

SHOOTING-INTERPOLATION ALGORITHM FOR SOLVING BOUNDARY VALUE PROBLEMS IN SECOND-ORDER SINGULAR ODEs

NATASHA CLARISSA MAHARANI, SAFRIZAL ARDANA ARDIYANSA,
MOHAMAD MUSLIKH* & UMMU HABIBAH

ABSTRACT

This article presents a novel Shooting-Interpolation framework for solving second-order singular boundary value problems (BVPs). The proposed method transforms the BVP into a system of first-order ordinary differential equations (ODEs) and reformulates it as an initial value problem, which is solved using the classical fourth-order Runge–Kutta (RK4) method. The unknown initial condition is iteratively updated through an interpolation-based shooting strategy to achieve boundary satisfaction without requiring analytical solutions. Theoretical analysis result confirms convergence under Lipschitz continuity assumptions, stability under perturbations in the shooting parameter, and fourth-order global accuracy due to the RK4 discretization. In addition, the interpolation mechanism ensures progressive refinement of the shooting parameter while maintaining numerical consistency. Complexity analysis shows quasi-linear time complexity $\mathcal{O}(N \cdot \text{iter}_{\max})$ and linear memory complexity $\mathcal{O}(N)$. Numerical results indicate that the proposed framework provides an accurate, stable, and efficient approach for solving nonlinear singular BVPs with Dirichlet, Neumann, and Robin boundary conditions.

Keywords: boundary value problems; shooting-interpolation algorithm; second-order singular ODEs.

1. Introduction

Many physical phenomena can be represented using mathematical models that generate differential equations (DEs) (Khresat *et al.* 2023). A DE is defined as a mathematical formulation that incorporates derivatives of an unknown function (Moruzzi 2020; Oliveira 2016). DEs describe the relationship between the rate of change of a continuously varying quantity and the influence of one variable on another (Gockenbach 2011). DEs have numerous applications in various scientific, technological, and engineering disciplines (Sahani *et al.* 2024), particularly in modeling relationships involving continuously varying quantities whose rates of change are represented by mathematical functions. DEs are fundamental in various fields such as physics, economics, engineering, and biology, where they are used to analyze and predict the behavior of dynamic systems and natural phenomena (Linge & Langtangen 2020), while also serving as the theoretical foundation for many scientific and technological models (Marjona 2024). Their applications include modeling temperature control in chemical reactions, mechanical systems, electrical circuits, diffusion processes, population dynamics, predator–prey interactions, and curvature analysis (Temple 1967; Abell & Braselton 2016). In addition, DE concepts have been integrated into metaheuristic algorithms (MAs) to improve optimization processes (Zhang *et al.* 2025), which are important for solving various complex optimization problems, including feature selection, classification performance improvement, and diagnosis optimization involving Random Forest, Support Vector Machine, XGBoost, and hybrid approaches (Anam *et al.* 2026; Ardiyansa *et al.* 2025; Fajri *et al.* 2025; Ardiyansa *et al.* 2024).

DEs can be categorized into two types: ordinary differential equations (ODEs) and partial differential equations (PDEs) (Gockenbach 2011). ODEs involve derivatives of an unknown function that depends on a single independent variable, and their solutions can be formulated as initial value problems (IVPs) or boundary value problems (BVPs) (Smirnov 1964; Madenci

et al. 2019). In contrast, PDEs involve unknown functions of multiple independent variables and their partial derivatives (Choi *et al.* 2024). PDEs are utilized to model multidimensional systems such as wave phenomena (Polyanin *et al.* 2008). However, solving PDEs is generally more complex than solving ODEs; therefore, various numerical and analytical techniques, such as homotopy perturbation (Mohyud-Din 2010) and separation of variables (Abell & Braselton 2016), can be utilized to transform certain PDE problems into ODEs (Latta 1961). These techniques can facilitate the derivation of solutions to PDEs (Rashed & Kassem 2023).

ODEs are used in various fields, and their solution is critical for various applications (Kelison & Appel 2022). However, ODEs with singularities lack closed-form analytical solutions due to their inherent mathematical complexity (Zhang *et al.* 2020). Singular ODEs are challenging to solve, even though they frequently emerge in physical problems such as compactons in nonlinear elastic rods (Dai *et al.* 2004). The singularities of ODE solutions exhibit regular spacing and dynamic movement along specific paths, introducing additional complexity. Consequently, numerical and semi-analytical methods are indispensable for solving differential and related integro-differential equations for which exact solutions are difficult to obtain. Various approaches have therefore been developed to obtain accurate approximate solutions for complex nonlinear problems, including homotopy-based methods for nonlinear multi-term fractional Volterra–Fredholm integro-differential equations (Eshkuvatov *et al.* 2025). Numerical methods offer several advantages, including reduced computational costs, improved accuracy, and enhanced efficiency (Rasedee *et al.* 2021). Numerical approaches also provide reliable and expedient solutions, making them well-suited for tackling singular ODEs (Ezhilarasi & Anbhuvizhi 2021). Common numerical techniques for addressing BVPs in ODEs include the shooting, finite difference, and collocation methods (Ascher *et al.* 1995).

The shooting method is a numerical approach for solving two-point BVPs in ODEs by iteratively estimating the initial conditions until the second boundary condition is satisfied (Filipov *et al.* 2017). This method can produce accurate numerical solutions for BVPs of ODEs with high convergence rates (Arefin *et al.* 2022). The shooting method can also effectively address unstable problems (Morrison *et al.* 1962). Previous research has demonstrated that the shooting method can effectively and efficiently solve BVPs derived from real-world phenomena (Edun & Akinlabi 2021). Additionally, Habibah *et al.* (2020) showed that the shooting method produced a better numerical approximation for the Cauchy–Euler DE than the finite-difference method due to its smaller numerical errors. The shooting method can solve various types of ODEs, including those with nonlinear or singular forms, as it does not require grouping coefficients into a system solvable by elementary row operations. Furthermore, the shooting method exhibits high convergence rates, providing accurate numerical solutions for BVPs in ODEs (Arefin *et al.* 2022). In contrast, finite difference methods have a significant drawback due to the complex stability treatment required at boundaries and interfaces (Svärd & Nordström 2013). Additionally, finite difference and collocation methods face challenges when applied to nonlinear singular ODEs, as the discretization process does not always result in a linear system that can be solved using elementary row operations.

However, the main limitation of the shooting method is its reliance on accurate initial guesses, which often need to be manually adjusted until the second boundary condition is satisfied. The conventional shooting method does not always guarantee successful solutions for BVPs because of possible difficulties in determining suitable initial guesses (Choi *et al.* 2021). Moreover, this trial-and-error process can be laborious and time-consuming. Therefore, improvements to the shooting method are necessary to determine suitable initial conditions more efficiently and systematically.

To address these limitations, this research develops a novel shooting-interpolation method by integrating interpolation techniques into the conventional shooting algorithm. The novelties of the proposed method are summarized as follows:

- (1) The development of a novel shooting-interpolation method that integrates interpolation techniques into the conventional shooting method for solving BVPs.

- (2) A systematic mechanism for approximating suitable initial conditions, reducing the dependence on repeated manual guessing.
- (3) The application of the proposed shooting-interpolation method to second-order singular ODEs with improved computational efficiency and fewer iterations compared to solve the conventional shooting method.

This article is organized as follows. Section 1 presents the introduction, outlining the motivation for the proposed framework. Section 2 describes the methodology. Section 3 presents the implementation of the algorithm for different types of boundary conditions, namely Dirichlet, Neumann, and Robin. Section 4 discusses the theoretical properties of the proposed algorithm, including convergence, stability, and error behavior. Section 5 analyzes the computational efficiency of the proposed algorithm in terms of time and memory complexity. Section 6 provides numerical examples to demonstrate the performance of the proposed algorithm. Finally, Section 7 concludes the paper and highlights possible directions for future work.

2. Research Method

The methodology adopted in this research follows a systematic sequence of stages as illustrated in the research flow diagram shown in Figure 1. The process begins with an in-depth literature review of the existing shooting method for solving BVPs of ODEs. This phase aims to establish a strong theoretical foundation by examining the mathematical properties of singular ODEs, the classical shooting technique and its limitations, as well as interpolation strategies commonly employed in numerical analysis.

Following the literature review stage, a novel shooting-interpolation algorithm is constructed in this article for solving BVPs of singular ODEs. The proposed framework transforms the original second-order singular ODE into an equivalent first-order system before applying numerical discretization. A novel shooting-interpolation algorithm is then formulated based on this framework. To discretize the continuous differential equations into a numerically solvable form, the fourth-order Runge–Kutta (RK4) method is employed in this study due to its higher accuracy and better numerical stability compared with lower-order explicit methods.

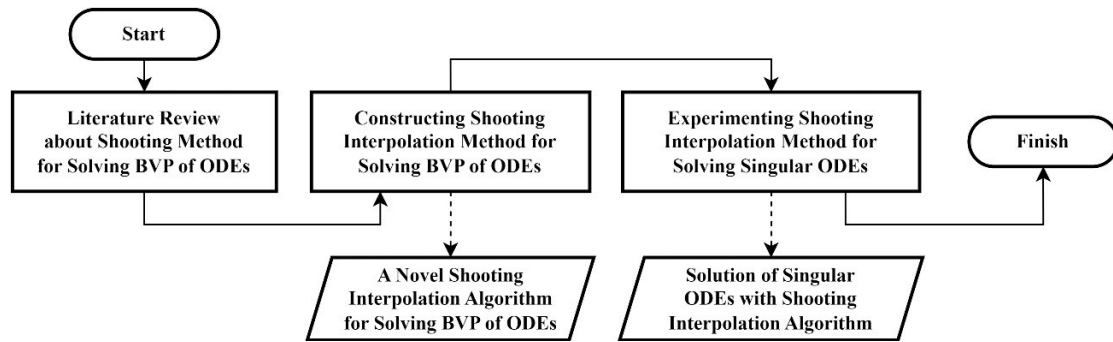


Figure 1: Research flow.

Special attention is given to the treatment of the singular point near $t \rightarrow 0$, since the governing equations contain singular coefficients of the form $1/t$ and $1/t^2$. Direct numerical evaluation at $t = 0$ may produce undefined operations and numerical instability. To overcome this problem, the computational domain is shifted slightly away from the singular point by replacing $t = 0$ with $t_0 = \varepsilon$, where ε is a small positive constant satisfying $0 < \varepsilon \ll 1$. Consequently, all singular coefficients are evaluated at $t + \varepsilon$ instead of t , such that:

$$\frac{1}{t} \rightarrow \frac{1}{t + \varepsilon}, \quad \frac{1}{t^2} \rightarrow \frac{1}{(t + \varepsilon)^2}. \quad (1)$$

This regularization eliminates division-by-zero singularities and transforms the original singular problem into a perturbed problem with bounded coefficients. In particular,

$$\lim_{t \rightarrow 0} \frac{1}{t + \varepsilon} = \frac{1}{\varepsilon}, \quad \lim_{t \rightarrow 0} \frac{1}{(t + \varepsilon)^2} = \frac{1}{\varepsilon^2}, \quad (2)$$

which remain finite for every ε . Therefore, the singular coefficients appearing in the governing equation become bounded throughout the computational domain.

Furthermore, the RK4 scheme improves numerical robustness near the singular region because its local truncation error is of order $\mathcal{O}(h^5)$ and its global truncation error is of order $\mathcal{O}(h^4)$. This higher-order accuracy reduces error propagation caused by the singular coefficients near the boundary. In addition, a sufficiently small step size h is selected to ensure numerical stability and to minimize the influence of the regularization parameter ε on the computed solution.

Once the algorithm is developed, it is tested on several singular ODE problems to evaluate its effectiveness, stability, and numerical accuracy. During this experimental phase, cubic spline interpolation is employed to estimate the unknown shooting parameters across iterations. This interpolation technique is selected because it provides smooth approximations with continuous first and second derivatives, thereby improving numerical stability and yielding more accurate estimates than linear interpolation methods.

3. Result and Discussion

This section explains the novel shooting-interpolation method for solving singular ODEs with Dirichlet, Neumann, and Robin boundary conditions. This section also presents the proposed algorithm.

