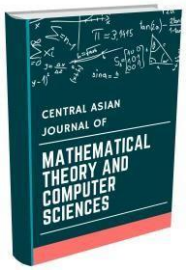

CENTRAL ASIAN JOURNAL OF MATHEMATICAL THEORY AND COMPUTER SCIENCES
https://cajmtcs.centralasianstudies.org/index.php/CAJMTCS
Volume: 06 Issue: 03 | July 2025 ISSN: 2660-5309

Article

Solving Differential Equations Using The Runge-Kutta Method

Zainab Talib Majeed*[1]

1. Islamic Azad University, Shiraz Branch, Iran
* Correspondence: ztalib238@gmail.com

Abstract: Differential equations have a wide-ranging impact on many disciplines, including physical sciences, engineering, economics, weather modeling, systems theory, and biology. The importance of this research is highlighted by the use of the fourth-order Runge-Kutta method in MATLAB and its comparison with traditional methods for solving fourth-order differential equations. Runge Kutta method and MATLAB. The study showed that the MATLAB method is effective and has a high degree of mathematical independence. The MATLAB code is similar to the fourth-order Runge-Kutta technique, and the use of this software in the research provides significant added value compared to traditional manual methods in terms of accuracy, speed, and performance efficiency.

Keywords: Problems, Ordinary Differential Equations, Fourth-Order Runge Kutta Method

1. Introduction

Differential equations are the vital tools for mathematical modeling of more than one phenomena and allow us to advantage perception into numerous problems [1]. However they may be in fashionable NP-difficult to clear up, especially if the range of variables is huge. Therefore, it's miles applicable to construct new techniques for solving those equations [2]. This demands a heavy dependency on state-of-the-art algorithms that offer solutions, in a specific and practical manner [3]. Howdy then take a look at the equation several instances at each step, resulting in correct answers without needing to comply with lengthy workflows [4]. This technique is more general and can be applied to numerous styles of equations as well [5]. The Runge-Kutta technique is a classic numerical technique for fixing regular differential equations (ODEs) and is extensively preferred for its accuracy and efficiency in the treatment of initial fee troubles [6]. They then check the equation numerous times at each step, resulting in correct answers without needing to observe long workflows [4]. This method is greater general and may be carried out to diverse kinds of equations as properly [5]. The Runge-Kutta method is a classic numerical technique for solving regular differential equations (ODEs) and is broadly appreciated for its accuracy and performance within the remedy of initial value problems [6]. In a more substantial comparative observe in which researchers taken into consideration several better-order Runge-Kutta strategies, including the smug (third order), Dormand–Prince (fourth and 5th order, respectively) and optimized (5th order) pairs [7]. Many studies have found that the classic fourth-order Runge-Kutta method is still a reliable and effective choice for scientific and engineering work. It offers good accuracy without requiring too much computational effort. Over time, different versions of Runge-Kutta methods have been created, some of which use higher-order derivatives to get better accuracy and lower

Citation: Majeed Z. T. Solving Differential Equations Using The Runge-Kutta Method. Central Asian Journal of Mathematical Theory and Computer Sciences 2025, 6(3), 702-713.

Received: 31th May 2025
Revised: 11th Jun 2025
Accepted: 18th Jun 2025
Published: 28th Jun 2025

errors. For example, the three-stage multiple derivative method has shown to be stable and matches well with exact solutions [9]. Recent studies emphasize how important it is to check the stability of numerical methods and how sensitive they are to changes in parameters. It's essential to test how reliable a method is when solving different problems to pick the best one for each situation. Tools like MATLAB are often used to run these methods and analyze the results, making calculations easier and more accurate.

A detailed assessment determined that higher-order Runge-Kutta techniques give accurate and reliable answers for initial cost problems. Adjusting the step size and reading errors can enhance their performance. This shows why ongoing studies into these strategies is critical to keep numerical solutions accurate and efficient in technology and engineering.

One have a look at checked out the fourth-order Runge-Kutta method for fixing everyday differential equations with initial values. It located that these methods paintings well and are realistic for those issues. The look at blanketed examples to expose how correct and clean the techniques are to use. The numerical consequences have been compared to precise answers, helping to apprehend their accuracy. The study additionally tested and calculated the mistakes worried.

