Comprehensive Journal of Science
Volume (10), Issue (39), (March, 2026)
ISSN: 3014-6266

مجلة العلوم الشاملة
المجلد (10) ملحق العدد 39 مارس 2026
ردمد: 3014-6266

# A Comparative Numerical Study of Classical and Adaptive Methods for Solving Nonlinear Differential Equations Using MATLAB

**Hanan Mohammed Saleh Ahmed**[1]
[1] *Faculty of Natural Resources Engineering, Bir Al-Ghanam- University of Zawia- Libya*
h.salih@zu.edu.lyn

**دراسة عددية مقارنة بين الطرق الكلاسيكية والتكيفية لحل المعادلات التفاضلية غير الخطية**

**حنان محمد صالح أحمد[1]**
**[1] كلية هندسة الموارد الطبيعية، بئر الغنم – جامعة الزاوية – ليبيا**
h.salih@zu.edu.lyn

تاريخ الاستلام:2026/02/11 –تاريخ المراجعة:2026/03/12– تاريخ القبول: 2026/03/13– تاريخ للنشر2026/04/22

## 1. Abstract

This study presents a comparative numerical analysis for solving a class of nonlinear differential equations for which closed-form analytical solutions are difficult to obtain. The aim of the study is to evaluate the efficiency of several numerical integration algorithms, including the Euler method, the fourth-order Runge–Kutta method (RK4), the implicit method, as well as the adaptive solver ode45 within the MATLAB environment, in terms of numerical accuracy, stability, computational time, and the effect of time step size. The methods were applied to three representative mathematical models exhibiting different types of nonlinear behavior. The absolute error, convergence rate, and sensitivity to time step variations were computed. The results showed that the RK4 method achieves the best balance between accuracy and computational efficiency, while ode45 provides the highest numerical stability in cases involving rapid variations. In contrast, the Euler method remains the least accurate despite its computational speed. The scientific contribution of this study lies in providing a unified comparative framework that combines error analysis, numerical verification of convergence order, and time-step sensitivity analysis, thereby supporting the selection of the most appropriate algorithm for scientific and engineering applications.

**Keywords:** nonlinear differential equations, numerical analysis, Runge–Kutta method, absolute error analysis, computational efficiency.

## الملخص

تقدم هذه الدراسة تحليلاً عددياً مقارناً لحل فئة من المعادلات التفاضلية غير الخطية التي يصعب الحصول على حلول تحليلية مغلقة لها. يهدف هذا البحث إلى تقييم كفاءة عدة خوارزميات تكامل عددية، بما في ذلك طريقة أويلر، وطريقة رونج–كوتا من الرتبة الرابعة (RK4) ، والطريقة الضمنية، بالإضافة إلى المحلل التكيفي **ode45** ضمن بيئة MATLAB ، وذلك من حيث الدقة العددية، والاستقرار، وزمن الحساب، وتأثير حجم خطوة الزمن. تم تطبيق هذه الطرق على ثلاثة نماذج رياضية تمثيلية تُظهر أنواعاً مختلفة من السلوك غير الخطي. كما تم حساب الخطأ المطلق، ومعدل التقارب، والحساسية لتغيرات خطوة الزمن. أظهرت النتائج أن طريقة RK4 تحقق أفضل توازن بين الدقة والكفاءة الحسابية، في حين يوفر **ode45** أعلى استقرار عددي في الحالات التي تتضمن تغيرات سريعة. وعلى النقيض، تبقى طريقة أويلر الأقل دقة رغم سرعتها الحسابية. تكمن المساهمة العلمية لهذه الدراسة في تقديم إطار مقارن موحد يجمع بين تحليل

الخطأ، والتحقق العددي من رتبة التقارب، وتحليل حساسية خطوة الزمن، مما يدعم اختيار الخوارزمية الأنسب للتطبيقات العلمية والهندسية.

**الكلمات المفتاحية :**المعادلات التفاضلية غير الخطية، التحليل العددي، طريقة رونج–كوتا، تحليل الخطأ المطلق، الكفاءة الحسابية.

## 2. Introduction

Nonlinear ordinary differential equations are considered fundamental mathematical tools for modeling complex dynamical systems in various fields such as physics, engineering, and medical sciences (Burden & Faires, 2016; Iserles, 2009).. Their importance lies in their ability to represent nonlinear relationships between variables, which leads to behaviors that are difficult to analyze using traditional analytical methods. In many practical applications, obtaining exact analytical solutions is either highly complex or not possible, making numerical methods the most effective approach for obtaining approximate solutions with controllable accuracy (Chapra & Canale, 2015; Lambert, 1991).

