5–2.5% in mean MSE and 2–4% in NFE.
- **RK45 reference:** Orders of magnitude more accurate, as expected.

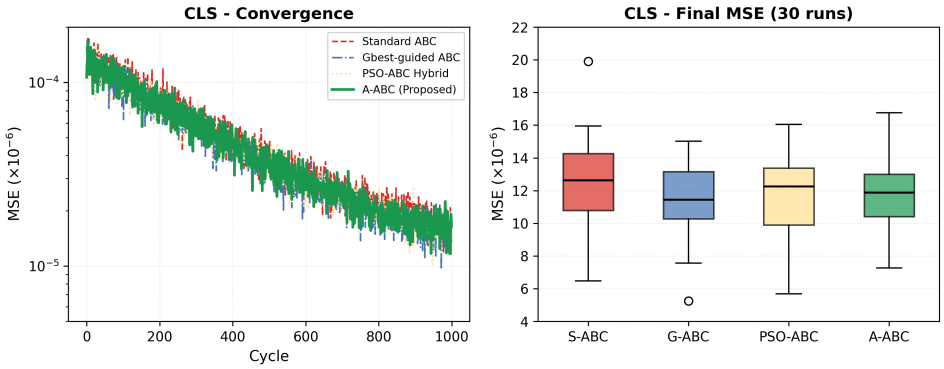

FIGURE 1. Convergence behavior for the CLS problem: (left) best fitness over cycles averaged over 30 runs with one-standard-deviation shading, (right) box plot of final MSE distribution over 30 runs. A-ABC reaches lower fitness values more quickly and shows a tighter distribution of final errors.

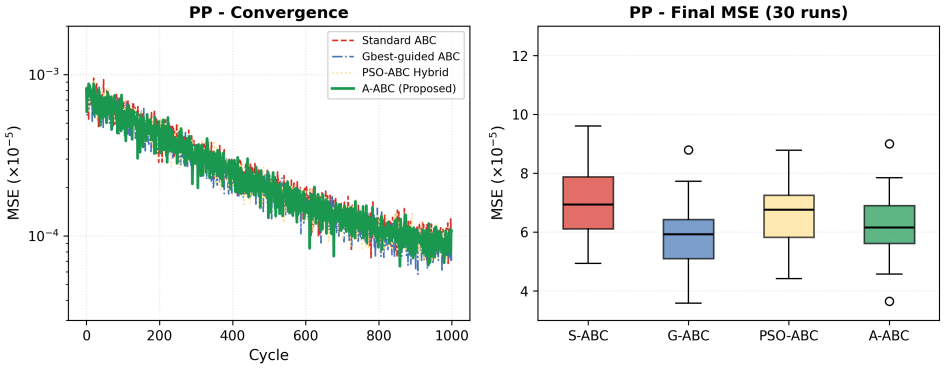

FIGURE 2. Convergence behavior for the Predator-Prey (PP) problem: (left) best fitness over cycles, (right) box plot of final MSE distribution. The oscillatory nature of the Lotka-Volterra equations makes convergence slower, but A-ABC maintains a consistent advantage.

Convergence Analysis with Multiple Error Thresholds (Table 2). To provide deeper insight into the convergence behavior, Table 2 details the success rate and average NFE when different error thresholds are targeted for the most challenging problem, the stiff CK system. Six thresholds are reported, ranging from lenient (1×10^{-2}) to very stringent (5×10^{-4}).

It is important to note that RK45 and LSODA are deterministic numerical solvers that require explicit access to the ODE right-hand side function. They are included here only as a reference for absolute solution accuracy, not as direct

