International Journal of Mathematics and Statistics Studies, 12 (1), 54-63, 2024
Print ISSN: 2053-2229 (Print),
Online ISSN: 2053-2210 (Online)
Website: https://www.eajournals.org/
Publication of the European Centre for Research Training and Development -UK

Solution of First Order Ordinary Differential Equations Using Fourth Order Runge-Kutta Method with MATLAB.

Shior, M.M.,[1] Agbata, B.C.,[2] Gbor, G. D.,[1] Ezugorie, I.U.,[3] Topman, N.N.,[2]
[1]Department of Mathematics/ Computer Science, Benue State University Makurdi, Nigeria
[2]Department of Mathematics University of Nigeria Nsukka, Nigeria
[3]Department of Industrial Mathematics/Applied Statistics, Enugu State University of Science and Technology, Nigeria.

doi: https://doi.org/10.37745/ijmss.13/vol12n15463 Published January 23, 2024

Citation: Shior, M.M., Agbata, B.C., Gbor, G. D., Ezugorie, I.U., Topman, N.N. (2024) Solution of First Order Ordinary Differential Equations Using Fourth Order Runge-Kutta Method with MATLAB, *International Journal of Mathematics and Statistics Studies*, 12 (1), 54-63

ABSTRACT: *Differential Equations are used in developing models in the physical sciences, engineering, mathematics, social science, environmental sciences, medical sciences and other numerous fields. This article examined solution of first ordinary differential equation using fourth order Runge-Kutta method with MATLAB. The fourth order Runge-Kutta method for modelling differential equations improves upon the Euler's method to obtain a greater accuracy without the necessity for higher-order derivatives of the given function. A first order differential equation was solved using fourth order Runge-Kutta method with MATLAB and the same problem was solved analytically in order to obtain the exact solution. The MATLAB commands match up quickly with the steps of the fourth order Runge-Kutta algorithm. Slight variation of the MATLAB code was used to show the effect of the size of h on the accuracy of the solution (see figure 4.1, 4.2, 4.3).The MATLAB and exact solutions are approximately equal though the MATLAB approach is easier and faster. The obtained results are in agreement with those in existing literature and improved the results obtained by [1]*

KEYWORDS: First order differential equations, MATLAB code, Numerical methods, Analytical methods, Fourth order Runge-Kutta Mehod, High order derivative, Simulation in numerical analysis.

INTRODUCTION

In numerical analysis, sometimes the analytical methods become obstinate when solving nonlinear differential equations which makes it difficult to obtain the exact solutions of such physical problems, numerical methods can then be used to obtain approximate solution of such problems [2]. A numerical method is a complete and monosemous set of algorithms for the solution of a problem including computable error estimate. It is an approximate technique for obtaining solution of a mathematical problem which cannot be easily solved analytically or has difficult analytical solution porecedure. Numerical methods for physical problems are of huge importance in many field of sciences because most real-life problems often lead to linear and nonlinear differential equations that cannot be solved analytically [1]. Generally speaking, two forms of analytical methods exist to solve the differential equation defined by (1), the two methods are known as the *single -step* and *multi-steps* methods. A single- step method can only be used when initial conditions are given while in multi-step one might needs the solution at several points for the method be implemented. There are several numerical methods in literature such as Newton Method, Simpson Law, Trapezoidal Law, Eulers Methods, Runge-Kutta methods, Adam's Methods, Predictor-Corrector Method, Milne's Method, Picard method. Euler and Runge-Kutta methods are mostly used to compute y over a limited range of x-values [1,2]. One of the simplest and direct method for the solution of first order differential equation is the Euler's method though it is not an efficient numerical method and it has limited applications when solving physical problems because a very small stepsize is required for a reasonable accuracy. Numerous authors have investigated numerical method for solving both ordinary and partial differential equations. [2] studied numerical solution of initial value problems of ordinary differential equations by Adams –Moultion predictor- corrector method, it was demonstrated in their studies that numerical methods have the ability tackle both linear and non- linear differential equations, it was concluded that Adm-Moulten predator-corrector method is strongly stable and strengthful. [1] studied formulation of Runge-Kutta;s method using MATLAB, various numerical methods were investigated, special attention was given to application of MALAB in formulation of forth order Runge-Kutta method and they obtained solution of first order ordinary differential equation using MATLAB. Other relevant works in the field differential equations are found in [3,4,6,7]. A stable numerical method yields a bounded solution which initiates the exact solution [2, 5].