Another observe compared 3 numerical strategies for solving fourth-order normal differential equations. The simulations confirmed that each technique has strengths and weaknesses depending at the sort of equation. When the use of very small step sizes, the numerical answers intently match the precise solutions. All 3 methods are right enough to remedy preliminary cost issues accurately and efficiently. Among them, the RK4 approach had the smallest average blunders, followed by way of the ABM4 technique.

Simple Different Equation (ODE) plays a vital role in modeling of many scientific and engineering issues. They demonstrate how an unknown function belongs to an independent variable via its derivatives. In the simplest example, a simple difference equation is expressed as follows:

$$\boldsymbol{F(y,z) = \frac{dy}{dz}}$$

Along with an initial condition

$$\mathbf{z(y_0) = z_0}$$

The fourth-order Runge-Kutta method is a widely used technique to calculate the initial values efficiently in solving equations. It provides a good balance between accuracy and computational effort by estimating the slope at several points within each stage and combining them to get a reliable connectivity [13].

This depends on calculating several approximate values within each time to reduce local error and ensure high accuracy in the final solution.

In the fourth order Range-Kutta method, the monopolitan values are calculated using the following equations:

$$\boldsymbol{m_1 = hf(y_n, z_n)}$$

$$\boldsymbol{m_2 = hf\left(y_n + \frac{h}{2}, z_n + \frac{k_1}{2}\right)}$$

$$\boldsymbol{m_3 = hf\left(y_n + \frac{h}{2}, z_n + \frac{k_2}{2}\right)}$$

$$\boldsymbol{m_4 = hf(y_n + h, z_n + m_3)}$$

Then the value is updated according to:

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

Here, hh is the size of the time phase, and (y_n, z_n) represents the current point. This method provides accurate results that can be applied in many real -world conditions [14]. When combined with software algorithms, it improves performance, helps analyze numerical errors, and allows the best possible results to reach the phase size to reach. This method is considered a relatively stable method, which has a good ability to handle nonlinear systems and equations with complex properties, which expands the scope of its practical applications [15].

The current research aims to find exact solutions for problems with time-dependent equations and to assess how well the MATLAB algorithm performs in solving differential equations using the fourth-order Runge-Kutta method, focusing on speed, accuracy, and overall performance

Accordingly, the applied experimental approach was employed to achieve the following research objectives

1. To use the fourth order Runge-Kutta method to solve the initial value problems in simple difference equations with high accuracy.
2. 2To facilitate calculation and to implement this method in matlab to improve the accuracy of results.
3. Comparing the performance of the proposed method with traditional methods in terms of speed and accuracy.
4. To test the effectiveness of the matlab algorithm in applying the Runge-Kutta method on independent equations of moderate complexity.
5. To ensure the compatibility of the code with the Runge-Kutta method to ensure accurate and reliable results in scientific and engineering applications.

2. Materials and Methods

Illustrative Examples

The calculations are performed sequentially using the equation as follows:

$$\boldsymbol{F(y,z)=\frac{dy}{dz}}$$

$$\boldsymbol{z(y_0)=z_0}$$

$$\boldsymbol{m_1=hf(y_0,z_0)}$$

$$\boldsymbol{m_2=hf\left(y_0+\frac{h}{2},z_0+\frac{m_1}{2}\right)}$$

$$\boldsymbol{m_3=hf\left(y_0+\frac{h}{2},z_0+\frac{m_2}{2}\right)}$$

$$\boldsymbol{m_4=hf(y_0+h,z_0+m_3)}$$

In the end we find

$$\boldsymbol{M=\frac{1}{6}(m_1+2m_2+2m_3+m_4)}$$

Hence the approximate value

$$\boldsymbol{Z=z_0+M}$$

Where m is the weighted mean of (m1, m2, m3, m4)

$$\boldsymbol{M=\frac{\sum_{n=1}^{4} w_n m_n}{\sum_{n=1}^{4} w_n}}$$

For values of n: 1, 2, 3, 4
where h is the step size
M1 is the slope at the start using z
M2 is the slope at the midpoint using z, m1
M3 is also the slope at the midpoint using z, m2
M4 is the slope at the end of the interval using z, m3
Analytical solution of the equation using the fourth-order Runge-Kutta method