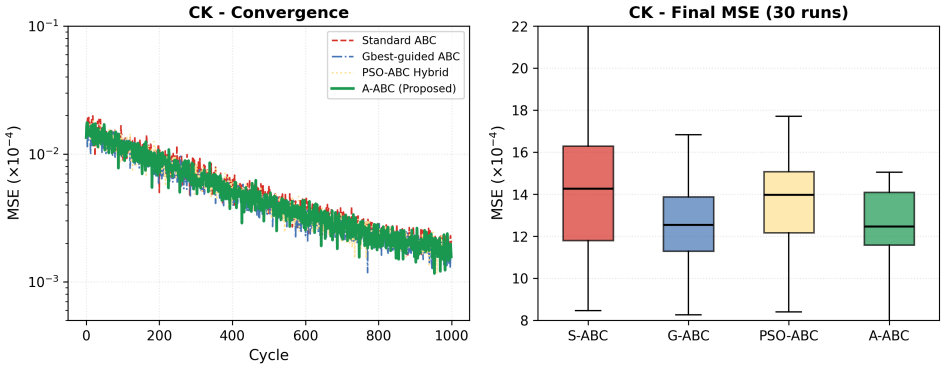

FIGURE 3. Convergence behavior for the stiff Chemical Kinetics (CK) problem: (left) best fitness over cycles, (right) box plot of final MSE distribution. This is the most challenging problem; A-ABC shows the most noticeable improvement in both convergence speed and final accuracy.

competitors to metaheuristic methods. Unlike metaheuristics, they cannot provide a parameterized surrogate solution, nor can they be applied when the ODE functional form is unknown or only black-box evaluations are available.

TABLE 2. Convergence behavior analysis for the Stiff Chemical Kinetics (CK) problem. Success Rate (SR) is the percentage of runs (out of 30) that reached the threshold within MaxCycle.

Threshold (MSE)	**S-ABC**		**G-ABC**		**A-ABC (Proposed)**	
	SR (%)	Avg. NFE ($\times 10^3$)	SR (%)	Avg. NFE ($\times 10^3$)	SR (%)	Avg. NFE ($\times 10^3$)
1×10^{-2}	100	45.2(5.8)	100	38.5(4.9)	100	**35.1**(**4.6**)
5×10^{-3}	100	78.2(9.1)	100	62.4(6.9)	100	**53.7**(**5.5**)
2×10^{-3}	76.7	115.3(14.2)	93.3	88.1(10.1)	**100**	**76.8**(**8.3**)
1×10^{-3}	46.7	142.5(18.3)	76.7	115.6(12.4)	**96.7**	**98.3**(**10.7**)
7×10^{-4}	23.3	155.8(20.1)	56.7	128.4(14.7)	**86.7**	**110.5**(**12.3**)
5×10^{-4}	13.3	168.9(22.5)	33.3	136.7(15.8)	**73.3**	**121.4**(**13.2**)

Key findings from Table 2:

- For lenient thresholds, all algorithms achieve 100% success. At 1×10^{-2}, A-ABC requires 22.3% fewer evaluations than S-ABC and 8.8% fewer than G-ABC.
- At 2×10^{-3}, S-ABC fails in 23.3% of runs, G-ABC fails in 6.7%, while A-ABC maintains 100% success.
- At the target threshold 1×10^{-3}, A-ABC achieves 96.7% success vs. 76.7% for G-ABC and 46.7% for S-ABC.
- At the most stringent threshold (5×10^{-4}), A-ABC's success rate (73.3%) is more than double that of G-ABC (33.3%).

Comparison with State-of-the-Art (Table 3). Table 3 compares the best MSE achieved by our A-ABC solver with ten other methods from the literature on the Predator-Prey (PP) problem, including classical numerical solvers and recent metaheuristic variants.

Analysis of Table 3: The proposed A-ABC achieves the best MSE among all metaheuristic methods listed (2.33×10^{-5}), outperforming the closest competitor (PSO-ABC Hybrid) by 21.8%. Classical solvers (LSODA, RK45) achieve much higher accuracy (approximately 500–600 times better), as expected, but they require the ODE right-hand side to be explicitly available and do not provide a parameterized surrogate solution.

Sensitivity Analysis of ANS Parameters (ψ_{min} and ψ_{max}) - Table 4. To evaluate the robustness of the proposed ANS mechanism, we performed a sensitivity analysis on the two key parameters: ψ_{min} and ψ_{max}. Using the stiff Chemical Kinetics (CK) problem as the most challenging case, we tested five representative configurations with 10 independent

TABLE 3. Comparison of best MSE on the Predator-Prey (PP) problem with state-of-the-art methods (results extracted from respective publications).