3.1. Dirichlet Boundary Condition

This subsection explains the novel shooting-interpolation method for solving the Dirichlet boundary condition. Consider the following second-order singular ODE as shown in Eq. (3),

$$u''(t) + \frac{a_1}{t + \varepsilon} u'(t) + \frac{a_0}{(t + \varepsilon)^2} u(t) = g(t, u(t), u'(t)), \quad (3)$$

for $0 < a \leq t \leq b$ with Dirichlet boundary conditions $u(a) = \alpha$ and $u(b) = \beta$, where $\varepsilon > 0$ is a sufficiently small positive constant introduced to regularize the singularity near $t \rightarrow 0$. This regularization avoids undefined numerical operations caused by the terms $1/t$ and $1/t^2$ while preserving the asymptotic behavior of the original singular ODE.

The first step to solve the ODE in Eq. (3) is to convert it into a system of first-order singular ODEs by defining $u'(t) = v(t)$, resulting in the following system of ODEs:

$$\begin{aligned} u'(t) &= v(t), \\ v'(t) &= g(t, u(t), v(t)) - \frac{a_1}{t + \varepsilon} v(t) - \frac{a_0}{(t + \varepsilon)^2} u(t). \end{aligned} \quad (4)$$

The subsequent step involves discretizing the system of equations to facilitate numerical implementation. In this study, the RK4 method is employed to transform the continuous DEs into a discrete form. Compared with lower-order methods, RK4 provides higher numerical accuracy and better stability properties, making it more suitable for solving singular ODEs where error propagation near the singular boundary may significantly affect the numerical solution. Let

$$f_1(t, u, v) = v, \text{ and } f_2(t, u, v) = g(t, u, v) - \frac{a_1}{t + \varepsilon} v - \frac{a_0}{(t + \varepsilon)^2} u. \quad (5)$$

The function $g(t, u, v)$ is assumed to be continuous with respect to t and locally Lipschitz continuous with respect to $u(t)$ and $v(t)$ on the computational domain. Under these assumptions, the transformed right-hand side in Eq. (5) becomes continuous in a neighborhood of $t = 0$. Consequently, no singular behavior is encountered during numerical integration, and the system satisfies the conditions required by the standard existence and uniqueness theorem for IVPs.

The presence of the singular coefficients $1/t$ and $1/t^2$ in the original DE potentially lead to unbounded derivatives as $t \rightarrow 0$. However, after regularization, the coefficients satisfy:

$$\left| \frac{a_1}{t + \varepsilon} \right| \leq \frac{|a_1|}{\varepsilon}, \quad \left| \frac{a_0}{(t + \varepsilon)^2} \right| \leq \frac{|a_0|}{\varepsilon^2}, \quad (6)$$

for all $t \geq 0$. Therefore, provided that $u(t)$, $v(t)$, and $g(t, u, v)$ remain bounded, the derivative

$$v'(t) = g(t, u, v) - \frac{a_1}{t + \varepsilon}v - \frac{a_0}{(t + \varepsilon)^2}u \quad (7)$$

also remains bounded throughout the computational interval. Hence, the regularized formulation prevents the occurrence of unbounded derivatives caused by the original coefficients.

It is important to note that the proposed method does not require the explicit evaluation of $v(0^+)$. Instead, numerical integration is initiated from the strictly positive point $t_0 = \varepsilon > 0$. Consequently, all quantities involved in the RK4 discretization remain finite, and the existence of the limit

$$v(0^+) = \lim_{t \rightarrow 0^+} u'(t) \quad (8)$$

is not required for the practical implementation of the shooting-interpolation algorithm. This avoids potential difficulties associated with evaluating derivatives directly at the singular point.

The regularization strategy adopted in this article assumes that the exact solution remains bounded and sufficiently smooth in the neighborhood of the singular point. Under this assumption, the perturbed problem provides a consistent approximation to the original singular ODE for sufficiently small values of ε . Consequently, the asymptotic behavior of the original equation near the singular boundary is preserved while maintaining numerical tractability.

The present approach is intended for weakly singular BVPs whose solutions remain bounded and sufficiently smooth near the singular boundary. For stronger singularities, such as coefficients of the form $1/t^p$ with $p > 2$, non-integrable singularities, or problems exhibiting solution blow-up near $t = 0$, the regularized problem may no longer accurately represent the original singular equation. In such cases, additional asymptotic analysis, singular perturbation techniques, or specialized numerical methods may be required. Therefore, the theoretical convergence and stability results established in this study should be interpreted within the class of weakly singular ODEs considered herein.

Using the RK4 scheme, the numerical approximations for $u(t)$ and $v(t)$ are then computed

through the following stages:

$$\begin{aligned}
 k_1^u &= hf_1(t_i, u_i, v_i), \\
 k_1^v &= hf_2(t_i, u_i, v_i), \\
 k_2^u &= hf_1\left(t_i + \frac{h}{2}, u_i + \frac{k_1^u}{2}, v_i + \frac{k_1^v}{2}\right), \\
 k_2^v &= hf_2\left(t_i + \frac{h}{2}, u_i + \frac{k_1^u}{2}, v_i + \frac{k_1^v}{2}\right), \\
 k_3^u &= hf_1\left(t_i + \frac{h}{2}, u_i + \frac{k_2^u}{2}, v_i + \frac{k_2^v}{2}\right), \\
 k_3^v &= hf_2\left(t_i + \frac{h}{2}, u_i + \frac{k_2^u}{2}, v_i + \frac{k_2^v}{2}\right), \\
 k_4^u &= hf_1(t_i + h, u_i + k_3^u, v_i + k_3^v), \\
 k_4^v &= hf_2(t_i + h, u_i + k_3^u, v_i + k_3^v).
 \end{aligned} \tag{9}$$

Thus, the discrete RK4 formulations become:

$$\begin{aligned}
 u_{i+1} &= u_i + \frac{1}{6} (k_1^u + 2k_2^u + 2k_3^u + k_4^u), \\
 v_{i+1} &= v_i + \frac{1}{6} (k_1^v + 2k_2^v + 2k_3^v + k_4^v).
 \end{aligned} \tag{10}$$

The interval $[a, b] \subset \mathbb{R}$ is divided into N subintervals with step size h , such that $N = (b - a)/h$. Let $t_i = t_0 + ih$ for $i = 1, 2, \dots, N$. Represent the values of $u(t)$ and $v(t)$ at the discrete points t_i as $u(t_i) = u_i$ and $v(t_i) = v_i$.

To ensure that the correct initial value ζ^* lies within a bounded interval, the method begins by selecting two preliminary guesses for $v(a)$, denoted as ζ_{p_1} and ζ_{p_2} . These initial guesses produce the corresponding terminal values $u(b) = \beta_1$ and $u(b) = \beta_2$, respectively. The selection of these values is refined iteratively until they satisfy the bracketing condition $\beta_1 < \beta$ and $\beta_2 > \beta$ or $\beta_1 > \beta$ and $\beta_2 < \beta$. This condition guarantees that the exact initial value ζ^* lies within the interval $[\zeta_{p_1}, \zeta_{p_2}]$, providing a valid domain for subsequent interpolation-based refinement.

In this context, the term prediction refers to the estimation of the initial condition ζ^* that leads to a numerical solution satisfying the boundary condition $u(b) = \beta$. The function F is defined as a numerical mapping from the initial guess ζ_p to the solution value $u(b)$, that is, $F(\zeta_p) = u(b)$. Hence, predicting the graph of the function $F(\zeta_p) = u(b)$ means observing how variations in the initial guess affect the value of the solution at the boundary.

The key innovation in this method lies in employing interpolation, specifically cubic spline or polynomial interpolation, to construct an approximate inverse of the function F based on a series of computed pairs $(\zeta_p, u(b))$. This allows for estimating the value $\zeta^* \approx F^{-1}(\beta)$, which is iteratively refined until the prediction error is small. This approach avoids the trial-and-error nature of the classical shooting method by guiding the selection of ζ^* using information from previous iterations. Mathematically, the convergence criterion is expressed as follows:

$$\text{err}(\zeta^*) = |F(\zeta^*) - u(b)| = |F(\zeta^*) - \beta| \approx 0^+. \tag{11}$$

Once the value of ζ^* satisfying this condition is obtained, it is used to compute the full solution u_i for $i = 1, 2, \dots, N - 1$, thereby solving the second-order singular BVP.

3.2. Neumann Boundary Condition

In practical scenarios, boundary conditions extend beyond just Dirichlet boundary conditions. For instance, the one-dimensional steady-state heat equation employs Neumann boundary con-

ditions, which require a different approach than Dirichlet. This subsection discusses the novel shooting-interpolation method for solving the first type of Neumann boundary conditions.

Consider the following second-order singular ODE such as shown in Eq. (3) for $0 < a \leq t \leq b$ with $u'(a) = \alpha$ and $u'(b) = \beta$. The first step involves the same as Dirichlet by converting the second-order singular ODE into a system of first-order ODEs as shown in Eq. (4). To discretize the system, the RK4 method is employed. Define f_1 and f_2 that defined in Eq. (5). Based on the RK4, the numerical approximations are computed through the stages as shown in Eq. (9). Thus, the discrete RK4 formulations become same as Eq. (10). The difference lies in the boundary condition, which becomes as follows:

$$\alpha = u'(a) = v(a) \rightarrow v_0 = \alpha, \quad (12)$$

$$\beta = u'(b) = v(b) \rightarrow v_N = \beta. \quad (13)$$

The interval $[a, b] \subset \mathbb{R}$ is divided into N subintervals with step size h , such that $N = (b-a)/h$. Let $t_i = t_0 + ih$ for $i = 1, 2, \dots, N$. Represent the values of $u(t)$ and $v(t)$ at the discrete points t_i as $u(t_i) = u_i$ and $v(t_i) = v_i$.

The next step requires making two initial guesses for the value of $u(a)$. Let the two initial guesses be $u(a) = \gamma_{p_1}$ and $u(a) = \gamma_{p_2}$, resulting the corresponding values $v(b) = \beta_1$ and $v(b) = \beta_2$. These guesses are not selected randomly, but chosen manually to satisfy the condition that $\beta_1 < \beta$ and $\beta_2 > \beta$, or $\beta_1 > \beta$ and $\beta_2 < \beta$. This ensures that the true solution γ^* lies within the interval $\gamma^* \in [\gamma_{p_1}, \gamma_{p_2}]$. If this condition is not initially satisfied, the guesses are adjusted until it is met. Furthermore, the initial guesses are refined iteratively by observing the trend in error reduction with each update. This ensures that the method exhibits consistent convergence behavior and approaches the correct solution for the initial value $u(a)$.

The next stage is to involve predicting the behavior of the function $F(\gamma_p) = v(b)$, where the horizontal axis represents the guess value γ_p , and the vertical axis corresponds to the resulting endpoint value $v(b)$. The desired value γ^* is then determined by computing $F^{-1}(\beta)$ through an iterative process, which continues until the error between $F(\gamma_p)$ and the target boundary value β falls below a predefined tolerance. This process is expressed mathematically in Eq. (14).

$$\text{err}(\gamma^*) = |F(\gamma^*) - v(b)| = |F(\gamma^*) - \beta| \approx 0^+. \quad (14)$$

The value of γ^* with a small error is then used to compute the values of u_i for $i = 0, 1, \dots, N$, thereby obtaining the solution to the second-order singular Neumann boundary condition.

3.3. Robin Boundary Condition Type I

Mixed boundary conditions, which extend the concepts of both Dirichlet and Neumann conditions, are common in real-world scenarios and call for a specialized approach. These are known as Robin boundary conditions, which come in several different forms. This part will be discuss the novel shooting-interpolation method for solving the first Robin boundary conditions.

Consider the following second-order singular ODE as shown in Eq. (3) for $0 < a \leq t \leq b$ with Robin boundary conditions $u'(a) = \alpha$ and $u(b) = \beta$. The step in this condition involves the same procedure as the previous method. The difference lies only in the boundary condition, which is:

$$\alpha = u'(a) = v(a) \rightarrow v_0 = \alpha. \quad (15)$$

The interval $[a, b] \subset \mathbb{R}$ is divided into N subintervals with step size h , such that $N = (b-a)/h$. Let $t_i = t_0 + ih$ for $i = 1, 2, \dots, N$. Represent the values of $u(t)$ and $v(t)$ at the discrete points t_i as $u(t_i) = u_i$ and $v(t_i) = v_i$.

