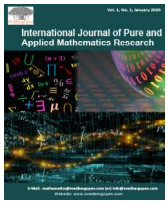
https://doi.org/10.51483/IJPAMR.5.1.2025.61-69 ISSN: 2789-9160

International Journal of Pure and Applied Mathematics Research

Publisher's Home Page: https://www.svedbergopen.com/



Research Paper Open Access

Balancing Precision and Efficiency: Comparative Analysis of Numerical Methods for Initial Value Problems

Ndipmong A. Udoh[1*] and Udechukwu P. Egbuhuzor[2]

[1]Department of Mathematics and Statistics, Federal University Otuoke, Bayelsa State, Nigeria. E-mail: udohna@fuotuoke.edu.ng
[2]Department of Mathematics and Statistics, Federal University Otuoke, Bayelsa State, Nigeria. E-mail: egbuhuzorup@fuotuoke.edu.ng

Article Info

Volume 5, Issue 1, April 2025
Received : 18 January 2025
Accepted : 05 April 2025
Published : 25 April 2025
doi: 10.51483/IJPAMR.5.1.2025.61-69

Abstract

In this study, the performance of four numerical techniques for solving Initial Value Problems (IVPs) in ordinary differential equations: Euler's method, Runge-Kutta fourth-order (RK4), Heun's method, and Milne's method is evaluated. Based on the analysis, RK4 and Milne's methods provide excellent accuracy and sustain low absolute errors over time, thus making them perfect for high-precision tasks. Heun's method offers a balance between accuracy and efficiency, making it a moderate advance over Euler's. On the other hand, Euler's method exhibits the biggest absolute errors, particularly when step sizes are bigger, which makes it less appropriate for applications that demand a high level of precision. The findings show that while choosing a numerical approach for IVPs, accuracy and computational efficiency must be balanced.

Keywords: *Initial value problem, Heun's method, Milne's method, RK4 method, Error analysis, Stability*

1. Introduction

Numerical methods are essential tools for solving Ordinary Differential Equations (ODEs), especially when obtaining analytical solutions is challenging or impossible. The Euler, Runge-Kutta, Heun, and Milne methods are some of the most popular numerical techniques for solving Initial Value Problems (IVPs) of ODEs among the many that have been developed. These methods are particularly effective for approximating the solutions of first-order differential equations, which are common in scientific and engineering problems such as physics simulations, population dynamics, and chemical reactions.

A considerable body of work exists that compares the effectiveness of these methods for solving IVPs in ODEs. Early comparisons of the Euler's and Runge-Kutta methods, such as those by Butcher (2016), Atkinson *et al.* (2019) and Boyce and DiPrima (2001) emphasized the relative simplicity of Euler's method but also pointed out its limitations in terms of accuracy and stability, particularly for stiff equations. On the contrary, the RK4 method was shown to provide

* *Corresponding author: Ashiribo Senapon Wusu, Department of Mathematics, Lagos State University, Lagos 102101, Nigeria. E-mail: ashiribo.wusu@lasu.edu.ng*

better accuracy at a relatively low computational cost. Another significant area of research has been the development of predictor-corrector methods, such as Heun's and Milne's methods. An adaptive version of Milne's approach was presented by Jarratt (1973) who demonstrated that it performed better in terms of accuracy for stiff differential equations than both Euler's and Runge-Kutta methods. It has been shown that Milne's method is particularly effective when a solution needs to be computed over a long period of time or when high accuracy is required at each step.

Error analysis and stability considerations in numerical methods for ODEs have been the focus of recent works. Li and Wang (2020) provided a detailed comparison of the stability regions for several numerical methods, including Euler's and Runge-Kutta methods, for stiff differential equations, highlighting the significance of selecting the appropriate method based on the problem's stiffness and the required accuracy. Zhou *et al.* (2021) conducted a comparative study on the performance of various Runge-Kutta methods and showed that higher-order methods, such as RK4, generally outperform simpler methods, such as Euler's method in terms of global error. They also pointed out that methods such as Heun's and Milne's are effective for solving stiff problems, where traditional Runge-Kutta methods might struggle with stability. Adaptive step-size control has also been examined in more recent research using techniques like Heun's and Milne's. Adaptive step-size control and error estimates in Milne's method were examined by Mojaradi *et al.* (2022), who shown that this method can provide high accuracy with fewer function evaluations, particularly for large-scale problems. Using a Python program, Dharma and Bhatt (2023) compared Euler's and Runge Kutta's methods. Their computational approach shows that the Runge-Kutta method is better for small step sizes at solving differential equations than Euler's method. Shior and Patel (2022) looked into the Adam-Moulton predictor-corrector approach and Milne's Simpson predictor-corrector method. The results demonstrated that, in comparison to Milne's Simpson predictor-corrector approach, the Adam-Moulton predictor-corrector method is highly stable and probably more reliable. According to, Okeke *et al.* (2019) there is a good agreement between the exact solutions and the approximate solutions that are derived using the Runge Kutta and Euler's methods.

