Numerical Investigation on System of Ordinary Differential Equations Absolute Time Inference with Mathematica®

Adeniji Adejimi[1], Surulere Samuel[2], Mkolesia Andrew[3], Shatalov Michael[4]
Department of Mathematics and Statistics
Tshwane University of Technology
Pretoria, South Africa

***Abstract*—The purpose of this research is to perform a comparative numerical analysis of an efficient numerical methods for second-order ordinary differential equations, by reducing the second-order ODE to a system of first-order differential equations. Then we obtain approximate solutions to the system of ODE. To validate the accuracy of the algorithm, a comparison between Euler's method and the Runge-Kutta method or order four was carried out and an exact solution was found to verify the efficiency, accuracy of the methods. Graphical representations of the parametric plots were also presented. Time inference analysis is taken to check the time taken to executes the algorithm in Mathematica®12.2.0. The obtained approximate solution using the algorithm shows that the Runge-Kutta method of order four is more efficient for solving system of linear ordinary differential equations.**

***Keywords*—Euler's method; Runge-Kutta method; System of ODE; Mathematica®; AbsoluteTiming**

I. Introduction

In recent times, computer algebraic system (CAS) have been employed to investigate the use of built-in functions and construction of algorithm to obtain solutions to initial value problems. Scientific problems arises in stem fields as Biology, Physics, Chemistry ,and Engineering [1], Ecology with respect to inverse problems [2]. They also arise in non-stem fields as humanities, virtual art, designs and films [3], [4]. Differential equations recently have been understood, that its application model life realities and numerical methods are used to investigate ODEs. Some problems have been modeled to explore dynamics of music, literature, poetry, prey-predator models, decay radiation, and numerical methods [5], have been used to understand its dynamics in terms of approximation. Some mathematical real life situations modeled by system of ODEs do not have exact solutions, hence, approximation and/or numerical techniques must be used. In this paper, we consider second-order ODEs, convert them to a system of ODEs and compare the exact and numerical solution. This exploration is investigated through evaluation using Mathematica®[6], [7], [8]. Euler's and Runge-Kutta [9], [5], [10] algorithm are implemented with its built-in function. Other researchers have implemented several methods to solve initial value problems and the analysis of the accuracy in areas such as Epidemiology [11], stability and efficiency to system of ODEs [12] but not with the algorithm implemented in this paper and the use of time inference obtained with Mathematica®. The Euler's method has large errors which is illustrated using Taylor's series expansion. Taylor's series converges within a small range and as a result, if the step size is not relatively small, it diverges. From Taylor's expansion, the first two terms represent the Euler's method. Through this, its imperative enough to know if h is not small enough, the method is not accurate and will not be a good use for practical implementation. The Runge-Kutta method of order four gives a better approximation than the Euler's method but its complicated in computing. However, it converges faster to the exact solution. This procedure explore the comparative analysis between the exact and numerical solution of the ODEs [13],[14]. The algorithm works robustly for obtaining solution to system of first order ODEs [15] and comparing the solution by computation and investigating the absolute timing it takes for each algorithm to compute, hence the algorithm can be applied to biological models. This algorithm will be applied to investigate nonlinear system of ODEs for further studies.

In this paper, we are more concerned about the accuracy of the algorithm and its reliability after comparing the exact solution, Euler's method and Runge-Kutta method on the system of first-order linear ODEs.

II. Methodology

In this section, we will consider a second-order ODEs by recasting it to system of first-order ODEs. In considering the Euler's method and Runge-Kutta method of order four, this methods investigated by coding an algorithm using the CAS, Mathematica and its built-in function. The approximate solution of both methods would be obtained and compared by the result of algorithm and the built-in function of Mathematica®. For Equation (1) and Equation (2), the explicit formula for the solution can't be obtained for an initial value problem of such form but can approximate the solution using the numerical methods which is based on the tangent line approximation. In this paper, the Euler's method would be used to approximate the system of ODEs with initial conditions. This technique will investigate numerically with the aid of the Mathematica® software.

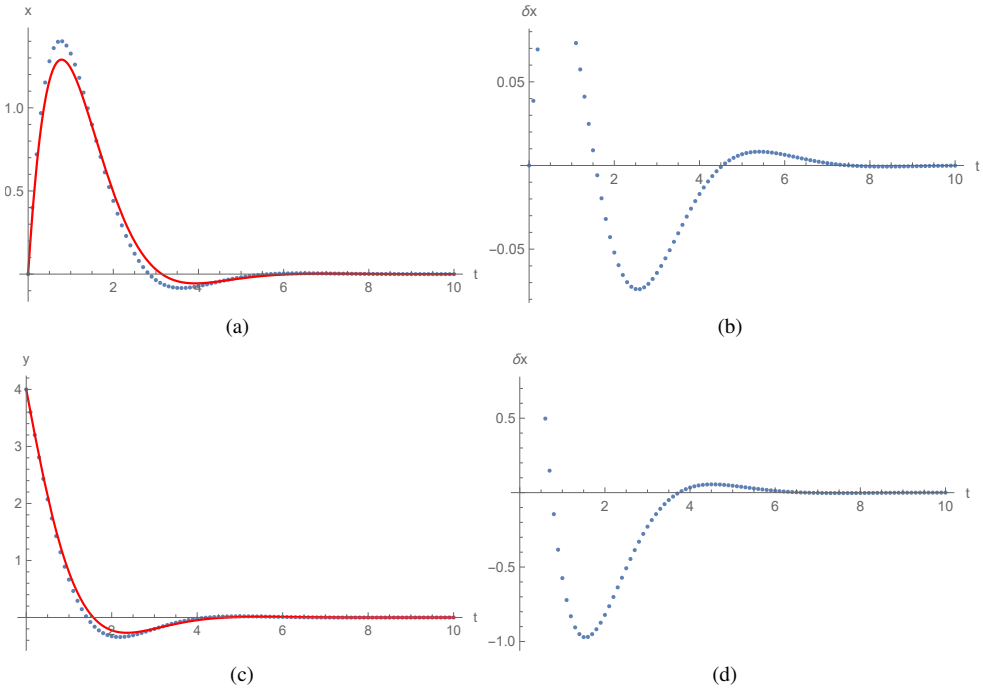


Fig. 1. (a) Graphical Representation: Exact and Numerical Solution of $x(t)$ (b) Error Plot of $x(t)$ (c) Graphical Representation: Exact and Numerical Solution of $y(t)$ (d) Error Plot of $y(t)$, using the Euler's Algorithm for Example 1.

