American Journal of Engineering Research (AJER)
e-ISSN: 2320-0847 p-ISSN : 2320-0936
Volume-8, Issue-8, pp-40-53
www.ajer.org
Research Paper Open Access

Analysis and Comparative Study of Numerical Solutions of Initial Value Problems (IVP) in Ordinary Differential Equations (ODE) With Euler and Runge Kutta Methods

Anthony Anya Okeke [1,*], Pius Tumba[1], Onyinyechi Favour. Anorue[2], Ahmed Abubakar Dauda[1]

[1] *Department of Mathematics, Federal University Gashua, P.M.B. 1005 Gashua, Yobe State, Nigeria*
[2] *Department of Mathematics, Michael Okpara University of Agriculture, Umudike, Abia State, Nigeria*
Corresponding Author: Anthony Anya Okeke

ABSTRACT : *In this paper, we present Euler's method and fourth-order Runge Kutta Method (RK4) in solving initial value problems (IVP) in Ordinary Differential Equations (ODE). These two proposed methods are quite efficient and practically well suited for solving these problems. For us to obtain and verify the accuracy of the numerical outcomes, we compared the approximate solutions with the exact solution. We found out that there is good agreement between the exact and approximate solutions. We also compared the performance and the computational effort of the two methods. In addition, to achieve more accuracy in the solutions, the step size needs to be very small. Lastly, the error terms have been analyzed for these two methods for different steps sizes and compared also by appropriate examples to demonstrate the reliability and efficiency.*

KEYWORDS *Ordinary Differential Equations (ODE), Initial value Problems (IVP), Euler Method, Runge Kutta Method, Error Analysis.*

Date of Submission: 20-07-2019 Date of Acceptance: 06-08-2019

I. INTRODUCTION

It is a common truth that Differential Equations are among the most important Mathematical tools used in producing models in the engineering, mathematics, physics, aeronautics, elasticity, astronomy, dynamics, biology, chemistry, medicine, environmental sciences, social sciences, banking and many other areas [1]. Many researchers have studied the nature of Differential Equations and many complicated systems that can be described quite precisely with mathematical expressions. A differential equation that has only one independent variable is called an Ordinary Differential Equation (ODE), and all derivatives in it are taken with respect to that variable. Most often, the variable is time, t; although, some authors use x as the independent variable. The differential equation where the unknown function depends on two or more variables is referred to as Partial Differential Equations (PDE). Although there are many analytic methods for finding the solution of differential equations, there exist quite a number of differential equations that cannot be solved analytically [2]. This means that the solution cannot be expressed as the sum of a finite number of elementary functions (polynomials, exponentials, trigonometric, and hyperbolic functions). For simple differential equations, it is possible to find closed form solutions [3]. But many differential equations arising in applications are so complicated that it is sometimes impractical to have solution formulas; even when a solution formula is available, it may involve integrals that can be calculated only by using a numerical quadrature formula. In either case, numerical methods provide a powerful alternative tool for solving the differential equations under the prescribed initial condition or conditions [3].

There are many types of practical numerical methods for solving initial value problems for ordinary differential equations. Historically, the ancestor of all numerical methods in use today was developed by Leonhard Euler between 1768 and 1770 [4], improved Euler's method and Runge Kutta methods described by Carl Runge and Martin Kutta in 1895 and 1905 respectively [5]. We have excellent and exhaustive books on this can be consulted, such as [1-3, 6-15]. In this work, we shall consider two standard numerical methods Euler and Runge Kutta for solving initial value problems of ordinary differential equations.

From the literature review, we found that many authors have worked on numerical solutions of initial value problems using the Euler method and the Runge Kutta method. Many researchers have attempted to solve initial value problems to get a higher accurate solution by applying numerous methods, such as the Euler method and the Runge Kutta method, and many other methods. In [16]**,** the authors studied on some numerical methods for solving initial value problems in ordinary differential equations and also in [17],the authors presented Euler's method for solving initial value problems in ordinary differential equations. Islam [18]discussed accurate solutions of initial value problems for ordinary differential equations with the fourth order Runge Kutta method, while [19] discussed accuracy analysis of numerical solutions of initial value problem for ordinary differential equations. [20-24]also studied numerical solutions of initial value problems in ordinary differential equations using different numerical techniques. In this paper, Euler's method and fourth order Runge Kutta methods are applied without discretization, transformation or restrictive assumptions for solving initial value problems in ordinary differential equations.

Euler's method historically is the first numerical technique. It is also called the tangent line method. It is the simplest to understand and geometrically easy to articulate. The method needs to take a smaller value of h, because of this restriction the method is unsuitable for practical use. If h is not small enough, this method is too inaccurate. A rigorous and elaborate numerical technique is the Runge Kutta method. This technique is the most widely used one since it gives starting values and is particularly suitable when the compilation of higher derivatives is complicated [19]**.** The numerical results are very encouraging.

Finally, we used two examples to illustrate the proposed formulae. The results obtained from each of the numerical examples show that the convergence and error analysis which we presented clearly illustrate the efficiency of the methods. In general, each of the numerical methods has its own advantages and disadvantages to use. Euler's method requires less time consumption, it is simple and single step. Also, in Euler's method $\frac{dy}{dx}$ changes rapidly over an interval, this gives a poor approximation at the beginning of the process in comparison with the average value over the interval. So the calculated value of y in this method occurs much error than the exact value, which reasonably increased in the succeeding intervals, then the final value of y differs on a large scale than the exact value.

In fact, Euler's method needs to take a smaller value of h, and as such, the method is suitable for practical use. If h is not too small enough, this method is inaccurate. But, the Runge Kutta method has the advantage of being the most widely used numerical weapon, since it gives reliable values, starting values and particularly suitable when the computation of higher order derivatives are complicated. It gives a greater accuracy than the Euler's method and also possesses the advantage of requiring only the function values at some selected points on the sub-intervals. Moreover, it is easy to change step-length for special procedures necessary for starting, which minimize the computing time. The Runge Kutta method is very useful and also very laborious. It is a lengthy procedure and needs to check back the values computed earlier. The inherent error in the Runge Kutta method is hard to be estimated and the method has its limitation in solving certain types of differential equations only and the step-length is the key factor of the computation.
Lastly, this paper is structured as follows: Section 2: problem formulations; Section 3: numerical examples; Section 4: discussion of results; and the last section, the conclusion of the paper.

II. PROBLEM FORMULATION

In this section, we consider two numerical methods for finding the approximate solutions of the initial value problem (IVP) of the first order ordinary differential equation of the form

$$y' = f(x,y),\ \ x \in (a,b), \qquad y(a) = y_0 \tag{2.1}$$

Where $y' = \frac{dy}{dx}$ and $f(x,y)$ is a given function and $y(x)$ is the solution of the equation (2.1).

Theorem 2.1 [1]

Let $f(x,y)$ be defined and continuous for all points (x,y) in the region D defined by $a \le x \le b, -\infty < y < \infty$, a and b finite, and let there exist a constant L such that, for every x, y, y^*, such that (x,y) and (x,y^*) are both in D,

$$|f(x,y) - f(x,y^*) \le L(y - y^*)|. \tag{2.2}$$

Then, if y_0 is any given number, there exists a unique solution $y(x)$ of the initial value problem (2.1), where $y(x)$ is continuous and differentiable for all (x,y)in D.
In this paper, we determine the solution of this equation in the range $a \le x \le b$, where a and b are finite, and we assume that f satisfies the Lipchitz conditions stated in Theorem 2.1.

A continuous approximation to the solution $y(x)$ will not be obtained; instead, an approximation to y will be generated at various values. Called mesh points, in the interval $a \le x \le b$**.** Numerical techniques for the solution of (2.1) is to obtain approximations to the values of the solution corresponding to the sequence of points $\{x_n\}$ defined by $x_n = a + nh,\ \ n = 0, 1, 2, 3, \ldots$.The parameter h is called the step size. The numerical solution of (1) is given by a set of points $\{(x_n, y_n): n = 0, 1, 2, 3, \ldots, N\}$ and each point (x_n, y_n) is an approximation to the corresponding point $(x_n, y(x_n))$ on the solution curve.

2.1.1. Euler's Method

Euler's method is also called the tangent method and it is the simplest one-step method. It is the most basic example method for numerical integration of ordinary differential equations. The Euler method is named after Leonhard Euler, who treated it in his book Institutiones Calculi Integralis published 1768-1870, republished in his collected works (Euler 1913) [2]. The Euler method is subdivided into three namely:

- Forward Euler's method
- Improved Euler's method
- Backward Euler's method

In this paper, we shall only consider the forward Euler's method.