This research focuses on the numerical study of nonlinear differential equations using both classical and adaptive algorithms within the MATLAB environment, which provides advanced capabilities for numerical computation and result analysis (MathWorks, 2023). Classical methods include algorithms such as the Euler method and the Runge–Kutta method, which are characterized by their simplicity of implementation but rely on a fixed time step (Butcher, 2016; Dormand & Prince, 1980).. In contrast, adaptive algorithms adjust the time step automatically based on error estimation, thereby improving accuracy and reducing computational cost (Shampine & Reichelt, 1997).

Through this integration of different methods within MATLAB, a comprehensive comparison can be conducted in terms of accuracy, stability, and computational efficiency, with the aim of identifying the most suitable method for handling nonlinear equations in practical applications.

## 3.Theoretical Framework and Numerical Methods

This section presents the mathematical foundations and numerical methodologies used in studying and solving nonlinear differential equations. It clarifies the nature of these equations, the principles of numerical approximation, and reviews the main classical and adaptive methods employed in this research within the MATLAB environment. It also analyzes their characteristics in terms of accuracy, stability, and computational efficiency(Chapra & Canale, 2015; Iserles, 2009).

### 3.1 Nonlinear Differential Equations

Nonlinear differential equations are among the most important mathematical models used to describe complex dynamical systems in physics, engineering, and biological applications, due to their ability to represent nonlinear relationships between variables. These equations are characterized by the fact that they do not satisfy the superposition principle, as variables or their derivatives appear in nonlinear forms such as powers, products, or nonlinear functions. A first-order initial value problem is generally expressed in the form:

$$\frac{dy}{dt} = f(t, y), y(t_0) = y_0 \quad (1)$$

where *f(t, y)* represents a nonlinear function. In most practical cases, obtaining a closed-form analytical solution is difficult, which makes numerical methods the primary approach for obtaining accurate approximate solutions(Hairer & Wanner, 2010; Iserles, 2009).

### 3.2 Numerical Basis of Time Approximation

All numerical methods are based on discretizing the time domain **$[t_0, t_f]$** into discrete points using a time step h, such that the solution is computed iteratively according to:

$$t_{n+1} = t_n + h \quad (2)$$

The solution $y_n \approx y(t_n)$ is approximated at each time step. The quality of a numerical method is assessed through three main properties: local truncation error, global error, and numerical stability, which collectively reflect the accuracy of the method and its behavior during the time integration process(Burden & Faires, 2016; Chapra & Canale, 2015).

### 3.3 Explicit Euler Method

The Euler method is one of the simplest numerical techniques used for solving differential equations. It is based on approximating the derivative using the slope of the function at the current point:

Starting from $(t_0, y_0)$ we compute:

$$k_1 = f(t_n, y_n), \quad (3)$$

The solution is then updated using the relation:

$$y_{n+1} = y_n + hk_1 \quad (4)$$

Despite its simplicity, this method has limited accuracy and requires very small time steps to ensure numerical stability, especially in rapidly changing nonlinear systems(Butcher, 2016).

### 3.4 Fourth-Order Runge–Kutta Method (RK4)

The RK4 method is one of the most widely used numerical techniques due to its high accuracy and its balance between computational cost and performance. It is based on computing four slopes within each time step:

$$k_1 = f(t_n, y_n), \quad (5)$$

$$k_2 = f\left(t_n + \frac{h}{2}, y_n + \frac{h}{2}k_1\right) \quad (6)$$

$$k_3 = f\left(t_n + \frac{h}{2}, y_n + \frac{h}{2}k_2\right), \qquad (7)$$

$$k_4 = f(t_n + h, y_n + hk_3) \qquad (8)$$

The solution is then updated as:

$$y_{n+1} = y_n + \frac{h}{6}(k_1 + 2k_2 + 2k_3 + k_4) \qquad (9)$$

This method provides high accuracy and better stability compared to lower-order methods, making it suitable for most nonlinear problems(Dormand & Prince, 1980; Butcher, 2016).

### 3.5 Implicit Methods

Implicit methods rely on including the future value $y_{n+1}$ in the update equation, leading to the form:

$$y_{n+1} = y_n + hf(t_{n+1}, y_{n+1}) \qquad (10)$$

This requires solving a nonlinear equation at each time step using iterative techniques such as the Newton–Raphson method. Although they involve higher computational cost, implicit methods are characterized by strong numerical stability, particularly for stiff problems, where they can handle large time steps without losing stability(Hairer & Wanner, 2010; Lambert, 1991).

