frontiers | Frontiers in Applied Mathematics and Statistics

TYPE Original Research
PUBLISHED 22 February 2024
DOI 10.3389/fams.2024.1360628

OPEN ACCESS

EDITED BY
Andrea Franceschini,
University of Padua, Italy

REVIEWED BY
Pengyu Chen,
Northwest Normal University, China
Waleed Mohamed Abd-Elhameed,
Jeddah University, Saudi Arabia

*CORRESPONDENCE
Habtamu Mekonnen
✉ habtamumekonnen2012@gmail.com

RECEIVED 23 December 2023
ACCEPTED 06 February 2024
PUBLISHED 22 February 2024

CITATION
Workineh Y, Mekonnen H and Belew B (2024) Numerical methods for solving second-order initial value problems of ordinary differential equations with Euler and Runge-Kutta fourth-order methods.
Front. Appl. Math. Stat. 10:1360628.
doi: 10.3389/fams.2024.1360628

Numerical methods for solving second-order initial value problems of ordinary differential equations with Euler and Runge-Kutta fourth-order methods

Yenesew Workineh, Habtamu Mekonnen* and Basaznew Belew

Department of Mathematics, College of Natural and Computational Sciences, Mekdela Amba University, Tulu Awuliya, Ethiopia

This paper presents two standard numerical methods for solving second order initial value problems for ordinary differential equations (ODEs). The Euler and the Runge-Kutta fourth-order methods are applied without any discretization or restrictive assumptions for solving ODEs. The numerical solutions obtained by the two methods are in good agreement with the exact solutions. The convergence and error analysis which are discussed demonstrate the effectiveness of the methods. The results obtained from the two numerical methods show that the RK4 method is appropriate, consistent, convergent, quite stable, and more accurate than the Euler's method.

KEYWORDS

ordinary differential equation (ODE), second order initial value problem (IVP), Euler method, fourth order Runge-Kutta method, error analysis

1 Introduction

Ordinary differential equations (ODEs) are mathematical equations that describe the relationship between a function and its derivatives. They are widely used in various fields of science, engineering and mathematics to model physical, biological, and dynamical systems [1–5]. Solving ODEs analytically can often be challenging or even impossible for complex differential equations [6–8]. Therefore, numerical methods play a crucial role in approximating solutions to these equations. The initial value problem (IVP) is a specific type of ODE problem where the values of the unknown function and its derivative(s) are specified at a given initial point. Many researchers developed different methods for solving ordinary differential equations (ODEs) with the initial value problem. Many authors have attempted to solve initial value problems (IVPs) to obtain high accuracy rapidly by using different methods such as the spactral method, the, the Euler's method and Runge Kutta fourth (RK4) order method and some other methods.

Jhnson and Lee explored the application of spectral methods to solve second-order IVP ODEs [9]. The authors employed Legendre polynomials as basis functions and developed spectral schemes to approximate the solution with high accuracy and rapid convergence rates. The advantages of spectral methods

in terms of solution accuracy and numerical examples to validate their approach are discussed. Spectral methods are based on representing the solution as a series of basis functions, such as Legendre polynomials, Fourier series or Chebyshev polynomials [6, 10, 11]. By discretizing the problem on a set of collocation points, the unknown function can be approximated by a truncated Legendre polynomial series. These methods provide exponential convergence and are particularly effective for smooth or periodic solutions. It is worth noting that spectral methods, such as the Fourier or Chebyshev methods, have their own advantages for solving ODEs. They are typically more accurate, rapid convergence rates and efficient for problems with only smooth and periodic solutions. Euler's method is straightforward to implement and understand, making it a popular choice for beginners in numerical analysis [5, 11]. It involves only simple arithmetic operations. Euler's method requires fewer computations compared to more complex methods, such as RK4. Hence, it can be computationally more efficient for simple and low-dimensional problems. Since Euler's method only uses information from the previous step to approximate the next step, it can provide a quick estimate of the solution, especially when speed is prioritized over accuracy. RK4 is a higher-order method, which means it provides more accurate approximations compared to Euler's method for a given step size [6, 11, 12]. It achieves this by using multiple evaluations of the derivative at different points within the step, resulting in a smaller truncation error. RK4 is a versatile method that can handle various types of problem, including those with stiff equations and irregular behavior [9]. It is widely used and considered to be one of the most accurate and reliable numerical methods for solving ODEs. Compared to Euler's method, RK4 exhibits better stability properties, making it more suitable for problems that require higher accuracy over longer integration intervals [6, 12]. While there are more advanced and sophisticated numerical methods available, Euler's method and the Runge-Kutta fourth-order method strike a balance between simplicity, efficiency, accuracy, and stability when solving second-order IVPs of ODEs. Euler's method is a basic, easy-to-implement method, while the Runge-Kutta fourth-order method provides higher accuracy, numerical stability, and better control over the solution's accuracy. The choice between these methods depends on the specific problem characteristics, accuracy requirements, available computational resources, and desired trade-offs between accuracy and computational efficiency. Euler's method and Runge-Kutta methods provide a practical and reliable alternative for general second order IVPs of ODEs but spectral methods are accurate and efficient for problems with smooth and periodic solutions. Although extensive research has been conducted on numerical methods for solving first-order IVPs of ODEs, there is a noticeable gap in the literature regarding specialized methods for solving second-order IVPs. Second-order IVPs are more complex due to their involving second derivatives, and applying first-order methods may result in inaccurate solutions or numerical instability. The lack of comprehensive studies on numerical methods tailored for second-order IVPs highlights the need to bridge this gap in the literature. It is crucial to develop and evaluate specialized numerical methods that can reliably and efficiently solve second-order IVPs of ODEs. The aim of this study is to investigate and compare the performance of two widely used numerical methods, namely the Euler method and the Runge-Kutta fourth-order method, in approximating solutions to second-order IVP of ODEs. The study of numerical methods for solving second-order IVP of ODEs with Euler's method and the Runge-Kutta fourth-order method holds significant importance in the field of numerical analysis and scientific computing. Understanding the strengths and limitations of these methods can help researchers, engineers, and scientists select the most appropriate approach for solving ODEs based on specific requirements such as accuracy, efficiency and stability. This knowledge can also aid in optimizing computational algorithms to solve ODEs and providing reliable solutions for real-world problems. In this paper we apply Eulers and fourth order Runge-Kutta method for solving initial value problem of second order ordinary differential equation. A more robust and intricate numerical technique is the Runge-Kutta fourth-order methods. The Runge-Kutta fourth-order (RK4) method generally exhibits better convergence properties compared to Euler's method when solving second-order initial value problems (IVPs) of ordinary differential equations (ODEs). We take an example of a second-order ordinary differential equation to verify our proposed formulation.

2 Problem formulation

In this section, we consider two numerical methods to find approximate solutions of the initial value problem (IVP) of the second-order ordinary differential equation having the following form [9].

$$y'' = f\left(x,\, y\left(x\right),\, y'\left(x\right)\right), y\left(d\right) = \alpha, y'\left(d\right) = \beta, d \le x \le e \quad (1)$$

Where $y'' = \frac{d^2y}{dx^2}$, $y' = \frac{dy}{dx}$ and $f\left(x,\, y\left(x\right),\, y'\left(x\right)\right)$ is the given function and $y(x)$ is the solution of Equation (1). To solve the second-order IVP of ODE by using the Euler and Runge-Kutta fourth-order methods, the second-order initial value problems of ODE can be transformed into a system of first-order initial value problems, which allows the use of standard numerical methods that are widely employed. This approach is commonly known as the first-order system approach and it is indeed not new. In the context of the current study focusing on Euler's method and the Runge-Kutta fourth order (RK4) method, the novelty lies in the analysis and comparison of these specific numerical methods for solving the first-order system derived from the original second-order initial value problem. Although the idea of transforming a second-order initial value problem into a system of first-order problems is not innovative, the analysis and comparison of specific numerical methods applied to the derived first-order system can provide new insights and understanding of their effectiveness, accuracy, stability, convergence properties, and computational efficiency.