The Runge-Kutta Method

The Euler's method has limited applications in practical problems since it requires a small stepsize to generat e reasonable accuracy. The Taylor's series method of higher-order differential equations is difficult to use because of the task involved to obtain higher total derivatives of $y(x)$. Highly relevant group of methods called Runge-Kutta methods which only require initial values of $y(x)$ associated with the differential equation and allow us to generate greater accuracy at each step. It used for finding the increment k of y corresponding to an increment h of x.

$$\frac{dy}{dx} = f(\mathrm{x}, \mathrm{y}), \quad y(\mathrm{x}_0) = \mathrm{y}_0 \tag{1}$$

is given below:

Calculate successively,

$$k_1 = hf(\mathrm{x}_0, \mathrm{y}_0)$$

$$k_2 = hf(\mathrm{x}_0 + \frac{h}{2}, \mathrm{y}_0 + \frac{\mathrm{k}_1}{2})$$

$$k_3 = hf(x_0 + \frac{h}{2}, y_0 + \frac{k_2}{2})$$

and

$$k_4 = hf(x_0 + h, y_0 + k_3)$$

Finally compute

$$k = \frac{1}{6}(k_1 + 2k_2 + 2k_3 + k_4)$$

Which yields the approximate value $y_1 = y_0 + k$.

Where k is the weighted mean of k_1, k_2, k_3 and k_4 given by

$$\bar{k} = \frac{\sum W_n k_n}{\sum W_n}, \text{ for } n = 1, 2, 3, 4$$

h is the step size,

k_1 is the slope at the beginning using y

k_2 is the midpoint slope using y and k_1

k_3 is again the midpoint slope using y and k_2

k_4 is the slope at the end of the interval using y and k_3

Analytical Solution of $x + y^3$ Using The Fourth-Order Runge-Kutta Algorithm

Example

Solve the following differential equation $\frac{dy}{dx} = x + y^3$ with initial condition $y(0) = 1$, using fourth order Runge-Kutta method from $t = 0$ to $t = 0.4$ taking the step $h = 0.1$

Solution

The fourth order Runge-Kutta method is described as

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

Where,

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

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

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

$$k_4 = hf(x_n + h, y_n + k_3)$$

From the question, $f(x, y) = x + y^3$, $h = 0.1$, $x_0 = 0$, $y_0 = 1$.

For the first step, we calculate

$$k_1 = hf(x_0, y_0) = 0.1f(0,1) = 0.1$$
$$k_2 = hf(x_0 + \frac{h}{2}, y_0 + \frac{k_1}{2}) = 0.1f(0.05, 1.05) = 0.1207$$
$$k_3 = hf(x_0 + \frac{h}{2}, y_0 + \frac{k_2}{2}) = 0.1f(0.05, 1.0604) = 0.1242$$
$$k_4 = hf(x_0 + h, y_0 + k_3) = 0.1f(0.1, 1.1242) = 0.1521$$
$$y_1 = y_0 + \frac{1}{6}(k_1 + 2k_2 + 2k_3 + k_4) = 1.1237$$
$$y_1 = y(0.1) = 1.1237$$

For the second step, we need to find $y(0.2)$

$$x_1 = x_0 + h = 0.1$$
$$k_1 = hf(x_1, y_1) = 0.1f(0.1, 1.1236) = 0.1519$$
$$k_2 = hf(x_1 + \frac{h}{2}, y_1 + \frac{k_1}{2}) = 0.1f(0.15, 1.20) = 0.1776$$
$$k_3 = hf(x_1 + \frac{h}{2}, y_1 + \frac{k_2}{2}) = 0.1f(0.15, 1.2125) = 0.1933$$
$$k_4 = hf(x_1 + h, y_1 + k_3) = 0.1f(0.2, 1.3170) = 0.2384$$
$$y_2 = y_1 + \frac{1}{6}(k_1 + 2k_2 + 2k_3 + k_4) = 1.3124$$
$$y_2 = y(0.2) = 1.3124$$