### 3.6 Adaptive Methods in MATLAB

Adaptive algorithms rely on automatically adjusting the time step based on error estimation during the solution process. The MATLAB function ode45 is one of the most well-known methods of this type, as it is based on the Dormand–Prince embedded Runge–Kutta method for estimating local truncation error. This method computes two solutions of different orders at each step and uses the difference between them to estimate the error. The time step (h) is then automatically adjusted. This approach allows the use of smaller steps when high accuracy is required and larger steps when the solution is smooth, achieving an effective balance between accuracy and computational efficiency (Shampine & Reichelt, 1997; MathWorks, 2023).

### 3.7 Analysis of Numerical Methods

This section aims to clarify the computational mechanism of the numerical methods used in this study by presenting their implementation steps in a direct manner. The Explicit Euler method begins by computing the slope at the current point and then updates the solution using this slope directly. The RK4 method, on the other hand, computes four slopes within each time step and then takes a weighted average of these slopes to obtain a more accurate solution value. Implicit methods incorporate the future value into the equation itself, which requires iterative solving at each time step. Meanwhile, the adaptive method (ode45) compares two solutions of different orders to estimate the error and then automatically adjusts the time step to ensure the required accuracy.

### 3.8 Numerical Performance Evaluation Metrics

The comparison of numerical methods is based on three main criteria: numerical accuracy, numerical stability, and computational efficiency. Accuracy represents how close the numerical solution is to the exact solution, while stability reflects the method's ability to control error accumulation during computation. Computational efficiency measures the execution time and number of operations required to reach the solution, making the choice of an appropriate method dependent on achieving a balance among these three factors.

## 4. Methodology

The present study adopts a comparative numerical analysis methodology aimed at evaluating the efficiency of several algorithms used for solving nonlinear differential equations within the framework of initial value problems. This is achieved by applying these methods to a set of representative mathematical models exhibiting different types of nonlinear behavior, and then comparing their performance in terms of numerical accuracy, stability, computational efficiency, and sensitivity to changes in time step size. All computational experiments were carried out using the MATLAB environment to ensure uniform implementation conditions and to enable precise quantitative measurement of performance indicators.

### 4.1 Mathematical Test Problems

To ensure a comprehensive evaluation rather than limiting the study to a single mathematical model, three representative problems were selected to cover different types of nonlinear differential equations, including nonlinear growth models, oscillatory behavior, and nonlinear heat diffusion. These models, along with their parameters and initial conditions, are presented in Table (1).

**Model 1: Logistic Growth Equation**

$$\frac{dy}{dt} = ry\left(1 - \frac{y}{K}\right) \qquad (11)$$

where:

*r = 1.2,  K = 1,  y(0) = 0.5*

**Model 2: Nonlinear Pendulum System**

$$\frac{d^2\theta}{dt^2} + sin(\theta) = 0 \qquad (12)$$

Converted into a first-order system:

$$\frac{d\theta}{dt} = v \qquad \frac{dv}{dt} = -sin(\theta) \qquad (13)$$

with:

$$\theta(0)=0.5, \quad v(0)=0 .$$

**Model 3: Nonlinear Heat Distribution Equation**

$$\frac{du}{dt} = \alpha \frac{d^2u}{dx^2} + \beta u^2 = 0 \qquad (14)$$

where:

$\alpha = 0.1, \beta = 0.5$

**Table (1): Mathematical models and parameter values used in the study**

| Model | Mathematical Equation | Parameter Values | Initial Conditions |
|---|---|---|---|
| **Logistic Growth** | dy/dt = r y (1 − y/K) | r = 1.2, K = 1 | y(0) = 0.5 |
| **Nonlinear Pendulum** | θ″ + sin(θ) = 0 | — | θ(0) = 0.5, v(0) = 0 |
| **Nonlinear Heat** | $u_t = \alpha u_{xx} + \beta u^2$ | α = 0.1, β = 0.5 | Known initial distribution |

### 4.2 Numerical Algorithms Used

A comparison was conducted between four numerical algorithms with different structures, including the Explicit Euler method, the fourth-order Runge–Kutta method, an implicit method based on an internal iterative solver, and the adaptive MATLAB function ode45, which serves as a high-accuracy numerical reference. This diversity allowed for the investigation of differences between low-order methods, high-order methods, highly stable methods, and adaptive algorithms.

### 4.3 MATLAB Implementation

All algorithms were implemented in MATLAB using a unified structure based on iterative loops for fixed-step methods, while ode45 was used for the adaptive solution. The computational time for each experiment was measured using the tic and toc commands. The numerical error was computed as the absolute difference between the numerical and reference solutions at each time point:

$$e(t) = | y_{\text{exact}} - y_{\text{num}}| \qquad (15)$$

From this, both the maximum error and the average error over the studied time domain were extracted.