At this stage, it is necessary to provide two initial estimates for the unknown value of $u(a)$, denoted as γ_{p_1} and γ_{p_2} . These estimates lead to the computed boundary outputs $u(b) = \beta_1$ and

$u(b) = \beta_2$, respectively. The selection of these estimates is performed iteratively, aiming to ensure that the resulting boundary values that satisfy the condition $\beta_1 < \beta$ and $\beta_2 > \beta$, or vice versa. Meeting this requirement guarantees that the true value γ^* resides within the interval $[\gamma_{p_1}, \gamma_{p_2}]$. When this condition is not fulfilled, the initial guesses should be revised accordingly to bracket the solution properly.

Following this, the functional relationship $F(\gamma_p) = u(b)$ is approximated, where γ_p serves as the input along the horizontal axis and the corresponding terminal value $u(b)$ forms the output. The objective is to determine the input γ^* such that $F(\gamma^*) = \beta$. This is achieved by applying an iterative inverse estimation procedure, which adjusts γ_p step by step until the discrepancy between $F(\gamma_p)$ and the target β falls within an acceptable tolerance. The formulation of this process is provided in Eq. (16).

$$\text{err}(\gamma^*) = |F(\gamma^*) - u(b)| = |F(\gamma^*) - \beta| \approx 0^+. \quad (16)$$

The value of γ^* with a small error is then used to compute the values of u_i for $i = 0, 1, \dots, N - 1$, thereby obtaining the solution to the second-order singular equation for the first type of Robin boundary conditions.

3.4. Robin Boundary Conditions Type II

As discussed earlier in the previous subsection, Robin boundary conditions come in various forms. This section focuses on novel shooting-interpolation method for solving the second type of Robin boundary conditions. Consider the following ODE as shown in Eq. (3) for $0 < a \leq t \leq b$ with Robin boundary conditions $u(a) = \alpha$ and $u'(b) = \beta$.

The step in second types of Robin condition involves the same procedure as the previous method. The difference lies only in the boundary condition. Since the system satisfies $u'(t) = v(t)$, the Robin boundary condition at the right endpoint can be written directly as $v(b) = \beta$.

The next phase involves selecting two preliminary estimates for the unknown value $v(a)$, denoted as ζ_{p_1} and ζ_{p_2} . These initial values produce corresponding outcomes at the penultimate time step, namely $v(b) = \beta_1$ and $v(b) = \beta_2$. The estimates are refined until the resulting terminal values satisfy $\beta_1 < \beta$ and $\beta_2 > \beta$, or $\beta_1 > \beta$ and $\beta_2 < \beta$. Satisfying this criterion ensures that the true solution ζ^* lies within the bounded interval $[\zeta_{p_1}, \zeta_{p_2}]$. If the condition is not initially met, the guesses are updated accordingly until the bracketing condition is fulfilled.

Subsequently, the relation between the input guess ζ_p and the output $v(b)$ is treated as a function $F(\zeta_p) = v(b)$, where the horizontal axis denotes the estimated initial value and the vertical axis corresponds to the final computed result. The main goal in this stage is to identify ζ^* such that $F(\zeta^*) = \beta$. This inverse estimation is carried out through an iterative scheme that minimizes the difference between $F(\zeta_p)$ and the target boundary value β until the error falls below the specified tolerance. The formulation of this procedure is presented as follows.

$$\text{err}(\zeta^*) = |F(\zeta^*) - v(b)| = |F(\zeta^*) - \beta| \approx 0^+. \quad (17)$$

The value of ζ^* with a small error is then used to compute the values of u_i for $i = 1, 2, \dots, N$, thereby obtaining the solution to the second-order singular for the first type of Robin conditions.

Algorithm 3.1 provides a unified computational framework for solving second-order singular BVPs under Dirichlet, Neumann, and Robin boundary conditions. This method transforms the ODE into a first-order system and employs the RK4 scheme to obtain highly accurate numerical approximations. The unknown initial condition is treated as a shooting parameter and iteratively refined through an interpolation-based inverse estimation process until the prescribed terminal boundary condition is satisfied within a tolerance. Although the numerical integration procedure remains identical for all boundary-condition types, the distinction among the four cases lies in the selection of the known boundary values, the unknown initial quantity to be estimated, and the shooting function used during the shooting process. These differences are summarized in

Algorithm 1 Shooting-Interpolation Algorithm with RK4

Require: $\text{iter}_{\max}$, a , b , h , left and right boundary conditions

Ensure: Numerical solution u_i , $i = 0, 1, \dots, N$

- 1: Rewrite the second-order singular ODE as $u' = v$ and $v' = g(t, u, v) - \frac{a_1}{t + \varepsilon}v - \frac{a_0}{(t + \varepsilon)^2}u$.
 - 2: Define $f_1(t, u, v) = v$ and $f_2(t, u, v) = g(t, u, v) - \frac{a_1}{t + \varepsilon}v - \frac{a_0}{(t + \varepsilon)^2}u$.
 - 3: Set $t_i = a + ih$, $i = 0, 1, \dots, N$.
 - 4: Initialize $\theta_p = []$ and $\beta_p = []$.
 - 5: **Define one RK4 step:**
 - 6: $k_1^u = hf_1(t_i, u_i, v_i)$, $k_1^v = hf_2(t_i, u_i, v_i)$.
 - 7: $k_2^u = hf_1\left(t_i + \frac{h}{2}, u_i + \frac{k_1^u}{2}, v_i + \frac{k_1^v}{2}\right)$, $k_2^v = hf_2\left(t_i + \frac{h}{2}, u_i + \frac{k_1^u}{2}, v_i + \frac{k_1^v}{2}\right)$.
 - 8: $k_3^u = hf_1\left(t_i + \frac{h}{2}, u_i + \frac{k_2^u}{2}, v_i + \frac{k_2^v}{2}\right)$, $k_3^v = hf_2\left(t_i + \frac{h}{2}, u_i + \frac{k_2^u}{2}, v_i + \frac{k_2^v}{2}\right)$.
 - 9: $k_4^u = hf_1(t_i + h, u_i + k_3^u, v_i + k_3^v)$, $k_4^v = hf_2(t_i + h, u_i + k_3^u, v_i + k_3^v)$.
 - 10: Update $u_{i+1} = u_i + \frac{k_1^u + 2k_2^u + 2k_3^u + k_4^u}{6}$ and $v_{i+1} = v_i + \frac{k_1^v + 2k_2^v + 2k_3^v + k_4^v}{6}$.
 - 11: Choose two initial shooting parameters θ_{p_1} and θ_{p_2} .
 - 12: **for** $j = 1$ **to** 2 **do**
 - 13: Set $\theta \leftarrow \theta_{p_j}$.
 - 14: Set the corresponding unknown initial condition using Table 1.
 - 15: **for** $i = 0$ **to** $N - 1$ **do**
 - 16: Apply the RK4 step to obtain (u_{i+1}, v_{i+1}) .
 - 17: **end for**
 - 18: Store the terminal value as β_j .
 - 19: $\theta_p[j] \leftarrow \theta_{p_j}$, $\beta_p[j] \leftarrow \beta_j$.
 - 20: **end for**
 - 21: Set $\text{iter} \leftarrow 2$.
 - 22: **while** $\text{iter} < \text{iter}_{\max}$ **do**
 - 23: Construct the interpolation set $\{(\theta_p[k], \beta_p[k]) : k = 1, \dots, \text{iter}\}$.
 - 24: Determine the new shooting parameter $\theta_{\text{new}} = F^{-1}(\beta)$.
 - 25: Set the corresponding unknown initial condition using θ_{new} .
 - 26: **for** $i = 0$ **to** $N - 1$ **do**
 - 27: Apply the RK4 step to obtain (u_{i+1}, v_{i+1}) .
 - 28: **end for**
 - 29: $\text{iter} \leftarrow \text{iter} + 1$.
 - 30: $\theta_p[\text{iter}] \leftarrow \theta_{\text{new}}$.
 - 31: Store the computed terminal value in $\beta_p[\text{iter}]$.
 - 32: **end while**
 - 33: Set $\theta^* \leftarrow \theta_p[\text{iter}_{\max}]$.
 - 34: Set the corresponding unknown initial condition using θ^* .
 - 35: **for** $i = 0$ **to** $N - 1$ **do**
 - 36: Apply the RK4 step to obtain (u_{i+1}, v_{i+1}) .
 - 37: **end for**
 - 38: Compute $\text{err}(\theta^*) = |\beta_{\text{comp}} - \beta|$.
 - 39: **return** $u(t_i)$, $i = 0, 1, \dots, N$.
-

Table 1, which demonstrates that the proposed framework can accommodate multiple boundary-condition configurations within a single generalized algorithmic structure.

Table 1: Boundary-condition types in the RK4 shooting-interpolation algorithm

Boundary Condition	Known Left Condition	Target Right Condition	Unknown Initial Value	Shooting Parameter	Shooting Function
Dirichlet	$u(a) = \alpha$	$u(b) = \beta$	$v(a)$	$\theta = v(a)$	$F(\theta) = u(b)$
Neumann	$v(a) = \alpha$	$v(b) = \beta$	$u(a)$	$\theta = u(a)$	$F(\theta) = v(b)$
Robin I	$v(a) = \alpha$	$u(b) = \beta$	$u(a)$	$\theta = u(a)$	$F(\theta) = u(b)$
Robin II	$u(a) = \alpha$	$v(b) = \beta$	$v(a)$	$\theta = v(a)$	$F(\theta) = v(b)$

4. Convergence, Stability, and Error Analysis

This section presents the theoretical justification of the proposed Shooting-Interpolation algorithm for solving second-order singular BVPs with Dirichlet, Neumann, and Robin conditions using the RK4 method. The proposed algorithm iteratively estimates an unknown initial shooting parameter through interpolation.

Let the exact shooting parameter be denoted generally by θ^* , where $\theta^* \in \{\zeta^*, \gamma^*\}$ depending on the imposed boundary condition. Define the shooting function as:

$$F(\theta) = \Phi(b; \theta), \quad (18)$$

where $\Phi(b; \theta)$ represents the numerical quantity evaluated at the boundary point corresponding to the considered boundary condition. The algorithm constructs a sequence:

$$\{\theta^{(k)}\}_{k=1}^{\text{iter}_{\max}} = \{\theta^{(1)}, \theta^{(2)}, \dots, \theta^{(\text{iter}_{\max})}\}, \quad (19)$$

using interpolation of previously computed pairs $(\theta^{(k)}, F(\theta^{(k)}))$.

4.1. Convergence Analysis

A numerical algorithm is said to be convergent if the numerical approximation approaches the exact solution as the step size and error tend to zero. Assume the following conditions hold:

- (1) The function $g(t, u, v)$ is continuous on $[a, b]$.
- (2) The function $g(t, u, v)$ satisfies the Lipschitz condition:

$$|g(t, u_1, v_1) - g(t, u_2, v_2)| \leq L_1|u_1 - u_2| + L_2|v_1 - v_2|, \quad (20)$$

for some constants $L_1, L_2 > 0$.

- (3) The shooting function $F(\theta)$ is continuously differentiable near θ^* .
- (4) The interpolation sequence satisfies $\theta^{(k)} \rightarrow \theta^*$.

By Picard–Lindelöf theorem, the transformed IVP admits a unique solution because the governing function satisfies the Lipschitz condition. Consequently, the shooting function $F(\theta)$ is continuous and locally smooth.