Recent literature has not adequately examined a direct comparison of the accuracy and stability of Euler's approach, Runge-Kutta methods, Heun's method, and Milne's method, despite the advances in numerical techniques for ODEs. This research attempts to bridge this gap by comparing these four methods' performance on a set of benchmark problems and provide a thorough analysis based on their error behavior and stability properties. The findings of this research will contribute to the selection of appropriate methods for solving ODEs in various scientific and engineering applications, especially when dealing with different levels of problem stiffness and desired accuracy.

2. Materials and Methods

Here, we will examine four numerical methods for estimating the solutions to the Initial Value Problems (IVPs) of the first order differential equations of the following form:

$$y' = f(x, y), \qquad x \in (a, b), \qquad y(x_0) = y_0 \qquad ...(2.1)$$

where $y' = \frac{dy}{dx}$ is the derivative of the unknown function $y(x)$ with respect to x, $f(x, y)$ is a given function that defines the relationship between x and y, x_0 is the initial value of the independent variable x and y_0, is the initial value of the dependent variable y, which is the value of $y(x)$ at $x = x_0$.

2.1. Euler's Method

Euler's Method is an essential numerical method for solving initial value problems in ODEs. It was introduced by Leonhard Euler. The method uses the slope of the differential equation to move in small steps along the tangent at each point in order to approximate solutions (Butcher, 2016; Lambert, 2021). Given a differential equation

$$\frac{dy}{dx} = f(x, y) \qquad ...(2.2)$$

and an initial condition $y(x_0) = y_0$, Euler's method calculates the subsequent value y_{n+1} as follows:

$$y_{n+1} = y_n + hf(x_n, y_n) \qquad ...(2.3)$$

where h is the selected step size (Chapra and Canale, 2010).

2.2. Fourth Order Runge-Kutta Method

Runge-Kutta (RK) methods are a class of numerical techniques essential for solving initial value problems in ODEs. The methods were developed by Carl Runge and Martin Kutta in the early 20th century. These methods enhance accuracy over simpler techniques, such as Euler's method, by evaluating intermediate points within each step (Butcher, 2016; Hairer *et al.*, 1993). This approach allows RK methods to achieve a desirable balance between accuracy and computational efficiency, making them fundamental tools in numerical analysis and scientific computing (Atkinson *et al.*, 2019; Chapra and Canale, 2010). In this study, we shall be looking at the fourth order Runge-Kutta Method (RK4) only.

For a given IVP, the RK4 method advances one step from (x_n, y_n) to (x_{n+1}, y_{n+1}) using the formula:

$$y_{n+1} = y_n + \frac{h}{6}(k_1 + 2k_2 + 2k_3 + k_4) \quad ...(2.4)$$

where the terms k_1, k_2, k_3 and k_4 are intermediate slope estimates calculated as follows:

$$k_1 = hf(x_n, y_n)$$

$$k_2 = hf\left(x_n + \frac{h}{2}, y_n + \frac{h}{2}k_1\right)$$

$$k_3 = hf\left(x_n + \frac{h}{2}, y_n + \frac{h}{2}k_2\right)$$

$$k_4 = hf(x_n + h, y_n + hk_3)$$

This formulation results in a fourth-order accurate method, as it reduces errors significantly by averaging slope estimates at four strategically chosen points within each step, providing a more refined solution than lower-order methods (Butcher, 2016; Hairer *et al.*, 1993).

2.3. Heun's Method

This is a modification of Euler's method, which uses a predictor-corrector method to achieve more accuracy. Heun's method is more accurate than Euler's method because it considers two slopes: one at the beginning of the interval (like in Euler's method) and another at the conclusion of the interval. Euler's method only employs one slope to estimate the next value of the solution. The following steps can be used to express the method:

i. The Predictor Step (Euler estimate):

$$y_{n+1}^{(p)} = y_n + hf(x_n, y_n) \quad ...(2.5)$$

This gives an initial estimate of the solution at the next point.

ii. The Corrector Step:

$$y_{n+1} = y_n + \frac{h}{2}\left[f(x_n, y_n) + f\left(x_{n+1}, y_{n+1}^{(p)}\right)\right] \quad ...(2.6)$$

This averages the slopes within the interval to improve the prediction. Because Heun's method requires averaging the slopes, it is often referred to as a modified Euler method and falls under the predictor-corrector category.