III. Euler's Method

Consider the initial value problem in

$$\frac{dx}{dt} = f(t, x, y), \tag{1}$$

$$\frac{dy}{dt} = f(t, x, y), \tag{2}$$

with initial condition $x(0) = 0$, and $y(0) = 0$ valid for $a \leq t \leq b$
Let

$$m_k = f(x_k, y_k), \tag{3}$$

$$n_k = g(x_k, y_k), \tag{4}$$

Such that

$$x_{k+1} = x_k + m_k \Delta t \Rightarrow x_{k+1} = x_k + f(x_k, y_k)\Delta t, \tag{5}$$

$$y_{k+1} = y_k + m_k \Delta t \Rightarrow y_{k+1} = y_k + g(x_k, y_k)\Delta t. \tag{6}$$

where $\Delta t = \dfrac{t_n - t_0}{n}$ is known as the step size. The rule of the thumb to the above algorithm is the smaller the step size (over the increased number of length interval), the more accurate the approximate solution. However, even when extremely small step sizes are used, over a large number of steps the error starts to accumulate and the estimate diverges from the actual functional value which denotes its limitation and the requires more time to compute the approximate solution using the CAS.

IV. Runge-Kutta Method of Order 4(RK4)

The Runge-Kutta method are an important family of methods for approximate solutions of ODEs which were developed by mathematics duo C Runge (1856-1927) and M.W Kutta (1867-1944). In this paper, the Runge-Kutta method (RK4) was considered for a initial value problem (IVP), for a system of the first order ODEs, such that

$$x'(t) = f(t, x) \text{ where } x = x(t) = [(x_1(t), x_2(t), \ldots, x_n(t)]^T, \quad f \in [a, b] \times \mathbb{R}^n \to \mathbb{R}^n, \tag{7}$$

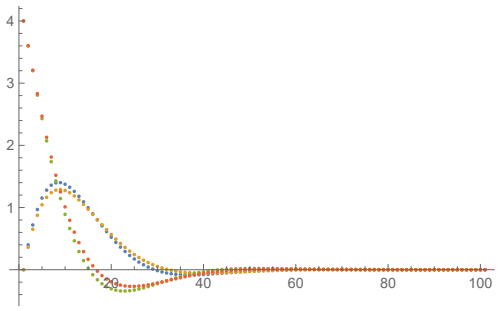
$$y'(t) = f(t, y) \text{ where } y = y(t) = [(y_1(t), y_2(t), \ldots, y_n(t)]^T, \quad f \in [a, b] \times \mathbb{R}^n \to \mathbb{R}^n, \tag{8}$$

with initial condition $x(0) = x_0$. To implement the RK4 method of solution of $x(t)$ and $y(t)$ of the IVP over the time interval $t \in [a, b]$. The time interval was subdivided into n equal intervals and we selected the step size such that $h = \dfrac{(b-a)}{n}$ where h is called the step size.
Consider a problem for the implementation of RK4

$$x' = f(t, x, y); x(t_0) = x_0, \tag{9}$$

$$y' = f(t, x, y); y(t_0) = y_0. \tag{10}$$

To obtain the solution through investigation of built-in algorithm of Mathematica® 12.2.0, it imperative to note that $k's$ obtains the solution for $x(t)$ and $l's$ for solutions to $y(t)$.



(a)

Fig. 2. (a) Graph of Exact and Numerical Solution, Euler's Algorithm for Example 1.

V. Numerical Examples

A. Using Euler's Algorithm

Consider the system of ODE

$$x' = -x + y, \tag{11}$$
$$y' = -x - y, \tag{12}$$
$$x(0) = 0, y(0) = 4, h = 0.1.$$

The exact solution was obtained as $x(t) = 4e^{-t}\sin(t)$, $y(t) = 4e^{-t}\cos(t)$. Results generated using the Euler's algorithm were represented on the Table I
Let $x1 = 4e^{-t}\sin(t)$,$y1 = 4e^{-t}\cos(t)$ be the exact solution, and $X_n \Rightarrow x' = -z + y$, $Y_n \Rightarrow y' = -x - y$ be the numerical solution.

TABLE I. Results Obtained using Euler's Algorithm

n	$x1$	X_n	$y1$	Y_n
0	0	0	4	4
1	0.361332	0.4	3.60127	3.6
2	0.650627	0.72	3.20964	3.2
3	0.875707	0.968	2.83092	2.808
4	1.04414	1.152	2.46962	2.4304
5	1.16315	1.27984	2.12912	2.07216
6	1.23953	1.359072	1.81182	1.7369
7	1.27964	1.3968608	1.51924	1.42736
8	1.28932	1.3999104	1.2522	1.14494
9	1.27391	1.374412864	1.01091	0.89045
10	1.23824	1.3260166272	0.795064	0.663964

The graphs in Figure 1 represents the exact solution, numerical solution of $x(t)$ & $y(t)$ and the error plots.