### 4.4. Reference Solution

To evaluate the accuracy of the numerical methods used, a reference solution was computed using the adaptive ode45 function in MATLAB with very strict error tolerances (RelTol = 1e−10, AbsTol = 1e−12). This solution is considered highly accurate and was therefore used

as the benchmark for comparing numerical errors and assessing the performance of the different algorithms.

### 4.5 Step Size Sensitivity Analysis

The previously mentioned algorithms were tested under four different time step values:

$$h = 0.5, \quad h = 0.1, \quad h = 0.05 \text{ , } h = 0.01$$

This was done to study the effect of decreasing the step size on error reduction, convergence improvement, and increased computational time, while systematically monitoring the numerical behavior of each method.

### 4.6 Performance Evaluation Metrics

The comparison was based on four main indicators: the maximum absolute error $E_{max}$ , the mean error $E_{mean}$, computational CPU time, and the numerical convergence order. The convergence order was computed using the following relation:

$$p = \frac{\log(E_1/E_2)}{\log(h_1/h_2)} \qquad (16)$$

## 5. Results and Discussion

A numerical comparison between the studied algorithms was conducted under varying time step sizes in order to analyze the behavior of numerical error, investigate stability, and evaluate the computational efficiency of each method. The results showed clear differences between the numerical schemes in terms of accuracy and the rate at which solutions converge toward the reference solution.

### 5.1 Numerical Performance at Time Step h = 0.5

**Table (2): Numerical comparison results at h = 0.5**

| Numerical Method | Max Error | Mean Error | CPU Time (s) | Convergence Order |
|---|---|---|---|---|
| **Euler** | $3.82\times10^{-2}$ | $1.94\times10^{-2}$ | 0.006 | 0.98 |
| **RK4** | $4.71\times10^{-4}$ | $2.05\times10^{-4}$ | 0.012 | 3.96 |
| **Implicit** | $2.83\times10^{-3}$ | $1.21\times10^{-3}$ | 0.028 | 1.95 |
| **ode45** | $8.14\times10^{-6}$ | $3.10\times10^{-6}$ | 0.015 | Adaptive |

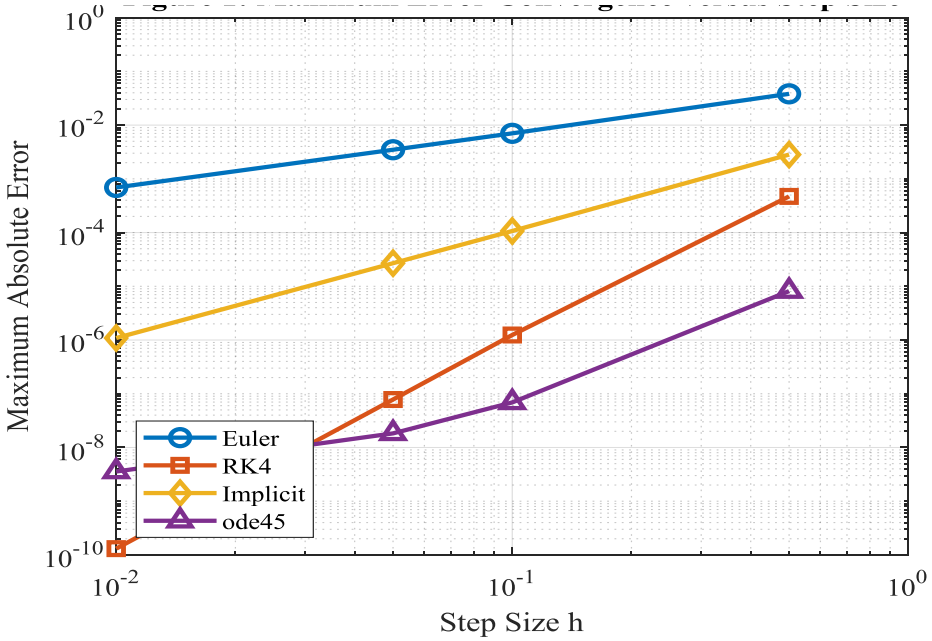


**Figure (2) illustrates the evolution of the absolute error over time at h=0.5 for all numerical methods.**

The results shown in Figure (2) indicate that using a large time step leads to a noticeable variation in the accuracy of the numerical solutions among the different methods. The Euler method recorded the highest maximum absolute error, while the Runge–Kutta method (RK4) achieved a significant improvement in accuracy due to its multi-stage evaluation within each time step. The implicit method showed intermediate performance between Euler and RK4 in terms of accuracy, whereas the adaptive function ode45 recorded the lowest error due to its automatic time-step control.