We can transform Equation (1) into a system of two first-order ODEs that are grouped as

$$\frac{dy}{dx} = z = f(x,\, y,\, z), \quad (2)$$

$$\frac{dz}{dx} = z' = f(x,\, y,\, z) \quad y\left(d\right) = \alpha,\, y'\left(d\right) = \beta,\ d \le x \le e.$$

TABLE 1 Comparison between Runge–Kutta fourth order and Euler method with exact solution for step size $h = 0.1$.

	Euler method	RK4_method	Error Euler	Error_RK4	Exact solution
x_n	$y(x_n)$	$y(x_n)$	e_r	e_r	y_n
0	−1	−1	0	0	−1
0.1	−1	−0.988941666666667	0.0110609220088744	2.58867554114861e-06	−0.988939077991126
0.2	−0.98	−0.950987181701389	0.0290191813209306	6.36302231959984e-06	−0.950980818679069
0.3	−0.934	−0.877610537781075	0.0564011852385027	1.17230195781914e-05	−0.877598814761497
0.4	−0.8546	−0.75812765438281	0.0964915332099271	1.91875927370022e-05	−0.758108466790073
0.5	−0.7327	−0.579190140587741	0.153539287058789	2.94276465303556e-05	−0.579160712941211
0.6	−0.557138	−0.324163985933377	0.233017321955529	4.33078889066074e-05	−0.324120678044471
0.7	−0.3142534	0.0276326121855938	0.341947951903721	6.19397181275128e-05	0.0276945519037213
0.8	0.0126393400000001	0.50186381938946	0.489311227410179	8.6748020719174e-05	0.501950567410179
0.9	0.44388497	1.13032168665261	0.686556272099047	0.000119555446434161	1.13044124209905
1	1.0042515022	1.95232975338913	0.948240939812559	0.000162688623432494	1.95249244201256

Thus, instead of solving Equation (1), we can solve Equation (2)

We note that, with $y' = f(x, y, z) = z$, Equation (2) represents the second-order initial value problem.

$$y'' = f(x, y, y') : y(x_0) = y_0, y'(x_0) = z_0$$

2.1 Euler method

The Euler method is a simple numerical technique for solving ordinary second-order differential equations. It approximates the solution by taking small steps along the curve defined by the differential equation. It is a basic explicit method for the numerical integration of an ordinary differential equations. Euler proposed his method for initial value problems (IVP) in 1768 [5, 9, 11]. It is the first numerical method to solve IVP and serves to illustrate the concepts involved in advanced methods. It is important to study this because error analysis is easier to understand.

Now let's consider a general second-order ODE with initial value problem (IVP):

$$y'' = f(x, y(x), y'(x)), y(d) = \alpha, y'(d) = \beta \quad (3)$$

To apply Euler's method, we first transform Equation (3) in to a system of two first order ODEs. Transform Equation (3) to systems of two first order ODE, let $y' = z$ and $z' = f(x, y, z)$, then Equation (3) becomes

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

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

With initial condition, $y(d) = \alpha, y'(d) = \beta$ and $z(x) = y'(x)$.

Now, we can apply the Euler method to this system of first-order ODEs. The Euler method updates the functions y and z at each step as follows:

$$y_{n+1} = y_n + hz_n \quad for\ n = 0, 1, 2, \ldots,$$

$$z_{n+1} = z_n + hf(x_n, y_n, z_n)$$

This is the general formula to solve the second-order ODE with IVP using the Euler method.

2.2 Runge-Kutta fourth order method

This method was devised by two German mathematicians, Runge about 1894 and was extended by Kutta a few years later [6, 11–13]. The Runge-Kutta method is the most popular because it is quite accurate, stable, and easy to programme. This method is distinguished by its order in the sense that it agrees with Taylor's series solution up to terms of h^r where r is the order of the method. It does not require a prior computational analysis of higher derivatives of $y(x)$ as in Taylor's series method. The Runge-Kutta fourth-order method (RK4) is widely used for solving second-order initial value problems (IVP) for an ordinary differential equation (ODE). Now let's consider a general second-order ODE with initial value problem (IVP):

$$y'' = f(x, y(x), y'(x)), y(d) = \alpha, y'(d) = \beta \quad (4)$$

To apply the Runge-Kutta method, we first transform Equation (4) into a system of two first-order ODEs. Transform equation (4) to systems of two first order ODE, let $y' = z$ and $z' = f(x, y, z)$, then Equation (4) becomes

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

10.3389/fams.2024.1360628

TABLE 2 Comparison between Runge–Kutta-fourth order and Euler's method with exact solution for step size $h = 0.05$.

	Euler method	RK4_method	Error Euler	Error_RK4	Exact solution
x_n	$y(x_n)$	$y(x_n)$	e_r	e_r	y_n
0	−1	−1	0	0	−1
0.05	−1	−0.997371354166667	0.00262872532359948	7.94902661471752e-08	−0.997371274676401
0.1	−0.995	−0.988939254257948	0.00606092200887443	1.76266822582427e-07	−0.988939077991126
0.15	−0.98425	−0.973809970980796	0.010440322119437	2.93100233306198e-07	−0.973809677880563
0.2	−0.9669125	−0.950981251831617	0.0159316813209306	4.33152547607563e-07	−0.950980818679069
0.25	−0.942053125	−0.919330162704315	0.0227235623246453	6.00028960517918e-07	−0.919329562675355
0.3	−0.90863028125	−0.877599612597494	0.0310314664885026	7.9783599693517e-07	−0.877598814761497
0.35	−0.8654837453125	−0.824383420963075	0.0411013555964619	1.03124703665358e-06	−0.824382389716038
0.4	−0.811322077578125	−0.758109772366138	0.0532136107880521	1.30557606525805e-06	−0.758108466790073
0.45	−0.744708740957031	−0.677022886684052	0.0676874811336436	1.62686066473849e-06	−0.677021259823388
0.5	−0.664046793454883	−0.579162714896582	0.084886080513672	2.00195537081171e-06	−0.579160712941211
0.55	−0.567562010122627	−0.462342450425013	0.10522199833427	2.43863665627675e-06	−0.462340011788357
0.6	−0.453284275323258	−0.324123623765422	0.129163597278788	2.94572095160817e-06	−0.324120678044471
0.65	−0.319027070253371	−0.161788523605838	0.157242079844823	3.53319728940704e-06	−0.161784990408548
0.7	−0.162364863046385	0.0276903395273841	0.190059414950106	4.21237633715932e-06	0.0276945519037213
0.75	0.0193918105929166	0.247684041054918	0.228297226519798	4.9960577966357e-06	0.247689037112715
0.8	0.229223809593345	0.501944668691785	0.272726757816834	5.89871839462575e-06	0.501950567410179
0.85	0.470433649390873	0.794646751152289	0.324220038484344	6.93672292828573e-06	0.794653687875218
0.9	0.746678846110064	1.1304331135379	0.383762395988983	8.1285611486237e-06	1.13044124209905
0.95	1.06200865409018	1.51446562853401	0.452466469557398	9.49511357029031e-06	1.5144751236475
1	1.42090453903676	1.95248138206287	0.531587902975799	1.10599496858654e-05	1.95249244201256

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

With initial condition, $y(d) = \alpha$, $y'(d) = \beta$ and $z(x) = y'(x)$.

Now, we can apply the Runge-Kutta fourth-order method to this system of first-order ODEs. The Runge-Kutta fourth order method updates the functions y and z at each step as follows:

The general formula for the Runge-Kutta approximation to solve systems of equations is given by

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

$$z_{n+1} = z_n + \frac{1}{6}(l_1 + 2l_2 + 2l_3 + l_4)$$

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

$$l_1 = hf(x_n, y_n, z_n)$$
$$k_2 = hf(x_n + \frac{1}{2}h, y_n + \frac{1}{2}k_1, z_n + \frac{1}{2}l_1)$$
$$l_2 = hf(x_n + \frac{1}{2}h, y_n + \frac{1}{2}k_1, z_n + \frac{1}{2}l_1)$$
$$k_3 = hf(x_n + \frac{1}{2}h, y_n + \frac{1}{2}k_2, z_n + \frac{1}{2}l_2)$$
$$l_3 = hf(x_n + \frac{1}{2}h, y_n + \frac{1}{2}k_1, z_n + \frac{1}{2}l_2)$$
$$k_4 = hf(x_n + h, y_n + k_3, z_n + l_3)$$
$$l_4 = hf(x_n + h, y_n + k_3, z_n + l_3)$$
$$for\ n = 0, 1, 2, 3, \ldots$$

3 Error analysis

Error analysis is an essential component when studying numerical methods for solving second-order Initial Value Problems (IVPs) of Ordinary Differential Equations (ODEs) using Euler's method and the Runge-Kutta fourth-order (RK4) method [9]. Error analysis allows us to quantify the accuracy of these methods and understand their convergence properties. In numerical methods, the truncation error and the global error are commonly

TABLE 3 Comparison between Runge–Kutta-fourth order and Eulers method with exact solution for step size $h = 0.025$.