Figure 1 (b) and (d) shows that the order of errors observed in the graphs for Figure 1 (a) and (c) are 10^{-2} and 10^{0} respectively. For example 1, we observed that the solutions for $x(t)$ yielded better approximations than the solutions for $y(t)$. Although, from a graphical perspective, Figure 1 (c) shows better agreement (between the exact and numerical solutions) as compared to Figure 1

The graph in Figure 2 represents the exact solution and numerical solution of $x(t)$ & $y(t)$

B. Using Runge-Kutta Algorithm

Now, let us consider the system of ODEs from equations in Equation (11)-Equation (12). The exact solution was obtained as $x(t) = 4e^{-t}\sin(t)$, $y(t) = 4e^{-t}\cos(t)$. The results generated by the Runge-Kutta algorithm are represented in the Table II.
Let $L = 4e^{-t}\sin(t), P = 4e^{-t}\cos(t)$ be the exact solution, and $X_n \Rightarrow x' = -x + y$, $Y_n \Rightarrow y' = -x - y$ be the numerical solution

TABLE II. Table II Represent the Result Obtained using Runge-Kutta Algorithm.

n	L	X_n	P	Y_n
0	0	0	4	4
1	0.361332	0.318733	3.60127	3.6
2	0.650627	0.579032	3.20964	3.21121
3	0.875707	0.78666	2.83092	2.8378
4	1.04414	0.94723	2.46962	2.48297
5	1.16315	1.06615	2.12912	2.14912
6	1.23953	1.14855	1.81182	1.83792
7	1.27964	1.19929	1.51924	1.55039
8	1.28932	1.22289	1.2522	1.28704
9	1.27391	1.22354	1.01091	1.04789
10	1.23824	1.20507	0.795064	0.832592

The graph in Figure 3 represents the exact solution and numerical solution of $x(t)$ & $y(t)$ and error plots.

The same phenomenon we observed under the Euler's algorithm section is also observed in this section. Figure 3 (b) and (d) are error plots having order of magnitudes 10^{-1} and 10^{-1} respectively. However, Figure 3 (a) and (c) shows that the agreement between the exact and numerical solutions for $y(t)$ is better compared to the agreements between the exact and numerical solutions for $x(t)$.

The graph in Figure 4 represents the exact solution and numerical solution of $x(t)$

C. Example 2

Consider the IVP of the form

$$x'' + 2x' + x = 0, \tag{13}$$
$$x(0) = 1, x'(0) = 0.$$

In what follows, we reduce Equation (13) to a system of first-order ODEs

$$x' = y, \tag{14}$$
$$y' = -x - 2y \tag{15}$$
$$x(0) = 1, y(0) = 0$$

The exact solution of the system was obtained as

$$x(t) = e^{-t}(1 + t), \tag{16}$$
$$y(t) = -e^{-t} \cdot t, \tag{17}$$
$$x(0) = 1, y(0) = 0,$$

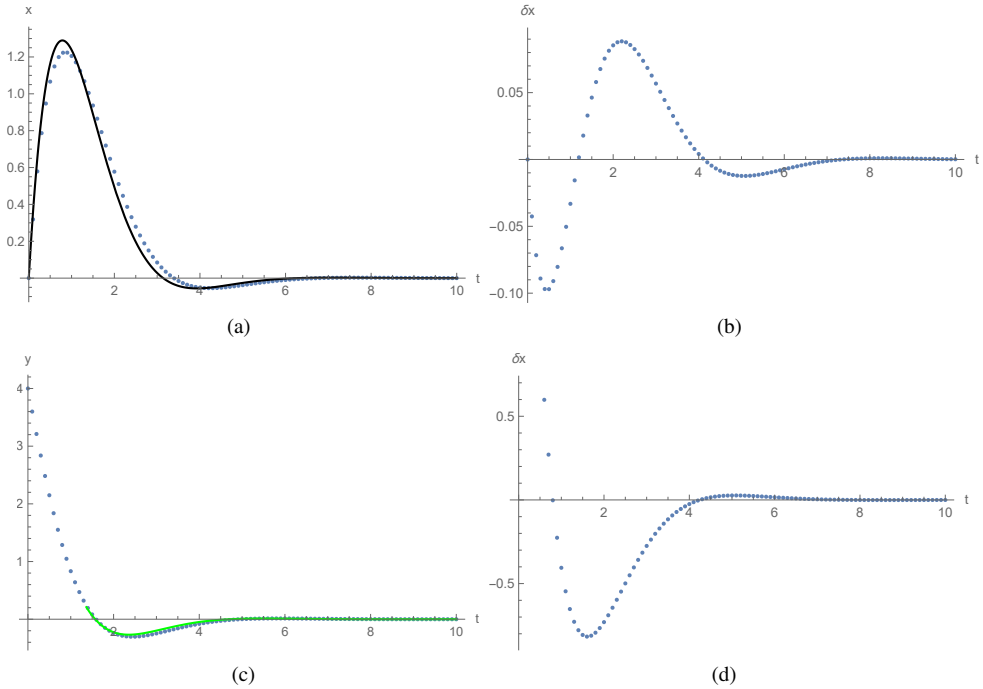
$$\text{Analytical solution} y(t) = e^{-t}(1 + t). \tag{18}$$

The result generated using Euler's algorithm by comparative analysis is presented in Table III.
We define the following

$$\left.\begin{array}{l} x2 \to e^{-t}(1 + t), \\ y2 \to -e^{-t} \cdot t. \end{array}\right\} = \text{Exact solution} \tag{19}$$

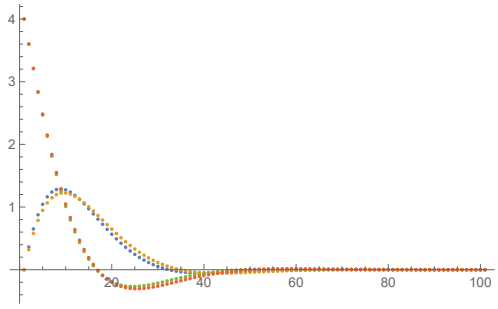
and

$$\left.\begin{array}{l} X_n \to y, \\ Y_n \to -x - 2y. \end{array}\right\} = \text{Numerical solution} \tag{20}$$



(a) (b) (c) (d)

Fig. 3. (a) Graphical Representation: Exact and Numerical Solution of $x(t)$ (b) Error Plot of $x(t)$ (c) Graphical Representation: Exact and Numerical Solution of $y(t)$ (d) Error Plot of $y(t)$, using the Runge-Kutta Algorithm for Example 1.



(a)

Fig. 4. (a) Graph of Exact and Numerical Solution, Runge-Kutta Algorithm for Example 1.