1.1.1. Derivative of Euler's Method

Let us consider the initial value problem

$$y' = \frac{dy}{dx} = f(x,y); \quad y(x_0) = y_0 \tag{2.3}$$

We know that if the function f is continuous in the open interval $a < x < b$containing $x = x_0$, there exists a unique solution of the equation (2.3) as

$$y_n = y(x_n); n = 1, 2, 3, \ldots \tag{2.4}$$

The solution is valid for throughout the interval $a < x < b$. We wish to determine the approximate value of y_n of the exact solution $y = y(x)$ in the given interval for the value $x = x_n = x_0 = nh$; $n = 0, 1, 2, \ldots$.Now, the equation of the tangent line through (x_0, y_0) of (3) is

$$y(x) = y_n + f(x_0, y_0)(x - x_0),$$

Setting $x = x_1$, we have

$$y_1 = y_0 + hf(x_0, y_0),$$

Similarly, we get the next approximation as

$$y_2 = y_1 + hf(x_1, y_1), \text{at} x = x_2$$
$$y_3 = y_2 + hf(x_2, y_2), \text{at} x = x_3$$

In general, the $(n + 1)^{th}$ approximation at $x = x_{n+1}$ is given by

$$y_{n+1} = y_n + hf(x_n, y_n); n = 0, 1, 2, 3, \ldots \tag{2.5}$$

1.1.2. Truncation Error for Euler's method

Numerical stability and errors are well discussed in depth in [10, 12-14]. There are two types of errors in the numerical solution of ODEs: Round-off errors and truncation errors. Round-off error occurs because computers use a fixed number of bits and hence fixed the number of binary digits to represent numbers. In a numerical computation round-off errors are introduced at every stage of computation. Hence though an individual round-off error due to a given number at a given numerical step may be small but the cumulative effect can be significant. When the number of bits required for representing a number is less than the number is usually rounded to fit the available number of bits. This is done either by chopping or by symmetric rounding.

Also, truncation errorarises when you use an approximation in place of an exact expression in a mathematical procedure. To estimate the truncation error for the Euler method, we first recall Taylor's Series approximation of a function.

Essentially, Taylor's Theorem State, that, any smooth function can be approximated by a polynomial. The Taylor Series Expansion of f(x) at a is

$$f(x) = f(a) + f'(a)(x - a) + \frac{f''(a)(x - a)^2}{2!} + \frac{f'''(a)(x - a)^3}{3!} + \cdots$$
$$= \sum_{n=0}^{\infty} \frac{f^{(n)}(a)(x - a)^n}{n!} \tag{2.6}$$

We note that computers are discrete, finite machine. They can't perform infinite calculation like these, so we have to cut the calculation somewhere. This result in truncation error (we truncate the expression). When we truncate the Taylor's Series expression to n terms, there's error left over, and we can include a remainder tern R_n to keep the = sign exact.

$$f(x) = f(a) + f'(a)(x - a) + \frac{f''(a)(x-a)^2}{2!} + \frac{f'''(a)(x-a)^3}{3!} + \cdots + \frac{f^{(n)}(a)(x-a)^n}{n!} + R_n \tag{2.7}$$

Where $R_n = \frac{f^{(n+1)}(\zeta).h^{n+1}}{(n+1)!}$, and$x \le \beta \le a$.

In (2.7), let $x = x_{n+1}$ and $x = a$, in which

$$y(x_{n+1}) = y(x_n) + hy'(x_n) + \frac{1}{2}h^2y''(\beta_n) \tag{2.8}$$

Since y satisfies the ordinary differential equation (3), which can be written as

$$y'(x_n) = f(x_n, y(x_n)) \tag{2.9}$$

Hence,'

$$y(x_{n+1}) = y(x_n) + hf(x_n, y(x_n)) + \frac{1}{2}h^2y''(\beta_n) \qquad (2.10)$$

By considering (2.10) to Euler's approximation in (2.5), it is very clear that Euler's method is obtained by omitting the remainder term $\frac{1}{2}h^2y''(\beta_n)$ in the Taylor's expansion of $y(x_{n+1})$ at the point x_n. Therefore, the truncation T_{n+1} error is given by

$$T_{n+1} = y(x_{n+1}) - y(x_n) = hy'(x_n) + \frac{1}{2}h^2y''(\beta_n) \qquad (2.11)$$

Thus, the truncation error is of $O(h^2)$: $h \longrightarrow 0$, i.e. the truncation error is proportional to h^2. By diminishing the size h, the error can be minimized. If M is positive constant such as $|y''(x)| < \frac{M}{2}$

$$|T_{n+1}| < \frac{Mh^2}{2} \qquad (2.12)$$

Here the right hand size is an upper bound of the truncation error. The absolute value of T_{n+1} is taken for the magnitude of the error only.

1.2. Runge Kutta Method

Runge Kutta method is a technique for approximating the solution of ODEs. This technique was developed by two German Mathematicians, Karl Runge by 1894 and extended by Wilhelm Kutta a few years later. The technique is popular because it is efficient, quite accurate, stable, and used in most computer programmes for differential equations. The Runge Kutta methods are distinguished by their order in the sense that they agree with Taylor's series solution up to terms of h^r, where r is the order of the method. It does not demand prior computational of higher derivatives of $y(x)$ as in Taylor's series method. The under-listed are the order of the Runge Kutta Method:

- Runge Kutta Method of order one is called Euler's Method,
- Runge Kutta Method of order two is the same as modified Euler's or Heun's Method,
- The fourth order Runge Kutta Method is called Classical Runge Kutta Method.

In this paper, we shall only focus on the fourth order Runge Kutta Method.

1.2.1. The Derivative of the Fourth Order Runge Kutta method

We shall derive the formula of fourth order Runge Kutta method to obtain an approximate numerical solution of the first order differential equation $y' = f(x,y)$ with the initial condition $y(x_0) = y_0$ and it is assumed that is not a singular point.

Let us take the first order differential equation

$$y' = \frac{dy}{dx} = f(x,y); \quad y(x_0) = y_0 \qquad (2.13)$$

Let $h = x_1 - x_0$, from Taylor's series expansion, we have

$$y(x+h) = y(x) + hy'(x) + \frac{h^2}{2!}y''(x) + \frac{h^3}{3!}y'''(x) + \cdots$$

$$\text{or}\, y(x+h) - y(x) = hy'(x) + \frac{h^2}{2!}y''(x) + \frac{h^3}{3!}y'''(x) + \cdots \qquad (2.14)$$

Differentiating (2.1) partially with respect to variables $x\&y$, we get

$y' = f(x,y) = f$

$y'' = f'(x,y) = f_x + ff_y$

$y'' = f''(x,y) = f_{xx} + 2ff_{xy} + f^2f_{yy} + f_xf_y + ff_y{}^2$

$y^{iv} = f'''(x,y) = (f_{xxx} + 3ff_{xxy} + f^2f_{xyy} + f^3f_{yyy} + f_y(f_{xx} + 2ff_{xy} + f^2f_{yy}) + 3(f_x + ff_y)(f_{xy} + ff_{yy})$
$\qquad + f_y{}^2(f_x + ff_y)$

Let us introduce the following convenient form

$F_1 = f_x + ff_y, \quad F_2 = f_{xx} + 2ff_{xy} + f^2f_{xy}, \quad F_3 = f_{xxx} + 3ff_{xxy} + f^2f_{xxy} + f^2f_{yyy}$

Then we get as

$y' = f, \quad y'' = F_1, \quad y''' = F_2 + f_yF_1$

$$y^{i4} = F_3 + f_yF_2 + 3F_1(f_{xy} + ff_{yy}) + F_1(f_{xy} + ff_{yy}) + F_1f_y{}^2$$

If we now put them into (2.14), we obtain

$$y(x+h) - y(x) = hf + \frac{h^2}{2!}F_1 + \frac{h^3}{3!}(F_1 + f_yF_1)$$
$$+ \frac{h^4}{4!}\{F_3 + f_yF_2 + 3F_1(f_{xy} + ff_{xy}) + F_1(f_{xy} + ff_{yy}) + F_1f_y{}^2\} \qquad (2.15)$$

Now, we shall develop a fourth-order formula. In order to develop the Runge Kutta formula to find the co-efficient $a, b, c, d, m, n\&p$ from below