For the third step,

$$x_2 = x_1 + h = 0.2$$
$$k_1 = hf(x_2, y_2) = 0.1f(0.2, 1.3124) = 0.2460$$
$$k_2 = hf(x_2 + \frac{h}{2}, y_2 + \frac{k_1}{2}) = 0.1f(0.25, 1.4354) = 0.3007$$
$$k_3 = hf(x_2 + \frac{h}{2}, y_2 + \frac{k_2}{2}) = 0.1f(0.25, 1.4628) = 0.3380$$
$$k_4 = hf(x_2 + h, y_2 + k_3) = 0.1f(0.3, 1.6504) = 0.4595$$
$$y_3 = y_2 + \frac{1}{6}(k_1 + 2k_2 + 2k_3 + k_4) = 1.6428$$
$$y_3 = y(0.3) = 1.6428$$

For the fourth step,

$$x_3 = x_2 + h = 0.4$$

$$k_1 = hf(x_3, y_3) = 0.1f(0.3, 1.6428) = 0.4734$$

$$k_2 = hf(x_3 + \frac{h}{2}, y_3 + \frac{k_1}{2}) = 0.1f(0.35, 1.8795) = 0.6690$$

$$k_3 = hf(x_3 + \frac{h}{2}, y_3 + \frac{k_2}{2}) = 0.1f(0.35, 1.9773) = 0.8081$$

$$k_4 = hf(x_3 + h, y_3 + k_3) = 0.1f(0.4, 2.4509) = 1.4823$$

$$y_4 = y_3 + \frac{1}{6}(k_1 + 2k_2 + 2k_3 + k_4) = 2.4611$$

$$y_4 = y(0.4) = 2.4611$$

Which is the required result.

MATLAB Code and Solution

```
%Runge-Kutta Order 4th Algorithm
%Approximate the solution of initial value problem
function[] = Runga_Kutta_Method()
clc
format compact
format short g
% Enter the function as you desire
f=input('Enter the function : ')
%initial values x0 = input
('Enter the value of x0 : ');
y0 = input ('Enter the value of
y0 : ');
% step size h= input ('Enter
the value of h : ');
%calulation point
xn = input ('Enter the value of
xn : ');
x = x0;
y = y0;
% calculation loop
 While (1)
 if(x==xn)
 break
end
 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+(k1+k3)*2+k4
)/6
x=x+h
y=y+k
fprintf('When     x=%f
y=%f \n',x,y);
end
end
```

Result :-

```
Enter
the function:@(x,y)x+y^3
f = @(x,y)x+y^3
Enter the value of x0:0
x0 =0
Enter the value of y0:1
y0 =1
Enter the value of h:0.1
h = 0.1
Enter the value of xn: 0.4
xn =0.4
k1 =0.1
 k2 =0.12076
k3 =0.12423
k4 =0.15209
 k =0.11676
 x =0.1000
  y =1.1168
When           x=0.100000
y=1.116759
k1 =0.14928
 k2 =0.18411
k3 =0.19164
 k4 =0.24398
 k =0.17918
x =0.2
y =1.2959
When           x=0.200000
y=1.295939
k1 =0.23765
 k2 =0.30817
```

k3 =0.32988
k4 =0.45975
k =0.30541
x =0.3 y =1.6013
When x=0.300000
y=1.601347
k1 = 0.44064
k2 =0.63951
k3 =0.74401
k4 =1.3301
k =0.69001
x =0.4 y =2.2914
When x=0.400000
y=2.291354

Table of values for $x+y^3$ using the Exact Solution fourth-order Runge-Kutta algorithm

i	x_i	y_i
0	0.0000	1.0000
1	0.1000	1.1237
2	0.2000	1.3124
3	0.3000	1.6428
4	0.4000	2.4611

Table of values for $x+y^3$ using the MATLAB simulated fourth-order Runge-Kutta approach

i	x_i	y_i
0	0.0000	1.0000
1	0.1000	1.1168
2	0.2000	1.2959
3	0.3000	1.6013
4	0.4000	2.2914

1. Error Analysis of Runge-Kutta Method

From equation (1)

$$\frac{dy}{dx} = f(\mathrm{x},\mathrm{y}), \quad y(\mathrm{x}_0) = \mathrm{y}_0$$