	Euler method	RK4_method	Error Euler	Error_RK4	Exact solution
x_n	$y(x_n)$	$y(x_n)$	e_r	e_r	y_n
0	−1	−1	0	0	−1
0.025	−1	−0.999359147135417	0.000640855327166401	2.4625830263858e-09	−0.999359144672834
0.05	−0.99875	−0.99737127986257	0.00137872532359951	5.18616916078685e-09	−0.997371274676401
0.075	−0.99615625	−0.993934067232134	0.0022221909590201	8.19115364425471e-09	−0.99393405904098
0.1	−0.99211953125	−0.988939089490462	0.0031804532588745	1.1499336483034e-08	−0.988939077991126
0.125	−0.98653486328125	−0.982271504579925	0.00426337383533881	1.51340135978728e-08	−0.982271489445911
0.15	−0.979291195800781	−0.973809697000635	0.00548151792021834	1.91200721921803e-08	−0.973809677880563
0.175	−0.970271084680176	−0.963424908115551	0.00684620004871694	2.34840922264112e-08	−0.963424884631459
0.2	−0.959350351230774	−0.950980846933525	0.00836953255170458	2.82544557750342e-08	−0.950980818679069
0.225	−0.946397723916817	−0.936333280355021	0.0100644770232566	3.34614612684803e-08	−0.93633324689356
0.25	−0.931274461615274	−0.9193296018128	0.0119448989399197	3.91374457286986e-08	−0.919329562675355
0.275	−0.91383395748622	−0.899808377184759	0.0140256256183723	4.53169115566254e-08	−0.899808331867848
0.3	−0.893921322470468	−0.877598866798164	0.0163225077089705	5.20366669753969e-08	−0.877598814761497
0.325	−0.871372947381676	−0.852520522283577	0.0188524844340696	5.93359703593421e-08	−0.852520462947607
0.35	−0.846016042508137	−0.824382456972721	0.0216336527920992	6.72566832227162e-08	−0.824382389716038
0.375	−0.817668153584856	−0.792982888467164	0.024685340961128	7.58434365311089e-08	−0.792982812623728
0.4	−0.786136652939193	−0.758108551933874	0.0280281861491201	8.51438011206795e-08	−0.758108466790073
0.425	−0.751218204553124	−0.719534082609241	0.031684217152358	9.52084749927806e-08	−0.719533987400766
0.45	−0.712698201721927	−0.677021365914865	0.035676941898539	1.06091477158898e-07	−0.677021259823388
0.475	−0.670350175922698	−0.630318853506074	0.0400314402669782	1.17850354142135e-07	−0.630318735655719
0.5	−0.623935175436374	−0.579160843487613	0.0447744624951633	1.30546401910792e-07	−0.579160712941211
0.525	−0.573201112193673	−0.523266722939915	0.049934533498654	1.44244895805556e-07	−0.523266578695019
0.55	−0.517882075238474	−0.462340170803698	0.0555420634501166	1.59015340672752e-07	−0.462340011788357
0.575	−0.457697609121393	−0.396068319070038	0.0616294649830836	1.74931729102212e-07	−0.396068144138309
0.6	−0.392351955451482	−0.324120870117292	0.0682312774070116	1.92072821758593e-07	−0.324120678044471
0.625	−0.321533255744927	−0.246149167925044	0.0753842983423234	2.10522441229655e-07	−0.246148957402603
0.65	−0.244912713616065	−0.161785220778329	0.0831277232075167	2.30369780807038e-07	−0.161784990408548
0.675	−0.162143714257858	−0.070640672952461	0.0915032930151381	2.51709741189154e-07	−0.0706404212427199
0.7	−0.0728608990557651	0.0276942772604496	0.100555450959486	2.74643271638841e-07	0.0276945519037213
0.725	0.023320806929307	0.133652015958094	0.110331508306537	2.99277750415161e-07	0.133652315235844
0.75	0.126807216987079	0.247688711385344	0.120881820125636	3.25727370714679e-07	0.247689037112715
0.775	0.238025956790523	0.370285574110294	0.132259971433334	3.54113563194591e-07	0.370285928223857
0.8	0.357427593057991	0.501950182844749	0.144522974352188	3.84565429834005e-07	0.501950567410179
0.825	0.485486819599531	0.64321787930102	0.157731476921711	4.1722022170454e-07	0.643218296521241
0.85	0.622703703640364	0.794653235651393	0.171949984234854	4.52223824165543e-07	0.794653687875218
0.875	0.76960499545976	0.956851598340245	0.187247092611774	4.89731289077255e-07	0.956852088071534
0.9	0.92674550453606	1.13044071219166	0.203695737562987	5.29907388857609e-07	1.13044124209905
0.925	1.09470954554876	1.31608242895834	0.221373456336778	5.72927200570561e-07	1.31608300188554
0.95	1.27411245775674	1.51447450467084	0.240362665890839	6.1897673542255e-07	1.51447512364758
0.975	1.46560220144838	1.72635249037048	0.260750957175699	6.68253599345192e-07	1.72635315862408
1	1.6698610353447	1.95249172104487	0.282631406667858	7.20967687106722e-07	1.95249244201256

TABLE 4 Comparison between Runge–Kutta fourth order and Euler's method with exact solution for step size $h = 0.0125$.