$k_1 = hf(x,y) = hf,$

$k_2 = hf(x + mh, y + mk_1)$

$k_3 = hf(x+nh, y+nk_2)$
$k_4 = hf(x+ph, y+pk_3)$ (2.16)
Our aim then is Δy will be expressed in the form
$\Delta y = y(x,y) - y(x) = ak_1 + bk_2 + ck_3 + dk_4$ (2.17)
At this stage, we may use Taylor's series expansion for two variables as the followings
$k_1 = hf,$
$k_2 = h[f + mhF_1 + \frac{1}{2}m^2h^2F_2 + \frac{1}{6}m^3h^3F_3 + \cdots\cdots\cdots$
$k_3 = h[f + nhF_1 + \frac{1}{2}h^2(n^2F_2 + 2mnf_yF_1)$
$+\frac{1}{6}h^3\{n^3F_3 + 3m^2nf_yF_2 + 6mn^2F_1(F_{xy} + ff_{yy})\} + \cdots\cdots\cdots]$
$k_4 = h[f + phF_1 + \frac{1}{2}h^2(p^2F_2 + 2npf_yF_1)$
$+\frac{1}{6}h^3\{p^3F_3 + 3n^2pf_yF_2 + 6np^2F_1(F_{xy} + ff_{yy}) + 6mnpF_1{f_y}^2\} + \cdots\cdots\cdots]$
Substituting the values of $k_1, k_2, k_3 \& k_4$ in (2.17), we obtain
$y(x+h) - y(x) = ahf + bh[f + mhF_1 + \frac{1}{2}m^2h^2F_2 + \frac{1}{6}m^3h^3F_3 + \cdots\cdots\cdots +$
$ch[f + nhF_1 + \frac{1}{2}h^2(n^2F_2 + 2mnf_yF_1) + \frac{1}{6}h^3\{n^3F_3 + 3m^2nf_yF_2 + 6mn^2F_1(F_{xy} + ff_{yy})\} + \cdots\cdots\cdots]$
$+dh[f + phF_1 + \frac{1}{2}h^2(p^2F_2 + 2npf_yF_1)$
$+\frac{1}{6}h^3\{p^3F_3 + 3n^2pf_yF_2 + 6np^2F_1(F_{xy} + ff_{yy}) + 6mnpF_1{f_y}^2\} + \cdots\cdots\cdots]$
This can be represented as
$y(x+h) - y(x) = (a+b+c+d)hf + (bm+cn+dp)h^2F_2 +$
$(bm^2+cn^2+dp^2)\frac{h^3F_1}{2} + (bm^3+cn^3+dp^3)\frac{h^4F_2}{6} +$
$(cmn+dnp)h^3f_yF_1 + (cm^2n+dn^2p)h^4f_yF_2 +$
$(cmn^2+dnp^2)h^4F_1(f_{xy} + ff_{yy}) + dmnph^4{f_y}^2F_1 + \cdots\cdots\cdots$ (2.18)
When we compare (2.15) and (2.18), we get
$a+b+c+d = 1,$ $bm+cn+dp = \frac{1}{2},$ $bm^2+cn^2+dp^2 = \frac{1}{3},$ $bm^3+cn^3+dp^3 = \frac{1}{4}$,
$cmn+dnp = \frac{1}{6},$ $cm^2n+dn^2p = \frac{1}{12},$ $cmn^2+dnp^2 = \frac{1}{8},$ $dmnp = \frac{1}{24}$
By solving the above equations, we obtain
$m = n = \frac{1}{2},\ p = 1,\ a = d = \frac{1}{6},\quad b = c = \frac{1}{3}$
Now we put these values in (2.16) and (2.17), we get the fourth-order Runge Kutta formulae as follows:
$k_1 = hf(x,y) = hf,\ k_2 = hf\left(x+\frac{h}{2}, y+\frac{k_1}{2}\right), k_3 = hf(x+\frac{h}{2}, y+\frac{k_2}{2}), k_4 = hf(x+h, y+k_3)$
$\Delta y = y(x+h) - y(x) = \frac{1}{6}(k_1 + 2k_2 + 2k_3 + k_4)$
When the initial values are (x_0, y_0), then, the first increment in y is computed from the given formulae below
$k_1 = hf(x_0,y_0) = hf, k_2 = hf\left(x_0+\frac{h}{2}, y_0+\frac{k_1}{2}\right), k_3 = hf(x_0+\frac{h}{2}, y_0+\frac{k_2}{2}),$
$k_4 = hf(x_0+h, y_0+k_3)$
$\Delta y = \frac{1}{6}(k_1 + 2k_2 + 2k_3 + k_4)$
or, $y(x_0+h) = y(x_0) + \frac{1}{6}(k_1 + 2k_2 + 2k_3 + k_4)$
or, $y_1 = y_0 + \frac{1}{6}(k_1 + 2k_2 + 2k_3 + k_4)$
Hence, the general fourth-order Runge Kutta formulae for the n^{th} interval is given by the followings:
$k_1 = hf(x_n,y_n) = hf,\quad k_2 = hf\left(x_n+\frac{h}{2}, y_n+\frac{k_1}{2}\right),\quad k_3 = hf(x_n+\frac{h}{2}, y_n+\frac{k_2}{2}),$
$k_4 = hf(x_n+h, y_n+k_3)$
$y_{n+1} = y_n + \frac{1}{6}(k_1 + 2k_2 + 2k_3 + k_4)$ (2.19)

1.2.2. Truncation Error in Runge Kutta Method

One of the serious drawbacks of Runge Kutta method is error estimation [17]. The direct method of estimating the error of higher order Runge Kutta formulae are very complicated and time consuming. Moreover, it is possible to computing the errors in laborious ways, are very hard, involving higher order partial derivatives.

We shall first estimate the error in second-order Runge Kutta formulae and the errors for higher orders can be obtained by generalizing the computed error. We get the second order Runge Kutta formulae as follows:

$$y_{n+1} = y_n + \frac{1}{2}(k_1 + k_2)$$
$$k_1 = hf(x_n, y_n)$$
$k_2 = hf(x_n + h, y_n + k_1)$ (2.20)

Now, the truncated error is given by the following formula

$$E_r = y(x_{n+1}) - y_{n+1} \tag{2.21}$$

Now expanding $y(x_{n+1})$ by Taylor's series expansion, we get

$$y(x_{n+1}) = y(x_n + h) = y(x_n) + hy'(x_n) + \frac{h^2}{2!}y''(x_n) + \frac{h^3}{3!}y'''(x_n) + \frac{h^4}{4!}y''''(x_n)\cdots\cdots\cdots$$
$$= y_n + hf + \frac{h^2}{2!}(f_x + ff_y) + \frac{h^3}{3!}(f_{xx} + 2ff_{xy} + f^2f_{yy} + f_xf_y + ff_y{}^2) + o(h^4) \tag{2.22}$$

We may use Taylor's series expansion in (2.20), we get

$$k_1 = hf$$
$$k_2 = h\left[f + h(f_x + ff_y) + \frac{h^2}{2}(f_{xx} + 2ff_{xy} + f^2f_{yy}) + \cdots\cdots\cdots\right]$$
$$= hf + h^2(f_x + ff_y) + \frac{h^3}{2}(f_{xx} + 2ff_{xy} + f^2f_{yy}) + \cdots\cdots\cdots$$
$$y_{n+1} = y_n + \frac{1}{2}\left[hf + hf + h^2(f_x + ff_y) + \frac{h^3}{2}(f_{xx} + 2ff_{xy} + f^2f_{yy}) + \cdots\cdots\cdots\right]$$
$$or y_{n+1} = y_n + hf + \frac{h^2}{2}(f_x + ff_y) + \frac{h^3}{4}(f_{xx} + 2ff_{xy} + f^2f_{yy}) + \tag{2.23}$$

Now, substituting (2.22) and (2.23) in (2.21), we get

$$E_r = \left[y_n + hf + \frac{h^2}{2}(f_x + ff_y) + \frac{h^3}{6}(f_{xx} + 2ff_{xy} + f^2f_{yy} + f_xf_y + ff_y{}^2) + \cdots\right] -$$
$$\left[y_n + hf + \frac{h^2}{2}(f_x + ff_y) + \frac{h^3}{4}(f_{xx} + 2ff_{xy} + f^2f_{yy}) + \cdots\right]$$
$$= \left(\frac{h^3}{6} - \frac{h^2}{4}\right)(f_{xx} + 2ff_{xy} + f^2f_{yy}) + \frac{h^3}{6}(f_xf_y + ff_y{}^2 + \cdots$$
$$= -\frac{h^3}{12}(f_{xx} + 2ff_{xy} + f^2f_{yy}) + \frac{h^3}{6}(f_xf_y + ff_y{}^2) + \cdots$$
$$= -\frac{h^3}{12}(f_{xx} + 2ff_{xy} + f^2f_{yy}) + 2f_y - 2ff_y{}^2\cdots \tag{2.24}$$

Hence, (2.24) shows that the truncation error of the second-order Runge Kutta formula is of order h^3. Similarly, we can show that the truncation errors in the third-order, fourth-order Runge Kutta formula are of h^4&h^5 respectively.

Hence, by applying Taylor's series expansion as above manner, we get the truncation error of the nth -order Runge Kutta formulae of order h^{n+1} as follows

$$E_r = ch^{n+1}y^{n+1} \tag{2.25}$$

III. NUMERICAL EXAMPLES

In this section, we present two numerical examples prove which of the two numerical methods converge faster to the analytical solutions. The numerical results and errors are computed and the findings are represented graphically. The computations were done using MATLAB programing language. The convergence of the IVP is calculated $e_n = |y(x_n) - y_n| < \delta$ where $y(x_n)$ represents the exact solution and δ depends on the problem which varies from 10^{-6} while the absolute error is computed by $|y(x_n) - y_n|$.

Example 1: We consider the initial value problem $y' = 2y + 4 - x,\ y(0) = 0.5,$ on the interval $0 \le x \le 1$. The exact solution of the given problem is given by $y(x) = -\frac{7}{4} + \frac{1}{2}x + \frac{9}{4}e^{2x}$. The approximate results and the absolute errors are derived and shown in Tables 1(a)–(d) while the graphs of the numerical solutions are displayed in Figures 1-7.