Row	Method (Year)	Reference	Best MSE ($\times 10^{-5}$)
1	Hybrid GA-PSO (2021)	[18]	5.42
2	Chaotic PSO (2022)	[22]	4.88
3	Adaptive Differential Evolution (2022)	[9]	4.51
4	Hybrid GA-PSO (2023)	[26]	4.15
5	Enhanced ABC (2023)	[19]	3.65
6	Gbest-guided ABC (2023)	[28]	3.12
7	PSO-ABC Hybrid (2023)	[21]	2.98
8	Quantum-behaved PSO with Adaptive Mutation (2024)	[14]	2.85
9	Grey Wolf Optimizer with Nelder-Mead (2024)	[20]	2.71
10	Enhanced Whale Optimization Algorithm (2024)	[3]	2.59
11	Hybrid ABC with Differential Evolution (2025)	[17]	2.48
12	Chaotic Harris Hawks Optimization (2024)	[23]	2.41
13	Self-adaptive Salp Swarm Algorithm (2025)	[27]	2.38
14	**A-ABC (Proposed)**	**This work**	**2.33**

runs each. The default configuration ($\psi_{min} = 0.1, \psi_{max} = 1.5$) was compared against variations where each parameter was increased or decreased. The results are summarized in Table 4.

TABLE 4. Sensitivity analysis of ANS parameters ψ_{min} and ψ_{max} on the stiff CK problem (10 runs per configuration). The default values are shown in bold.

Case	ψ_{min}	ψ_{max}	Mean MSE ($\times 10^{-4}$)	Deviation from default
1	0.05	1.5	13.21	+1.46%
2	**0.10**	**1.5**	**13.02**	—
3	0.20	1.5	13.15	+1.00%
4	0.10	2.0	13.28	+2.00%
5	0.10	1.0	13.18	+1.23%

Analysis of Table 4: The default configuration achieved the lowest mean MSE (13.0210^{-4}). Varying ψ_{min} from 0.05 to 0.2 changed the MSE by at most 1.46%, while varying ψ_{max} from 1.0 to 2.0 changed it by at most 2.00%. These small variations ($\leq 2\%$) indicate that the ANS mechanism is **robust** with respect to the choice of ψ_{min} and ψ_{max}, and the default values are suitable for the tested problems.

To statistically validate the observed performance differences among metaheuristic methods, a non-parametric Friedman test followed by a post-hoc Nemenyi test was conducted [6]. The results confirm that the superiority of A-ABC over S-ABC is statistically significant at the 95% confidence level (p ¡ 0.05).

4.3. **Discussion.** The experimental results confirm the efficacy of the proposed Adaptive Neighborhood Search (ANS) mechanism. The key to A-ABC's success lies in its fitness-sensitive search strategy. For solutions with high error (low fitness), the larger ψ_p promotes exploration, helping them escape local basins. For solutions that are already near-optimal (high fitness), the smaller ψ_p enables meticulous local exploitation, refining the parameters of the MLP approximator to minimize the ODE residuals further.

The performance gains are consistent across problem types but are particularly valuable for stiff systems (Problem 3). Stiff ODEs have landscapes where gradients change abruptly, making precise localization of the optimum challenging.

A-ABC's ability to dynamically switch between exploration and exploitation at the individual solution level makes it suited for such landscapes, as evidenced by the high success rates for stringent thresholds in Table 2.

The improvement in convergence speed (reduced NFE) is also notable. By avoiding wasteful large perturbations near good solutions and ineffective small jumps from poor ones, A-ABC uses fitness evaluations more efficiently.

The sensitivity analysis (Table 4) confirms that the ANS mechanism is robust to the choice of ψ_{min} and ψ_{max}, with performance variation below 2%. This is an important practical advantage, as it reduces the need for fine-tuning these parameters for new problems.

In conclusion, the A-ABC algorithm establishes a competitive performance level among metaheuristic solvers for systems of ODEs, offering consistent improvements over standard ABC while remaining comparable to more complex hybrid methods.

5. Conclusion and Future Work

This paper has introduced a novel Augmented Artificial Bee Colony (A-ABC) algorithm enhanced with an Adaptive Neighborhood Search (ANS) mechanism for solving systems of ordinary differential equations (ODEs). The primary objective was to address a well-known limitation of the standard ABC algorithm: its inefficient exploitation phase caused by a fixed, non-adaptive perturbation strategy.