	Euler method	RK4_method	Error Euler	Error_RK4	Exact solution
x_n	$y(x_n)$	$y(x_n)$	e_r	e_r	y_n
0	−1	−1	0	0	−1
0.0125	−1	−0.999841782633464	0.00015821744316 0025	7.66235963567397e−11	−0.99984178255684
0.025	−0.9996875	−0.99935914483009	0.00032835532716 6352	1.57256652144611e−10	−0.999359144672834
0.0375	−0.99905078125	−0.998539843521072	0.00051093797098 1229	2.42053488364036e−10	−0.998539843279019
0.05	−0.998077783203125	−0.997371275007575	0.00070650852672 4381	3.31174421219771e−10	−0.997371274676401
0.0625	−0.996756094360352	−0.995840465193677	0.00091562959145 8744	4.24784540875578e−10	−0.995840464768893
0.075	−0.995072942878723	−0.993934059564035	0.00113888383774 308	5.23055265766459e−10	−0.99393405904098
0.0875	−0.993015186937046	−0.991638312899708	0.00137687466350 045	6.26162788286422e−10	−0.991638312273545
0.1	−0.990569304852906	−0.988939078725416	0.00163022686178 083	7.3429029523453e−10	−0.988939077991126
0.1125	−0.987721384944694	−0.985821798481323	0.00189958731099 726	8.47626968614179e−10	−0.985821797633696
0.125	−0.984457115132157	−0.982271490412278	0.00218562568624 536	9.66366764387772e−10	−0.982271489445911
0.1375	−0.980761772268854	−0.978272738167235	0.00248903519233 112	1.09071252030191e-09	−0.978272737076523
0.15	−0.976620211199699	−0.973809679101436	0.00281053331913 594	1.22087295828521e-09	−0.973809677880563
0.1625	−0.972016853536616	−0.968865992273692	0.00315086261998 687	1.35706257342605e-09	−0.96886599091663
0.175	−0.966935676145168	−0.963424886130964	0.00351079151370 959	1.49950496464157e-09	−0.963424884631459
0.1875	−0.961360199334811	−0.957469085872189	0.00389111511105 078	1.64842939298637e-09	−0.95746908422376
0.2	−0.95527347474527	−0.950980820483144	0.00429265606620 022	1.80407422334383e-09	−0.950980818679069
0.2125	−0.948658072921328	−0.943941809433876	0.00471626545413 761	1.96668470398009e-09	−0.943941807467191
0.225	−0.941496070568132	−0.936333249030075	0.00516282367457 144	2.13651463187858e-09	−0.93633324689356
0.2375	−0.933769037478902	−0.928135798409482	0.00563324138324 639	2.31382679682923e-09	−0.928135796095656
0.25	−0.925458023126774	−0.919329565174245	0.00612846045141 902	2.49889064996012e-09	−0.919329562675355
0.2625	−0.916543542912228	−0.909894090649877	0.00664945495433 755	2.69198641156265e-09	−0.909894087957891
0.275	−0.90700556405741	−0.899808334761251	0.00719723218956 259	2.89340318371245e-09	−0.899808331867848
0.2875	−0.896823491138377	−0.889050660515806	0.00777283372600 968	3.10343872822472e-09	−0.889050657412367
0.3	−0.885976151246112	−0.877598818083899	0.00837733648461 425	3.3224019091449e-09	−0.877598814761497
0.3125	−0.874441778766906	−0.865429928465981	0.00901185385153 525	3.55061036128035e-09	−0.86542992491537
0.325	−0.862197999772465	−0.852520466735999	0.00967753682485 828	3.78839271064635e-09	−0.852520462947607
0.3375	−0.849221816009868	−0.838846244850187	0.010375575195769	4.03608801935462e-09	−0.838846240814099
0.35	−0.835489588481245	−0.824382394010085	0.0111071987652069	4.29404656276944e-09	−0.824382389716038
0.3625	−0.820977020602795	−0.809103346568393	0.0118736785970336	4.56263071768603e-09	−0.809103342005762
0.375	−0.805659140932504	−0.792982817465941	0.012676328308776	4.84221285290687e-09	−0.792982812623728
0.3875	−0.789510285455638	−0.775993785187767	0.013516505401049	5.13317843786609e-09	−0.775993780054589
0.4	−0.772504079416848	−0.758108472225999	0.0143956126267748	5.43592659774106e-09	−0.758108466790073
0.4125	−0.754613418687398	−0.739298325036908	0.0153150994013572	5.75086722687246e-09	−0.739298319286041
0.425	−0.735810450655777	−0.71953399347919	0.0162764632550103	6.07842387534419e-09	−0.719533987400766
0.4375	−0.716066554629629	−0.698785309720203	0.0172812513284611	6.41903497022867e-09	−0.698785303301168
0.45	−0.695352321736672	−0.67702126659654	0.018331061913284	6.77315237229692e-09	−0.677021259823388
0.4625	−0.673637534311906	−0.65420999541498	0.0194275460381675	7.14124126499627e-09	−0.654209988273739
0.475	−0.650891144758169	−0.630318743179503	0.0205724091024501	7.52378404023091e-09	−0.630318735655719
0.4875	−0.627081253866695	−0.60531384922969	0.0217674125582822	7.92127663462594e-09	−0.605313841308413

(Continued)

TABLE 4 (Continued)

	Euler method	RK4_method	Error Euler	Error_RK4	Exact solution
x_n	$y(x_n)$	$y(x_n)$	e_r	e_r	y_n
0.5	−0.602175088584054	−0.579160721275443	0.0230143756428427	8.33423174917414e-09	−0.579160712941211
0.5125	−0.57613897921148	−0.551823810812613	0.0243151771620446	8.76317751696831e-09	−0.551823802049435
0.525	−0.548938336022252	−0.52326658790368	0.0256715737273326	9.20866050080349e-09	−0.523266578695019
0.5375	−0.520537625282424	−0.49345151530727	0.0270861196463964	9.67124291761934e-09	−0.493451505636028
0.55	−0.490900344659846	−0.462340021939864	0.0285603328714887	1.01515071904146e-08	−0.462340011788357
0.5625	−0.45998899800602	−0.429892475652617	0.0300965330034543	1.06500506191765e-08	−0.429892465002566
0.575	−0.42776506949497	−0.396068155305802	0.031696925356661	1.11674930414196e-08	−0.396068144138309
0.5875	−0.394188997102878	−0.360825222122928	0.0333637866844218	1.17044715031156e-08	−0.360825210418457
0.6	−0.359220145411866	−0.324120690306115	0.0350994673673957	1.22616446995849e-08	−0.324120678044471
0.6125	−0.322816777720846	−0.285910396893872	0.0369063936666661	1.2839691865274e-08	−0.28591038405418
0.625	−0.284936027445972	−0.246148970841916	0.0387870700433691	1.34393123574217e-08	−0.246148957402603
0.6375	−0.245533868792753	−0.204789801307189	0.0407440815467922	1.40612288479502e-08	−0.20478978724596
0.65	−0.20456508668146	−0.161785005114735	0.0427800962729118	1.47061868793763e-08	−0.161784990408548
0.6625	−0.161983245906997	−0.117085393386553	0.0448978678953992	1.53749557252336e-08	−0.117085378011597
0.675	−0.117740659513903	−0.0706404373110469	0.0471002382711829	1.60683270439277e-08	−0.0706404212427199
0.6875	−0.0717883563667219	−0.0223982330311248	0.0493901401227153	1.67871181816492e-08	−0.0223982162440066
0.7	−0.0240760478954237	0.0276945343715484	0.051770599799145	1.7532172907897e-08	0.0276945519037213
0.7125	0.0254479060049128	0.0796926278192158	0.0542447401186637	1.83043606938327e-08	0.0796926461235765
0.725	0.0768365319414793	0.133652296131265	0.0568157832943649	1.91045788910937e-08	0.133652315235844
0.7375	0.13014427888004	0.189631312892295	0.0594870539460066	1.99337518713705e-08	0.189631332826047
0.75	0.185427054912566	0.247689016319883	0.062261982200149	2.07928315953954e-08	0.247689037112715
0.7625	0.242742264959161	0.307886350157565	0.0651441068812088	2.16828046073481e-08	0.30788637184037
0.775	0.302148849427843	0.370285905619175	0.0681370787960134	2.26046822926484e-08	0.370285928223857
0.7875	0.363707323856302	0.434951964411365	0.0712446641145731	2.35595102315855e-08	0.434951987970875
0.8	0.427479819560381	0.501950542861812	0.0744707478497984	2.45483675609393e-08	0.501950567410179
0.8125	0.493530125314658	0.571349437181299	0.0778193374390066	2.55723657804907e-08	0.571349462753664
0.825	0.561923730091108	0.643218269888593	0.0812945664301338	2.66326488640445e-08	0.643218296521241
0.8375	0.632727866882513	0.717628537427769	0.0849006982756543	2.77303989770772e-08	0.717628565158168
0.85	0.706011557637944	0.794653659008383	0.088642130237274	2.88668347003807e-08	0.794653687875218
0.8625	0.781845659338301	0.874369026699666	0.0925233974045763	3.00432116961957e-08	0.874369056742878
0.875	0.860302911240661	0.956852056810713	0.0965491768308727	3.12608208208331e-08	0.956852088071534
0.8875	0.941457983320816	1.04218224258944	0.100724291789624	3.2521000781216e-08	1.04218227511044
0.9	1.02538752594421	1.13044120827392	0.105053716154833	3.38251242570919e-08	1.13044124209905
0.9125	1.1121702207962	1.22171276453059	0.109542578909002	3.51746087812188e-08	1.2217127997052
0.925	1.20188683310328	1.31608296531462	0.114196168782255	3.65709158511862e-08	1.31608300188554
0.9375	1.29462026517788	1.41364016618871	0.119019939026382	3.80155509294156e-08	1.41364020420426
0.95	1.39045561131993	1.51447508413751	0.124019512327651	3.95100665517845e-08	1.51447512364758
0.9625	1.48948021410943	1.61868085891571	0.129200685862344	4.10560634378498e-08	1.61868089997178
0.975	1.59178372212501	1.72635311596889	0.13456943649907	4.2655188714491e-08	1.72635315862408
0.9875	1.69745814912426	1.83759003096718	0.140131926152066	4.43091430213372e-08	1.83759007527633
1	1.80659793472282	1.95249239599288	0.145894507289739	4.60196762919196e-08	1.95249244201256