Table 1. (a) Numerical approximations and absolutes errors for step size $h = 0.1$; (b) Numerical approximations and absolutes errors for step size $h = 0.05$; (c) Numerical approximations and absolutes errors for step size$h = 0.025$; (d) Numerical approximations and absolutes errors for step size $h = 0.0125$.

(a)

n	x_n	Exact Solution y_n	Runge Kutta Method $h = 0.1$		Euler Method $h = 0.1$	
			$y(x_n)$	$errors$	$y(x_n)$	$errors$
0	0.0	0.5000000000	0.5000000000	0.0000000000	0.5000000000	0.0000000000
1	0.1	1.0481562059	1.0481500000	0.0000062059	1.0000000000	0.0481562059
2	0.2	1.7066055697	1.7065904100	0.0000151597	1.5900000000	0.1166055697
3	0.3	2.4997673009	2.4997395268	0.0000277741	2.2880000000	0.2117673009
4	0.4	3.4574670891	3.4574218580	0.0000452311	3.1156000000	0.3418670891
5	0.5	4.6161341140	4.6160650574	0.0000690567	4.0987200000	0.5174141140
6	0.6	6.0202630762	6.0201618611	0.0001012151	5.2684640000	0.7517990762
7	0.7	7.7241999254	7.7240556971	0.0001442283	6.6621568000	1.0620431254
8	0.8	9.7943229549	9.7941216284	0.0002013264	8.3245881600	1.4697347949
9	0.9	12.3117067949	12.3114301570	0.0002766380	10.3095057920	2.0022010029
10	1.0	15.3753762226	15.3750007937	0.0003754289	12.6814069504	2.6939692722

(b)

n	x_n	Exact Solution y_n	Runge Kutta Method $h = 0.05$		Euler Method $h = 0.05$	
			$y(x_n)$	$errors$	$y(x_n)$	$errors$
0	0.0	0.5000000000	0.5000000000	0.0000000000	0.5000000000	0.0000000000
1	0.1	1.0481562059	1.0481557844	0.0000004214	1.0225000000	0.0256562059
2	0.2	1.7066055697	1.7066045402	0.0000010295	1.6442250000	0.0623805697
3	0.3	2.4997673009	2.4997654147	0.0000018862	2.3860122500	0.1137550509
4	0.4	3.4574670891	3.4574640174	0.0000030717	3.2730748225	0.1843922666
5	0.5	4.6161341140	4.6161294243	0.0000046897	4.3359205352	0.2802135788
6	0.6	6.0202630762	6.0202562025	0.0000068737	5.6114638476	0.4087992285
7	0.7	7.7241999254	7.7241901306	0.0000097948	7.1443712556	0.5798286698
8	0.8	9.7943229549	9.7943092825	0.0000136724	8.9886892193	0.8056337356
9	0.9	12.3117067949	12.3116880080	0.0000187869	11.2098139554	1.1018928396
10	1.0	15.3753762226	15.3753507266	0.0000254960	13.8868748860	1.4885013370

(c)

n	x_n	Exact Solution y_n	Runge Kutta Method $h = 0.025$		Euler Method $h = 0.025$	
			$y(x_n)$	$errors$	$y(x_n)$	$errors$
0	0.0	0.5000000000	0.5000000000	0.0000000000	0.5000000000	0.0000000000
1	0.1	1.0481562059	1.0481561784	0.0000000275	1.0348890625	0.0132671434
2	0.2	1.7066055697	1.7066055026	0.0000000671	1.6742747485	0.0323308212
3	0.3	2.4997673009	2.4997671780	0.0000001229	2.4406767335	0.0590905673
4	0.4	3.4574670891	3.4574668890	0.0000002001	3.3614678239	0.0959992652
5	0.5	4.6161341140	4.6161338085	0.0000003056	4.4699198366	0.1462142775
6	0.6	6.0202630762	4.6161338085	0.0000004478	5.8064748734	0.2137882028
7	0.7	7.7241999254	7.7241992872	0.0000006382	7.4202905615	0.3039093639
8	0.8	9.7943229549	9.7943220641	0.0000008908	9.3711183044	0.4232046505
9	0.9	12.3117067949	12.3117055709	0.0000012240	11.7315863059	0.5801204890
10	1.0	15.3753762226	15.3753745614	0.0000016612	14.5899746023	0.7854016203

(d)

n	x_n	Exact Solution y_n	Runge Kutta Method $h = 0.0125$		Euler Method $h = 0.0125$	
			$y(x_n)$	errors	$y(x_n)$	errors
0	0.0	0.5000000000	0.5000000000	0.0000000000	0.5000000000	0.0000000000
1	0.1	1.0481562059	1.0481562041	0.0000000018	1.0414065194	0.0067496865
2	0.2	1.7066055697	1.7066055654	0.0000000043	1.6901376465	0.0164679232
3	0.3	2.4997673009	2.4997672930	0.0000000078	2.4696333866	0.0301339143
4	0.4	3.4574670891	3.4574670763	0.0000000128	3.4084531100	0.0490139791
5	0.5	4.6161341140	4.6161340945	0.0000000195	4.5413936364	0.0747404777
6	0.6	6.0202630762	6.0202630476	0.0000000286	5.9108515116	0.1094115646
7	0.7	7.7241999254	7.7241998847	0.0000000407	7.5684828098	0.1557171156
8	0.8	9.7943229549	9.7943228980	0.0000000568	9.5772254418	0.2170975131
9	0.9	12.3117067949	12.3117067168	0.0000000781	12.0137631400	0.2979436549
10	1.0	15.3753762226	15.3753761166	0.0000001060	14.9715275865	0.4038486361