The interpolation stage approximates the mapping $F^{-1}(\beta)$. Let $\tilde{\theta}^{(k)}$ denote the interpolated estimate at the k -th iteration. Based on interpolation theory, the interpolation error satisfies

$$|\theta^* - \tilde{\theta}^{(k)}| \leq C \prod_{i=1}^k |\theta^* - \theta^{(i)}|, \quad (21)$$

where C is a bounded constant depending on higher-order derivatives of F^{-1} . Since the interpolation points progressively approach the exact parameter and $|\theta^* - \theta^{(i)}| < 1$, the interpolation error decreases iteratively, yielding

$$\lim_{k \rightarrow \infty} |\theta^* - \tilde{\theta}^{(k)}| = 0, \quad \tilde{\theta}^{(k)} \rightarrow \theta^*. \quad (22)$$

Furthermore, the classical RK4 method is both consistent and convergent with local truncation error $\mathcal{O}(h^5)$ and global discretization error $\mathcal{O}(h^4)$. Therefore,

$$u_h(t; \tilde{\theta}^{(k)}) \rightarrow u(t), \quad \text{as } h \rightarrow 0 \text{ and } k \rightarrow \infty. \quad (23)$$

Hence, the proposed Shooting-Interpolation algorithm converges for Dirichlet, Neumann, and Robin boundary conditions. $\square$

4.2. Stability Analysis

Stability analysis is essential to ensure that the numerical solution remains bounded when small perturbations arise from round-off errors, discretization errors, or perturbations in the initial shooting parameter. A numerical algorithm is said to be stable if small perturbations introduced during the iterative process do not grow uncontrollably as the computation progresses.

The proposed Shooting-Interpolation algorithm employs the classical RK4 integration scheme for the transformed first-order system. Let

$$\mathbf{y}(t) = \begin{bmatrix} u(t) \\ v(t) \end{bmatrix}, \quad \mathbf{f}(t, \mathbf{y}) = \begin{bmatrix} f_1(t, u, v) \\ f_2(t, u, v) \end{bmatrix}. \quad (24)$$

The RK4 update formula may be written compactly as:

$$\mathbf{y}_{i+1} = \mathbf{y}_i + \frac{1}{6} (\mathbf{k}_1 + 2\mathbf{k}_2 + 2\mathbf{k}_3 + \mathbf{k}_4), \quad (25)$$

where:

$$\begin{aligned} \mathbf{k}_1 &= h\mathbf{f}(t_i, \mathbf{y}_i), \\ \mathbf{k}_2 &= h\mathbf{f}\left(t_i + \frac{h}{2}, \mathbf{y}_i + \frac{\mathbf{k}_1}{2}\right), \\ \mathbf{k}_3 &= h\mathbf{f}\left(t_i + \frac{h}{2}, \mathbf{y}_i + \frac{\mathbf{k}_2}{2}\right), \\ \mathbf{k}_4 &= h\mathbf{f}(t_i + h, \mathbf{y}_i + \mathbf{k}_3). \end{aligned} \quad (26)$$

To analyze stability, consider two nearby numerical solutions $\mathbf{y}_i$ and $\tilde{\mathbf{y}}_i$ generated from slightly perturbed initial conditions. Define the perturbation vector as $\mathbf{e}_i = \mathbf{y}_i - \tilde{\mathbf{y}}_i$. Assume that $\mathbf{f}(t, \mathbf{y})$ satisfies the Lipschitz condition

$$\|\mathbf{f}(t, \mathbf{y}_1) - \mathbf{f}(t, \mathbf{y}_2)\| \leq L\|\mathbf{y}_1 - \mathbf{y}_2\|, \quad (27)$$

for some constant $L > 0$. By using the stability property of the RK4 method, the perturbation propagation satisfies:

$$\|\mathbf{e}_{i+1}\| \leq (1 + Ch)\|\mathbf{e}_i\|, \quad (28)$$

where $C > 0$ is a bounded constant depending on the Lipschitz constant and the RK4 coefficients. Applying the inequality recursively for n iterations gives

$$\|\mathbf{e}_n\| \leq (1 + Ch)^n \|\mathbf{e}_0\| \leq e^{Cnh} \|\mathbf{e}_0\| = e^{C(b-a)} \|\mathbf{e}_0\|. \quad (29)$$

The bound above remains finite for all $t \in [a, b]$. Therefore, perturbations introduced during the numerical iteration remain bounded throughout the computation. This result demonstrates that the proposed Shooting-Interpolation algorithm possesses numerical stability for sufficiently smooth singular boundary value problems satisfying the Lipschitz condition.

Furthermore, because the interpolation process only updates the shooting parameter using previously computed bounded numerical solutions, the interpolation process does not introduce unbounded amplification of errors. Consequently, the algorithm remains stable under small perturbations in both the initial shooting parameter and the RK4 numerical integration process.

4.3. Error Analysis

Error analysis in this study is important to quantify the accuracy of the numerical approximation produced by the proposed algorithm. In numerical analysis, the total numerical error consists of the local truncation error and the global discretization error.

Let the exact solution of the transformed first-order system at t_i in Eq. (4) be denoted by $\mathbf{y}(t_i) = [u(t_i), v(t_i)]^T$, and let the RK4 numerical approximation be denoted by $\mathbf{y}_i$. Consider the Taylor series expansion of the exact solution around t_i as follows.

$$\mathbf{y}(t_{i+1}) = \mathbf{y}(t_i) + h\mathbf{y}'(t_i) + \frac{h^2}{2}\mathbf{y}''(t_i) + \frac{h^3}{6}\mathbf{y}^{(3)}(t_i) + \frac{h^4}{24}\mathbf{y}^{(4)}(t_i) + \frac{h^5}{120}\mathbf{y}^{(5)}(\xi_i), \quad (30)$$

for some $\xi_i \in [t_i, t_{i+1}]$. The RK4 method reproduces the Taylor expansion terms exactly up to order four. Therefore, subtracting the RK4 approximation from the exact Taylor expansion yields the local truncation error $\tau_i = \mathcal{O}(h^5)$.

Assume that the exact solution possesses bounded fifth derivatives on $[a, b]$, namely

$$\|\mathbf{y}^{(5)}(t)\| \leq M, \quad t \in [a, b], \quad (31)$$

for some constant $M > 0$. Then, there exists a constant $C > 0$ such that $\|\tau_i\| \leq Ch^5$. Hence, the local truncation error of the RK4 integration procedure satisfies $\tau_i = \mathcal{O}(h^5)$, showing that the RK4 method is locally fourth-order.

Next, define the global discretization error by $E_i = \|\mathbf{y}(t_i) - \mathbf{y}_i\|$. Since the RK4 method is stable and the local truncation error satisfies $\tau_i = \mathcal{O}(h^5)$, the accumulated global discretization error satisfies $E_i = \mathcal{O}(h^4)$. Therefore, the Shooting-Interpolation algorithm combined with the RK4 integration scheme achieves fourth-order global accuracy. Furthermore, because the interpolation stage only updates the shooting parameter using bounded numerical approximations, the interpolation process preserves the overall convergence behavior and high-order accuracy of the proposed algorithm.

5. Computational Complexity Analysis

The computational complexity of the Shooting-Interpolation algorithm with the RK4 scheme is analyzed in terms of both time complexity and space complexity. Let N denote the number of discretization points and let $\text{iter}_{\max}$ represent the maximum number of interpolation iterations.

The proposed algorithm consists of three major computational stages, namely discretization, RK4-based shooting integration, and interpolation-based refinement of the shooting parameter. In each shooting process, the algorithm performs forward numerical integration from $i = 0$ to $N - 1$ using the RK4 method applied to the transformed first-order system $u'(t) = f_1(t, u, v)$ and $v'(t) = f_2(t, u, v)$.

For each discretization step, the RK4 procedure computes four intermediate coefficients for both variables u and v , namely $(k_1^u, k_2^u, k_3^u, k_4^u)$ and $(k_1^v, k_2^v, k_3^v, k_4^v)$. Each RK4 stage requires only a constant number of arithmetic and function-evaluation operations. Consequently, one

RK4 integration step requires $\mathcal{O}(1)$ computational cost. Since the forward integration contains N discretization steps, one complete shooting process requires $\mathcal{O}(N)$ computational time.

The interpolation stage is repeated iteratively until the maximum iteration $\text{iter}_{\max}$ is reached. During the k -th interpolation iteration, the algorithm constructs interpolation data from previously computed pairs:

$$(\theta_p^{(j)}, \beta^{(j)}) \text{ for } j = 1, 2, \dots, k,$$

followed by one additional RK4 forward integration over all discretization points. Therefore, the computational cost for the k -th iteration becomes $\mathcal{O}(N + k)$, where the term $\mathcal{O}(k)$ corresponds to the interpolation construction process and $\mathcal{O}(N)$ corresponds to the RK4 shooting integration. By summing all interpolation iterations, the total computational complexity of the proposed algorithm is obtained as follows:

$$\begin{aligned} T(N) &= \sum_{k=1}^{\text{iter}_{\max}} \mathcal{O}(N + k) \\ &= \mathcal{O}(N \cdot \text{iter}_{\max} + \sum_{k=1}^{\text{iter}_{\max}} k) \\ &= \mathcal{O}(N \cdot \text{iter}_{\max} + \text{iter}_{\max}^2). \end{aligned} \quad (32)$$

Because the number of discretization points is generally much larger than the interpolation iterations, namely $N \gg \text{iter}_{\max}$, the dominant computational term becomes

$$T(N) \approx \mathcal{O}(N \cdot \text{iter}_{\max}). \quad (33)$$

Although the RK4 requires four intermediate evaluations per discretization step, this only affects the constant factor of the computation and does not alter the asymptotic order of complexity. Therefore, the time complexity is quasi-linear with respect to the discretization size.

For space complexity, the proposed algorithm stores several numerical vectors, including the discretization points $t[i]$, the solution vectors $u[i]$ and $v[i]$, and the interpolation history vectors θ_p and β . Additional RK4 intermediate coefficients k_1, k_2, k_3, k_4 are computed locally during each iteration and require only constant auxiliary memory. Hence, the dominant memory requirement arises from storing the numerical solution vectors of length proportional to N . Therefore, the overall space complexity of the proposed algorithm is $S(N) = \mathcal{O}(N)$.

These results indicate that the proposed Shooting-Interpolation algorithm combined with the RK4 integration scheme maintains quasi-linear computational growth while preserving linear memory consumption. Consequently, the method remains computationally efficient and scalable for solving second-order singular boundary value problems with Dirichlet, Neumann, and Robin boundary conditions over large discretization domains.

6. Numerical Example

This section presents numerical examples to demonstrate the applicability, accuracy, and effectiveness of the proposed Shooting-Interpolation method combined with the RK4 scheme for solving singular second-order BVPs with various boundary conditions.

6.1. Novel Shooting-Interpolation Method for First ODE

Given the singular second-order ODE problem as shown in Eq. (34),

$$u''(t) - \frac{u'(t)}{t} - 2 = 0, \quad (34)$$

for $0 < t \leq 1$ with Dirichlet boundary conditions $u(0^+) = 0$ and $u(1) = 0$. The exact solution of Eq. (34) is $u(t) = t^2 \ln(t)$. By transforming Eq. (34) into a system of first-order ODEs, define $u'(t) = v(t)$. Then, the transformed system becomes

$$u'(t) = v(t), \quad (35)$$

$$v'(t) = \frac{v(t)}{t + \varepsilon} + 2. \quad (36)$$

Next, define the RK4 functions as follows:

$$f_1(t, u, v) = v, \quad (37)$$

$$f_2(t, u, v) = \frac{v}{t + \varepsilon} + 2, \quad (38)$$

where $\varepsilon > 0$ is a small positive constant introduced to avoid singularity at $t = 0$. The numerical integration is performed using the classical RK4 scheme. For each discretization point t_i , the RK4 coefficients are computed same as Eq. (9). Thus, the RK4 update equations are obtained same as Eq. (10). The next step is to discretize the interval domain $(0, 1] \subseteq \mathbb{R}$ into N subintervals with step size $h = 0.01$, such that $N = 100$. Let $t_i = 0.01i$ for $i = 0, 1, 2, \dots, N$. Furthermore, let $u(t_i) = u_i$ and $v(t_i) = v_i$. The proposed Shooting-Interpolation method is then applied to determine the unknown initial shooting parameter satisfying the prescribed Dirichlet boundary condition at the right endpoint.