International Journal of Mathematics and Statistics Studies, 12 (1), 54-63, 2024
Print ISSN: 2053-2229 (Print),
Online ISSN: 2053-2210 (Online)
Website: https://www.eajournals.org/
Publication of the European Centre for Research Training and Development -UK

Then

$$k = y(x_0 + h) = y(x_0) + \frac{1}{6}(\mathrm{k}_1 + 2\,\mathrm{k}_2 + 2\,\mathrm{k}_3 + \mathrm{k}_4) + 0(\mathrm{h}^5)$$

The error induced for each successive step of the iterated algorithm i.e local truncation error is almost

$err = Ch^5$

C denotes some constant which is dependent on x_0 and the fourth derivative of the exact solution $\tilde{y}(x)$ at x_0 and does not depend on *h***.** If we assume that C does not change much as *x* varies from x_0 to $x_0 + h$, we can estimate Ch^5.

Suppose p denotes the approximate solution to $\tilde{y}(x)$ at $x_0 + h$ which is obtained after carrying out a step fourth order Runge-Kutta approximation

$\tilde{y}(x) = p + Ch^5$

Suppose r denotes the approximate solution to $\tilde{y}(x)$ at $x_0 + h$ which is obtained by carrying out a two- step fourth order Runge-Kutta approximation (with step of $\frac{1}{2}h$)

$\tilde{y}(x) = r + 2C(\frac{h}{2})^5$

The difference of these two equations yields

$0 = p + r + C(1 - 2^{-4})\,\mathrm{h}^5$

Local truncation $err = Ch^5 = \dfrac{p-r}{1-h^{-4}}$ □ $p - r$

i	x_i	MATLAB Solution	EXACT Solution	ERROR
0	0.0000	1.0000	1.0000	0.0000
1	0.1000	1.1168	1.1237	0.0069
2	0.2000	1.2959	1.3124	0.0165
3	0.3000	1.6013	1.6428	0.0415
4	0.4000	2.2914	2.4611	O.1697