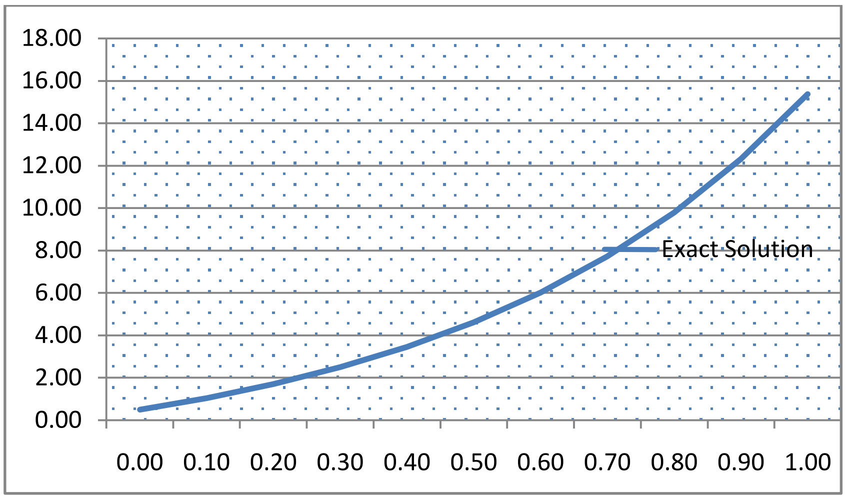

Figure 1: Exact Numerical Solution

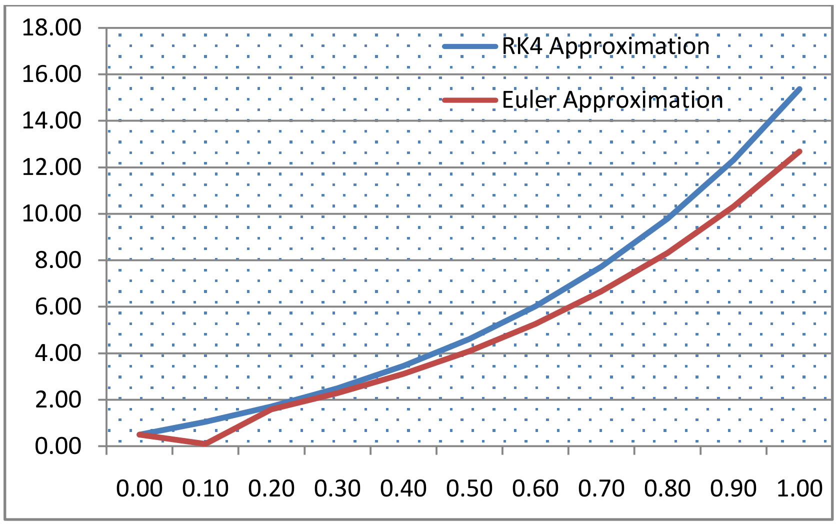

Figure 2: Numerical Approximation for step size h=0.1

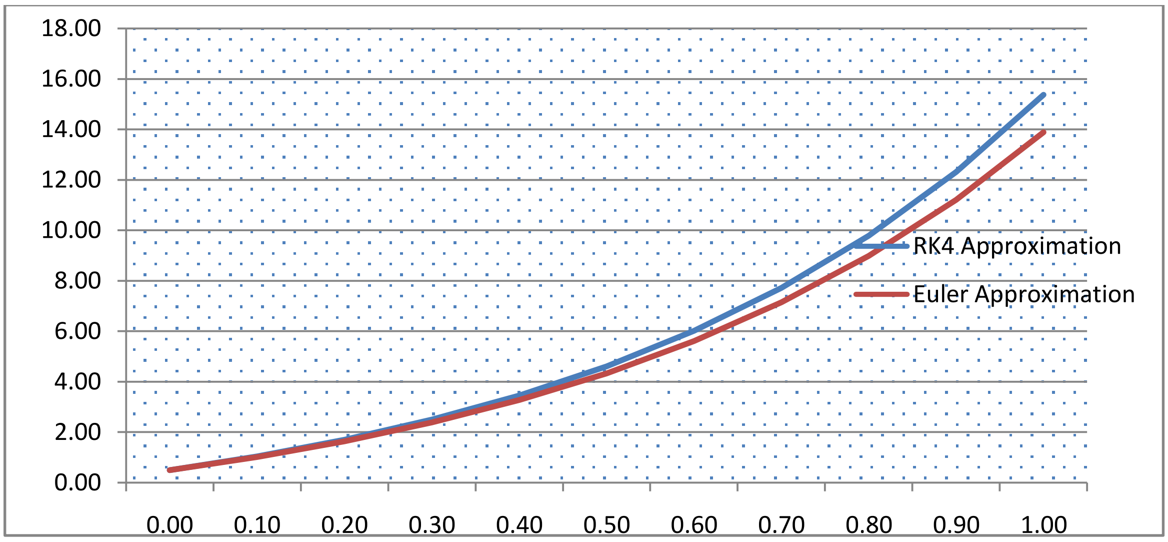

Figure 3: Numerical Approximation for step size h=0.05

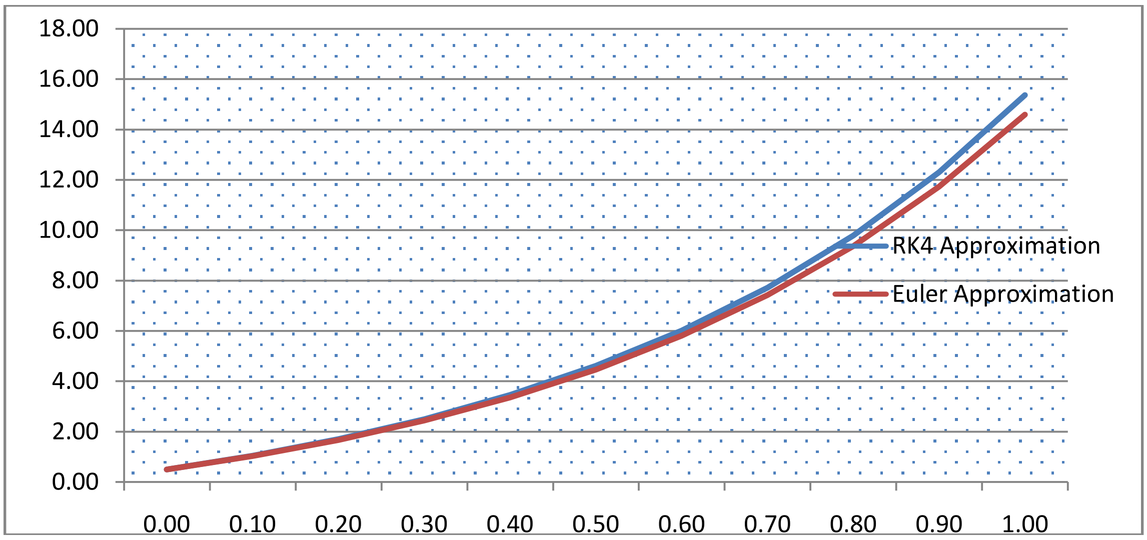

Figure 4: Numerical Approximation for step size h=0.025

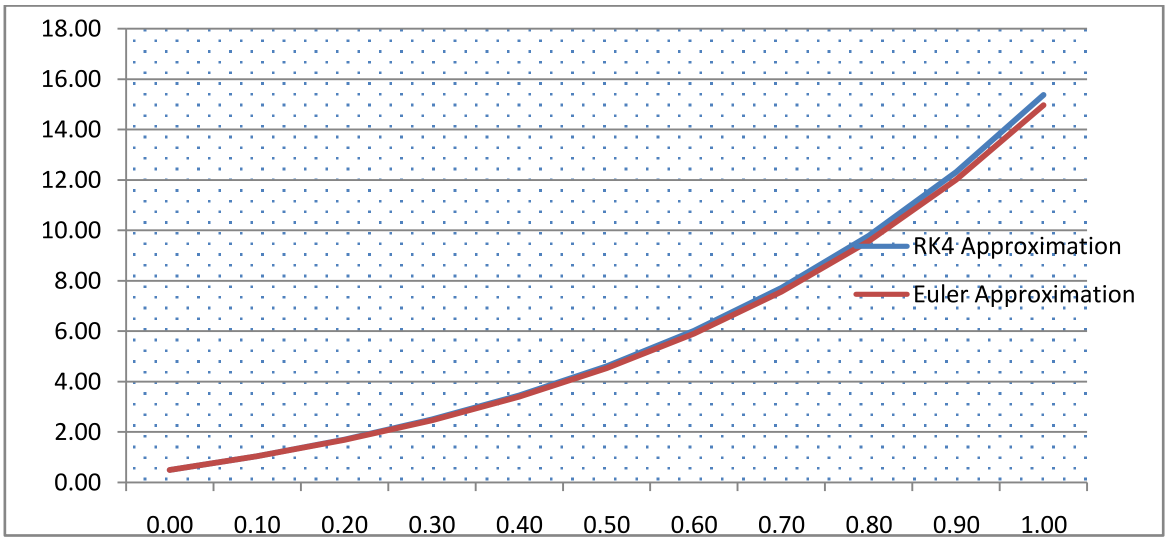

Figure 5: Numerical Approximation for step size h=0.0125

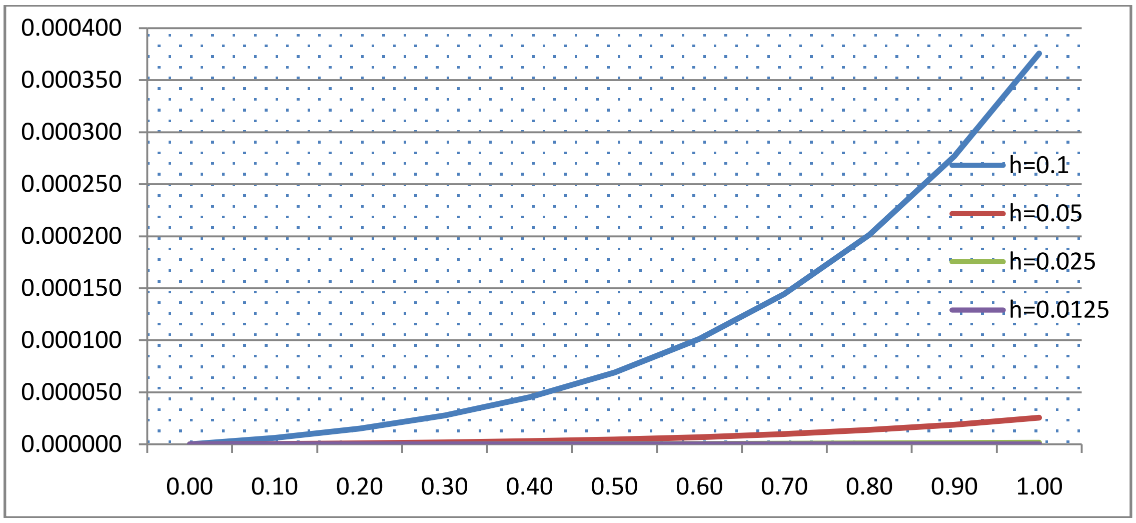

Figure 6: Error for different step size using RK4 method

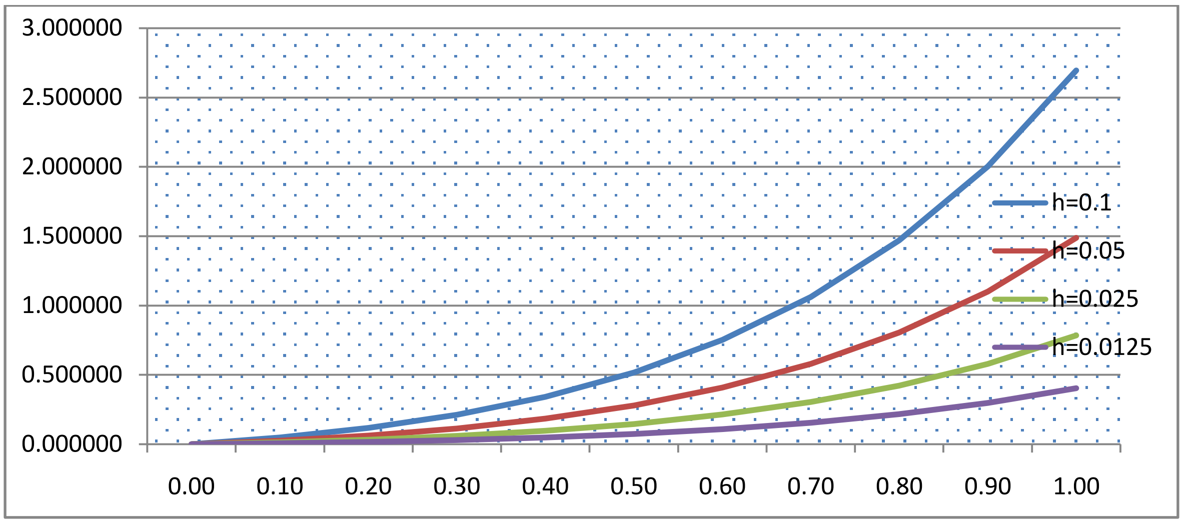

Figure 7: Error for different step size using Euler's method

Example 2: We consider the initial value problem $y' = y - x^2 + 1,\ y(0) = 0.5$, on the interval$0 \le x \le 2$. The exact solution of the given problem is given by $y(x) = x^2 + 2x + 1 - 0.5e^x$. The approximate results and the absolute errors are obtained and shown in Tables 2(a)-(d) while the graphs of the numerical solutions are displayed in Figures 8-14.