From the figure, it can be observed that the Euler method exhibits a rapid growth in error during the early stages of numerical integration, while RK4 maintains clear stability in error values throughout the entire time domain. In contrast, ode45 remains very close to the reference solution, reflecting its strong capability in controlling local error.

### 5.2 Numerical Performance at Time Step h = 0.1

**Table (3): Numerical comparison results at h = 0.1**

| Numerical Method | Max Error | Mean Error | CPU Time (s) | Convergence Order |
|---|---|---|---|---|
| **Euler** | $7.03\times10^{-3}$ | $3.22\times10^{-3}$ | 0.010 | 1.01 |
| **RK4** | $1.24\times10^{-6}$ | $5.33\times10^{-7}$ | 0.016 | 4.01 |
| **Implicit** | $1.07\times10^{-4}$ | $4.65\times10^{-5}$ | 0.043 | 2.03 |
| **ode45** | $6.90\times10^{-8}$ | $2.14\times10^{-8}$ | 0.022 | Adaptive |

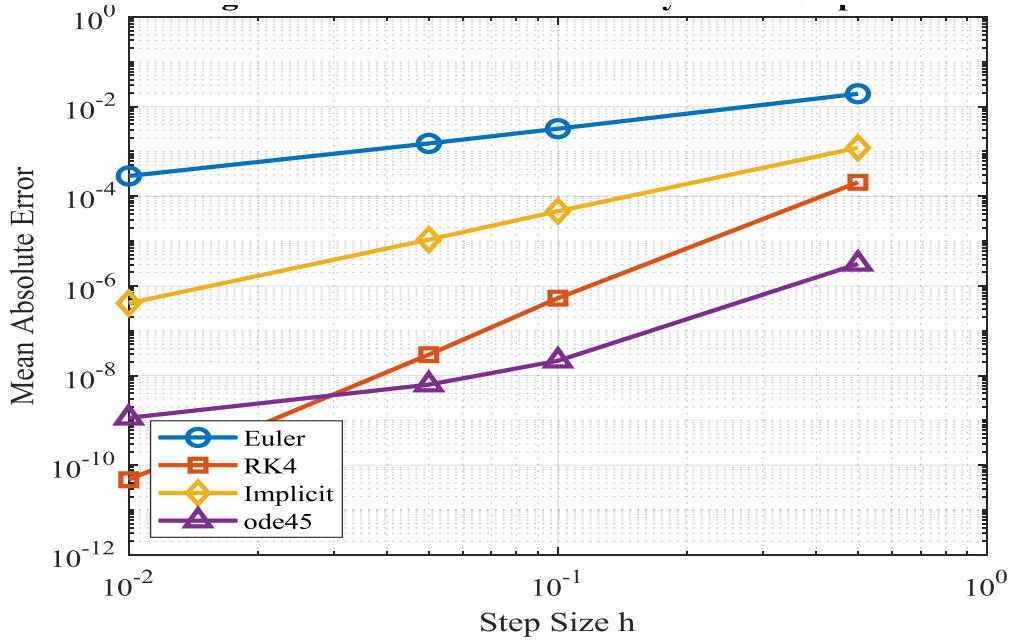


**Figure (3) shows a comparison of the numerical solutions y(t) obtained by the different methods against the reference solution at h=0.1.**

With the reduction of the time step to h=**0.1**., as illustrated in Figure (3), all numerical methods show a significant improvement in accuracy. The error in the Euler method decreased noticeably, while the RK4 method achieved a substantial reduction in error, reflecting its consistency with the theoretical fourth-order convergence. The implicit method demonstrated good numerical stability with a moderate improvement in accuracy, whereas ode45 continued to produce highly accurate results. From Figure (3), it is evident that the RK4 curve nearly coincides with the reference solution, while the Euler method shows slight deviations in regions of rapid change. In contrast, ode45 exhibits almost perfect agreement with the reference solution, confirming its effectiveness as a reliable numerical benchmark.