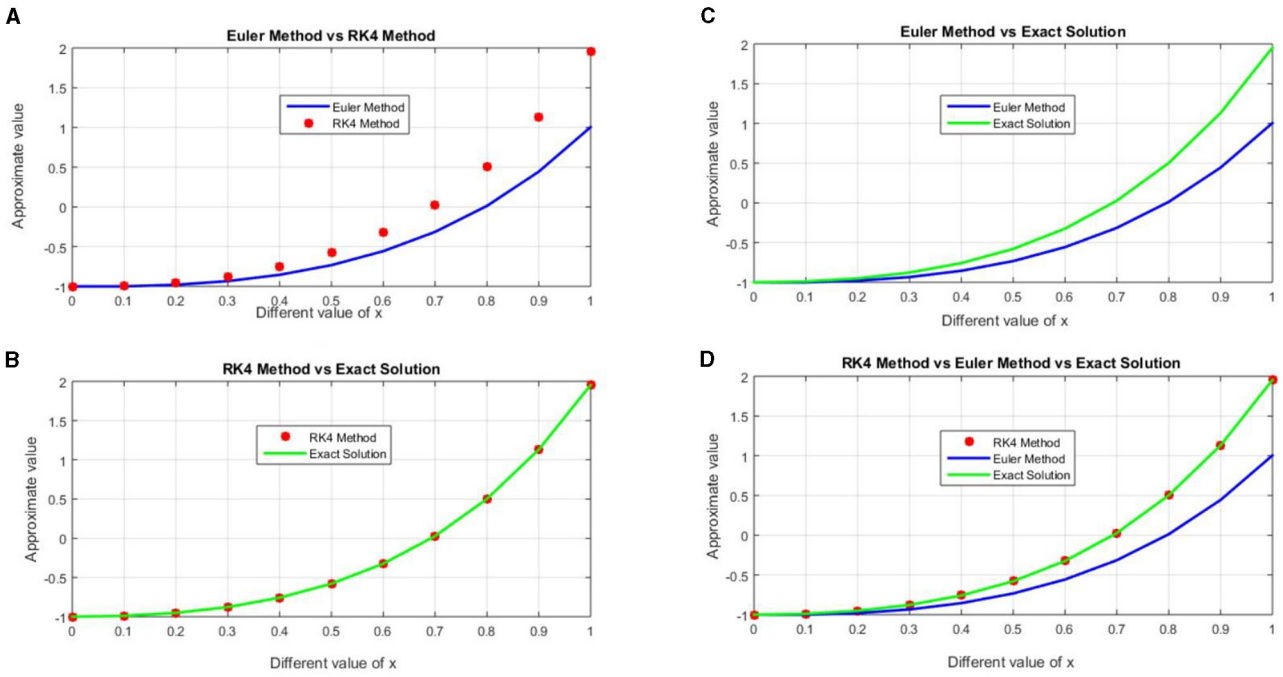

FIGURE 1
Exact and approximate numerical Solutions for $h = 0.1$, **(A)** graph of the approximate solution for Eulers and Runge-Kutta fourth order, **(B)** graph of the approximate solution of Exact and Runge-Kutta fourth order, **(C)** graph of the approximate solution for Eulers and Exact, and **(D)** graph of approximate solutions for Euler, Runge-Kutta forth order methods and exact solution.

evaluated. The truncation error in Euler's method arises from the linear approximation of the derivative [14, 15]. It is proportional to the step size h, used in the numerical scheme. Specifically, the truncation error is of order $O(h)^2$, meaning that by halving the step size, the error typically decreases by a factor of four. The truncation error in RK4 arises from the approximation of the derivative at various internal points within the step. It is proportional to the step size h, increased to the power of five. Therefore, the truncation error of RK4 is of order $O(h)^5$, which is significantly smaller than Euler's method [16]. The global error in Euler's method is the cumulative effect of the truncation error at each step throughout the integration interval [17]. As the number of steps increases, the global error accumulates, resulting in a larger discrepancy between the numerical solution and the exact solution. The global error in RK4 is significantly smaller compared to Euler's method [15]. Due to its higher order of accuracy, the cumulative effect of the truncation error is reduced, resulting in a more accurate approximation of the exact solution. It is important to note that while RK4 has a smaller truncation error and is typically more accurate than Euler's method, the step size h, also plays a role. The accuracy of the solution will depend on how small we make the step size h [18]. If $|y(x_n)_- y_n| = 0$, then a numerical technique is considered as convergent. Where the exact solution is denoted by y_n and the approximate solution by $y(x_n)$.

Using a smaller step size can improve the accuracy of both methods; however, it can also increase computational cost [19]. In error analysis, it is common to compare the numerical solutions obtained using Euler's method and RK4 to an analytically available exact solution or a solution obtained using a more accurate method (such as a higher-order Runge-Kutta method). By comparing the errors, we can assess the convergence properties of the methods and determine their suitability for a specific problem. In order to verify the accuracy of the suggested methods, we examine first-order initial value problem ODE in this study. MATLAB software is used to obtain the approximate solution for the two numerical methods that are proposed, at different step sizes. The formula for calculating the maximum error is defined by $e_r = \max_{0 \le x \le steps} (|y(x_n)_- y_n|)$.

3.1 Numerical example

This section examines a numerical example to verify which numerical techniques converge to an analytical solution more quickly. Errors and numerical solutions are calculated.

Example 1: We consider the initial value problem $y'' - 3y' + 2y = 0$, $y(0) = -1$, $y'(0) = 0$ on the interval $0 \le x \le 1$ with $h = 0.1$. Then the exact solution to the given problem is given by $y(x) = e^{2x} - 2e^x$.

The approximate results and maximum errors are obtained and shown in Tables 1–4 and the graphs of the numerical solutions are shown as follows in Figures 1–6 from (a)-(r).

4 Discussion of results

In this work the obtained results are shown in the Tables 1–4 and graphically representations are show in the Figures 1–7. The Tables 1–4 shows that the comparison of the two desired methods Euler's and Runge-Kutta fourth order method with the exact solution and also the Figures 1–4 shows that the graph of