TABLE III. RESULTS OBTAINED USING EULER'S ALGORITHM

n	$x2$	X_n	$y2$	Y_n
0	1	1	0	0
1	0.995321	1	-0.0904837	0
2	0.982477	0.99	-0.163746	-0.1
3	0.963064	0.972	-0.222245	-0.18
4	0.938448	0.9477	-0.268128	-0.243
5	0.909796	0.885735	-0.303265	-0.2916
6	0.878099	0.850306	-0.329287	-0.32805
7	0.844195	0.813105	-0.34761	-0.354294
8	0.808792	0.774841	-0.359463	-0.372009
9	0.772482	0.736099	-0.365913	-0.382638
10	0.735759	0.697357	-0.367879	-0.38742

The graph in Figure 5 represents the exact solution, numerical solution of $x(t)$ & $y(t)$ and error plots.

For exmaple 2, we can observe a stark contrast to example 1. Figure 5 (a) and (c) shows that the agreement between the exact and numerical solutions for $x(t)$ is better compared to the agreements between the exact and numerical solutions for $y(t)$. The error plots for the graph in Figure 5 (b) shows that the approximation is quite good, as the error has order of magnitude 10^{-2}. The error plots for the solutions of $y(t)$ is not quite good (it has dimensions of order 10^0).

The graph in Figure 6 represents the exact solution and numerical solution of $x(t)$

D. Using Runge-Kutta Algorithm

Consider the system of ODEs in Equation (14)-Equation (15), result generated by RK4. The numerical solution is presented in Equation (20). The comparative Runge-Kutta algorithm is presented in the Table IV

TABLE IV. RESULTS OBTAINED USING RUNGE-KUTTA ALGORITHM

n	L	X_n	P	Y_n
0	1	1	0	0
1	0.995321	0.996738	-0.0904837	-0.0905542
2	0.982477	0.984877	-0.163746	-0.163981
3	0.963064	0.966075	-0.222245	-0.222686
4	0.938448	0.941753	-0.268128	-0.268776
5	0.909796	0.913129	-0.303265	-0.304097
6	0.878099	0.88124	-0.329287	-0.330261
7	0.844195	0.846968	-0.34761	-0.348673
8	0.808792	0.811058	-0.359463	-0.36056
9	0.772482	0.774135	-0.365913	-0.366986
10	0.735759	0.736721	-0.367879	-0.368873

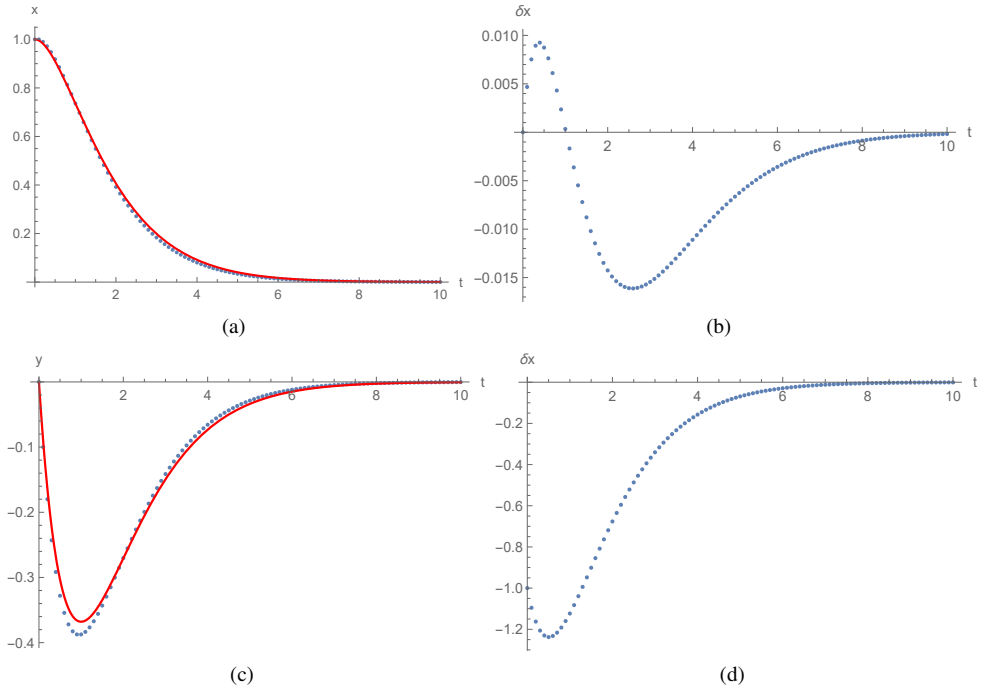
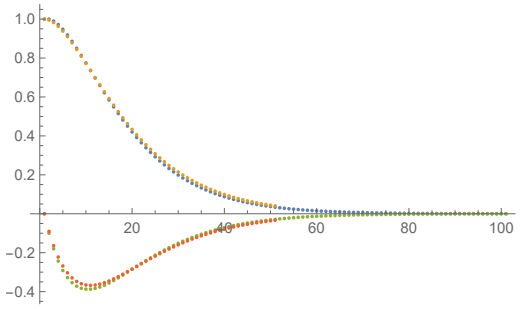


Fig. 5. (a) Graphical Representation: Exact and Numerical Solution of $x(t)$ (b) Error Plot of $x(t)$ (c) Graphical Representation: Exact and Numerical Solution of $y(t)$ (d) Error Plot of $y(t)$ using the Euler's Algorithm for Example 2.



(a)

Fig. 6. (a) Graph of Exact and Numerical Solution, Euler's Algorithm for Example 2.

The graph in Figure 7 represents the exact solution, numerical solution of $x(t)$ & y (t), and error plots.

For this section, the error plots for $x(t)$ has order of magnitude 10^{-3} (see Figure 7 (b)). The magnitude of order for the error plots of $y(t)$ still maintains the same value as it was in the section for Euler's algorithm (see Figure 7 (d)).