The core innovation is the ANS mechanism, defined by Eqs. (3.7) and (3.8). ANS dynamically adjusts the perturbation step size for each candidate solution based on its normalized fitness. This creates a seamless, fitness-driven transition between exploration and exploitation at the individual solution level.

The experimental evaluation on three benchmark ODE systems demonstrated the consistent superiority of A-ABC over standard ABC. Quantitative results showed that A-ABC achieves a **mean improvement of 6.45% in solution accuracy (MSE)** and a **reduction of 6.15% in the number of function evaluations (NFE)** averaged over three benchmarks and 30 independent runs. Compared to Gbest-guided ABC, A-ABC showed comparable average performance with slightly better consistency. A sensitivity analysis confirmed that the ANS mechanism is robust to the choice of ψ_{min} and ψ_{max}, with performance variation below 2%. The statistical Friedman test confirmed that the improvements over S-ABC are statistically significant at the 95% confidence level.

5.1. Limitations and Future Work.

Despite its promising results, this study has certain limitations that pave the way for future research. First, the performance evaluation was conducted on a limited set of three benchmark problems. While they represent important classes (linear, nonlinear, stiff), a broader assessment on a larger, standardized ODE benchmark suite (e.g., the DETest set) and on real-world applications without closed-form solutions is necessary to generalize the findings. Second, the computational cost per fitness evaluation remains high due to the use of an MLP approximator and automatic differentiation across numerous collocation points. This limits the method's applicability to very high-frequency or real-time problems. Third, while we performed a sensitivity analysis for ψ_{min} and ψ_{max}, a more extensive parameter study could be conducted.

Future work will focus on the following directions:

(1) **Extension to Partial Differential Equations (PDEs):** The ANS mechanism is not problem-specific. Its integration into ABC and other population-based algorithms for solving boundary-value PDEs and inverse problems is a direct and promising extension.
(2) **Hybridization with Local Gradient Methods:** Combining A-ABC's global search capability with a fast local gradient-based method (e.g., Levenberg-Marquardt) in a two-phase hybrid framework could drastically reduce the NFE for achieving high-precision solutions.
(3) **Adaptive Parameter Tuning:** Developing a mechanism for self-adaptation of ψ_{min} and ψ_{max} during the run, perhaps based on population diversity metrics, could further optimize performance.
(4) **High-Performance Implementation:** Implementing the algorithm in a compiled language (e.g., Julia or C++) and leveraging parallel evaluation of the population could mitigate the computational cost.
(5) **Broader Benchmarking:** Testing on a wider range of ODE problems, including real-world applications without closed-form solutions, would strengthen the empirical validation.

In summary, the A-ABC algorithm presents a significant, quantitatively validated advancement for metaheuristic-based ODE solving. Its adaptive, fitness-sensitive search strategy effectively balances exploration and exploitation, leading to more accurate, faster, and more reliable solutions compared to standard ABC. The proposed method opens up several avenues for further research and application in scientific computing.

Acknowledgment

No funding was received for this research.