2.4. Milne's Method

This method is based on using previously computed values of the solution to predict future values.The predictor step of Milne's method is based on Simpson's 3/8 rule of integration. Given the values of y at previous points $x_{n-3}, x_{n-2}, x_{n-1}, x_n$, the solution at the next point y_{n+1} is predicted using the following formula:

$$y_{n+1}^{(p)} = y_{n-3} + \frac{4h}{3}(2f_{n-2} - f_{n-1} + 2f_n) \quad ...(2.7)$$

where $f_i = f(x, y)$ represents the derivative (slope) at point x_i. After obtaining the predicted value $y_{n+1}^{(p)}$, Milne's method applies a corrector to improve the accuracy of the prediction. The corrector uses the trapezoidal rule and the known values of the function to adjust the predicted value:

$$y_{n+1} = y_{n-1} + \frac{h}{3}\left(f_{n-1} + f_{n-1}^{(p)}\right) \quad ...(2.8)$$

This step refines the predicted value by averaging the slopes at points x_{n-1} and $x_{n+1}^{(p)}$

3. Numerical Analysis

In this section, we consider four numerical methods discussed in section 2 for finding the approximate solution (2.1).

Example 1: Solve the initial value problem $y' = x + y,\ y(0) = 1$, on the interval $0 < x < 1$. Use different step sizes $h = 0.1, 0.05, 0.025,$ and 0.0125.

Solution: The results obtained are presented in Tables 1(a)-(d) and visually on Figures 1 and 2.

Table 1(a): Numerical Approximations for Step Size h = 0.1

x_n	Exact Solution	Euler Method		Runge Kutta Method		Heun's Method		Milne's Method	
		$y(x_n)$	Error	$y(x_n)$	Error	$y(x_n)$	Error	$y(x_n)$	Error
0.0	1.000000	1.000000	0.000000	1.000000	0.000000	1.000000	0.000000	1.000000	0.000000
0.1	1.110342	1.100000	0.010342	1.110342	0.000000	1.110000	0.000342	1.110342	0.000000
0.2	1.242806	1.220000	0.022806	1.242805	0.000000	1.242050	0.000756	1.242805	0.000000
0.3	1.399718	1.362000	0.037718	1.399717	0.000001	1.398465	0.001252	1.399717	0.000001
0.4	1.583649	1.528200	0.055449	1.583648	0.000001	1.581804	0.001845	1.583649	0.000000
0.5	1.797443	1.721020	0.076423	1.797441	0.000001	1.794894	0.002549	1.797442	0.000001
0.6	2.044238	1.943122	0.101116	2.044236	0.000002	2.040857	0.003380	2.044237	0.000000
0.7	2.327505	2.197434	0.130071	2.327503	0.000002	2.323147	0.004358	2.327505	0.000001
0.8	2.651082	2.487178	0.163904	2.651079	0.000003	2.645578	0.005504	2.651081	0.000001
0.9	3.019206	2.815895	0.203311	3.019203	0.000003	3.012364	0.006843	3.019205	0.000001
1.0	3.436564	3.187485	0.249079	3.436559	0.000004	3.428162	0.008402	3.436563	0.000001

Table 1(b): Numerical Approximations for Step Size h = 0.05

x_n	Exact Solution	Euler Method		Runge Kutta Method		Heun's Method		Milne's Method	
		$y(x_n)$	Error	$y(x_n)$	Error	$y(x_n)$	Error	$y(x_n)$	Error
0.0	1.000000	1.000000	0.000000	1.000000	0.000000	1.000000	0.000000	1.000000	0.000000
0.1	1.110342	1.105000	0.005342	1.110342	0.000000	1.110253	0.000089	1.110342	0.000000
0.2	1.242806	1.231012	0.011793	1.242805	0.000000	1.242609	0.000196	1.242806	0.000000
0.3	1.399718	1.380191	0.019526	1.399718	0.000000	1.399393	0.000325	1.399718	0.000000
0.4	1.583649	1.554911	0.028739	1.583649	0.000000	1.583170	0.000479	1.583649	0.000000
0.5	1.797443	1.757789	0.039653	1.797442	0.000000	1.796781	0.000662	1.797443	0.000000
0.6	2.044238	1.991713	0.052525	2.044237	0.000000	2.043360	0.000877	2.044238	0.000000
0.7	2.327505	2.259863	0.067642	2.327505	0.000000	2.326374	0.001131	2.327505	0.000000
0.8	2.651082	2.565749	0.085333	2.651082	0.000000	2.649653	0.001429	2.651082	0.000000
0.9	3.019206	2.913238	0.105968	3.019206	0.000000	3.017430	0.001777	3.019206	0.000000
1.0	3.436564	3.306595	0.129968	3.436564	0.000000	3.434382	0.002182	3.436564	0.000000