The graph in Figure 8 represents the exact solution and numerical solution of $x(t)$

E. Example 3

In what follows, we will consider the well-known model of a vibrating particle attached to a stationary wall through a spring

$$mx''(t) + dx'(t) + kx(t) = 0, \tag{21}$$

where m is the mass of the particle, d is the damping factor, and k is the stiffness constant of the spring. Equation (21) is rewritten as

$$x''(t) + 2\delta x'(t) + \omega^2 x(t) = 0, \tag{22}$$

where $2\delta = d/m$ and $\omega^2 = k/m$. Equation (22) is reduced to the system of first-order ODEs

$$x' = y \tag{23}$$

$$y' = -2\delta y - \omega^2 x \tag{24}$$

$$x(0) = 0.1, y(0) = 0, \delta = 0, \omega = 2\pi$$

From Equation (23)-Equation (24), the exact solution was obtained as

$$x(t) = (0.1 - 2.6963 \times 10^{-34i} \cos[6.28319t] + (3.15475 \times 10^{-17} + 8.22996 \times 10{-}34i \sin[6.28319t], \tag{25}$$

$$y(t) = (-0.628319 + 2.38256^{-18i})\,((0 + 0.i) + (1 + 0.i)\sin[6.28319t])\,, \tag{26}$$

$$x(0) = 1, y(0) = 0.$$

The results generated from the Euler's algorithm is presented in Table V by defining the following: $x3$ to be Equation (25) and $y3$ represented as Equation (26), while the numerical solution of Equation (23) and Equation (24) are represented as (X_n, Y_n) (which were obtained using the Euler's algorithm).

TABLE V. RESULTS OBTAINED USING EULER'S ALGORITHM.

n	$x3$	X_n	$y3$	Y_n
0	$0.1 - 2.6963 \times 10^{-34i}$	0.1	$0. + 0i$	0
1	$0.0998027 - 2.63931 \times 10^{-34i}$	0.1	$-0.0394524 + 1.49602 \times 10^{-19i}$	-0.0394784
2	$0.0992115 - 2.57189 \times 10^{-34i}$	0.0996052	$-0.0787492 + 2.98614 \times 10^{-19i}$	-0.0789568
3	$0.0982287 - 2.49433 \times 10^{-34i}$	0.0988156	$-0.117735 + 4.46447 \times 10^{-19i}$	-0.118279
4	$0.0968583 - 2.40692 \times 10^{-34i}$	0.0976329	$-0.156256 + 5.92518 \times 10^{-19i}$	-0.15729
5	$0.0951057 - 2.31002 \times 10^{-34i}$	0.09606	$-0.194161 + 7.36251 \times 10^{-19i}$	-0.195834
6	$0.0929776 - 2.20399 \times 10^{-34i}$	0.0941016	$-0.231299 + 8.77078 \times 10^{-19i}$	-0.233757
7	$0.0904827 - 2.08927 \times 10^{-34i}$	0.091764	$-0.267525 + 1.01444 \times 10^{-18i}$	-0.270907
8	$0.0876307 - 1.96631 \times 10^{-34}$	0.089055	$-0.302695 + 1.14781 \times 10^{-18i}$	-0.307134
9	$0.0844328 - 1.83558 \times 10^{-34i}$	0.0859836	$-0.33667 + 1.27664 \times 10^{-18i}$	-0.342291
10	$0.0809017 - 1.69761 \times 10^{-34i}$	0.0825607	$-0.369316 + 1.40043 \times 10^{-18i}$	-0.376236

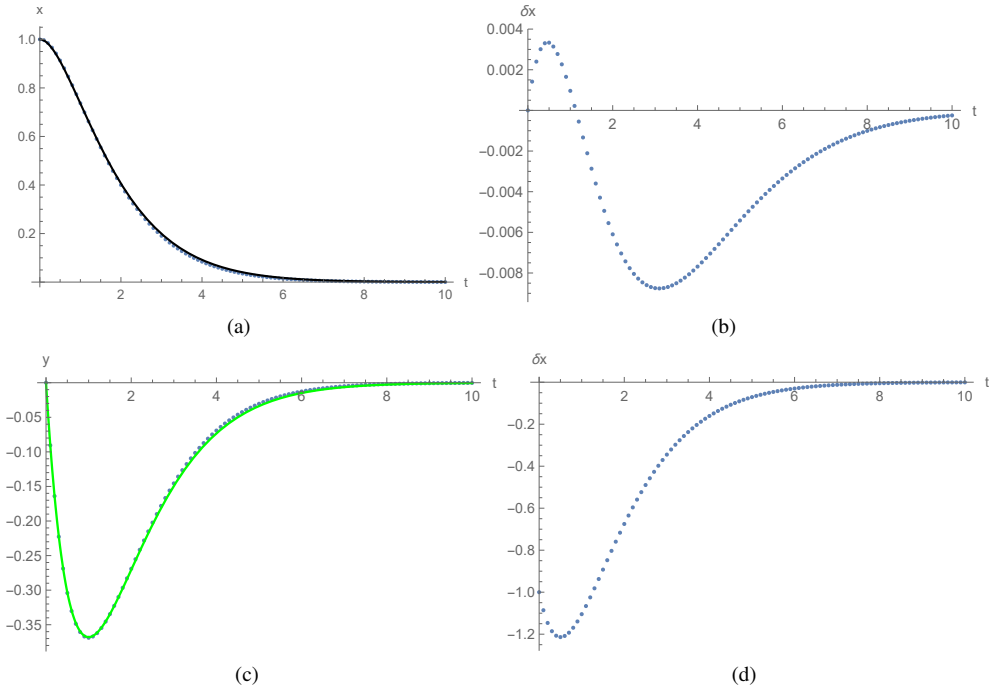
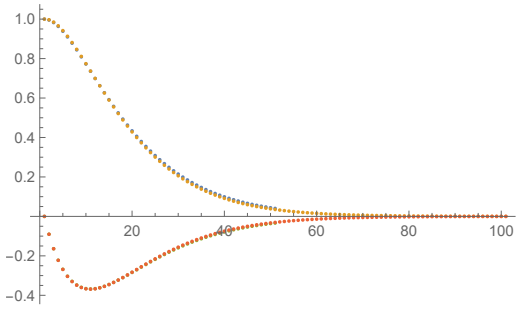


Fig. 7. (a) Graphical Representation: Exact and Numerical Solution of $x(t)$ (b) Error Plot of $x(t)$ (c) Graphical Representation: Exact and Numerical Solution of $y(t)$ (d) Error Plot of $y(t)$, using the Runge-Kutta Algorithm for Example 2.