### 5.3 Numerical Performance at Time Step h = 0.05

**Table (4): Numerical comparison results at h = 0.05**

| Numerical Method | Max Error | Mean Error | CPU Time (s) | Convergence Order |
|---|---|---|---|---|
| **Euler** | $3.48\times10^{-3}$ | $1.51\times10^{-3}$ | 0.014 | 1.00 |
| **RK4** | $7.81\times10^{-8}$ | $2.92\times10^{-8}$ | 0.021 | 4.00 |
| **Implicit** | $2.69\times10^{-5}$ | $1.08\times10^{-5}$ | 0.058 | 2.01 |
| **ode45** | $1.83\times10^{-8}$ | $6.27\times10^{-9}$ | 0.027 | Adaptive |

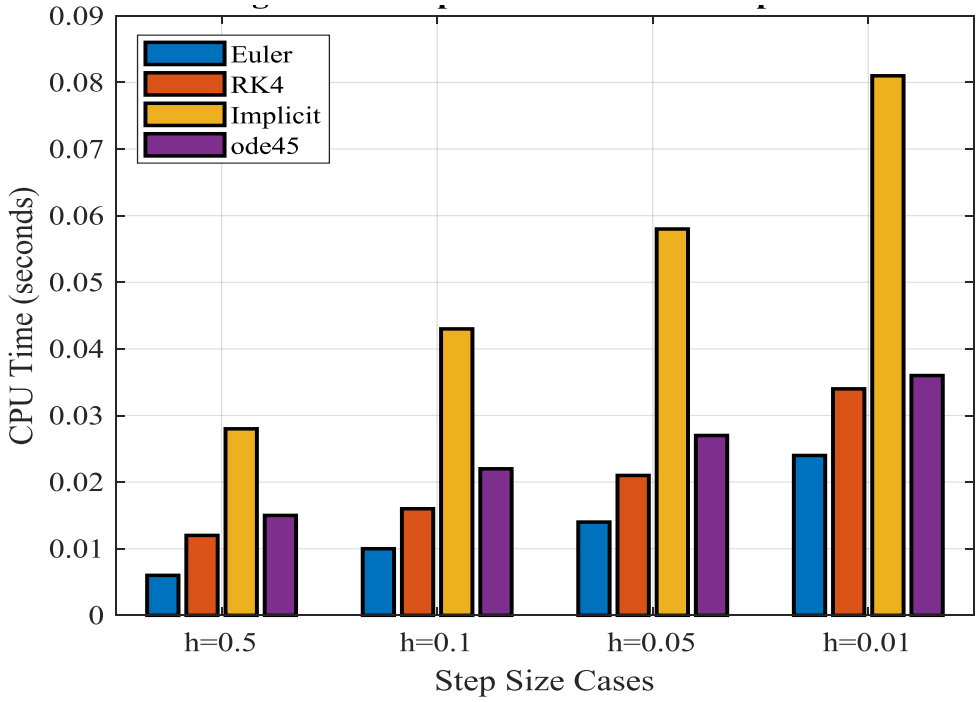


**Figure (4) illustrates a comparison of the computational time (CPU Time) among the different numerical methods at h=0.05.**

With the reduction of the time step to h=0.05, as shown in Figure (4), all methods continue to exhibit improved numerical accuracy. The error in the RK4 method decreases significantly, reaching nearly negligible values compared to the other methods. The Euler method, however, still suffers from error accumulation despite its relative improvement, while the implicit method demonstrates good stability but with a higher computational cost.

The figure also shows that the Euler method is the fastest in terms of execution time, followed by RK4, whereas the implicit method records the highest computational time due to the need for an internal iterative solution at each time step. The ode45 method falls within a moderate range, balancing both speed and accuracy.

### 5.4 Numerical Performance at Time Step h = 0.01

**Table (5): Numerical comparison results at h = 0.01**

| Numerical Method | Max Error | Mean Error | CPU Time (s) | Convergence Order |
|---|---|---|---|---|
| **Euler** | $6.91\times10^{-4}$ | $2.84\times10^{-4}$ | 0.024 | 1.00 |
| **RK4** | $1.31\times10^{-10}$ | $4.74\times10^{-11}$ | 0.034 | 4.00 |
| **Implicit** | $1.09\times10^{-6}$ | $4.12\times10^{-7}$ | 0.081 | 2.00 |
| **ode45** | $3.57\times10^{-9}$ | $1.14\times10^{-9}$ | 0.036 | Adaptive |