The next stage involves making two initial guesses for $v(0^+)$. Two possible initial guesses are $v(0^+) = -1$ and $v(0^+) = 1$. These yield $u(1) \approx 5 \cdot 10^9$ and $u(1) \approx -5 \cdot 10^9$ respectively, resulting $\beta_1 < 0$ and $\beta_2 > 0$. The next stage is to predict the graph of the function $F(\zeta_p) = u(1)$ over 1000 iterations as shown in the Figure 3, resulting in the estimated value of $v(0^+)$, that is $\zeta^* = -1.3203 \cdot 10^{-9}$ with $\text{err}(\zeta^*) \approx 2.1011 \cdot 10^{-12}$. The value of ζ^* is supposed to be guessed in the conventional shooting method. By contrast, the proposed shooting-interpolation method iteratively finds this value. The iteration's error and the solution of ODE can be obtained using ζ^* , and it can be seen in Figure 4.

To examine the long-term performance of the first ODE problem, the iteration process was extended until a specified tolerance error threshold of $\text{err}(\zeta^*) < 1 \times 10^{-11}$ was achieved. This convergence criterion can be achieved at 460-th iteration as shown in Figure (4b). At 1000-th iteration, the predicted value of $v(0^+)$ was refined to $\zeta^* = -1.3203 \times 10^{-9}$, and the resulting solution satisfied the boundary condition $u(1) \approx 4.9978 \times 10^{-10}$, corresponding to an absolute error of approximately 2.1011×10^{-12} . This result affirms the accuracy and robustness of the proposed shooting-interpolation method in efficiently solving singular BVPs.

Figure 2 illustrates the squared error distribution, where the proposed shooting-interpolation approach exhibits a consistently lower error profile compared with the benchmark algorithms. This behavior is consistent with the quantitative results reported in Table 2, which confirms the superior performance of the RK4-based shooting-interpolation method across all accuracy metrics. In particular, the RK4 Shoot-Inter method achieves the lowest errors, with MSE, RMSE, MAE, and MAXE values of 1.65×10^{-15} , 4.07×10^{-8} , 7.87×10^{-9} , and 3.69×10^{-7} , respectively, significantly outperforming both the Euler-based method and the conventional shooting method. While the Euler-based approach also shows improved performance over the conventional shooting algorithm, its error magnitudes remain several orders higher than those of the RK4 variant. In terms of computational cost, the RK4 Shoot-Inter method requires a CPU time of 2.79×10^2 seconds, which is slightly higher than the Euler-based method but substantially justified by the marked improvement in accuracy, whereas the conventional shooting method exhibits negligible computational time but significantly poor accuracy, indicating a clear trade-off between computational efficiency and solution quality.

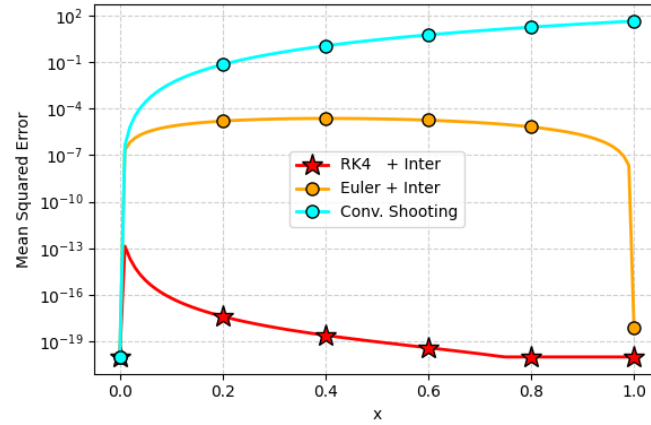


Figure 2: Comparative of error distribution for the first ODE

Table 2: Performance comparison of the considered numerical methods for the first ODE.

Method	MSE	RMSE	MAE	MAXE	CPU
RK4 Shoot-Inter.	1.65×10^{-15}	4.07×10^{-8}	7.87×10^{-9}	3.69×10^{-7}	2.79×10^2
Euler Shoot-Inter.	1.28×10^{-5}	3.57×10^{-3}	3.27×10^{-3}	4.83×10^{-3}	2.71×10^2
Conv Shooting	8.85×10^0	2.97×10^0	2.21×10^0	6.60×10^0	9.78×10^{-4}

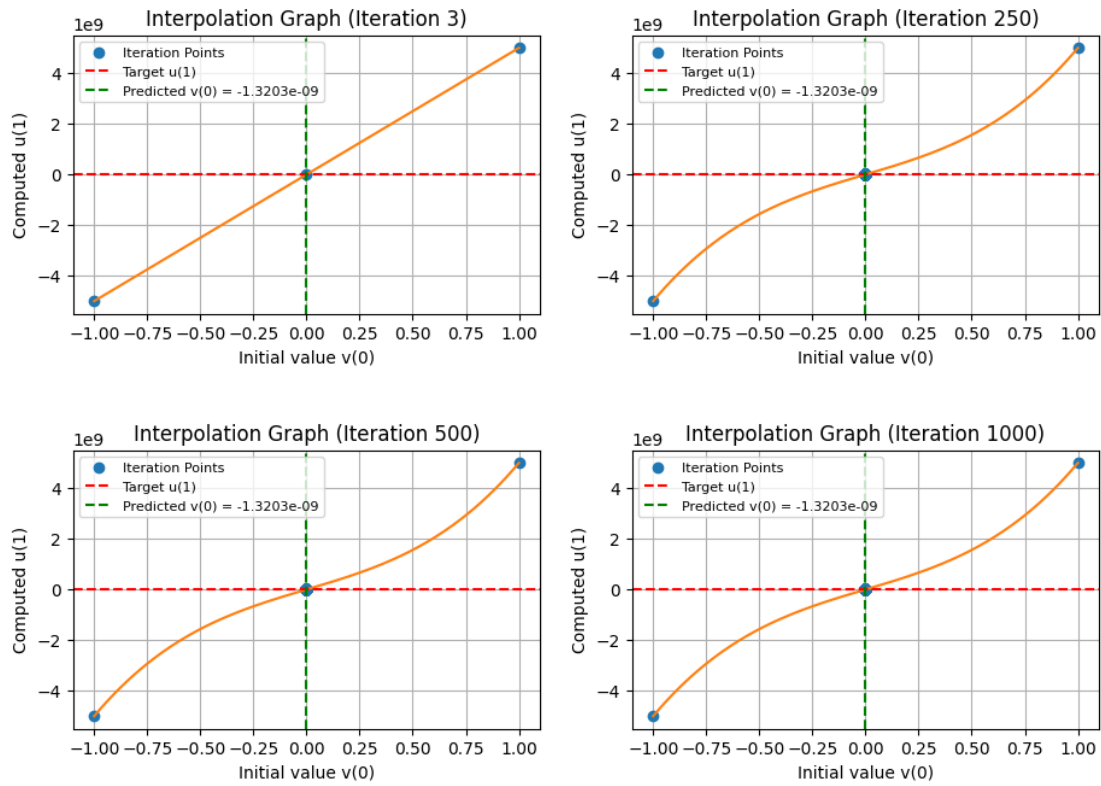


Figure 3: Interpolation graphs at iterations 3, 250, 500, and 1000 for the first ODE.

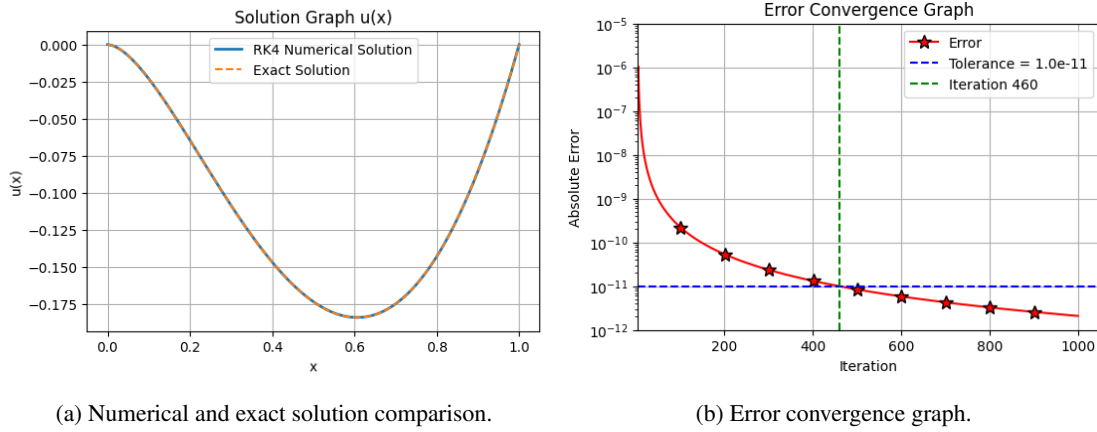


Figure 4: Performance visualization of the proposed method for the first ODE.

6.2. Shooting-Interpolation Method for Second ODE

Given the singular second-order ODE problem as shown in Eq. (39).

$$u''(t) + \frac{2}{t}u'(t) - \phi^2 u(t) \exp\left(\frac{\psi\varphi(1-u(t))}{1+\varphi(1-u(t))}\right) = 0, \quad (39)$$

with $\phi = 0.6$; $\psi = 40$; and $\varphi = 0.2$ for $0 < t \leq 1$ with boundary conditions $u'(0^+) = 0$ and $u(1) = 1$. The exact solution of this problem is almost impossible to be obtained, therefore it will be computed numerically. By transforming Eq. (39) into a system of first-order ODEs and converting it into an IVP, the following system is obtained:

$$u'(t) = v(t), \quad (40)$$

$$v'(t) = -\frac{2}{t+\varepsilon}v(t) + \phi^2 u(t) \exp\left(\frac{\psi\varphi(1-u(t))}{1+\varphi(1-u(t))}\right). \quad (41)$$

The next step is to discretize the domain interval $(0, 1] \subseteq \mathbb{R}$ into N subintervals with a step size of $h = 0.01$; such that $N = 100$. Let $t_i = 0.01i$ for $i = 0, 1, 2, \dots, N$. The values of $u(t)$ and $v(t)$ can be represented as $u(t_i) = u_i$ and $v(t_i) = v_i$. To obtain a more accurate numerical approximation, the RK4 method is employed. For each iteration step, the RK4 formulation is defined same as Eq. (9), where the RK4 coefficients are computed by:

$$\begin{aligned} k_1^u &= h v_i, \\ k_1^v &= h \left[-\frac{2}{t_i + \varepsilon} v_i + \phi^2 u_i \exp\left(\frac{\psi\varphi(1-u_i)}{1+\varphi(1-u_i)}\right) \right], \\ k_2^u &= h \left(v_i + \frac{k_1^v}{2} \right), \\ k_2^v &= h \left[\frac{-2}{t_i + \varepsilon + h/2} \left(v_i + \frac{k_1^v}{2} \right) + \phi^2 \left(u_i + \frac{k_1^u}{2} \right) \exp\left(\frac{\psi\varphi(1-(u_i + k_1^u/2))}{1+\varphi(1-(u_i + k_1^u/2))}\right) \right], \\ k_3^u &= h \left(v_i + \frac{k_2^v}{2} \right), \end{aligned}$$

$$\begin{aligned}
 k_3^v &= h \left[-\frac{2}{t_i + \varepsilon + h/2} \left(v_i + \frac{k_2^v}{2} \right) \right. \\
 &\quad \left. + \phi^2 \left(u_i + \frac{k_2^u}{2} \right) \exp \left(\frac{\psi \varphi (1 - (u_i + k_2^u/2))}{1 + \varphi (1 - (u_i + k_2^u/2))} \right) \right], \\
 k_4^u &= h (v_i + k_3^v), \\
 k_4^v &= h \left[-\frac{2}{t_i + \varepsilon + h} (v_i + k_3^v) + \phi^2 (u_i + k_3^u) \exp \left(\frac{\psi \varphi (1 - (u_i + k_3^u))}{1 + \varphi (1 - (u_i + k_3^u))} \right) \right]. \quad (42)
 \end{aligned}$$

The next step involves making two initial guesses for $u(0^+)$. Two possible initial guesses are $u(0^+) = 0.01$ and $u(0^+) = 0.6$. These yield $u(1) \approx 1.0952$ and $u(1) \approx 9.5197 \cdot 10^{-2}$ respectively, resulting in $\beta_1 > 1$ and $\beta_2 < 1$. The next stage is to predict the graph of the function $F(\gamma_p) = u(1)$ over 1000 iterations as shown in Figure 5, resulting in $\gamma^* = 0.3636$ with $\text{err}(\gamma^*) \approx 3.2467 \cdot 10^{-9}$. The second problem also has another solution obtained through two different initial guesses.