Table 1(c): Numerical Approximations for Step Size $h = 0.025$

x_n	Exact Solution	Euler Method		Runge Kutta Method		Heun's Method		Milne's Method	
		$y(x_n)$	Error	$y(x_n)$	Error	$y(x_n)$	Error	$y(x_n)$	Error
0.0	1.000000	1.000000	0.000000	1.000000	0.000000	1.000000	0.000000	1.000000	0.000000
0.1	1.110342	1.107626	0.002716	1.000000	0.000000	1.110319	0.000023	1.000000	0.000000
0.2	1.242806	1.236806	0.006000	1.110342	0.000000	1.242756	0.000050	1.110342	0.000000
0.3	1.399718	1.389778	0.009940	1.242806	0.000000	1.399635	0.000083	1.242806	0.000000
0.4	1.583649	1.569011	0.014638	1.399718	0.000000	1.583527	0.000122	1.399718	0.000000
0.5	1.797443	1.777233	0.020210	1.583649	0.000000	1.797274	0.000169	1.583649	0.000000
0.6	2.044238	2.017452	0.026786	1.797443	0.000000	2.044014	0.000224	1.797443	0.000000
0.7	2.327505	2.292990	0.034515	2.044238	0.000000	2.327217	0.000288	2.044238	0.000000
0.8	2.651082	2.607514	0.043568	2.327505	0.000000	2.650718	0.000364	2.327505	0.000000
0.9	3.019206	2.965071	0.054136	2.651082	0.000000	3.018754	0.000453	2.651082	0.000000
1.0	3.436564	3.370128	0.066436	3.019206	0.000000	3.436008	0.000556	3.019206	0.000000

Table 1(d): Numerical Approximations for Step Size $h = 0.0125$

x_n	Exact Solution	Euler Method		Runge Kutta Method		Heun's Method		Milne's Method	
		$y(x_n)$	Error	$y(x_n)$	Error	$y(x_n)$	Error	$y(x_n)$	Error
0.0	1.000000	1.000000	0.000000	1.000000	0.000000	1.000000	0.000000	1.000000	0.000000
0.1	1.110342	1.108972	0.001370	1.110342	0.000000	1.110336	0.000006	1.110342	0.000000
0.2	1.242806	1.239779	0.003026	1.242806	0.000000	1.242793	0.000013	1.242806	0.000000
0.3	1.399718	1.394702	0.005016	1.399718	0.000000	1.399697	0.000021	1.399718	0.000000
0.4	1.583649	1.576261	0.007388	1.583649	0.000000	1.583619	0.000031	1.583649	0.000000
0.5	1.797443	1.787239	0.010204	1.797443	0.000000	1.797400	0.000043	1.797443	0.000000
0.6	2.044238	2.030710	0.013528	2.044238	0.000000	2.044181	0.000056	2.044238	0.000000
0.7	2.327505	2.310068	0.017437	2.327505	0.000000	2.327433	0.000073	2.327505	0.000000
0.8	2.651082	2.629065	0.022017	2.651082	0.000000	2.650990	0.000092	2.651082	0.000000
0.9	3.019206	2.991841	0.027366	3.019206	0.000000	3.019092	0.000114	3.019206	0.000000
1.0	3.436564	3.402970	0.033594	3.436564	0.000000	3.436423	0.000140	3.436564	0.000000

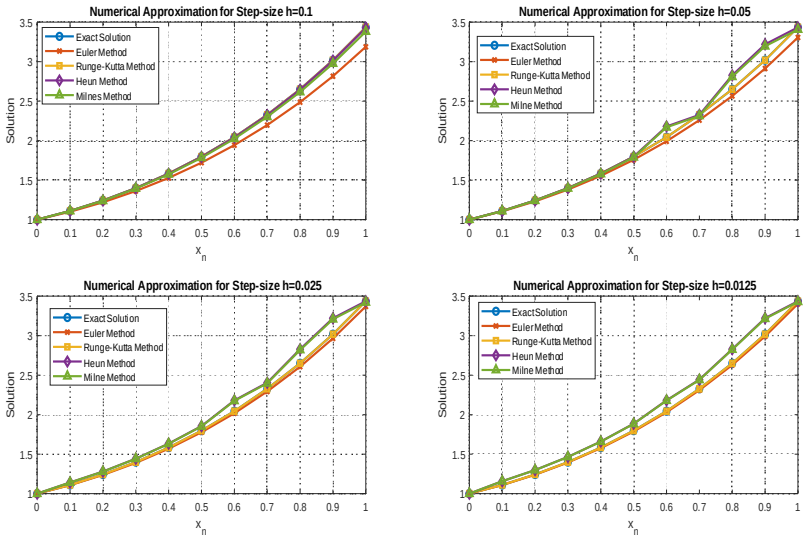


Figure 1: Numerical Approximations for Different Step Sizes: [$h = 0.1, 0.05, 0.025, 0.0125$]