Table 2. (a) Numerical approximations and absolutes errors for step size $h = 0.1$; (b) Numerical approximations and absolutes errors for step size $h = 0.05$; (c) Numerical approximations and absolutes errors for step size $h = 0.025$, (d) Numerical approximations and absolutes errors for step size $h = 0.0125$.

(a)

n	**x_n**	**Exact Solution y_n**	**Runge Kutta Method $h = 0.1$**		**Euler Method $h = 0.1$**	
			$y(x_n)$	**errors**	**$y(x_n)$**	**errors**
0	0.0	0.5000000000	0.5000000000	0.0000000000	0.5000000000	0.0000000000
1	0.1	0.6574145401	0.6574143750	0.0000001660	0.6500000000	0.0074145410
2	0.2	0.8292986209	0.8292982760	0.0000003449	0.8140000000	0.0152986209
3	0.3	1.0150705962	1.0150700584	0.0000005378	0.9914000000	0.0236705962
4	0.4	1.2140876511	1.2140869057	0.0000007455	1.1815400000	0.0325476512
5	0.5	1.4256393646	1.4256383956	0.0000009690	1.3836940000	0.0419453646
6	0.6	1.6489405998	1.6489393904	0.0000012094	1.5970634000	0.0518771998
7	0.7	1.8831236463	1.8831221786	0.0000014677	1.8207697400	0.0623539063
8	0.8	2.1272295358	2.1272277907	0.0000017451	2.0538467140	0.0733828218
9	0.9	2.3801984444	2.3801964018	0.0000020426	2.2952313854	0.0849670590
10	1.0	2.6408590858	2.6408567242	0.0000023616	2.5437545239	0.0971045619

(b)

n	**x_n**	**Exact Solution y_n**	**Runge Kutta Method $h = 0.05$**		**Euler Method $h = 0.05$**	
			$y(x_n)$	**errors**	**$y(x_n)$**	**errors**
0	0.0	0.5000000000	0.5000000000	0.0000000000	0.5000000000	0.0000000000
1	0.1	0.6574145401	0.6574145304	0.0000000106	0.6536250000	0.0037895410
2	0.2	0.8292986209	0.8292985989	0.0000000220	0.8214715625	0.0078270584
3	0.3	1.0150705962	1.0150705619	0.0000000343	1.0029473977	0.0121231986
4	0.4	1.2140876511	1.2140876036	0.0000000475	1.1973995059	0.0166881453
5	0.5	1.4256393646	1.4256393029	0.0000000618	1.4041079553	0.0215314094
6	0.6	1.6489405998	1.6489405227	0.0000000771	1.6222790207	0.0266615791
7	0.7	1.8831236463	1.8831235527	0.0000000935	1.8510376203	0.0320860260
8	0.8	2.1272295358	2.1272294246	0.0000001111	2.0894189764	0.0378105594
9	0.9	2.3801984444	2.3801983144	0.0000001300	2.3363594215	0.0438390230
10	1.0	2.6408590858	2.6408589355	0.0000001503	2.5906862622	0.0501728236

(c)

n	x_n	Exact Solution y_n	Runge Kutta Method $h = 0.025$		Euler Method $h = 0.025$	
			$y(x_n)$	**errors**	$y(x_n)$	**errors**
0	0.0	0.5000000000	0.5000000000	0.0000000000	0.5000000000	0.0000000000
1	0.1	0.6574145401	0.6574145403	0.0000000007	0.6554982324	0.0019163085
2	0.2	0.8292986209	0.8292986195	0.0000000014	0.8253384788	0.0039601421
3	0.3	1.0150705962	1.0150705940	0.0000000022	1.0089333673	0.0061372289

4	0.4	1.2140876511	1.2140876482	0.0000000030	1.2056345492	0.0084531020
5	0.5	1.4256393646	1.4256393608	0.0000000039	1.4147263688	0.0109129958
6	0.6	1.6489405998	1.6489405949	0.0000000049	1.6354188765	0.0135217233
7	0.7	1.8831236463	1.8831236404	0.0000000059	1.8668401152	0.0162835311
8	0.8	2.1272295358	2.1272295287	0.0000000070	2.1080276077	0.0192019281
9	0.9	2.3801984444	2.3801984362	0.0000000082	2.3579189592	0.0222794852
10	1.0	2.6408590858	2.6408590763	0.0000000095	2.6153414848	0.0255176010

(d)

n	x_n	Exact Solution y_n	Runge Kutta Method h = 0.0125		Euler Method h = 0.025	
			$y(x_n)$	**errors**	$y(x_n)$	**errors**
0	0.0	0.5000000000	0.5000000000	0.0000000000	0.5000000000	0.0000000000
1	0.1	0.6574145401	0.6574145409	0.0000000000	0.6564508731	0.0009636678
2	0.2	0.8292986209	0.8292986208	0.0000000001	0.8273066068	0.0019920141
3	0.3	1.0150705962	1.0150705961	0.0000000001	1.0119825867	0.0030880095
4	0.4	1.2140876511	1.2140876510	0.0000000002	1.2098331143	0.0042545368
5	0.5	1.4256393646	1.4256393644	0.0000000002	1.4201450250	0.0054943397
6	0.6	1.6489405998	1.6489405995	0.0000000003	1.6421306378	0.0068099620
7	0.7	1.8831236463	1.8831236459	0.0000000004	1.8749199705	0.0082036758
8	0.8	2.1272295358	2.1272295353	0.0000000004	2.1175521396	0.0096773962
9	0.9	2.3801984444	2.3801984439	0.0000000005	2.3689658626	0.0112325818
10	1.0	2.6408590858	2.6408590852	0.0000000006	2.6279889679	0.0128701179