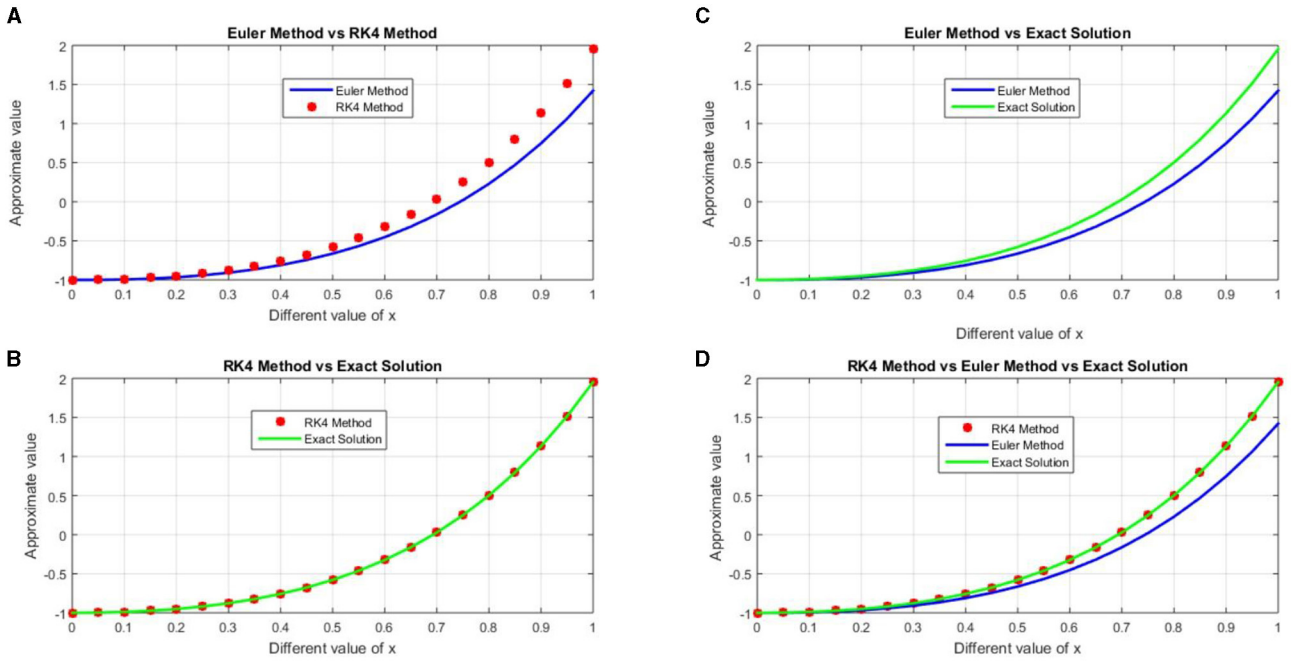

FIGURE 2
Exact and approximate numerical solutions for $h = 0.05$, **(A)** graph of the approximate solution for Eulers and Runge-Kutta fourth order, **(B)** graph of the approximate solution of Exact and Runge-Kutta fourth order, **(C)** graph of the approximate solution for Eulers and Exact, and **(D)** graph of approximate solutions for Euler, Runge-Kutta forth order methods and exact solution.

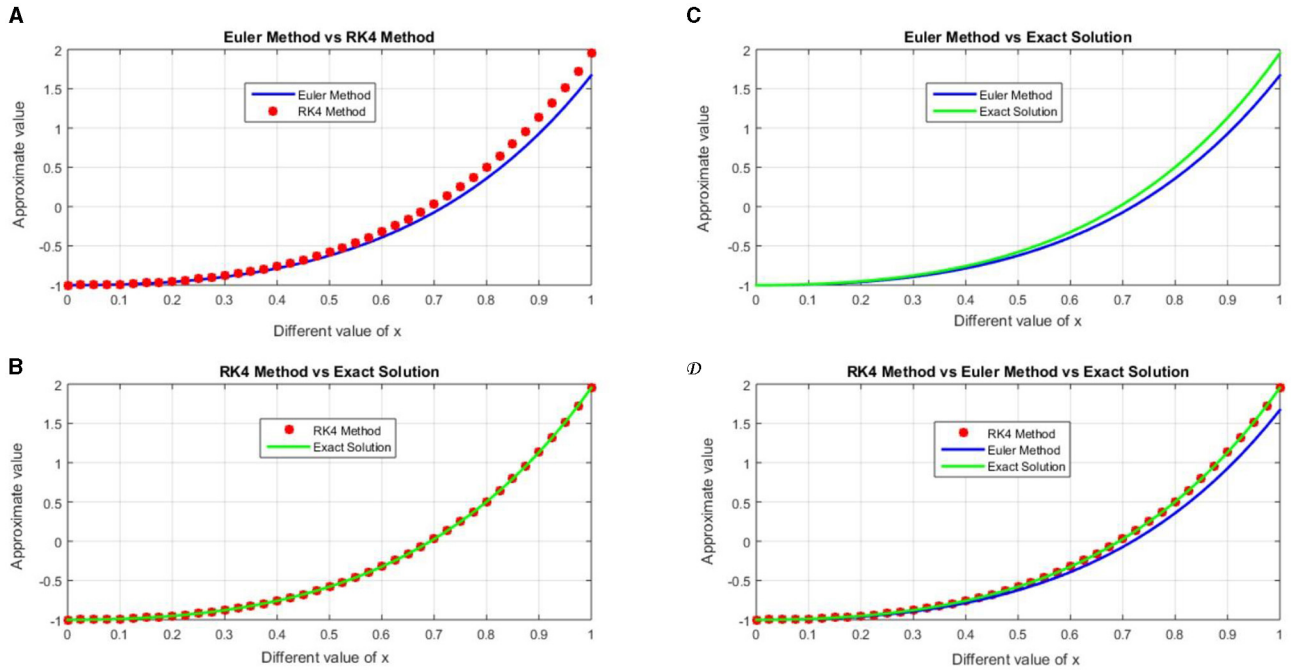

FIGURE 3
Exact and approximate numerical solutions for $h = 0.025$, **(A)** graph of the approximate solution for Euler's and Runge-Kutta fourth order, **(B)** graph of the approximate solution of Exact and Runge-Kutta fourth order, **(C)** graph of the approximate solution for Euler's and Exact, and **(D)** graph of approximate solutions for Euler, Runge-Kutta forth order methods and exact solution.

the approximate solution for Euler and Runge-Kutta methods for each step size h. The approximated numerical solution is calculated with the step size $h = 0.1,\ 0.05,\ 0.025,\ 0.125$. The approximate solution of Euler's and Runge-Kutta fourth order methods have different values for the same step size h for each iteration for example when we compared the accuracy of the Euler

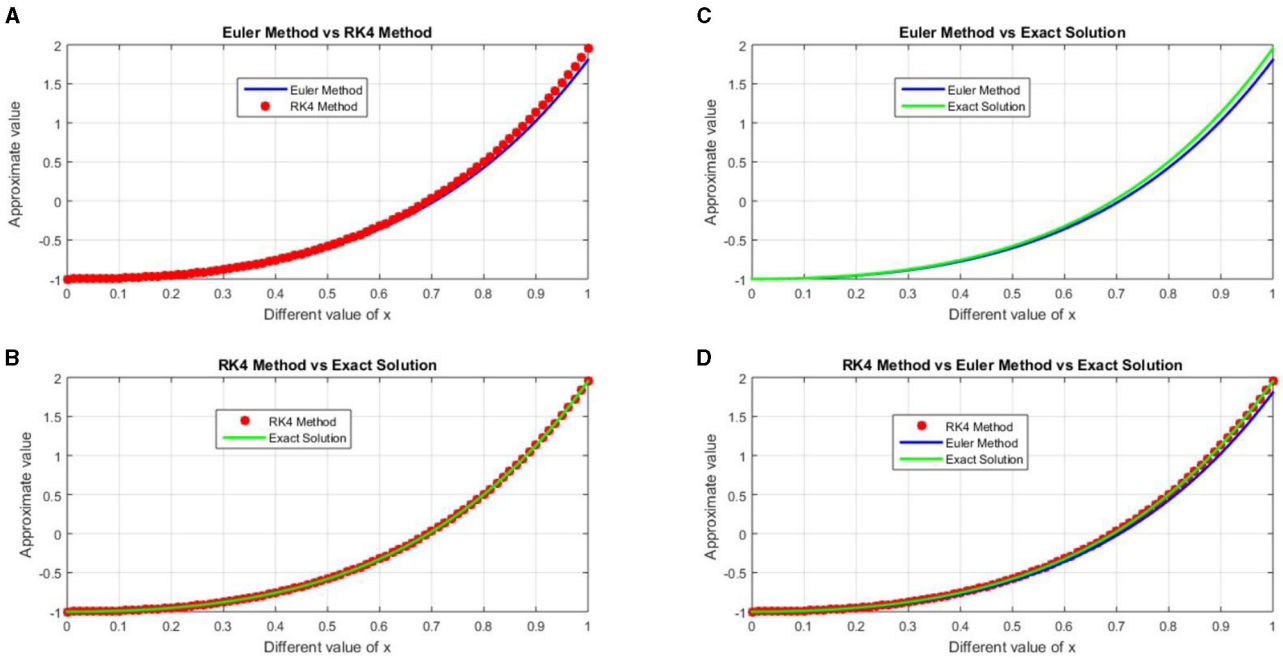

FIGURE 4
Exact and approximate numerical Solutions for $h = 0.0125$, **(A)** graph of the approximate solution for Euler's and Runge-Kutta fourth order, **(B)** graph of the approximate solution of Exact and Runge-Kutta fourth order, **(C)** graph of the approximate solution for Eulers and Exact, and **(D)** graph of approximate solutions for Euler, Runge-Kutta forth order methods and exact solution.