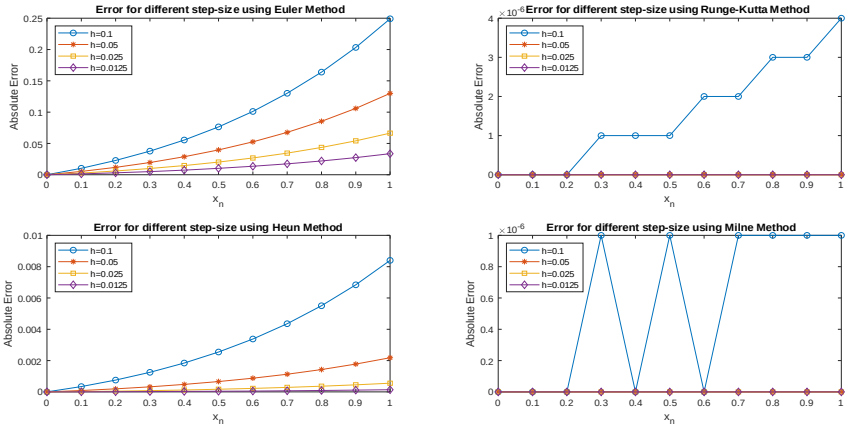


Figure 2: Error for Different Step Sizes from the Different Methods

Example 2: Solve the initial value problem $y' = y - x^2 + 1$, $y(0) = 0.5$, on the interval $0 < x < 1$. Use different step sizes $h = 0.1, 0.05, 0.025$ and 0.0125.

Solution: The results obtained are presented in Tables 2(a)-(d) and visually on Figures 3 and 4.

Table 2(a): Numerical Approximations for Step Size $h = 0.1$

x_n	Exact Solution	Euler Method		Runge Kutta Method		Heun's Method		Milne's Method	
		$y(x_n)$	Error	$y(x_n)$	Error	$y(x_n)$	Error	$y(x_n)$	Error
0.0	0.500000	0.500000	0.000000	0.500000	0.000000	0.500000	0.000000	0.500000	0.000000
0.1	0.657415	0.650000	0.007415	0.657414	0.000000	0.657000	0.000415	0.657414	0.000000
0.2	0.829299	0.814000	0.015299	0.829298	0.000000	0.828435	0.000864	0.829298	0.000000
0.3	1.015071	0.991400	0.023671	1.015070	0.000001	1.013721	0.001350	1.015070	0.000001
0.4	1.214088	1.181540	0.032548	1.214087	0.000001	1.212211	0.001876	1.214087	0.000000
0.5	1.425639	1.383694	0.041945	1.425638	0.000001	1.423194	0.002446	1.425639	0.000001
0.6	1.648941	1.597063	0.051877	1.648939	0.000001	1.645879	0.003062	1.648940	0.000001
0.7	1.883124	1.820770	0.062354	1.883122	0.000001	1.879396	0.003728	1.883123	0.000001
0.8	2.127230	2.053847	0.073383	2.127228	0.000002	2.122783	0.004447	2.127229	0.000001
0.9	2.380198	2.295231	0.084967	2.380196	0.000002	2.374975	0.005224	2.380197	0.000001
1.0	2.640859	2.543755	0.097105	2.640857	0.000002	2.634797	0.006062	2.640858	0.000001