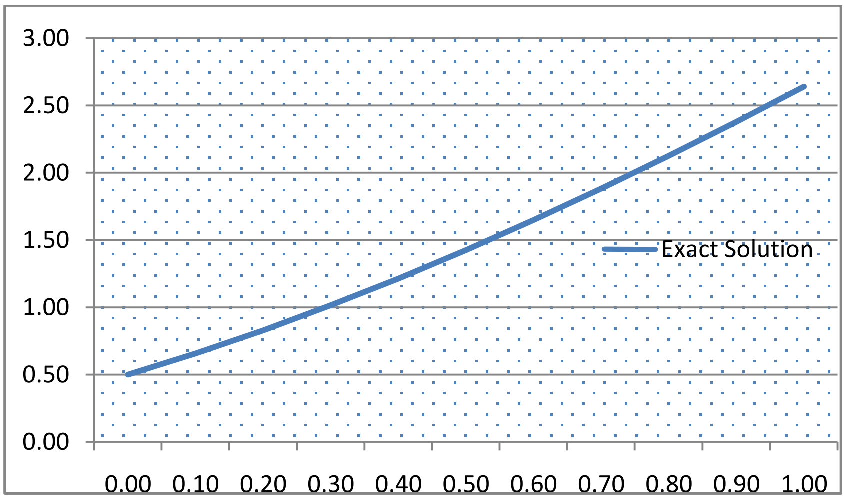

Figure 8: Exact Numerical Solution

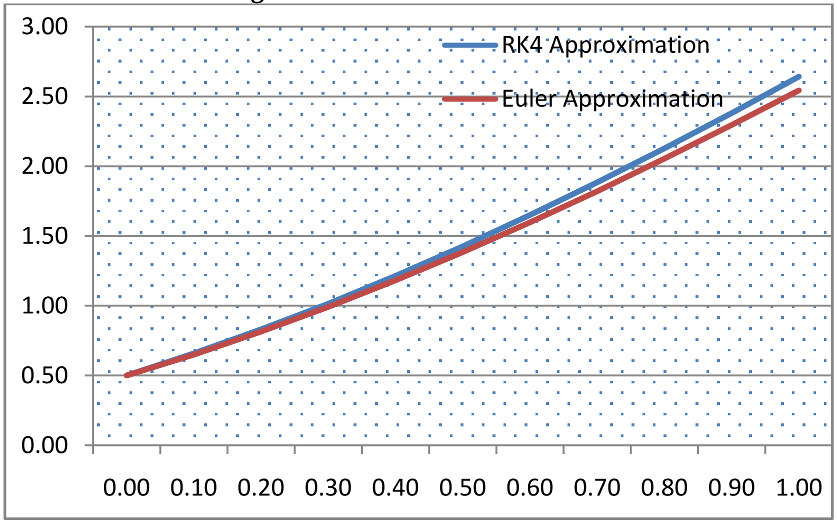

Figure 9: Numerical Approximation for step size h=0.1

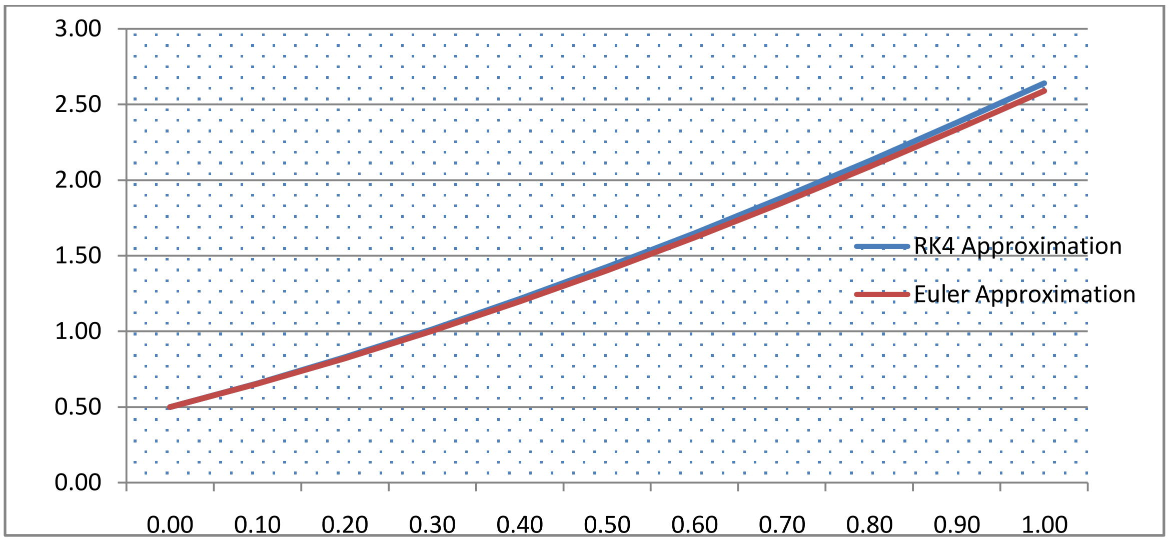

Figure 10: Numerical Approximation for step size h=0.05

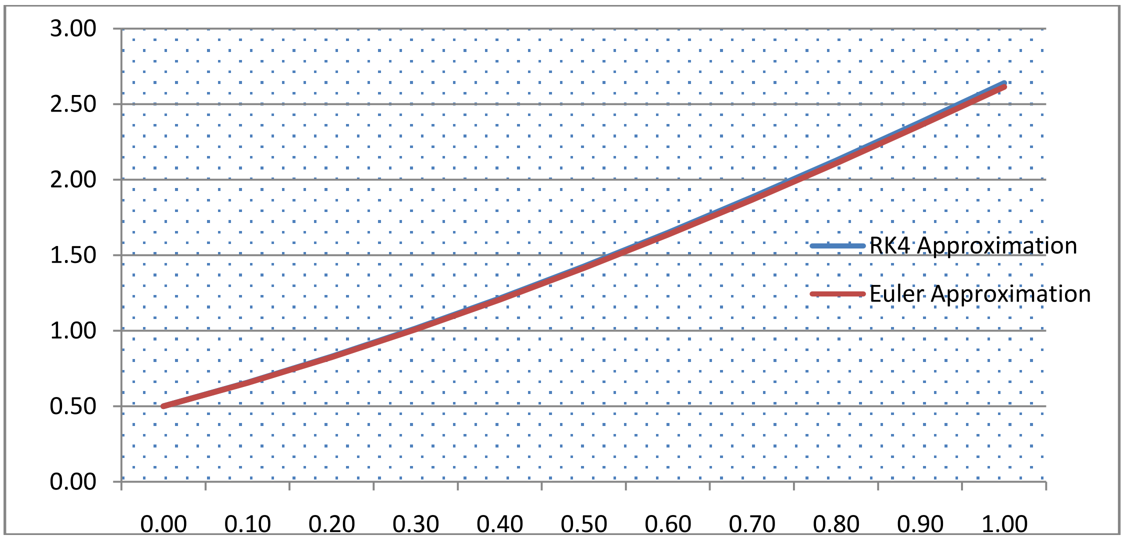

Figure 11: Numerical Approximation for step size h=0.025

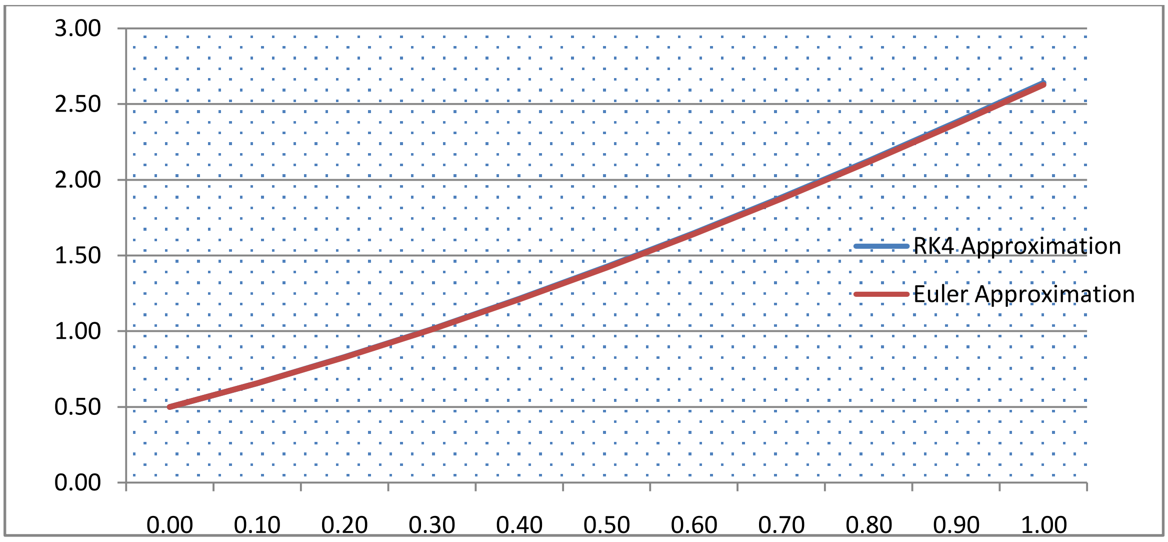

Figure 12: Numerical Approximation for step size h=0.0125

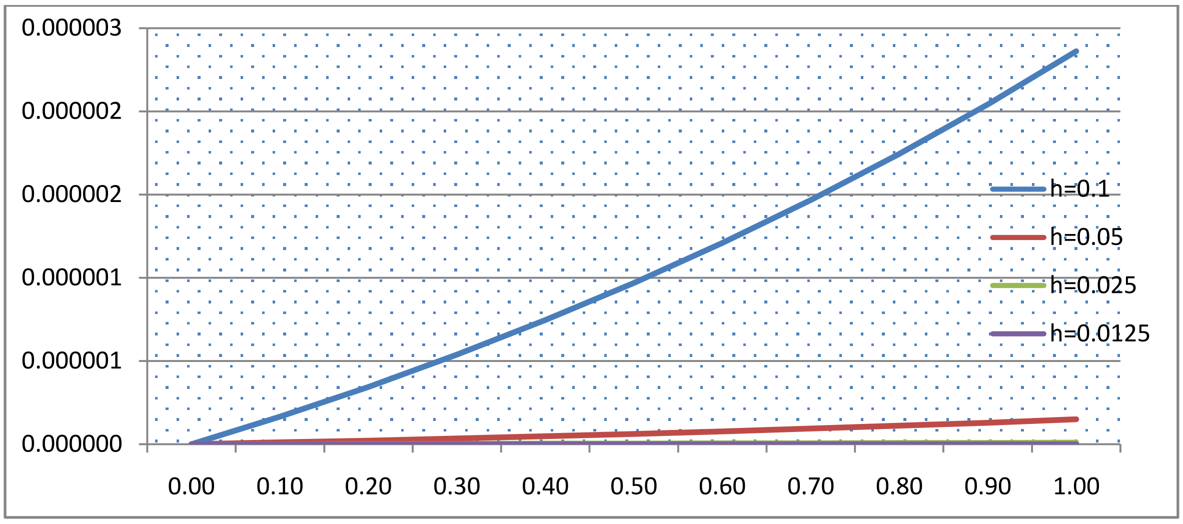

Figure 13: Error for different step size using RK4 method

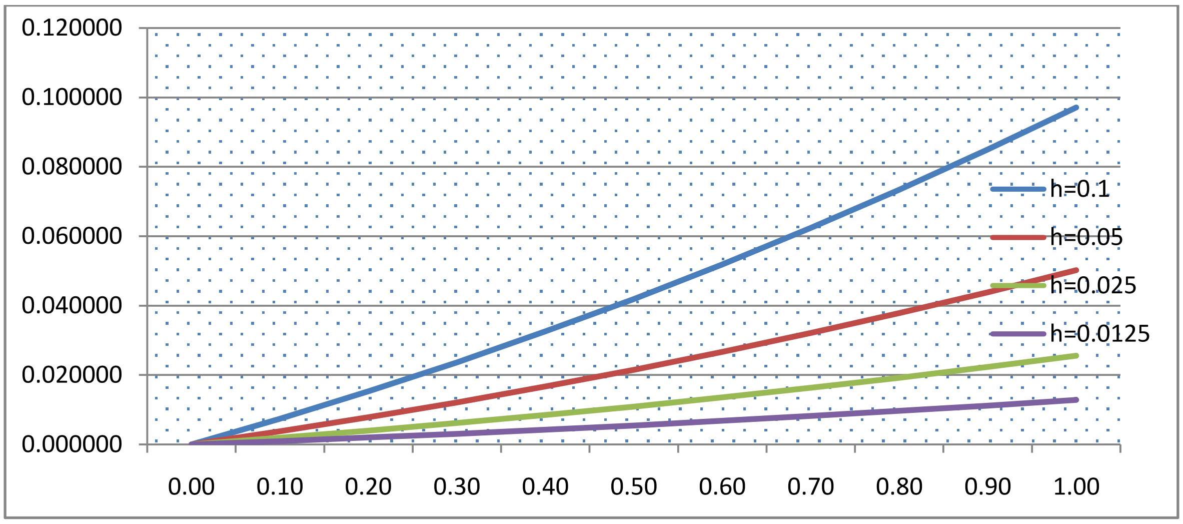

Figure 14: Error for different step size using Euler's method

IV. DISCUSSION AND RESULTS

The obtained results are displayed in Table 1(a)-(d) and Table 2(a)-(c) and graphically represented in Figures (1-7) and Figures (8-14) respectively. The approximate solutions and absolute errors are calculated using Matlab programming language with the step sizes 0.1, 0.05, 0.025, and 0.0125 and also computed with the exact solution. From the tables for each of the methods, we observed that the numerical solutions converge to the exact solution and the errors incurred in the Euler's method are greater than that of the Runge Kutta method. We also observed that the Runge Kutta approximations for the same size converge firstly to the exact solution. This indicates that the small step size provides a better approximation. The fourth order Runge Kutta method is laborious, it requires four evaluations per step size, but it gives more accurate results than the Euler's method with only one-fourth the step size. We equally observed that the fourth order Runge Kutta Method converges faster, more accurate and cost effective than the Euler's method (as can be seen in the tables and figures) in solving initial value problems in ordinary differential equations.

V. CONCLUSION

In this paper, the fourth order Runge Kutta and Euler's methods are used for solving initial value problems (IVP) in Ordinary Differential Equations (ODE). To find more accurate results, we reduced the step size for both methods. From our tables and figures, we analyzed that the solution for both methods converges to the exact solution for decreasing the step size h. The numerical solutions obtained by the two methods are in good agreement with the exact solutions. However, by comparing the results of the two methods, we state that the RK4 Method is appropriate, consistent, convergent, quite stable, and more accurate than the Euler's method and it is widely used in numerical solutions of initial value problems in ordinary differential equations. In our subsequent research, we shall examine the comparison of RK4 method with other existing method like the Adomian decomposition.

REFERENCES

[1]. Lambert,J. (2000). Computational Methods in Ordinary Differential Equations. New York: Wiley & Sons: 21-205IEA:

[2]. Butcher ,J. (2003).Numerical Methods for Ordinary Differential Equations. West Sussex: John Wiley & Sons Ltd. 45-95.

[3]. Atkinson K, Han W, Stewart D, (2009) Numerical Solution of Ordinary Differential Equations. New Jersey: John Wiley & Sons, Hoboken: 70-87.

[4]. Euler L, (1768) Institutiones Calculi IntegralisVolumenPrimum, Opera Omnia. Vol. XI, B. G. TeubneriLipsiaeetBerolini MCMXIII: 21-228.

[5]. Euler, L. (1913).De integration aequationum di erentialium per approximationem, In Opera Omnia, 1st series, Vol II, Institutiones Calculi Integralis, Teubner, Leipzig and Berlin, 424434.