$$\mathbf{3y+z}$$

Solve the differential equation

$$\mathbf{3yz} = \frac{dy}{dx}$$

With an initial condition:

$$\mathbf{z(0)=1}$$

For t=0 up to t=0.4
with step size h=0.1
Solution:
The fourth-order Lagrange method is given by

$$z_{n+1} = z_n + \frac{1}{6}(m_1 + 2m_2 + 2m_3 + m_4)$$

where:

$$m_1 = hf(y_n, z_n)$$

$$m_2 = hf\left(y_n + \frac{h}{2}, z_n + \frac{m_1}{2}\right)$$

$$m_3 = hf\left(y_n + \frac{h}{2}, z_n + \frac{m_2}{2}\right)$$

$$m_4 = hf(y_n + h, z_n + m_3)$$

From the data:

$$f(y,z) = 3y + z, h = 0.1, y_0 = 0, z_0 = 1$$

Step one:

$$m_1 = hf(y_0, z_0) = 0.1 \times (3 \times 0 + 1) = 0.1 \times 1 = 0.1$$

$$m_2 = hf\left(y_0 + \frac{h}{2}, z_0 + \frac{k_1}{2}\right) = 0.1 \times \left(3 \times 0.05 + 1 + \frac{0.1}{2}\right)$$
$$= 0.1 \times (0.15 + 1 + 0.05) = 0.1 \times 1.2 = 0.12$$

$$m_3 = hf\left(y_0 + \frac{h}{2}, z_0 + \frac{m_2}{2}\right) = 0.1 \times \left(3 \times 0.05 + 1 + \frac{0.12}{2}\right)$$
$$= 0.1 \times (0.15 + 1 + 0.06) = 0.1 \times 1.21 = 0.121$$

$$m_4 = hf(y_0 + h, z_0 + m_3) = 0.1 \times (3 \times 0.1 + 1 + 0.121)$$

$$= 0.1 \times (0.3 + 1 + 0.121) = 0.1 \times 1.421 = 0.1421$$

Update value:

$$z_1 = z_0 + \frac{1}{6}(m_1 + 2m_2 + 2m_3 + m_4)$$

$$= 1 + \frac{1}{6}(0.1 + 2 \times 0.12 + 2 \times 0.121 + 0.1421)$$

$$= 1 + \frac{1}{6}(0.1 + 0.24 + 0.242 + 0.1421) = 1 + \frac{1}{6}(0.7241) = 1 + 0.12068$$

$$= 1.12068$$

The steps are repeated in order to:

$$\mathbf{y} = 0.2, 0.3, 0.4$$

As follows:

$$\mathbf{m_1} = hf(\mathbf{y_1}, \mathbf{z_1}) = 0.1 \times (3 \times 0.1 + 1.1206) = 0.1 \times (0.3 + 1.1206)$$

$$= 0.1 \times 1.4206 = 0.14206$$

$$\mathbf{m_2} = hf\left(0.1 + \frac{0.1}{2}, 1.1206 + \frac{0.14206}{2}\right)$$

$$= 0.1 \times (3 \times 0.15 + 1.1206 + 0.07103)$$

$$= 0.1 \times (0.45 + 1.19171) = 0.1 \times 1.64171 = 0.164171$$

$$\mathbf{m_3} = hf\left(0.1 + \frac{0.1}{2}, 1.1206 + \frac{0.164171}{2}\right)$$

$$= 0.1 \times (3 \times 0.15 + 1.1206 + 0.082085)$$

$$= 0.1 \times (0.45 + 1.202765) = 0.1 \times 1.652765 = 0.16527657$$

$$\mathbf{m_4} = hf(0.2, 1.1206 + 0.1652765) = 0.1 \times (3 \times 0.2 + 1.28595657)$$

$$= 0.1 \times (0.6 + 1.285956) = 0.1 \times 1.8859565 = 0.18859565$$

Value update

$$\mathbf{z_2} = \mathbf{z_1} + \frac{1}{6}(\mathbf{m_1} + 2\mathbf{m_2} + 2\mathbf{m_3} + \mathbf{m_4})$$