(a)

Fig. 8. (a) Graph of Exact and Numerical Solution, Runge-Kutta Algorithm for Example 2.

The graph in Figure 9 represents the exact solution, numerical solution of $x(t)$ & $y(t)$ and error plots. The graph in Figure 11 represents the exact solution and numerical solution of $x(t)$

F. Using Runge-Kutta Algorithm

Now, let us consider the system of ODEs in Equation (23)-Equation (24), for the results generated using RK4 and its numerical solution. The comparative analysis of RK4 algorithm is presented in the Table VI.

The graph in Figure 10 represents the exact solution, numerical solution of $x(t)$ & $y(t)$, and error plots.

The graph in Figure 12 represents the exact solution and numerical solution of $x(t)$

TABLE VI. RESULTS OBTAINED USING RUNGE-KUTTA ALGORITHM

n	L	X_n	P	Y_n
0	$0.1 - 2.6963 \times 10^{-34}i$	0.1	0. + 0.i	0
1	$0.0998027 - 2.63931 \times 10^{-34}i$	0.1	$-0.0394524 + 1.49602 \times 10^{-19}i$	-0.0394784
2	$0.0992115 - 2.57189 \times 10^{-34}i$	0.0996052	$-0.0787492 + 2.98614 \times 10^{-19}i$	-0.0789568
3	$0.0982287 - 2.49433 \times 10^{-34}i$	0.0988156	$-0.117735 + 4.46447 \times 10^{-19}i$	-0.118279
4	$0.0968583 - 2.40692 \times 10^{-34}i$	0.0976329	$-0.156256 + 5.92518 \times 10^{-19}i$	-0.15729
5	$0.0951057 - 2.31002 \times 10^{-34}i$	0.09606	$-0.194161 + 7.36251 \times 10^{-19}i$	-0.195834
6	$0.0929776 - 2.20399 \times 10^{-34}i$	0.0941016	$-0.231299 + 8.77078 \times 10^{-19}i$	-0.233757
7	$0.0904827 - 2.08927 \times 10^{-34}i$	0.091764	$-0.267525 + 1.01444 \times 10^{-18}i$	-0.270907
8	$0.0876307 - 1.96631 \times 10^{-34}i$	0.089055	$-0.302695 + 1.14781 \times 10^{-18}i$	-0.307134
9	$0.0844328 - 1.83558 \times 10^{-34}i$	0.0859836	$-0.33667 + 1.27664 \times 10^{-18}i$	-0.342291
10	$0.0809017 - 1.69761 \times 10^{-34}i$	0.0825607	$-0.369316 + 1.40043 \times 10^{-18}i$	-0.376236

VI. TIME INFERENCE

This section details the time taken by Mathematica® to evaluate the formulated algorithms. An inference can be drawn that Euler's algorithm solves the system of ODEs in less time than the RK4. By computation the average time for each algorithm is represented in Table VII. As seen in Table VII, the Euler's algorithm computes faster although, it does not give an accurate approximation but the RK4 algorithm gives a better approximation despite having a lower computational speed compared to the Euler's algorithm.

TABLE VII. THE COMPUTATIONAL TIME TAKEN IN EVALUATING THE EULER'S AND RUNGE-KUTTA ALGORITHM

	Example 1		Example 2	
	Euler's	RK4	Euler's	RK4
Absolute Timing	0.000553	0.014443	0.001603	0.003870
Average Timing	0.000732	0.002925	0.000684	0.002868

VII. CONCLUSION

In this paper, we considered an algorithm which was simulated using Mathematica® 12.2.0 installed on a 512 SSD, 64 bit laptop. The built-in function of Mathematica® with

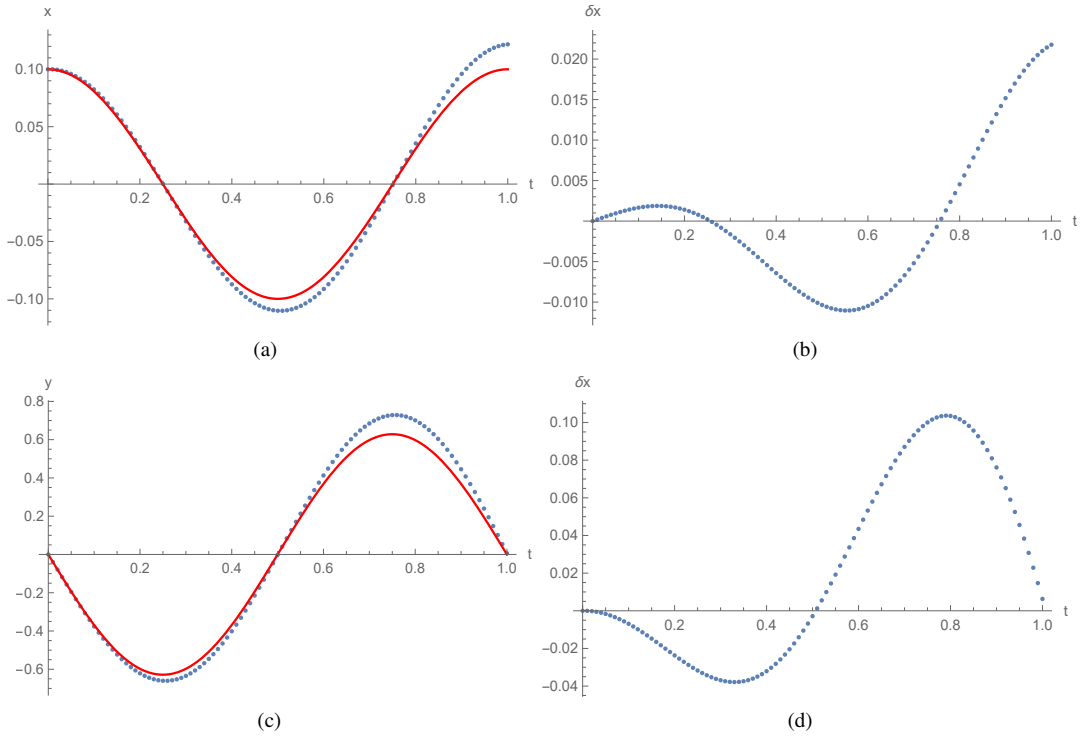


Fig. 9. (a) Graphical Representation: Exact and Numerical Solution of $x(t)$ (b) Error Plot of $x(t)$ (c) Graphical Representation: Exact and Numerical Solution of $y(t)$ (d) Error Plot of $y(t)$ using the Euler's Algorithm in Example 3.