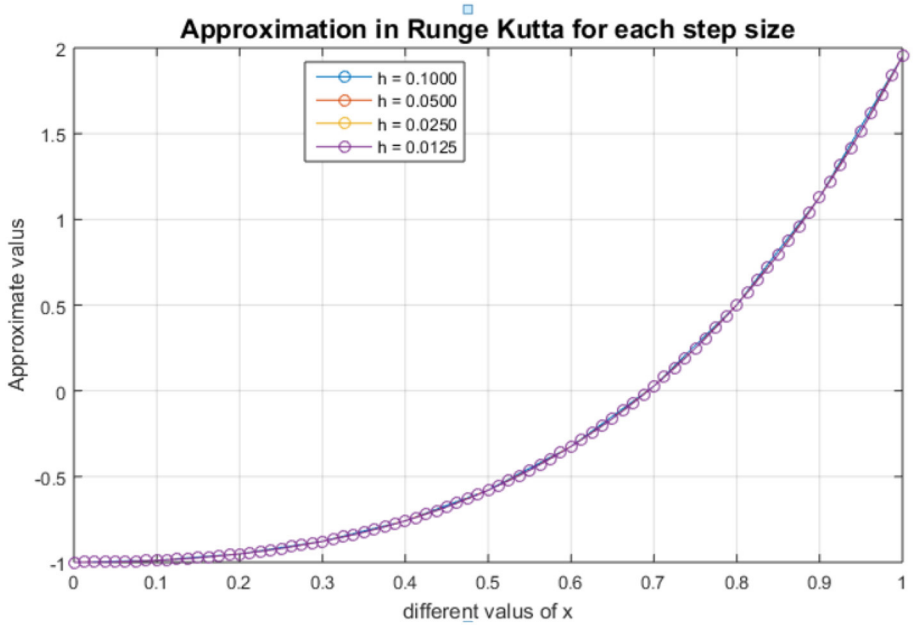

FIGURE 5
Approximate value for Runge-Kutta fourth order method for each step size.

and Runge-Kutta fourth order method with sizes $h = 0.1$ and $h = 0.05$, then approximated solution with the step size $h = 0.1$ has less accurate then the approximated solution with the step size $h = 0.05$ because the error of approximate values for $h = 0.1$ is greater than the error for $h = 0.05$ *in* each iterations. This shows that the Euler's method with $h = 0.1$ and $h = 0.05$ does not converges to the exact solution. Similarly for the step sizes $h = 0.025$ and $h = 0.0125$, then approximate solution with the step size $h = 0.025$ has less accurate than the approximate the solution with the step size $h = 0.0125$ because of

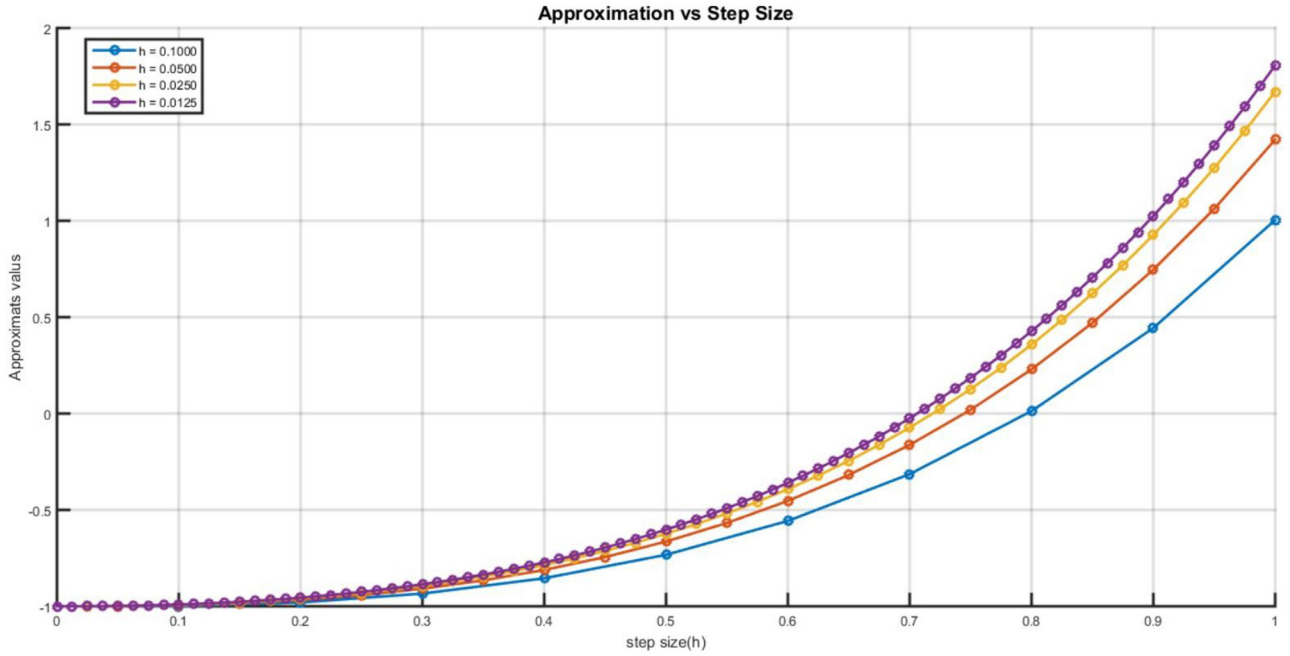

FIGURE 6
Approximate value for each step size in Euler method.

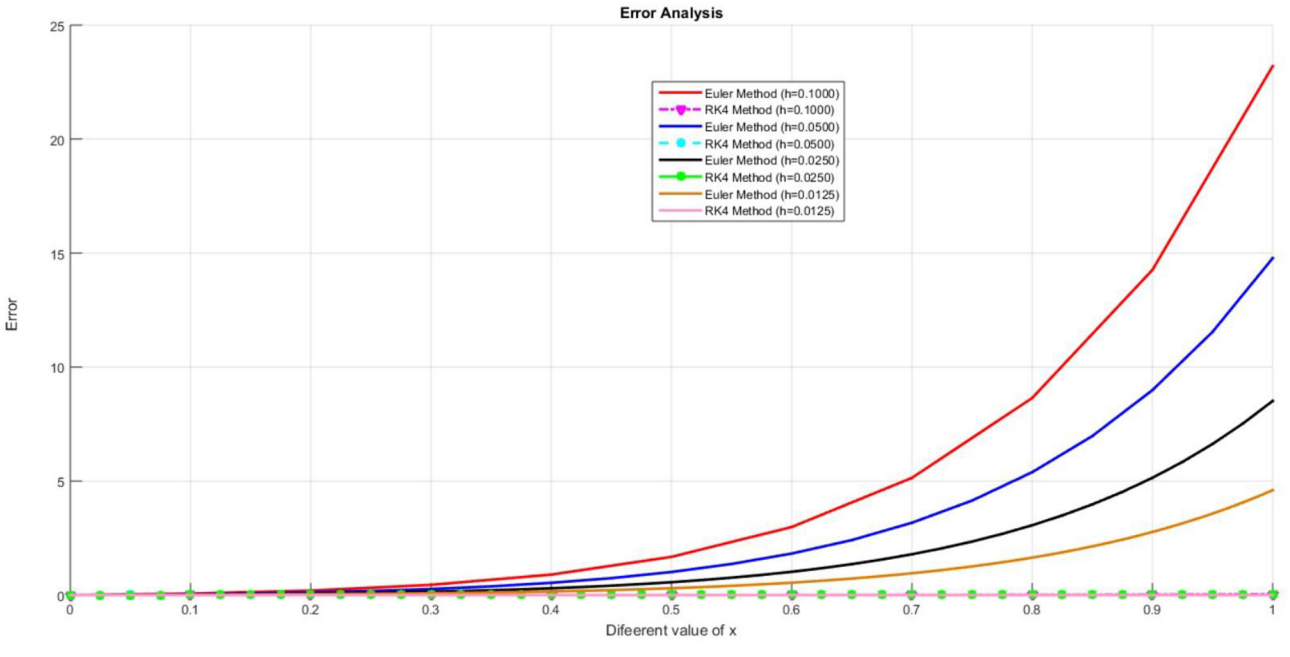

FIGURE 7
The approximated error for Euler and Runge-Kutta fourth order for different step size.