$$= 1.12068 + \frac{1}{6}(0.14206 + 2 \times 0.1641714 + 2 \times 0.1652765$$

$$+ 0.1885956)$$

$$= 1.12068 + \frac{1}{6}(0.14206 + 0.328342 + 0.3305531 + 0.18859565)$$

$$= 1.12068 + \frac{1}{6}(0.989559)$$

$$= 1.12068 + 0.1649266 = 1.2856066$$

Step 3

$$\mathbf{m_1 = 0.1 \times (3 \times 0.2 + 1.285606) = 0.1 \times (0.6 + 1.285606)}$$
$$\mathbf{= 0.1 \times 1.885606 = 0.1885606}$$

$$\mathbf{m_2 = 0.1 \times \left(3 \times 0.25 + 1.2856066 + \frac{0.1885606}{2}\right)}$$
$$\mathbf{= 0.1 \times (0.75 + 1.2856066 + 0.0942803)}$$
$$\mathbf{= 0.1 \times 2.1298869 = 0.21298869}$$

$$\mathbf{m_3 = 0.1 \times \left(3 \times 0.25 + 1.2856066 + \frac{0.212988693}{2}\right)}$$
$$\mathbf{= 0.1 \times (0.75 + 1.2856066 + 0.10649435)}$$
$$\mathbf{= 0.1 \times 2.1421009 = 0.21421009}$$

$$\mathbf{m_4 = 0.1 \times (3 \times 0.3 + 1.285606 + 0.21421009)}$$
$$\mathbf{= 0.1 \times (0.9 + 1.4998167) = 0.1 \times 2.3998167}$$
$$\mathbf{= 0.2399816}$$

Value update

$$\mathbf{z_3 = z_2 + \frac{1}{6}(m_1 + 2m_2 + 2m_3 + m_4)}$$
$$\mathbf{= 1.2856066 + \frac{1}{6}(0.1885606 + 2 \times 0.21298869 + 2 \times 0.21421009}$$
$$\mathbf{+ 0.2399816)}$$

$$\mathbf{= 1.285606 + \frac{1}{6}(0.1885606 + 0.42597738 + 0.42842019 + 0.23998167)}$$
$$\mathbf{= 1.2856066 + \frac{1}{6}(1.282939)}$$

$$\mathbf{= 1.2856066 + 0.213823 = 1.499429}$$

Step four:

$$\mathbf{m_1 = 0.1 \times (3 \times 0.3 + 1.4994289) = 0.1 \times (0.9 + 1.4994289)}$$
$$\mathbf{= 0.1 \times 2.3994289 = 0.23994289}$$

$$\mathbf{m_2 = 0.1 \times \left(3 \times 0.35 + 1.4994289 + \frac{0.23994289}{2}\right)}$$
$$\mathbf{= 0.1 \times (1.05 + 1.4994289 + 0.1198715) = 0.1 \times 2.6684014}$$
$$\mathbf{= 0.26684014}$$

$$\mathbf{m_3 = 0.1 \times \left(3 \times 0.35 + 1.499429 + \frac{0.26694014}{2}\right)}$$
$$\mathbf{= 0.1 \times (1.05 + 1.4994299 + 0.1334707) = 0.1 \times 2.6828997}$$
$$\mathbf{= 0.26828997}$$

$$\mathbf{m_4 = 0.1 \times (3 \times 0.4 + 1.494299 + 0.26828997) = 0.1 \times (1.2 + 1.7677199)}$$
$$\mathbf{= 0.1 \times 2.9677199 = 0.2967719}$$

Value update

$$\mathbf{z_4 = z_3 + \frac{1}{6}(k_1 + 2k_2 + 2k_3 + k_4)}$$
$$\mathbf{= 1.4994289 + \frac{1}{6}(0.23994289 + 2 \times 0.26684014}$$
$$\mathbf{+ 2 \times 0.268289997 + 0.29677189)}$$

$$\mathbf{= 1.4994299 + \frac{1}{6}(0.23994299 + 0.5338028 + 0.5365799 + 0.29677199)}$$
$$\mathbf{= 1.499429 + \frac{1}{6}(1.6071755)}$$

$$\mathbf{= 1.499429 + 0.2678625 = 1.7672924}$$

Solution using the MATLAB algorithm:

Figures 1, 2, AND 3 show the MATLAB algorithm.

```
% Runge-Kutta 4th Order Algorithm
% Approximate the solution of an initial value problem

function [] = Runge_Kutta_Method()

clc

% Get step size and end point from user
h = input('Enter the value of h: ');
xn = input('Enter the value of xn: ');

format compact
format short g

% Get the function from user
f = input('Enter the function (as a function handle, e.g. @(x,y) x + y): ');

% Get initial values from user
x0 = input('Enter the value of x0: ');
y0 = input('Enter the value of y0: ');
```

Figure 1. The first part of the MATLAB algorithm.

```
% إدخال حجم الخطوة والنقطة النهائية من المستخدم
h = input('Enter the value of h: ');
xn = input('Enter the value of xn: ');

format compact
format short g

% إدخال الدالة كدالة مجهولة (function handle)
f = input('Enter the function (as a function handle, e.g. @(x,y) x + y^3): ');

% إدخال القيم الابتدائية
x0 = input('Enter the value of x0: ');
y0 = input('Enter the value of y0: ');

% تهيئة المتغيرات
x = x0;
y = y0;

% حلقة الحساب
while x < xn
    k1 = h * f(x, y);
    k2 = h * f(x + h/2, y + k1/2);
```

Figure 2. The second part of the MATLAB algorithm.

```
                                    while x < xn
                          ;k1 = h * f(x, y)
            ;k2 = h * f(x + h/2, y + k1/2)
            ;k3 = h * f(x + h/2, y + k2/2)
                ;k4 = h * f(x + h, y + k3)

        ;k = (k1 + 2*k2 + 2*k3 + k4) / 6

                                    ;x = x + h
                                    ;y = y + k

        % عرض القيم الوسيطة والنتيجة لكل خطوة
              ;fprintf('Step results:\n')
              ;fprintf('k1 = %.5f\n', k1)
              ;fprintf('k2 = %.5f\n', k2)
              ;fprintf('k3 = %.5f\n', k3)
              ;fprintf('k4 = %.5f\n', k4)
               ;fprintf('k  = %.5f\n', k)
                ;fprintf('x = %.4f\n', x)
                ;fprintf('y = %.6f\n', y)
        ;fprintf('When y = %.6f\n\n', y)
                                        end
```

Figure 3. The third part of the MATLAB algorithm.

```
% خوارزمية طريقة رونج-كوتا من الرتبة الرابعة مع عرض القيم الوسيطة والنتائج

                                ()function [] = Runge_Kutta_Solution

                                                clc

                                % إدخال حجم الخطوة والنقطة النهائية
                                ;h = input('Enter the value of h: ')
                                ;xn = input('Enter the value of xn: ')

                                                format compact
                                                format short g

                                % إدخال الدالة كدالة مجهولة
;f = input('Enter the function (as a function handle, e.g. @(x,y) x + y^3): ')

                                % إدخال القيم الابتدائية
                                ;x0 = input('Enter the value of x0: ')
                                ;y0 = input('Enter the value of y0: ')
```

Figure 4. Final part of the algorithm.

3. Results and Discussion

The final solutions to the study sample equation were compiled using either the analytical method or the MATLAB algorithm, as shown in Table 1 and Table 2.

Table 1. Analytical solution to the study sample equation.

I	Yi	Zi
0	0.0	1.0
1	0.1	1.123
2	0.2	1.312
3	0.3	1.643
4	0.4	2.461

Table 2. Solution using MATLAB simulation.

I	Yi	Zi
0	0.0	1.0
1	0.1	1.117
2	0.2	1.296
3	0.3	1.601
4	0.4	2.291

Figures 4, 5, and 6 show the graph of the analytical solution and the solution using the algorithm.