absolute timing was used to generate a better approximation in the three examples considered in preceding sections. The accuracy of this algorithm depends on the system of ODEs under consideration. The algorithm was built to simulate system of ODEs by converting second-order ODE to a system of first-order IVP, with graphical representation of exact and numerical solution. The error plot with the same step size $h = 0.1$ was also graphically illustrated. Considering the parametric plot of example 1, the plot shows that the solution presented by the both algorithm, that RK4 algorithm gives a better approximation than the Euler's algorithm. This was also the case in example 2, the Runge-Kutta algorithm showed that the solution obtained from the exact and numerical solution by comparative analysis obtained accurate result and the same goes for example 3. It has been observed by the results shown in Figure 13, through the parametric plot that the algorithm works well. Table VII shows how the Euler's algorithm computes faster but does not give accurate approximations, like that of RK4 which computes slower but gives better approximation. In future work, we intend to harness the algorithm to solve higher orders of ODE that can be recasted to system of ODEs. We also aim to adopt the algorithms to solve other differential equations such as Lotka-Volterra model, and stiff equations, and investigate the algorithm time taken in execution, using Mathematica®. By investigation, it is shown in Figure 9 that the parametric plot of the system of ODEs using the (Euler's and RK4) algorithms concludes that RK4 algorithm works well and gives better approximation. In Equation (25) and Equation (26), we observed the appearance of complex conjugate values in the exact solution. This was due to the low value of damping ($\delta = 0.01$) used in the equation of motion, describing the particle attached to a stationary wall. This explanation also covers for the complex values observed in Table V and Table VI. The numerical values in the aforementioned tables showed close approximations, despite the complex part. The values of the complex parts are also negligible as they are really small (having dimensions of $10^{-34}, 10^{-19}$ and 10^{-18}). Comparing Figure 9 (b) and (d) against Figure 10 (b) and (d), we can observe that the Runge-Kutta algorithm also gave a better approximation to the Euler algorithm. The errors for the Runge-Kutta algorithm was in order 5×10^{-2} while Euler algorithm was in order 1×10^{-1}.

ACKNOWLEDGMENT

The authors wish to thank Tshwane University of Technology for their support and the Department of Higher Education and Training, South Africa.

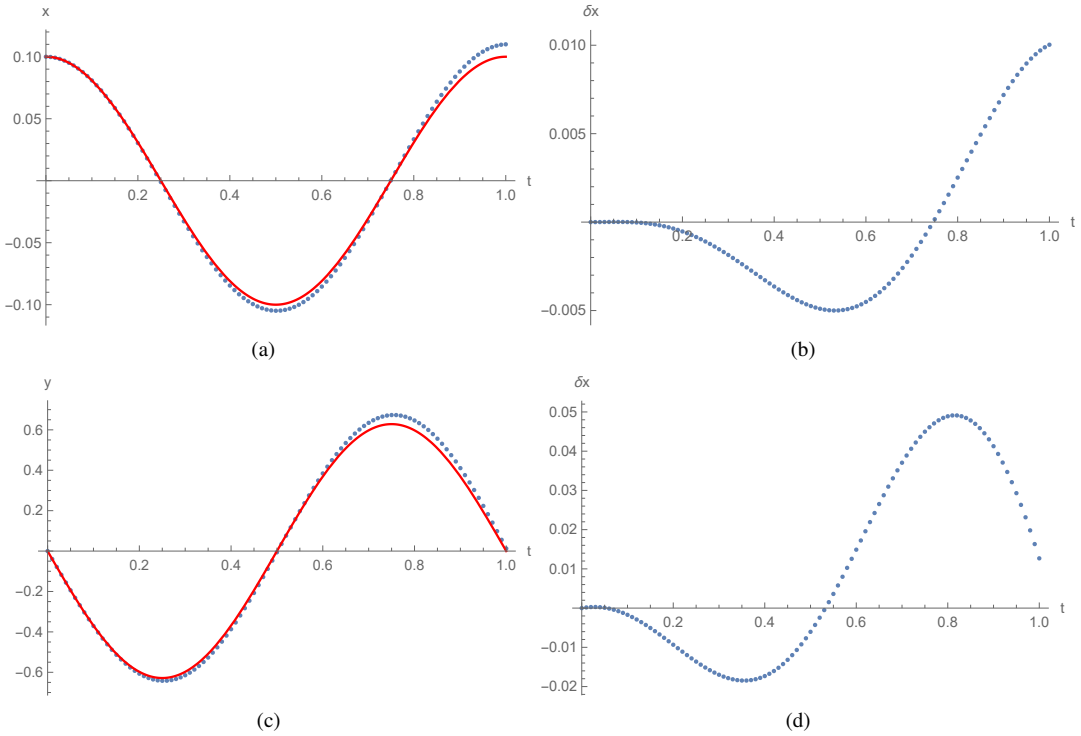


Fig. 10. (a) Graphical Representation: Exact and Numerical Solution of $x(t)$ (b) Error Plot of $x(t)$ (c) Graphical Representation: Exact and Numerical Solution of $y(t)$ (d) Error Plot of $y(t)$ using the Runge-Kutta Algorithm in Example 3.