the approximate solution with the step size $h = 0.0125$ has less absolute error then the approximated solution for $h = 0.025$. This shows that the Euler's method with $h = 0.1$ and $h = 0.05$ does not converges to the exact solution but for $h = 0.025$ and $h = 0.0125$ Converges slowly to exact solution. The Runge-Kutta fourth order method with the same step size also the approximate solution obtained for $h = 0.1$ and $h = 0.05$ converges gradually to exact solution but the approximate for $h = 0.025$ and $h = 0.0125$ converges fatly to exact solution. This shows that as the step size decreases the accuracy the approximate solution also increases. The Figures 5, 6 shows that the approximate solution of Euler and Runge-Kutta fourth order method with the same step size

respectively. According to Figure 5 the graph of the approximate solution for Runge-Kutta fourth order method are approximately all are overlapped because of little difference between approximate solutions of Runge-Kutta methods with Exact solution (i.e., error) for each h values. From the Figure 6 also the graph of approximated solutions for Euler's method has the gap between two consecutives step sizes this shows that Euler's method has maximum absolute error compared to Runge-Kutta fourth order for same step size. Graph of approximated solutions for $h = 0.0125$ is above the graph of approximated value for $h = 0.1,\ 0.05,\ 0.025$ this means that the graph of approximated solution approach's to the graph of exact solution as step size decreases but the graph for approximated solution for $h = 0.1$ is under the graph of approximated solution *for* $h = 0.05,\ 0.025,\ 0.0125$ this means that it away from the graph of exact solution and it has maximum error for $h = 0.1$. In the Figure 7 the graph of maximum absolute error for Euler and Runge-Kutta fourth order for each step size are plotted. The graph of error for Euler's method in Figure 7 for $h = 0.1$ is above $h = 0.05,\ 0.025,$ *and* 0.0125 this shows the approximate solution for h=0.1 has maximum error then all approximated solution for $h = 0.05,\ 0.025,\ 0.0125$ but for $h = 0.0125$ the error graph is below $h = 0.05,\ 0.025,$ *and* 0.1 this shows that the absolute error for $h = 0.0125$ is less then all other approximated solution for $h = 0.05,\ 0.025,$ *and* 0.1 for every iteration. Additionally, error analysis for Runge-Kutta method in Figure 7 the graph of error for each step size resembles overlapped to x-axis that means the error for Runge-Kutta is almost tends to zero. From the Figure 7 we observed that Runge-Kutta fourth order converges fastly then the Euler's method with the same step size. The absolute error for each step size h of Euler and Runge-Kutta fourth order methods tends to zero as the step size tends to zero. Although the fourth order Runge-Kutta method is more accurate than the Euler's approach, which only requires one-fourth the step size, it is arduous and takes four evaluations per step size. Finally, we noticed that, as the tables and figures show, the fourth order Runge-Kutta method is the most effective approach for solving second order initial value problem for ordinary differential equations, and that it is converging more quickly than the Euler method.

5 Conclusion

In this work, we have discussed Euler's and fourth order Runge-Kutta method for solving second order initial value problems that provides efficient solutions. To achieve the desired accuracy of the numerical solution it is necessary to take step size small. From the tables and figures, we can see that accuracy of the method obtained 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 second order initial value problems in ordinary differential equations. Our research will be helpful in many scientific areas where numerical computations are needed.

Data availability statement

The original contributions presented in the study are included in the article/supplementary material, further inquiries can be directed to the corresponding author.

Author contributions

YW: Conceptualization, Data curation, Investigation, Methodology, Resources, Software, Supervision, Validation, Visualization, Writing—original draft, Writing—review & editing, Formal analysis, Project administration. HM: Conceptualization, Data curation, Investigation, Methodology, Resources, Software, Supervision, Validation, Visualization, Writing—original draft, Writing—review & editing. BB: Conceptualization, Data curation, Formal analysis, Investigation, Methodology, Project administration, Resources, Software, Supervision, Validation, Visualization, Writing—original draft, Writing—review & editing.

Funding

The author(s) declare that no financial support was received for the research, authorship, and/or publication of this article.

Conflict of interest

The authors declare that the research was conducted in the absence of any commercial or financial relationships that could be construed as a potential conflict of interest.

Publisher's note

References

1. Sobczyk K. *Stochastic Differential Equations: With Applications to Physics and Engineering*. Cham: Springer Science & Business Media (2001).

2. Zhu SP. An exact and explicit solution for the valuation of American put options. *Quantitative Financ.* (2006) 6:229–42. doi: 10.1080/14697680600699811

3. Hamming R. *Numerical Methods for Scientists and Engineers*. Chelmsford, MA: Courier Corporation (2012).

4. Chapra SC. *Numerical Methods for Engineers*. New York, NY: Mcgraw-Hill. (2010).

5. Brissaud A, Frisch U. Solving linear stochastic differential equations. *J Mathematic Phys*. (1974) 15:524–34. doi: 10.1063/1.1666678

6. Denis B. *An Overview of Numerical and Analytical Methods for solving Ordinary Differential Equations*. Cornell University (2020). Available online at: http://arxiv.org/abs/2012.07558

7. Butcher JC. *The Numerical Analysis of Ordinary Differential Equations: Runge-Kutta and General Linear Methods*. London: Wiley-Interscience (1987).

8. Jator SN Li J. Boundary value methods via a multistep method with variable coefficients for second order initial and boundary value problems. *Int J Pure Appl Mathematics*. (2009) 50:403–20.

9. Hossain MJ, Alam MS, Hossain MB. A study on numerical solutions of second order initial value problems (IVP) for ordinary differential equations with fourth order and Butcher's fifth order Runge-Kutta methods. *Am J Comput Appl Mathematics*. (2017) 7:129–37. doi: 10.5923/j.ajcam.20170705.02

10. Hussain K, Ismail F, Senu N. Solving directly special fourth-order ordinary differential equations using Runge–Kutta type method. *J Comput Appl Math*. (2016) 306:179–99. doi: 10.1016/j.cam.2016.04.002

11. Kamruzzaman M, Nath MC. A comparative study on numerical solution of initial value problem by using Euler's method, modified Euler's method and Runge–Kutta method. *J Comput Mathematic Sci*. (2018) 9:493–500. doi: 10.29055/jcms/784

12. Cromer A. Stable solutions using the Euler approximation. *Am J Phys*. (1981) 49:455–9. doi: 10.1119/1.12478

13. Hossen M, Ahmed Z, Kabir R, Hossan Z. A comparative investigation on numerical solution of initial value problem by using modified Euler method and Runge-Kutta method. *IOSR-JM*. (2019) 12:2278–5728. doi: 10.9790/5728-1504034045

14. Hindmarsh AC, Petzold LR. Algorithms and software for ordinary differential equations and differential-algebraic equations, Part I: Euler methods and error estimation. *Computers in Physics*. (1995) 9:34–41. doi: 10.1063/1.168536

15. Okeke AA, Tumba P, Anorue OF, Dauda A. Analysis and comparative study of numerical solutions of initial value problems (IVP) in ordinary differential equations (ODE) with Euler and Runge-Kutta methods. *AJER*. (2019) 8:6–15.

16. Amir Taher K. *Comparison of Numerical Methods for Solving a System of Ordinary Differential Equations: Accuracy, Stability and Efficiency*. Lowa City, IA: Johan Wiley, and Sons (2009). p. 81.

17. Neto AR, Rao KR. A stochastic approach to global error estimation in ODE multistep numerical integration. *J Comput Appl Math*. (1990) 30:257–81. doi: 10.1016/0377-0427(90)90279-9

18. Gustafsson K, Söderlind G. Control strategies for the iterative solution of nonlinear equations in ODE solvers. *SIAM J Sci Comp*. (1997) 18:23–40. doi: 10.1137/S1064827595287109

19. Atkinson K, Han W, Stewart DE. *Numerical Solution of Ordinary Differential Equations*. New York, NY: John Wiley & Sons (2009).