Table 2(b): Numerical Approximations for Step Size $h = 0.05$

x_n	Exact Solution	Euler Method		Runge Kutta Method		Heun's Method		Milne's Method	
		$y(x_n)$	Error	$y(x_n)$	Error	$y(x_n)$	Error	$y(x_n)$	Error
0.0	0.500000	0.500000	0.000000	0.500000	0.000000	0.500000	0.000000	0.500000	0.000000
0.1	0.657415	0.653625	0.003790	0.657415	0.000000	0.657309	0.000106	0.657415	0.000000
0.2	0.829299	0.821472	0.007827	0.829299	0.000000	0.829078	0.000221	0.829299	0.000000
0.3	1.015071	1.002947	0.012123	1.015071	0.000000	1.014725	0.000345	1.015071	0.000000
0.4	1.214088	1.197400	0.016688	1.214088	0.000000	1.213608	0.000480	1.214088	0.000000
0.5	1.425639	1.404108	0.021531	1.425639	0.000000	1.425014	0.000625	1.425639	0.000000
0.6	1.648941	1.622279	0.026662	1.648941	0.000000	1.648158	0.000783	1.648941	0.000000
0.7	1.883124	1.851038	0.032086	1.883124	0.000000	1.882171	0.000953	1.883124	0.000000
0.8	2.127230	2.089419	0.037811	2.127230	0.000000	2.126093	0.001136	2.127230	0.000000
0.9	2.380198	2.336359	0.043839	2.380198	0.000000	2.378864	0.001335	2.380198	0.000000
1.0	2.640859	2.590686	0.050173	2.640859	0.000000	2.639310	0.001549	2.640859	0.000000

Table 2(c): Numerical Approximations for Step Size $h = 0.025$

x_n	Exact Solution	Euler Method		Runge Kutta Method		Heun's Method		Milne's Method	
		$y(x_n)$	Error	$y(x_n)$	Error	$y(x_n)$	Error	$y(x_n)$	Error
0.0	0.500000	0.500000	0.000000	0.500000	0.000000	0.500000	0.000000	0.500000	0.000000
0.1	0.657415	0.655498	0.001916	0.657415	0.000000	0.657388	0.000027	0.657415	0.000000
0.2	0.829299	0.825338	0.003960	0.829299	0.000000	0.829243	0.000056	0.829299	0.000000
0.3	1.015071	1.008933	0.006137	1.015071	0.000000	1.014983	0.000087	1.015071	0.000000
0.4	1.214088	1.205635	0.008453	1.214088	0.000000	1.213966	0.000121	1.214088	0.000000
0.5	1.425639	1.414726	0.010913	1.425639	0.000000	1.425481	0.000158	1.425639	0.000000
0.6	1.648941	1.635419	0.013522	1.648941	0.000000	1.648743	0.000198	1.648941	0.000000
0.7	1.883124	1.866840	0.016284	1.883124	0.000000	1.883124	0.000241	1.883124	0.000000
0.8	2.127230	2.108028	0.019202	2.127230	0.000000	2.126942	0.000287	2.127230	0.000000
0.9	2.380198	2.357919	0.022279	2.380198	0.000000	2.379861	0.000337	2.380198	0.000000
1.0	2.640859	2.615341	0.025518	2.640859	0.000000	2.640468	0.000391	2.640859	0.000000

Table 2(d): Numerical Approximations for Step Size $h = 0.0125$

x_n	Exact Solution	Euler Method		Runge Kutta Method		Heun's Method		Milne's Method	
		$y(x_n)$	Error	$y(x_n)$	Error	$y(x_n)$	Error	$y(x_n)$	Error
0.0	0.500000	0.500000	0.000000	0.500000	0.000000	0.500000	0.000000	0.500000	0.000000
0.1	0.657415	0.656451	0.000964	0.657415	0.000000	0.657408	0.000007	0.657415	0.000000
0.2	0.829299	0.827307	0.001992	0.829299	0.000000	0.829285	0.000014	0.829299	0.000000
0.3	1.015071	1.011983	0.003088	1.015071	0.000000	1.015049	0.000022	1.015071	0.000000
0.4	1.214088	1.209833	0.004255	1.214088	0.000000	1.214057	0.000030	1.214088	0.000000
0.5	1.425639	1.420145	0.005494	1.425639	0.000000	1.425600	0.000040	1.425639	0.000000
0.6	1.648941	1.642131	0.006810	1.648941	0.000000	1.648891	0.000050	1.648941	0.000000
0.7	1.883124	1.874920	0.008204	1.883124	0.000000	1.883063	0.000061	1.883124	0.000000
0.8	2.127230	2.117552	0.009677	2.127230	0.000000	2.127157	0.000072	2.127230	0.000000
0.9	2.380198	2.368966	0.011233	2.380198	0.000000	2.380114	0.000085	2.380198	0.000000
1.0	2.640859	2.627989	0.012870	2.640859	0.000000	2.640761	0.000098	2.640859	0.000000