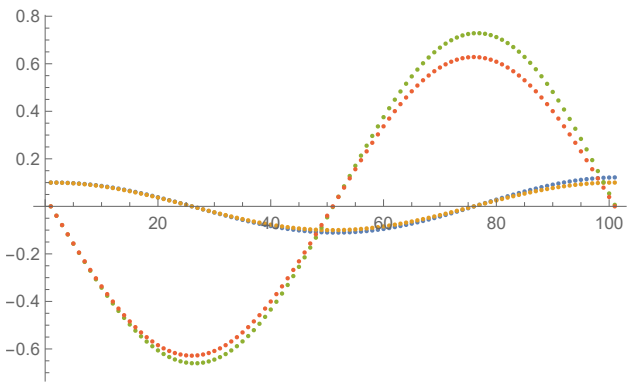


Fig. 11. (a) Graph of Exact and Numerical Solution, Euler's Algorithm in Example 3.

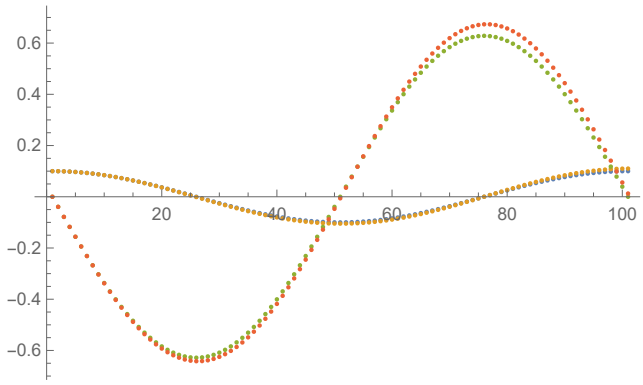


Fig. 12. (a) Graph of Exact and Numerical Solution, Runge-Kutta Algorithm in Example 3.

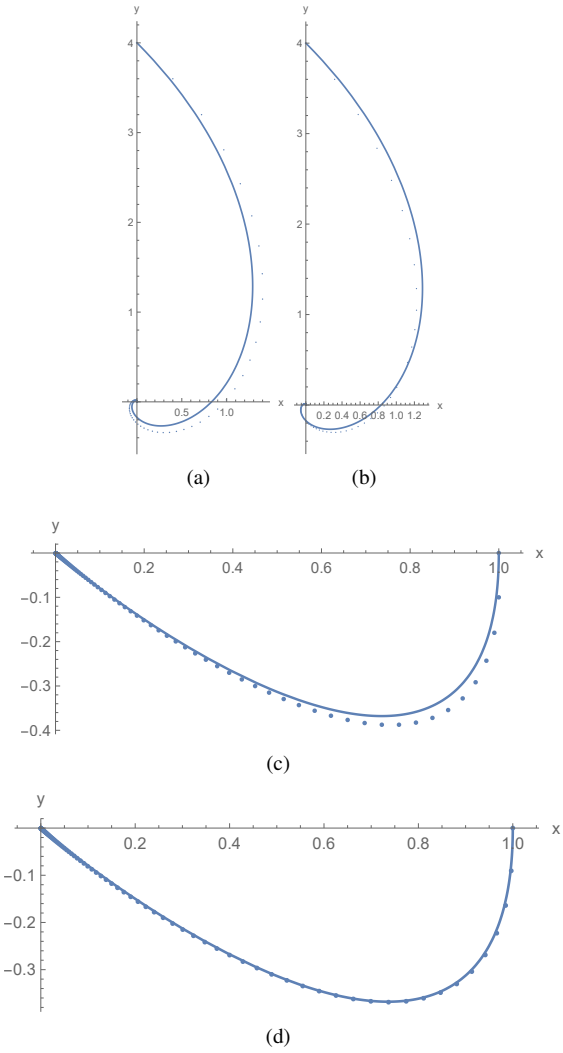


Fig. 13. [a,b] Parametric plot: Euler's and RK4 algorithm for Example 1 [c,d] Parametric plot: Euler's and RK4 algorithm for Example 2

REFERENCES

[1] F. C. Hoppensteadt and C. S. Peskin,. Mathematics in Medicine and the Life Sciences. *Springer-Verlag, New-York,*, 1992.

[2] AA. Adeniji, and MY. Shatalov, and I. Fedotov, and AC. Mkolesia, Introduction to numerical methods. *Discontinuity Nonlinearity and Complexity*, 10(3), 523-534, 2021

[3] L. Koss. Visual arts, design, and differential equations. *Journal of Mathematics and the Arts*, 11(3): 129–158, 2017.

[4] W. Brian. 2015-Koss-Differential Equations In Art and Film. *SIMIODE*.

[5] J. RChasnov. Introduction to numerical methods. *The Hong Kong University of Science and Technology*, 60, 2012.

[6] Ü. Göktaş, D. Kapadia. Methods in Mathematica for solving ordinary differential equations. *Mathematical and Computational Applications*, 16(4): 784–796, 2011.

[7] Abell, LL. Martha and Braselton, P. James. Differential equations with Mathematica. *Academic Press*, 2016.

[8] M. Sofroniou, and R. Knapp. Wolfram Mathematica Tutorial Collection-Advanced Numerical Differential Equation Solving in Mathematica. Wolfram Research. *Inc*, 2008.

[9] DF. Griffiths, DJ. Higham. Euler's method, Numerical Methods for Ordinary Differential Equations. *Springer, London*, 19-31, 2010.

[10] CR. Nathan, W. Dorathy Euler's Method for Systems of Differential Equations. *Applications of Calculus to Biology and Medicine*, 117-122, 2017.

[11] B. Yong The comparison of fourth order Runge-Kutta and homotopy analysis method for solving three basic epidemic models. *Journal of Physics: Conference Series*, 1317(1), 12-20, 2019.

[12] AT. Kolar Comparison of numerical methods for solving a system of ordinary differential equations: accuracy, stability and efficiency. *Mälardalen University, School of Education, Culture and Communication.*, 65, 2020.

[13] Blanchard, Paul and Devaney, Robert L and Glen, R Hall, and Jong-Eao Lee. *Differential Equations: A Contemporary Approach, Thomson Learning*, 2007.

[14] N. Shawagfeh, D. Kaya. Comparing numerical methods for the solutions of systems of ordinary differential equations. *Applied mathematics letters*, 17(3):323-8, 2004.

[15] P, B Conrad. Ordinary Differential Equations: A Systems Approach. , 2010.