For the second case when the initial guesses are $u(0^+) = 0.6$ and $u(0^+) = 1.2$, these yield $u(1) \approx 9.5218 \cdot 10^{-1}$ and $u(1) \approx 1.2131$ respectively, resulting in $\beta_1 > 1$ and $\beta_2 < 1$. With these initial guesses, the estimated value of $u(0^+)$ is $\gamma^* = 0.9079$ with $\text{err}(\gamma^*) \approx 3.3001 \cdot 10^{-8}$. Similar to the first problem, the conventional shooting method requires an initial guess for γ^* ; however, for this nonlinear ODEs, FDM and collocation methods are not suitable since the nonlinear structure leads to difficulties in constructing stable discrete linear systems. In contrast, the proposed shooting–interpolation method can determine γ^* iteratively without such restrictions. The two solutions of the ODE are obtained using different values of γ^* , and the results are shown in Figure 8. Finally, Figures 10 present the error behavior for each iteration.

To further examine the robustness and convergence behavior of the proposed method, extended iterations were conducted under predefined tolerance levels of 1×10^{-7} and 1×10^{-8} . When the initial guesses are $u(0^+) = 0.01$ and $u(0^+) = 0.6$, convergence is achieved at iteration 318. Based on interpolation, the predicted γ^* that results in $u(1) = 1$ is $\gamma^* = 0.3636$, where the corresponding value satisfies $u(1) = 1.0000$ with an absolute error of approximately 9.9940×10^{-9} . The computational time for this case is 285.04 seconds for 1000 iterations, reflecting the cost of achieving high-accuracy approximation over the interval $0.1 \leq u(0^+) \leq 0.6$.

In contrast, using initial guesses $u(0^+) = 0.6$ and $u(0^+) = 1.2$, convergence is reached at iteration 354. The interpolated estimate yields $\gamma^* = 0.9071$, producing $u(1) = 1.0000$ with an error of approximately 9.9868×10^{-8} . This result highlights the efficiency of the proposed method and confirms its ability to capture multiple solution branches of singular BVPs. Moreover, the computational time for the interval $0.6 \leq u(0^+) \leq 1.2$ is 274.29 seconds for 1000 iterations, indicating that the shooting–interpolation framework remains stable while maintaining high numerical accuracy.

Figures 11 present the quantified error distributions for the second ODE with different ranges, while the corresponding quantitative performance metrics are summarized in Table 4. Across both scenarios, the proposed method consistently achieves the highest accuracy, exhibiting significantly lower error magnitudes compared with the Euler approach and the conventional shooting algorithm. This improvement is reflected in the error distributions, where the RK4 solution remains stable and closely aligned with the reference behavior over the entire domain. In contrast, both alternative methods show noticeably higher and less stable error profiles. Moreover, the RK4 framework maintains this superior accuracy across different initial condition ranges, indicating strong robustness of the proposed approach, although at the expense of higher computational cost compared with the simpler methods.

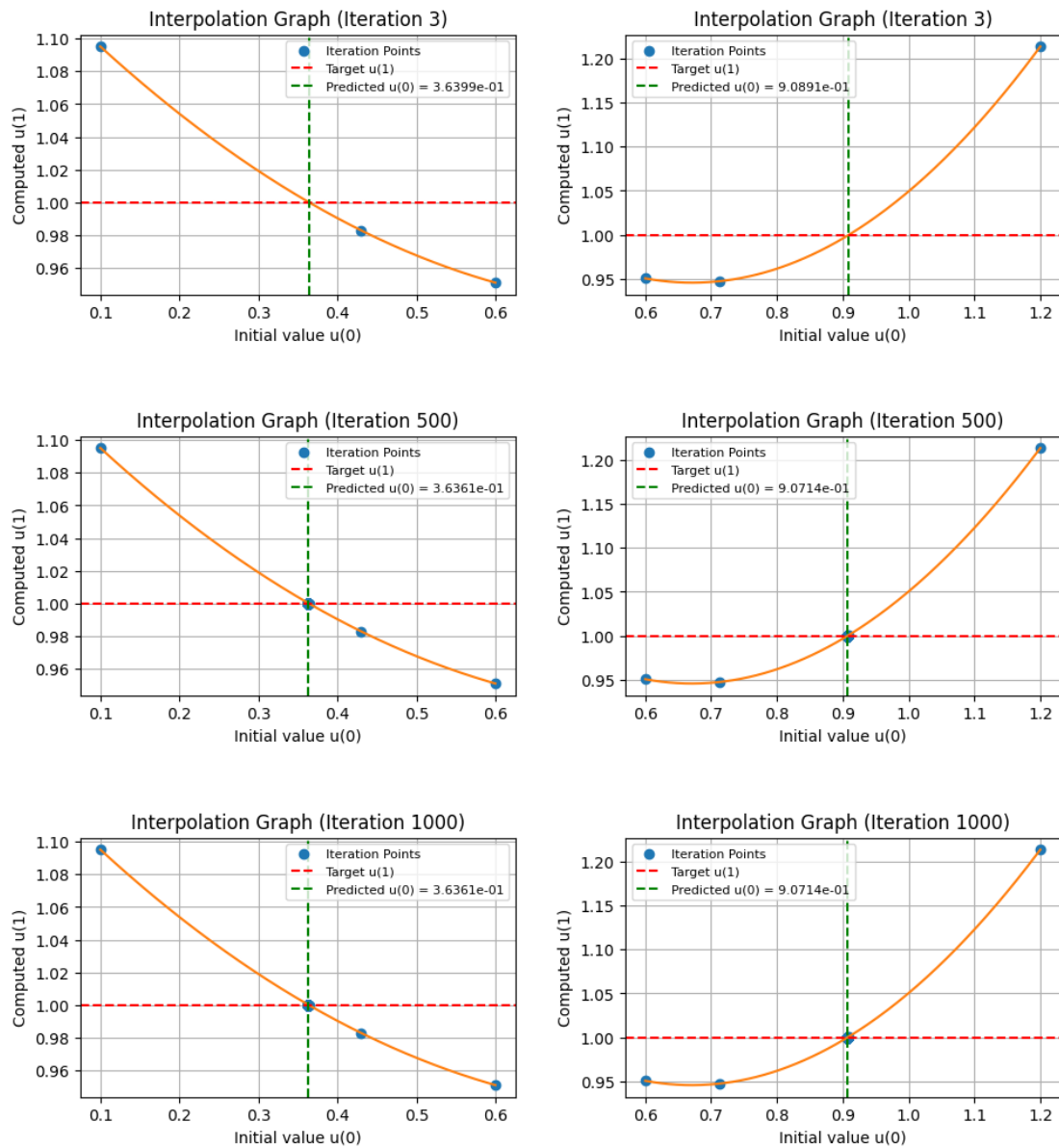


Figure 5: Interpolation graphs at iterations 3, 500, and 1000 for the second ODE.

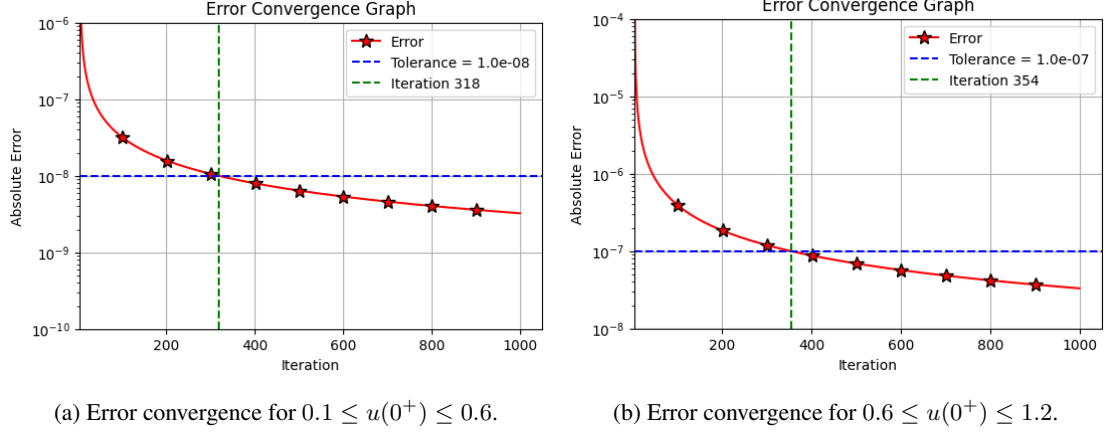


Figure 6: Error convergence of the second ODE for different intervals.

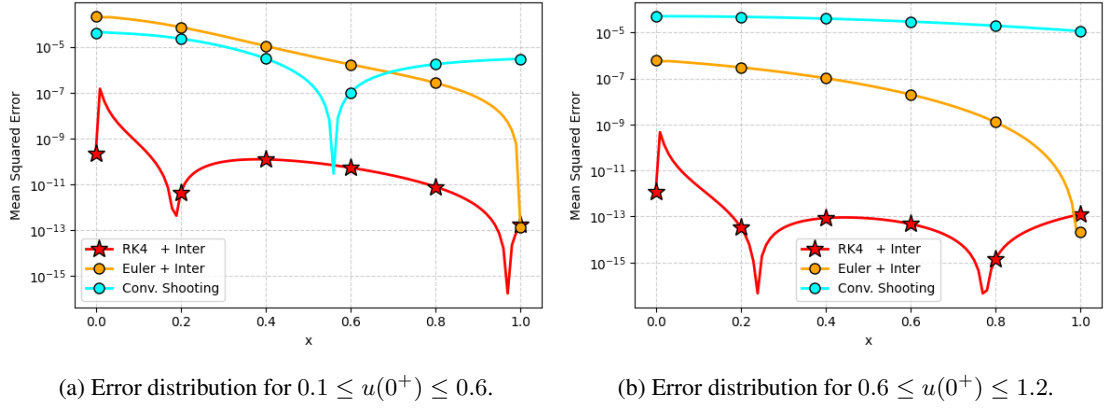


Figure 7: Comparison of error distributions for the second ODE.

 Table 3: Performance comparison with different ranges of $u(0^+)$ for the second ODE.

Range	Method	MSE	RMSE	MAE	MAXE	CPU
[0.1, 0.6]	RK4 + Shoot.	2.33×10^{-9}	4.83×10^{-5}	1.64×10^{-5}	3.84×10^{-4}	2.85×10^2
	Euler + Shoot.	3.75×10^{-5}	6.13×10^{-3}	4.18×10^{-3}	1.48×10^{-2}	2.82×10^2
	Conv. Shoot.	1.02×10^{-5}	3.19×10^{-3}	2.47×10^{-3}	6.76×10^{-3}	1.59×10^{-3}
[0.6, 1.2]	RK4 + Shoot.	7.18×10^{-12}	2.68×10^{-6}	8.17×10^{-7}	2.15×10^{-5}	2.74×10^2
	Euler + Shoot.	1.44×10^{-7}	3.79×10^{-4}	2.86×10^{-4}	7.91×10^{-4}	2.63×10^2
	Conv. Shoot.	3.38×10^{-5}	5.81×10^{-3}	5.69×10^{-3}	7.14×10^{-3}	2.97×10^{-3}

6.3. Novel Shooting-Interpolation Method for Third ODE

Given the singular second-order ODE from electromagnetic interaction as shown in Eq. (43).

$$u''(t) + \frac{4}{t}u'(t) + (tu(t) - 1)u(t) = 0, \quad (43)$$

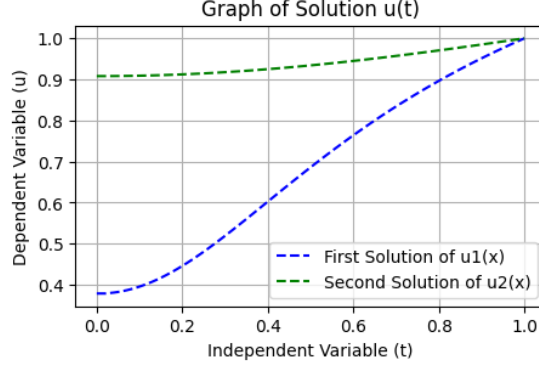


Figure 8: Two numerical solutions of the Second ODE.

with boundary conditions $u'(0^+) = 0$ and $u(5) = 0$ for $t \in (0, 5]$. The solution of this problem is hard to be obtained, therefore it will be computed numerically. By transforming Eq. (43) into a system of first-order ODEs and converting it into an IVP, the following system is obtained:

$$\begin{aligned} u'(t) &= v(t), \\ v'(t) &= -\frac{4}{t+\varepsilon}v(t) - (tu(t) - 1)u(t), \end{aligned} \quad (44)$$

for $0 < t \leq 5$. The next step is to discretize the domain interval $t \in (0, 5] \subseteq \mathbb{R}$ into N subintervals with a step size of $h = 0.01$; such that $N = 500$. Let $t_i = 0.01i$ for $i = 0, 1, 2, \dots, N$. The values of $u(t)$ and $v(t)$ can be represented as $u(t_i) = u_i$ and $v(t_i) = v_i$.