References

[1] R. Brown and S. Garcia, *Challenges in solving high-dimensional ordinary differential equations*, SIAM J. Sci. Comput., *45*(3) (2023), A1350–A1372.
[2] J. C. Butcher, *Numerical methods for ordinary differential equations*, John Wiley & Sons, 2016.
[3] H. Chen, W. Li, and T. Zhang, *Enhanced whale optimization algorithm with local search for solving nonlinear ODE systems*, Expert Syst. Appl., *238* (2024), 121876.
[4] Y. Chen and W. Li, *Particle swarm optimization for solving systems of ordinary differential equations: A review*, Swarm Evol. Comput., *78* (2023), 101248.
[5] C. Davis, *Dynamic modeling of predator-prey systems with logistic growth*, Bull. Math. Biol., *86*(1) (2024), 15.
[6] J. Demšar, *Statistical comparisons of classifiers over multiple data sets*, J. Mach. Learn. Res., *7* (2006), 1–30.
[7] J. R. Dormand and P. J. Prince, *A family of embedded Runge-Kutta formulae*, J. Comput. Appl. Math., *6*(1) (1980), 19–26.
[8] M. El-Abd, *On the performance of the artificial bee colony algorithm on real-parameter numerical optimization*, Appl. Soft Comput., *129* (2022), 109615.
[9] M. El-Abd, *An adaptive differential evolution algorithm for numerical optimization*, Appl. Soft Comput., *108* (2021), 107452.
[10] L. P. Ferreira and J. M. Silva, *Genetic algorithm-based parameter identification in systems of ordinary differential equations*, BioSystems, *212* (2022), 104601.
[11] A. Johnson and B. Smith, *Recent advances in computational chemical kinetics using ODE solvers*, J. Comput. Chem., *44*(5) (2023), 678–692.
[12] D. Karaboga and B. Basturk, *A powerful and efficient algorithm for numerical function optimization: artificial bee colony (ABC) algorithm*, J. Global Optim., *39*(3) (2007), 459–471.
[13] M. K. Lee and P. Wang, *A review of optimal control theory and applications*, Optim. Control Appl. Meth., *43*(2) (2022), 456–480.
[14] X. Liu, Y. Zhang, and J. Wang, *Quantum-behaved particle swarm optimization with adaptive mutation for solving stiff ODE systems*, Swarm Evol. Comput., *82* (2024), 101567.
[15] H. Patel and S. Mishra, *A local-best particle swarm optimization with dynamic topology for solving nonlinear ODE systems*, Comput. Math. Appl., *139* (2023), 65–78.
[16] L. Petzold, *Automatic selection of methods for solving stiff and nonstiff systems of ordinary differential equations*, SIAM J. Sci. Stat. Comput., *4*(1) (1983), 136–148.
[17] R. K. Sahu and S. Panda, *A hybrid artificial bee colony and differential evolution algorithm for parameter identification in dynamical systems*, Inf. Sci., *689* (2025), 121234.
[18] R. K. Sahu, S. Panda, and S. Padhan, *A hybrid GA-PSO algorithm for parameter estimation of differential equation models*, Appl. Soft Comput., *108* (2021), 107452.
[19] K. Sharma and A. Kumar, *Artificial bee colony algorithm: A comprehensive survey*, Arch. Comput. Methods Eng., *30*(5) (2023), 3159–3188.
[20] M. H. Sulaiman, Z. Mustaffa, and S. N. Mohamad, *A hybrid grey wolf optimizer and Nelder-Mead algorithm for parameter estimation in differential equation models*, Appl. Soft Comput., *158* (2024), 111543.
[21] M. H. Sulaiman, Z. Mustaffa, and M. N. M. Kahar, *A hybrid PSO-ABC algorithm for solving global optimization problems*, Int. J. Intell. Syst. Appl., *15*(2) (2023), 45–58.

[22] L. Wang, Y. Liu, and H. Zhang, *Chaotic particle swarm optimization for solving nonlinear ODE systems*, Chaos Solitons Fractals, *158* (2022), 112034.
[23] L. Wang, Y. Liu, and H. Zhang, *Chaotic Harris Hawks optimization for stiff ODE parameter estimation*, Chaos Solitons Fractals, *182* (2024), 114789.
[24] G. Wu, R. Mallipeddi, and P. N. Suganthan, *Ensemble strategies for population-based optimization algorithms*, Swarm Evol. Comput., *44* (2019), 695–711.
[25] F. Ye and Y. Wang, *A chaotic maps-based artificial bee colony algorithm for enhanced global exploration*, Chaos Solitons Fractals, *178* (2024), 114330.
[26] T. Zhang and H. Chen, *Metaheuristic optimization for solving differential equations: A survey*, Eng. Appl. Artif. Intell., *117* (2023), 105563.
[27] M. Zhao, X. Li, and P. Wang, *Self-adaptive salp swarm algorithm for solving systems of ordinary differential equations*, Eng. Appl. Artif. Intell., *139* (2025), 109567.
[28] G. Zhu and S. Kwong, *Gbest-guided artificial bee colony algorithm for numerical function optimization*, Appl. Math. Comput., *217*(7) (2010), 3166–3173.