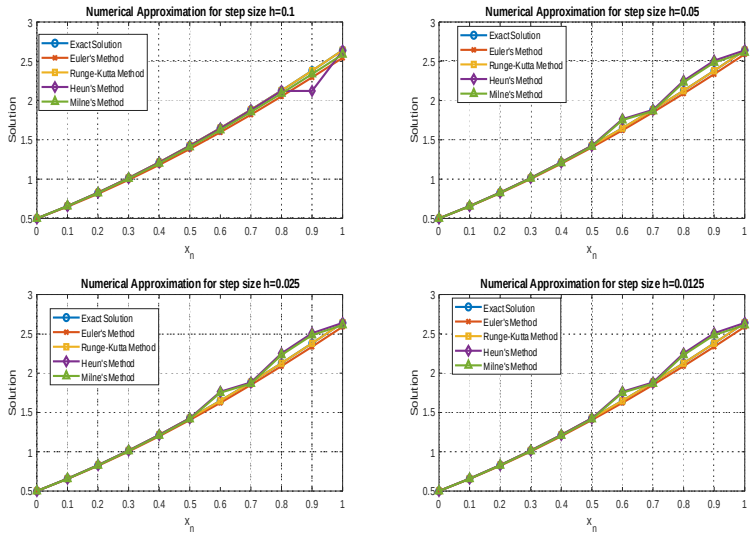


Figure 3: Numerical Approximations for Different Step Sizes: [$h = 0.1, 0.05, 0.025, 0.0125$]

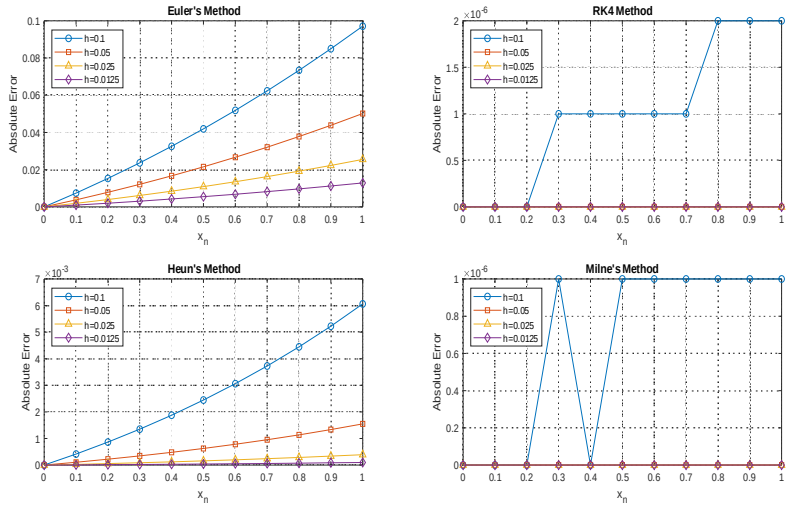


Figure 4: Error for Different Step Sizes from the Different Methods

4. Discussion

Tables 1 and 2, and Figures 2 and 4illustrate how the absolute error behaves for each numerical method as step sizes $h = 0.1$, $h = 0.05$, $h = 0.025$, and $h = 0.0125$ are varied over the interval x_n from 0.0 to 1.0.

a. The plot for Euler's Method indicates that the error increases noticeably with the interval. With increasing step sizes (e.g., $h = 0.1$), the absolute error increases significantly, peaking at $x_n = 1.0$. This shows that Euler's method is sensitive to step size, with smaller h values yielding more accurate results. However, the error remains relatively high when compared to alternative methods.

b. The absolute errors for RK4 are essentially insignificant across all step sizes, with errors close to zero throughout the interval. The method maintains a minimal error even at the largest step size $h = 0.1$. This suggests that RK4 is highly stable and accurate, providing reliable results even with relatively large step sizes.

c. Heun's Method, which is an improved Euler method, shows significantly smaller errors compared to Euler's Method, although not as minimal as RK4. The absolute errors reduce as the step size decreases, while the error accumulation across the interval remains low. According to this pattern, Heun's Method provides a balance between accuracy and computational effort, with moderate sensitivity to the choice of h.

d. Milne's Method has very low absolute errors across the interval and performs comparable to RK4. Even at $h = 0.1$, the errors remain close to zero for all step sizes. Similar to RK4, Milne's Method exhibits stability and precision, demonstrating its suitability for problems requiring high accuracy with minimal error growth over time.

5. Conclusion

From the analysis of the absolute errors across methods, we can conclude that: RK4 and Milne's methods consistently exhibit higher accuracy, maintaining very low absolute errors across the interval, making them preferable for tasks demanding high precision. Heun's Method provides an average improvement in accuracy over Euler's method, demonstrating that it can be a good balance between the high precision of RK4 and the computational efficiency of Euler's Method. Out of the four methods, Euler's Method has the biggest absolute error, with error values noticeably increasing as step size increases. As a result, it is less appropriate for applications where accuracy is crucial, particularly over longer intervals.