To obtain a high numerical accuracy, the RK4 method is employed. The RK4 coefficients are defined as follows:

$$\begin{aligned} k_1^u &= hv_i, \\ k_1^v &= h \left[-\frac{4}{t_i}v_i - (t_i u_i - 1)u_i \right], \\ k_2^u &= h \left(v_i + \frac{k_1^v}{2} \right), \\ k_2^v &= h \left[-\frac{4}{t_i + \varepsilon + h/2} \left(v_i + \frac{k_1^v}{2} \right) - \left(\left(t_i + \frac{h}{2} \right) \left(u_i + \frac{k_1^u}{2} \right) - 1 \right) \left(u_i + \frac{k_1^u}{2} \right) \right], \\ k_3^u &= h \left(v_i + \frac{k_2^v}{2} \right), \\ k_3^v &= h \left[-\frac{4}{t_i + \varepsilon + h/2} \left(v_i + \frac{k_2^v}{2} \right) - \left(\left(t_i + h/2 \right) \left(u_i + \frac{k_2^u}{2} \right) - 1 \right) \left(u_i + \frac{k_2^u}{2} \right) \right], \\ k_4^u &= h(v_i + k_3^v), \\ k_4^v &= h \left[-\frac{4}{t_i + \varepsilon + h} (v_i + k_3^v) - ((t_i + h)(u_i + k_3^u) - 1)(u_i + k_3^u) \right]. \end{aligned} \quad (45)$$

By using the RK4 formulation, the shooting-interpolation iteratively adjusts the unknown initial condition until the boundary condition at $t = 5$ is satisfied within a prescribed tolerance.

The next stage involves making two initial guesses for $u(0^+)$. Two possible initial guesses are $u(0^+) = 0.1$ and $u(0^+) = 5$. These yield $u(5) \approx 0.2605$ and $u(5) \approx -1.3080$ respectively,

resulting in $\beta_1 > 0$ and $\beta_2 < 0$. It produces $\gamma^* = 2.3520$ with $\text{err}(\gamma^*) \approx 5.0087 \cdot 10^{-6}$. The ODE in electromagnetic interaction given in Eq. (43) can also be defined over a different interval, such as $t \in (0, 10] \subseteq \mathbb{R}$ with boundary conditions $u'(0^+) = 0$ and $u(10) = 0$.

The solution with the new boundary setting is computed using the same approach as before, with a step size of $h = 0.01$ and initial guesses $u(0^+) = 2$ and $u(0^+) = 2.2$. These guesses yield $u(10) \approx 0.1293$ and $u(10) \approx -1.2254$, which satisfy $\beta_1 > 0$ and $\beta_2 < 0$. The method produces the predicted value of $u(0^+)$, namely $\gamma^* = 2.1199$, with $\text{err}(\gamma^*) \approx 3.168 \cdot 10^{-4}$, which is smaller if compared with Euler method of $\text{err}(\gamma^*) \approx 1.1492 \cdot 10^{-2}$.

In terms of computational efficiency, the proposed method requires a CPU computational time of 305.8060 seconds for 1000 iterations over the interval $t \in (0, 10]$, while for the shorter interval $t \in (0, 5]$, the CPU computational time is 273.1375 seconds under the same number of iterations. This indicates a moderate increase in computational cost as the domain length expands, while maintaining stable convergence behavior across both configurations.

To further investigate the convergence behavior, the iterative process was extended under a defined tolerance error of 1×10^{-5} . For the case defined over the interval $[0, 5]$, convergence was achieved after 460 iterations, as shown in Figure 11a. Based on the interpolation algorithm, the predicted value of $\gamma^* = 2.3520$ yields $u(5) = 0$, resulting in an absolute error of 9.9653×10^{-6} , which satisfies the convergence threshold. In contrast, for the ODE defined over the interval $[0, 10]$, the tolerance error was met after 484 iterations, as shown in Figure 11b. The interpolated value $\gamma^* = 2.1199$ yields $u(10) = 0$, resulting in an error of 4.9956×10^{-4} .

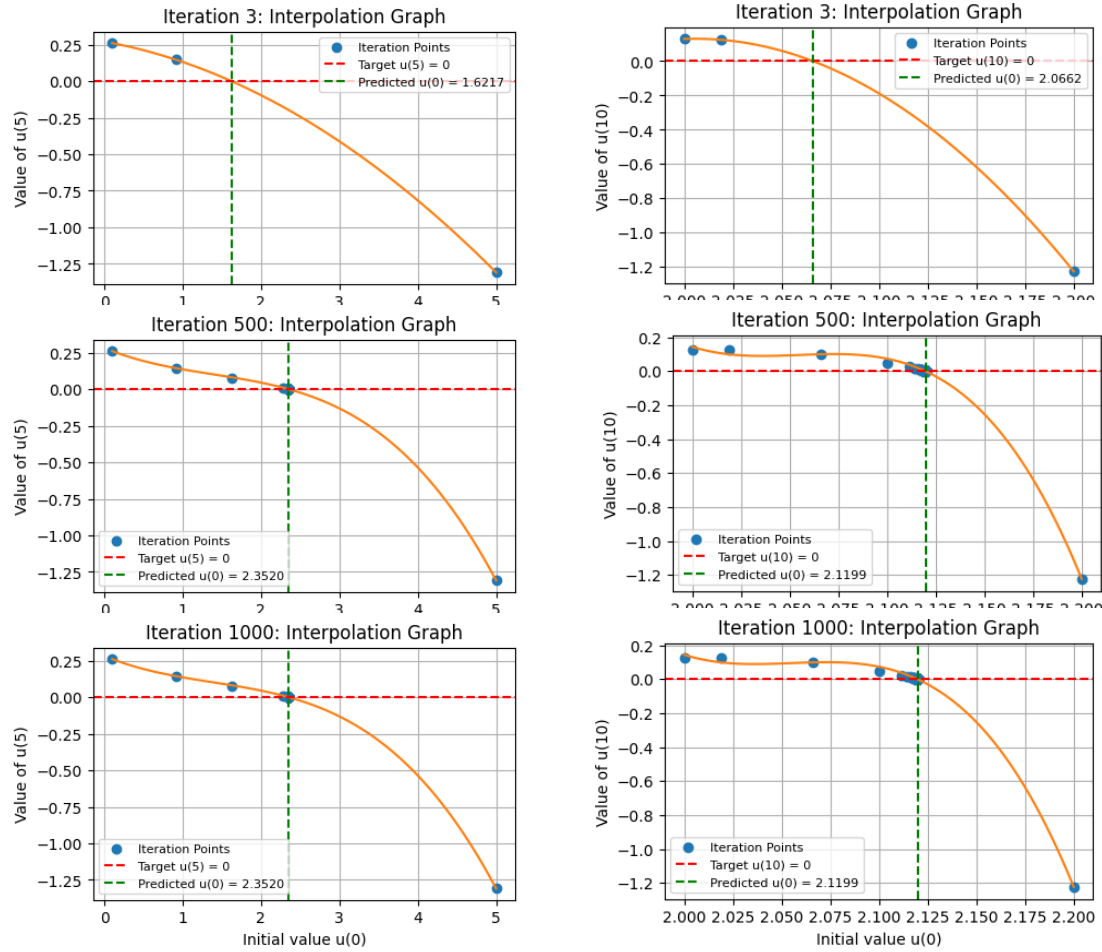


Figure 9: Interpolation graphs at iterations 3, 500, and 1000 for the third ODE.

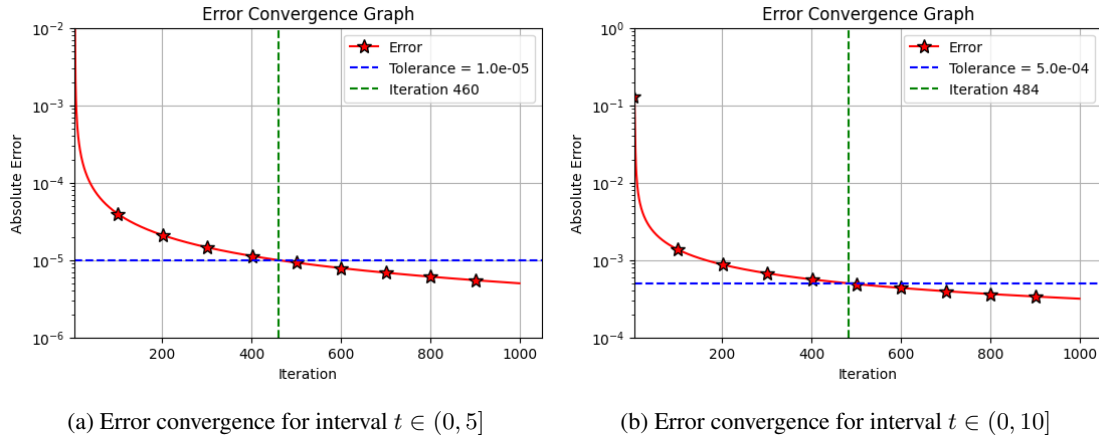


Figure 10: Error convergence of the third ODE for different intervals.

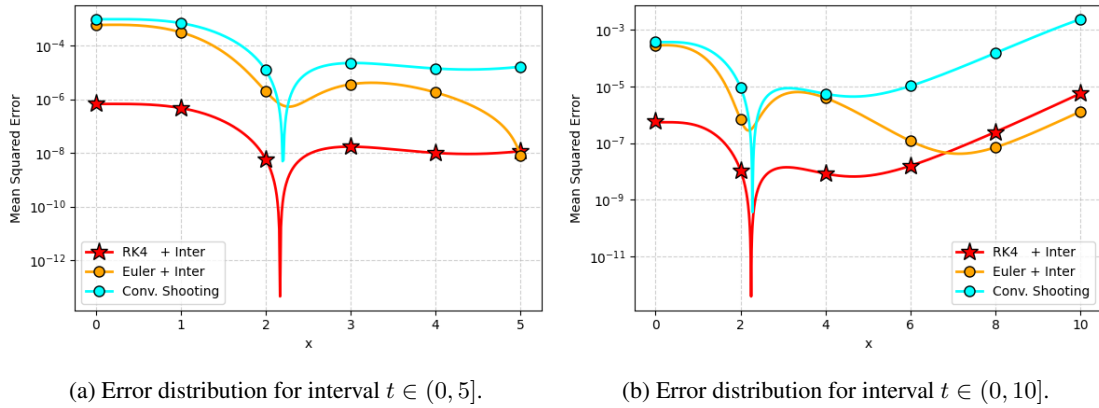


Figure 11: Comparative of error distribution for the third ODE.

Table 4: Performance comparison with different interval of t for the third ODE.