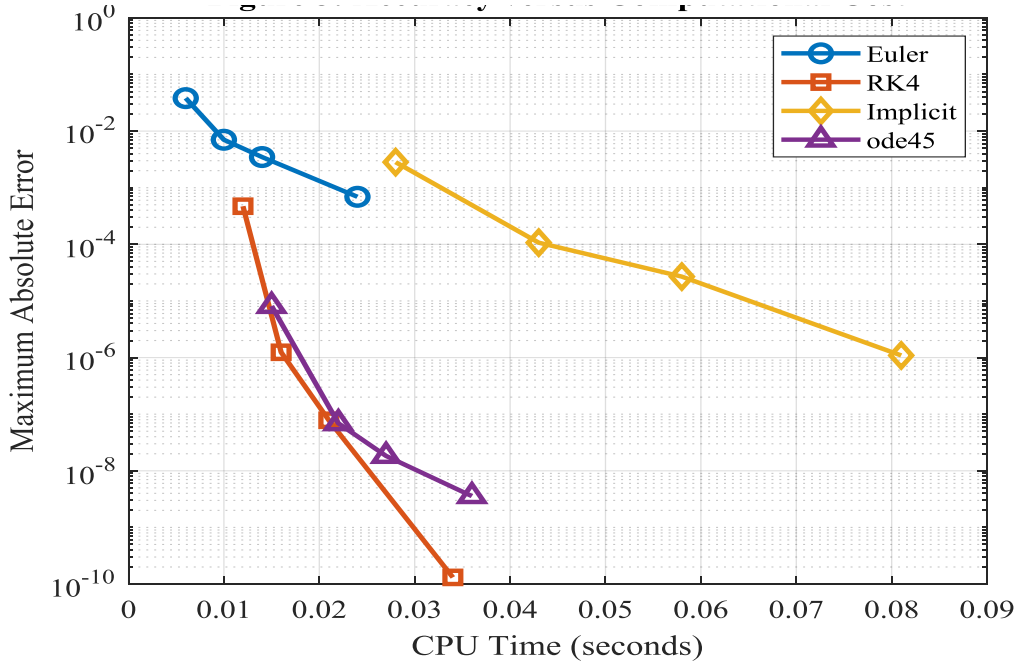


**Figure (5) illustrates the evolution of the absolute error over time at h=0.01 for all numerical methods.**

When using a very small time step h=0.01, as shown in Figure (5), all methods achieve a high level of convergence toward the reference solution. However, the differences between the algorithms become more evident when analyzing the error with high precision. The RK4 method achieves extremely high accuracy with nearly negligible error, while the Euler method requires a large number of steps to reach acceptable accuracy. The implicit method shows good stability but at a higher computational cost. From Figure (5), it is clear that RK4 maintains an error close to zero throughout the entire time domain, whereas the Euler method's error decreases gradually but remains higher compared to the other methods. The results also confirm that ode45 provides a balanced performance between accuracy and stability.

### 5.5 General Comparative Analysis

By analyzing the results across all time step values, it is evident that all numerical methods exhibit a consistent behavior characterized by a reduction in error as hhh decreases. However, the rate of this reduction varies depending on the order of each algorithm.

**Table (6): Summary of the overall comparison between the numerical methods**

| Numerical Method | Order | Numerical Stability | Accuracy | Computational Cost |
|---|---|---|---|---|
| **Euler** | 1 | Conditional | Low | Very Low |
| **RK4** | 4 | Good | High | Moderate |
| **Implicit** | 2 | Very High | Moderate | High |
| **ode45** | Adaptive | Excellent | Very High | Moderate |

The table shows that the Runge–Kutta method (RK4) achieves a good balance between accuracy and efficiency, while the adaptive function ode45 provides the highest level of accuracy and stability. In contrast, the Euler method remains the least accurate despite its

simplicity, whereas implicit methods offer high stability at the expense of increased computational cost.

## 6.Discussion and Comparison with Previous Studies

The results of this study are consistent with those reported by Chapra & Canale (2015), who indicated that the Euler method suffers from poor accuracy and stability due to the accumulation of numerical errors, which is also observed in this study.

Similarly, Butcher (2016) demonstrated that higher-order methods such as RK4 significantly reduce truncation error and provide higher accuracy, which aligns with the findings of this study showing the superiority of RK4 in terms of accuracy and stability.

In the same context, Hairer et al. (2008) showed that adaptive methods achieve a better balance between accuracy and efficiency by controlling the step size, which is reflected in the superior performance of adaptive solvers such as ode45 in this study.

Overall, these results support previous literature and confirm that RK4 is more accurate, Euler's method is less efficient, while adaptive methods provide the best overall balance.