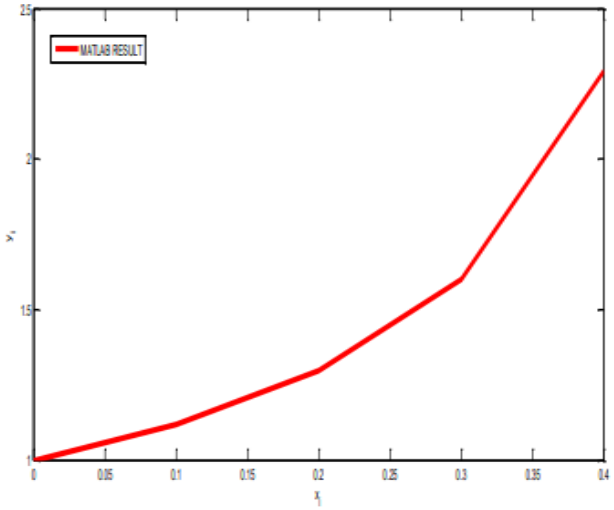

Figure 4. The analytical solution of the study sample equation.

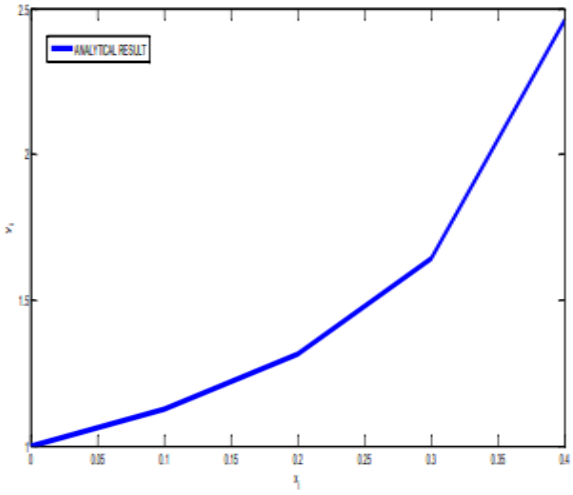

Figure 5. The solution using MATLAB for the study sample equation.

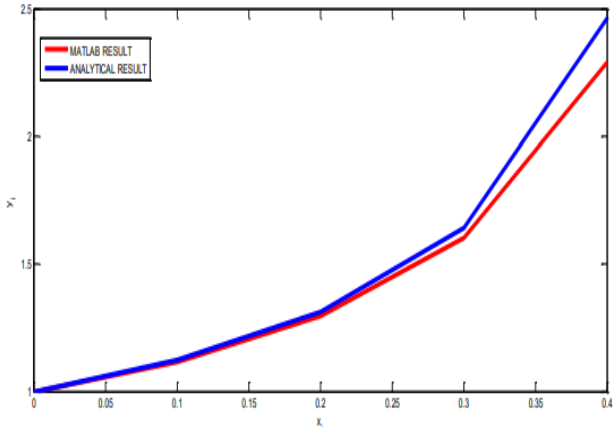

Figure 6. Comparative solution.

Error analysis and comparison between the two solution methods
By the initial value equation

$$\frac{dz}{dy} = f(y,z), z(y_0) = z_0$$

Through the basic equation of Rang Kuta

$$z_{n+1} = z_n + \frac{1}{6}(m_1 + 2m_2 + 2m_3 + m_4) + O(h^5)$$

Where

$$m_1 = hf(y_n, z_n)$$

$$m_2 = hf\left(y_n + \frac{h}{2}, y_n + \frac{m_1}{2}\right)$$

$$m_3 = hf\left(y_n + \frac{h}{2}, z_n + \frac{k_2}{2}\right)$$

$$m_4 = hf(y_n + h, z_n + m_3)$$

Local error per step:

$$\text{err} \approx Ch^5$$

And so

$$y(x_0 + h) = p + Ch^5 \ldots\ldots\ldots\ldots\ldots\ldots \quad (1)$$

$$y(x_0 + h) = r + C\left(\frac{h}{2}\right)^5 = r + \frac{Ch^5}{32} \quad \ldots\ldots\ldots\ldots\ldots\ldots (2)$$

By subtracting the two equations, we get:

$$p - r = Ch^5 - \frac{Ch^5}{32} = \frac{31}{32}Ch^5$$

Then the local error is estimated according to the relationship:

$$\text{err} \approx \frac{p-r}{\frac{31}{32}} = \frac{32}{31}(p-r)$$

Finding the local error using MATLAB, as shown in Table 4.