[6]. Brenan, K., Campbell S, Petzold L, (1989).Numerical Solution of Initial-Value Problems in Differential-Algebraic Equations. New York: Society for Industrial and Applied Mathematics: 76-127.

[7]. Lambert, J. (1999).Numerical Methods for Ordinary Differential Systems: The Initial value Problem. New York: John Wiley &Sons: 149-205.

[8]. Boyce W, DiPrima R, (2000) Elementary Differential Equations and Boundary Value Problems. New York: John Wiley & Sons, Inc.: 419-471.

[9]. Carnahan, B,, Luther, H., Wikes, J. (1990). Applied Numerical Methods, Florida: Krieger Publishing Company: 341-386.

[10]. Chapra, S.,Canale, R. (2006). Applied Numerical Methods with MATLAB for Engineers and Scientists, 6 Ed,. Boston: McGraw Hill: 707-742.

[11]. Esfandiari R, (2013) Numerical Methods for Engineers and Scientists using MATLAB. New York: CRC Press (CRC), Taylor & Francis Group: 329-351.

[12]. Fatunla, S., (1988).Numerical Methods for Initial Value Problems in Ordinary Differential equations. Boston: Academic Press, INC: 41-62.

[13]. Conte, S., Boor,C.Elementary Numerical Analysis: Algorithmic Approach. New York: McGraw-Hill Book Company: 356-387.

[14]. Kreyszig, E. (2011).Advanced Engineering Mathematics, 10 Eds,. Boston: John Wiley & Sons, Inc: 921-937.

[15]. Iserles, A. (1996).A First Course in the Numerical Analysis of Differential Equations, Cambridge: Cambridge University Press: 1-50.

[16]. Bosede, O., Emmanuel, F.,Temitayo, O. (2012). On Some Numerical Methods for Solving Initial Value Problems in Ordinary Differential Equations. IOSR Journal of Mathematics (IOSRJM) Vol. 1 (3): 25-31.

[17]. Fadugba, S.,Ogunrinde, B., Okunlola, T. (2012).Euler's Method for Solving Initial Value Problems in Ordinary Differential Equations. The Pacific Journal of Science and Technology, Vol. 13 (2): 152-158.

[18]. Islam, Md. A. (2015). Accurate Solutions of Initial Value Problems for Ordinary Differential Equations with the Fourth Order Runge Kutta Method. Journal of Mathematics Research, Vol. 7 (3): 41-45.

[19]. Islam, M. A. (2015). Accurate Analysis of Numerical Solutions of Initial Value Problems (IVP) for ordinary differential equations (ODE). IOSR Journal of Mathematics (IOSR-JM), Vol. 11 (3): 18-23.

[20]. Jamali N, (2019) Analysis and Comparative Study of Numerical Methods to Solve Ordinary Differential Equation with Initial Value Problem. International Journal of Advanced Research (IJAR). Vol. 7(5): 117-128: Available from: http://www.journalijar.com/uploads/536_IJAR-27303.pdf.

[21]. Hossain, B. B.,Hossain, M. J., MiahMd, et al. (2017) A Comparative Study on Fourth Order and Butcher's Fifth Order Runge Kutta Methods with Third Order Initial Value Problem (IVP). Applied and Computational Mathematics, Vol. 6(6): 243-253. Available from: doi:10.11648/j.acm.20170606.12.

[22]. Fadugba, S.E.,Olaosebikan, T.E, (2018). Comparative Study of a Class of One-Step Methods for the Numerical Solutions of Some Initial Value Problems in ordinary Differential Equations. Research Journal of Mathematics and Computer Science: 2-9: Available from: https://escipub.com/Articles/RJMCS/RJMCS-2017-12-1801.

[23]. Hamed, A. B., Alrhaman, I. Y.,Sani, I, (2017). The accuracy of Euler and modified Euler technique for First Order Ordinary Differential Equations with initial conditions. American journal of Engineering Research (AJER), Vol. 6(9): 334-338.

[24]. Oshinubi IK, Ogunjimi OA, Longe OB, (2017) On some Computational Methods for Solving an Ordinary Differential Equations with Initial Condition. Proceedings of the ISTEAMS Multidisciplinary Cross-Border Conference, University of Ghana, Legon: 73-80.