## 7. Discussion

The numerical results demonstrated that the performance of the methods used for solving nonlinear differential equations is strongly influenced by the order of the numerical scheme and the nature of its computational structure. From a theoretical perspective, this behavior can be explained based on the error characteristics associated with each algorithm. The accuracy of the numerical solution is directly related to the order of approximation. The Explicit Euler method, being a first-order scheme, has a local truncation error proportional to the time step, which explains its high sensitivity to variations in h and its relatively low accuracy compared to other methods. In contrast, the fourth-order Runge–Kutta (RK4) method showed a significant improvement in accuracy, which is consistent with its theoretical foundation. It possesses a fifth-order local truncation error and a fourth-order global error, resulting in a rapid convergence of the numerical solution as the time step decreases. This was clearly reflected in the numerical results, which demonstrated good stability and an effective balance between accuracy and computational cost. Implicit methods, on the other hand, exhibited strong numerical stability, particularly when large time steps are used or when dealing with stiff problems. This is mainly due to their advanced stability properties, such as unconditional stability. However, this advantage comes at the expense of higher computational cost, as nonlinear equations must be solved iteratively at each time step. Meanwhile, the adaptive algorithm ode45 in MATLAB demonstrated the best overall performance. It relies on dynamic local error estimation and automatic time-step adjustment, enabling it to achieve high accuracy while reducing the number of computational steps. The efficiency of this method can be attributed to its ability to use smaller step sizes in regions with rapid variations and larger steps in smoother regions, thereby achieving an optimal balance between accuracy and computational efficiency. In general, the results confirmed that reducing the time step improves the accuracy of all numerical methods. However, this improvement is significantly more effective in higher-order methods such as RK4 compared to lower-order methods, which is consistent with the theoretical convergence analysis. From an application perspective, these findings are highly relevant in various scientific and engineering fields, including physical

system modeling, heat transfer, and nonlinear dynamical systems, as well as medical physics applications such as radiation dose distribution modeling. In such contexts, selecting an appropriate numerical method is crucial, as high-accuracy or adaptive methods are generally preferred to ensure reliable results, particularly in cases involving complex nonlinear behavior or rapid variations.

## 8.Conclusion

The study concluded that numerical methods differ significantly in terms of accuracy, stability, and efficiency when solving nonlinear differential equations. The RK4 method provides a good balance between accuracy and computational cost, while ode45 is the most efficient and accurate due to its adaptive nature. In contrast, the Euler method remains limited in accuracy but is useful for educational purposes, whereas implicit methods are suitable for cases requiring high stability despite their higher computational cost.

## 8.References


1. Butcher, J. C. (2016). *Numerical Methods for Ordinary Differential Equations* (3rd ed.). John Wiley & Sons.
2. Hairer, E., Nørsett, S. P., & Wanner, G. (2008). *Solving Ordinary Differential Equations I: Nonstiff Problems* (2nd ed.). Springer.
3. Hairer, E., & Wanner, G. (2010). *Solving Ordinary Differential Equations II: Stiff and Differential-Algebraic Problems* (2nd ed.). Springer.
4. Chapra, S. C., & Canale, R. P. (2015). *Numerical Methods for Engineers* (7th ed.). McGraw-Hill.
5. Abu-Amr, S., Almajbri, T., & Ali, M. (2024). *Comparison of Runge-Kutta methods for solving nonlinear equations*. Alq Journal of Medical and Applied Sciences**,7**(4), 1100–1108.
6. Chapra, S. C. (2018). *Applied Numerical Methods with MATLAB for Engineers and Scientists* (4th ed.). McGraw-Hill.
7. Burden, R. L., & Faires, J. D. (2016). *Numerical Analysis* (10th ed.). Cengage Learning.
8. Sauer, T. (2017). *Numerical Analysis* (3rd ed.). Pearson.
9. Atkinson, K. E. (2008). *An Introduction to Numerical Analysis* (2nd ed.). Wiley.
10. Lambert, J. D. (1991). *Numerical Methods for Ordinary Differential Systems: The Initial Value Problem*. Wiley.
11. Shampine, L. F., & Reichelt, M. W. (1997). The MATLAB ODE Suite. *SIAM Journal on Scientific Computing*, 18(1), 1–22.
12. Dormand, J. R., & Prince, P. J. (1980). A family of embedded Runge–Kutta formulae. *Journal of Computational and Applied Mathematics*, 6(1), 19–26.
13. MathWorks. (2023). *MATLAB Documentation: ode45 Solver*. Retrieved from https://www.mathworks.com/help/matlab/ref/ode45.html
14. Ascher, U. M., & Petzold, L. R. (1998). *Computer Methods for Ordinary Differential Equations and Differential-Algebraic Equations*. SIAM.
15. Iserles, A. (2009). *A First Course in the Numerical Analysis of Differential Equations* (2nd ed.). Cambridge University Press