Range	Method	MSE	RMSE	MAE	MAXE	CPU
$t \in (0, 5]$	RK4 + Shoot.	1.69×10^{-7}	4.12×10^{-4}	2.96×10^{-4}	8.55×10^{-4}	2.73×10^2
	Euler + Shoot.	1.26×10^{-4}	1.12×10^{-2}	6.99×10^{-3}	2.45×10^{-2}	2.62×10^2
	Conv. Shoot.	2.47×10^{-4}	1.57×10^{-2}	1.13×10^{-2}	3.12×10^{-2}	1.59×10^{-3}
$t \in (0, 10]$	RK4 + Shoot.	4.31×10^{-7}	6.56×10^{-4}	4.38×10^{-4}	2.37×10^{-3}	3.05×10^2
	Euler + Shoot.	3.03×10^{-5}	5.50×10^{-3}	2.85×10^{-3}	1.68×10^{-2}	2.98×10^2
	Conv. Shoot.	2.25×10^{-4}	1.50×10^{-2}	1.05×10^{-2}	4.82×10^{-2}	3.97×10^{-3}

Figures 11 present the squared error distributions for the third ODE under two different intervals of t , namely $t \in (0, 5]$ and $t \in (0, 10]$, while the corresponding quantitative performance metrics are summarized in Table 4. In both cases, the proposed algorithm consistently provides the smallest error values among all considered approaches. For the interval $t \in (0, 5]$, the proposed method achieves an MSE of 1.69×10^{-7} , which is substantially lower than the Euler and conventional shooting methods. Similarly, for the interval $t \in (0, 10]$, the RK4-based approach still maintains the best performance with an MSE of 4.31×10^{-7} and the smallest RMSE, MAE, and MAXE values overall.

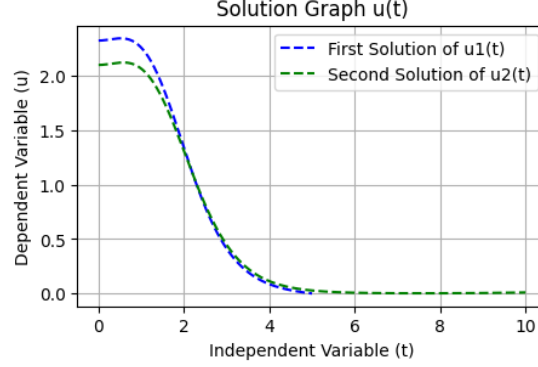


Figure 12: Numerical solutions of the third ODE on interval $t \in (0, 5]$ and $t \in (0, 10]$

The error distributions confirm these results. The proposed algorithm's curve remains significantly lower and more stable across the computational domain, indicating better numerical accuracy and stability. In contrast, the Euler and conventional shooting methods exhibit noticeably larger error magnitudes, particularly as the interval length increases. The conventional shooting method also shows the highest fluctuation and maximum error values in several regions of the domain. Although the RK4-based framework requires higher computational time compared with the conventional shooting approach, its superior accuracy and robustness demonstrate its effectiveness for solving the third ODE over different integration intervals.

7. Conclusions and Recommendations

This article introduces a novel Shooting-Interpolation framework for solving second-order singular boundary value problems (BVPs). The proposed method transforms the BVP into a system of first-order ordinary differential equations and reformulates it as an initial value problem, which is then solved using the RK4. The unknown initial condition is iteratively updated through an interpolation-based shooting strategy. Theoretical analysis confirms that the method is convergent under Lipschitz continuity assumptions, stable under small perturbations in the shooting parameter, and consistent with fourth-order global accuracy due to RK4 discretization.

The numerical and computational analyses further demonstrate the effectiveness of the proposed method. The RK4 integration ensures local truncation error of $\mathcal{O}(h^5)$ and global error of $\mathcal{O}(h^4)$, while the interpolation process progressively refines the shooting parameter toward the correct boundary satisfaction. The overall algorithm exhibits quasi-linear computational complexity $\mathcal{O}(N \cdot \text{iter}_{\max})$ and linear memory complexity $\mathcal{O}(N)$, making it suitable for large-scale discretization. In terms of performance, the proposed method remains computationally feasible even for large iteration and extended computational domains, while maintaining high numerical accuracy and stability throughout the solution process.

Recommendations for developing the shooting-interpolation method in future research are testing the novel shooting-interpolation method to solve non-singular BVPs, higher-order BVPs, and BVPs that represent real-world problems. It can also be applied to compare the performance between forward Euler, Heun, and Runge-Kutta methods in transforming BVP systems with boundary conditions into IVP systems within the novel shooting-interpolation method algorithm. Lastly, the proposed method can be further investigated by analyzing its stability and accuracy across various problem types. Expanding the method's application to more complex, non-linear BVPs and exploring adaptive step size techniques could enhance its efficiency. Additionally, integrating optimization algorithms for automatically selecting initial guesses could reduce dependency on manual interventions.

Acknowledgement

The authors gratefully acknowledge Brawijaya University for its ongoing support and encouragement during the course of this research. They also extend their appreciation to colleagues and reviewers for their constructive feedback, which has contributed to enhancing the quality of this work. This study was supported by a Professor Grant from Brawijaya University, Malang, East Java, Indonesia, under grant number 02172.9/UN10.F0901/B/KS/2024.

References

- Abell M. & Braselton J. 2016. *Differential Equations with Mathematica*, 4th Ed., Applications of Systems of Ordinary Differential Equations, pp. 565–612. Academic Press.
- Anam S., Ardiyansa S., Fitriah Z. & Muslikh M. 2026. Optimizing heart risk attack diagnosis by hybridizing xgboost and chaotic binary bat algorithm. *Applied Computational Intelligence and Soft Computing* p. 33.
- Ardiyansa S., Maharani N., Anam S. & Julianto E. 2024. Optimizing heart attack diagnosis using random forest with bat algorithm and greedy crossover technique. *BAREKENG: Jurnal Ilmu Matematika dan Terapan* **18**(2): 1053–1066.
- Ardiyansa S., Muslikh M. & Alghofari A. 2025. Binary hippopotamus algorithm with random forest for optimizing feature selection problem. *Numerical Algebra, Control and Optimization* **15**(4): 1176–1191.
- Arefin M., Nishu M., Dhali M. & Uddin M. 2022. Analysis of reliable solutions to the boundary value problems by using shooting method. *Mathematical Problems in Engineering* **188**(1): 538–548.
- Ascher U., Mattheij R. & Russell R. 1995. *Numerical solution of Boundary Value Problems for Ordinary Differential Equations*. Philadelphia: SIAM.
- Choi D., Leeghim H. & Kim D. 2021. Effective boundary value problem solver via bézier functions. *Symmetry* **13**(5): 736.
- Choi J., Chen Y., Lee H., Kim H. & Gel Y. 2024. Snn-pde: Learning dynamic pdes from data with simplicial neural networks. *Proceedings of the AAAI Conference on Artificial Intelligence*, pp. 11561–11569.
- Dai H., Huang D. & Liu Z. 2004. Singular dynamics with application to singular waves in physical problems. *Journal of the Physical Society of Japan* **73**: 1151–1155.
- Edun I. & Akinlabi G. 2021. Application of the shooting method for the solution of second order boundary value problems. *Journal of Physics: Conference Series*, p. 012020.
- Eshkuvatov Z., Laadjal Z. & Ismail S. 2025. Solving class of mixed nonlinear multi-term fractional volterra-fredholm integro-differential equations by new development of ham. *Mathematical Modelling and Analysis* **30**(1): 52–73.
- Ezhilarasi A. & Anbhuvizhi R. 2021. An analysis of solution of ordinary differential equations by new form of numerical methods. *International Conference on Advancements in Electrical, Electronics, Communication, Computing and Automation*, pp. 1–4.
- Fajri H., Ardiyansa S., Anam S., Maharani N. & Julianto E. 2025. Improving support vector machine performance with binary gaussian improved whale optimization algorithm: A case study on diabetes data. *BAREKENG: Jurnal Ilmu Matematika dan Terapan* **19**(4): 2531–2542.
- Filipov S., Gospodinov I. & Faragó I. 2017. Shooting-projection method for two-point boundary value problems. *Applied Mathematics Letters* **72**: 10–15.
- Gockenbach M. 2011. *Partial Differential Equations: Analytical and Numerical Methods*. 2nd Ed. Philadelphia: SIAM.
- Habibah U., Tuloli M., Rimanada V. & Ferreira T. 2020. Penyelesaian numerik masalah syarat batas robin pada persamaan diferensial cauchy-euler. *MAJAMATH: Jurnal Matematika dan Pendidikan Matematika* **3**(1): 32–40.
- Hohenegger M., Settanni G., Weinmüller E. & Wolde M. 2024. Numerical treatment of singular odes using finite difference and collocation methods. *Applied Numerical Mathematics* **205**: 184–194.
- Kellison A. & Appel A. 2022. *Verified Numerical Methods for Ordinary Differential Equations*. Cham: Springer.
- Khesat H., El-Ajou A., Al-Omari S., Alhazmi S. & Oqielat M. 2023. Exact and approximate solutions for linear and nonlinear partial differential equations via laplace residual power series method. *Axioms* **12**(7): 694.
- Latta G. 1961. Ordinary differential equations and their solutions. *American Mathematical Monthly* **68**(4): 392–393.
- Linge S. & Langtangen H. 2020. *Programming for Computations - Python*. Cham: Springer.

- Madenci E., Barut A. & Dorduncu M. 2019. *Peridynamic Differential Operator for Numerical Analysis*. Cham: Springer.
- Marjona K. 2024. Application of differential equations in various fields of science. *Journal of Applied Science and Technology* **4**(6): 76–81.
- Mohyud-Din S. 2010. On the conversion of partial differential equations. *Zeitschrift für Naturforschung A* **65**(11): 896–900.
- Morrison D., Riley J. & Zancanaro J. 1962. Multiple shooting method for two-point boundary value problems. *Communications of the ACM* **5**: 613–614.
- Moruzzi G. 2020. *Essential Python for the Physicist*. Cham: Springer.
- Oliveira C. 2016. *Options and Derivatives Programming in C++*. Berkeley: Apress.
- Polyanin A., Schiesser W. & Zhurov A. 2008. Partial differential equations. *Scholarpedia* **3**(10): 4605–4605.
- Rasedee A., Sathar M., Hamzah S., Ishak N., Wong T., Koo L. & Ibrahim S. 2021. Two-point block variable order step size multistep method for solving higher order ordinary differential equations directly. *Journal of King Saud University - Science* **33**(3): 101376.
- Rashed A. & Kassem M. 2023. Similarity transforming techniques of partial differential equations and its applications: A review. *Delta University Scientific Journal* **6**(2): 168–181.
- Sahani S., Sah A., Jha A. & Sahani K. 2024. Analytical frameworks: Differential equations in aerospace engineering. *ALSYSTECH Journal of Education Technology* **2**(1): 13–30.
- Smirnov V. 1964. Ordinary differential equations. *USSR Computational Mathematics and Mathematical Physics* **11**: 1–76.
- Svärd M. & Nordström J. 2013. Review of summation-by-parts schemes for initial-boundary-value problems. *Journal of Computational Physics* **268**: 17–38.
- Temple G. 1967. Differential-difference equations. *The Mathematical Gazette* **51**(377): 276–276.
- Yadav N., Yadav A. & Kumar M. 2015. *Applied Sciences and Technology*, An introduction to neural network methods for differential equations, pp. 1–10. Dordrecht: Springer.
- Zhang Y., Yong L., Chen Y., Yang J. & Zhang M. 2025. A hybrid multi-strategy differential creative search optimization algorithm and its applications. *Biomimetics* **10**: 356.
- Zhang Z., Cai Y. & Zhang D. 2020. Solving ordinary differential equations with adaptive differential evolution. *IEEE Access* **8**: 128908–128922.

Department of Mathematics

Faculty of Mathematics and Natural Sciences

Brawijaya University

65145 East Java

INDONESIA

E-mail: natashacm@student.ub.ac.id, safrizal@student.ub.ac.id, mslk@ub.ac.id*,
ummu_habibah@ub.ac.id

Received: 2 March 2026

Accepted: 13 June 2026

*Corresponding author