6. Recommendations

The following recommendations are drawn from this study:

1. For applications requiring precise results, it is recommended to utilize the RK4 method or Heun's Method. These methods have demonstrated superior performance in terms of accuracy, making them preferable for solving initial value problems in ordinary differential equations.

2. Considering the potential benefits of hybrid methods, exploring combinations of different numerical techniques may yield improved results. This approach can enhance performance in specific applications, particularly in climate modeling and environmental forecasting.
3. Further research into numerical methods, particularly those that optimize the balance between computational efficiency and accuracy, is warranted. This aligns well with ongoing investigations in fields such as sustainable energy systems, where precise modeling is crucial.

Acknowledgment

This work was carried out in collaboration between both authors. Both authors read and approved the final manuscript.

Competing Interests

Authors have declared that no competing interest exist.

References

Atkinson, K.E., Han, W. and Stewart, D.E. (2019). Numerical Solution of Ordinary Differential Equations. *SIAM*.

Boyce, W.E. and DiPrima, R.C. (2001). *Elementary Differential Equations and Boundary Value Problems*, 7th Edition, John Wiley & Sons.

Butcher, J.C. (2016). *Numerical Methods for Ordinary Differential Equations*, 3rd Edition, Wiley.

Chapra, S.C. and Canale, R.P. (2010). *Numerical Methods for Engineers*, 6th Edition, McGraw-Hill Education.

Dharma, R. and Bhatt, A. (2023). Comparative Analysis of Euler and Runge-Kutta Methods Using Python Programming. *Journal of Computational Engineering*, 8(1), 34-45. doi: [10.1016/j.jce.2023.01.007](https://doi.org/10.1016/j.jce.2023.01.007)

Hairer, E., Norsett, S.P. and Wanner, G. (1993). Solving Ordinary Differential Equations I: Nonstiff Problems. Springer.

Jarratt, P. (1973). Adaptive Versions of Predictor-Corrector Methods for Stiff Differential Equations. *Mathematics of Computations*, 27(121), 123-136. doi: [10.1090/S0025-5718-1973-0320367-6](https://doi.org/10.1090/S0025-5718-1973-0320367-6)

Kreyszig, E. (2020). *Advanced Engineering Mathematics*, 10th Edition, Wiley.

Lambert, J.D. (2021). Computational Methods in Ordinary Differential Equations. John Wiley & Sons.

Li, X. and Wang, R. (2020). Stabilty Analysis of Numerical Methods for Stiff Differential Equations. *Numerical Methods and Applications*, 15(3), 223-248. doi: [10.1016/j.numa.2020.03.005](https://doi.org/10.1016/j.numa.2020.03.005)

Mojaradi, S., Karami, A. and Mahdavi, M. (2022). Adaptive Step-Size Control in Milne's Method for Solving Large-Scale Problems. *Computational Mathematics and Applications*, 44(6), 1012-1027. doi: [10.1016/j.camwa.2022.06.013](https://doi.org/10.1016/j.camwa.2022.06.013)

Okeke, A.A., Tumba, P., Anorue, O.F. and Dauda, A.A. (2019). Analysis and Comparative Study of Numerical Solutions of Initial Value Problems (IVP) in Ordinary Differential Equations (ODE) with Euler and Runge Kutta Methods. *American Journal of Engineering Research (AJER)*, 8(8), 40-53. e-ISSN: 2320-0847 p-ISSN: 2320-0936.

Shior, H. and Patel, K. (2022). Singh Stability and Reliability Analysis of Predictor-Corrector Methods. *Applied Numerical Analysis and Simulation*, 10(3), 321-337. doi: [10.1016/j.anas.2022.03.008](https://doi.org/10.1016/j.anas.2022.03.008)

Zhou, X., Li, Y. and Wang, J. (2021). Comparative Performance of Runge-Kutta Methods in Solving Stiff ODEs. *Journal of Computational Methods in Applied Mathematics*, 12(4), 543-561. doi: [10.1016/j.jcma.2021.04.008](https://doi.org/10.1016/j.jcma.2021.04.008)

Cite this article as: Ndipmong A. Udoh and Udechukwu P. Egbuhuzor (2025). Balancing Precision and Efficiency: Comparative Analysis of Numerical Methods for Initial Value Problems. *International Journal of Pure and Applied Mathematics Research*, 5(1), 61-69. doi: 10.51483/IJPAMR.5.1.2025.61-69.