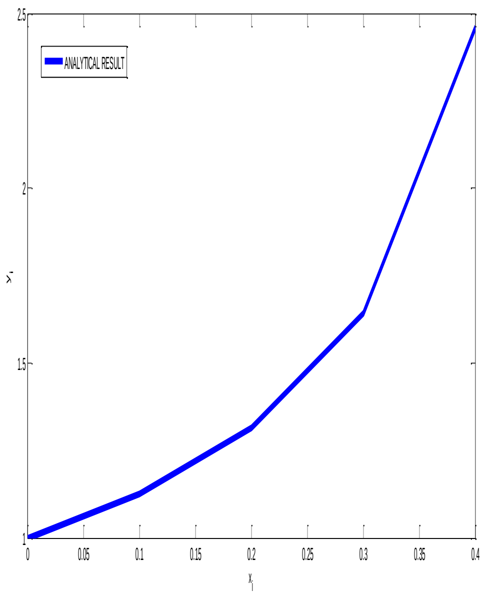

Figure 4.1 Analytical solution

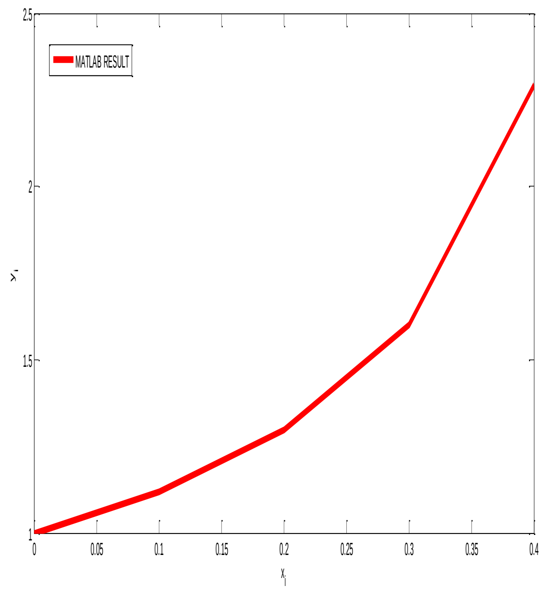

Figure 4.2 MATLAB solution

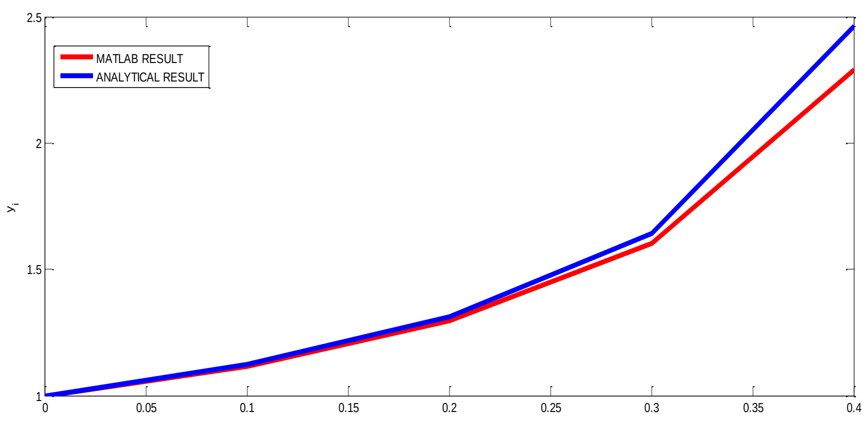

Figure 4.3 Analytical and MATLAB solution.

CONCLUSION

A first order differential equation is a differential equation used to show existing relationship between a given function and its derivatives. It is an equation which involves two variables usually x and y with its function f(x,y) defined on a region in the xy-plane.

In this article, solution of first order ordinary differentiation equation using Fourth order Runge-Kutta method with MATLAB is investigated and an illustrative example is presented to demonstrate efficiency of the method. It can be seen from figure 4.3 that the two solutions are identical at the initial condition of $y(0) = 1$, though the insignificant error varies directly as the value of x and the error eventually propagates through the solution to $x = 0.4$. Generally speaking, the smaller the step size the more it reduces error. Therefore, the above error can be reducing by choosing a smaller step size. The great achievement in this study is that, MATLAB code has been established to easy the stress of analytical method when solving first order differential equations and computer simulation is also performed for clear interpretations of analytical and MATLAB results.

REFERENCES

[1] R. Singh, N. Gupta, " Formulation of Runge-Kutta's method using MATLAB" *International Research Journal of Modernization in Engineering Technology and Science*, vol. 04, no. 11, pp. 739-743, 2022.

[2] M. M. Shior, C. E. Odo, B. C. Agbata, I.G. Ezugorie, S.S. Arivi, " Numerical solution of initial value problem of ordinary differential equations by Adams-Moulton predictor-Corrector method" *International Journal of Mathematics and computer science*, vol. 7 no. 3, pp. 17-30, 2022

[3] M. A. Ramadan, T. Radwan, M. A. Nassar, M. A. Abd El Salam, " A comparison study of numerical Techniques for solving ordinary differential equations defined on a semi-I nfinite domain using rational Chebyshev functions" *Hindawi Journal of function spaces*, vol. 2021, Article ID1111417, pp. 1-12

[4] E. Looez-Sandoval, A. Mello, J. J. Godina-Nava, A. R. Samana, " Power series solution for solving nonlinear Burgers-Type equations" *Hindawi Journal of Abstract and Applied Analysis,* vol. 2015, Article ID712584, pp. 1-9

[5] A. Mustafa, M .M. Hamza, "Derivation for the numerical solution of ordinary differential equations via computer algebra" *Applied and computational mathematics,* vol. 6, no. 4, pp. 167-170, 2017

[6] B. C. Agbata, B. N. Ani, M. M. Shior, I. G. Ezugorie, R. V. Paul, P. K. Meseda, " Analysis of Adomian decomposition method and its application for solving linear and nonlinear differential equations. *ARJOM,* vol. 18, no. 2, pp. 56-70, 2022.

[7] C. Lin, M. M. Lin "Series solution of the differential equation determining the nth- shell one-electron density of a bare coulomb problem in quantum physics," *communications in Nonlinear Science and Numerical Si mulation,* vol 13 pp. 677-681, 2008