Table 4. Local error using MATLAB.

N	Yi	Solution using MATLAB	Exact solution	Error
0	0.0	1.0	1.0	0.0
1	0.1	1.1	1.1	0.007
2	0.2	1.2	1.3	0.017
3	0.3	1.6	1.6	0.042
4	0.4	2.2	2.4	0.169

4. Conclusion

This research has resolved the study sample differential equation using the traditional Runge-Kutta method, as well as the programmatic method using the Matlab algorithm. The error rate of both methods was studied. The results showed that both solution methods were identical in the initial condition. As the step value decreases, the error rate decreases and the accuracy of the solution may be increased by choosing smaller steps. The use of Matlab algorithms saves time and provides high accuracy in the solution.

REFERENCES

[1] D. W. Zingg and T. T. Chisholm, "Runge-Kutta methods for linear ordinary differential equations," NASA Ames Research Center Technical Report, NASA, 1997.

[2] Runge-Kutta Method - Articles," Scientific Research Publishing, 2023. [Online].

[3] V. Chauhan and P. K. Srivastava, "Computational techniques based on Runge-Kutta method of various order and type for solving differential equations," Int. J. Math. Eng. Manage. Sci., vol. 4, no. 2, pp. 375–386, 2019, doi: 10.33889/IJMEMS.2019.4.2-030.

[4] U. M. Ascher and L. R. Petzold, *Computer methods for ordinary differential equations and differential-algebraic equations*, vol. 61. SIAM, 1998.

[5] R. J. LeVeque, *Finite difference methods for ordinary and partial differential equations: steady-state and time-dependent problems*. SIAM, 2007.

[6] K. J. Audu, V. J. Udoh, and J. Garba, "An investigative comparison of higher-order Runge-Kutta techniques for resolving first-order differential equations," Turk. J. Sci. Technol., vol. 20, no. 1, pp. 141–158, 2025, doi: 10.55525/tjst.1433935.

[7] C. Vuik, F. J. Vermolen, M. B. van Gijzen, and T. Vuik, *Numerical methods for ordinary differential equations*. TU Delft OPEN Publishing, 2023.

[8] "The Runge-Kutta Method," UNT Digital Library, 2025. [Online]. Available: https://digital.library.unt.edu/ark:/67531/metadc108136/. [Accessed: Jun. 17, 2025]

[9] A three-stage multiderivative explicit Runge-Kutta method," Scientific Research Publishing, 2013. [Online]. Available: https://www.scirp.org/journal/paperinformation?paperid=33543. [Accessed: Jun. 17, 2025].

[10] A. Islam, "Accurate solutions of initial value problems for ordinary differential equations with the fourth order Runge Kutta method," J. Math. Res., vol. 7, no. 3, pp. 41–41, 2015, doi: 10.5539/jmr.v7n3p41.

[11] M. Marsudi and I. Darti, "A comparative study on numerical solutions of initial values problems of differential equations using the three numerical methods," *BAREKENG: J. Math. & App.*, vol. 19, no. 2, pp. 1263–1278, Jun. 2025.

[12] Y. A. Lesnussa, *BAREKENG: Journal of Mathematics and Its Applications*, vol. 19, no. 2. Pattimura University, 2025, doi: 10.30598/barekengvol19iss2pp1263-1278.

[13] M. Redmann and S. Riedel, "Runge-Kutta methods for rough differential equations," arXiv preprint arXiv:2003.12626, 2020, doi: 10.48550/arXiv.2003.12626.

[14] P. M. Burrage, *Runge-Kutta methods for stochastic differential equations*, Ph.D. dissertation, Dept. Math., The University of Queensland, Brisbane, Australia, 1999.

[15] D. R. Paudel and M. R. Bhatta, "Comparative study of Euler's method and Runge-Kutta method to solve an ordinary differential equation through a computational approach," *Acad. J. Math. Educ.*, vol. 6, no. 1, pp. 81–85, Dec. 2023, doi: 10.3126/ajme.v6i1.63802.