v. 1, n. 1, 2024

International Journal of
Exact
Sciences

CLASSICAL NUMERICAL METHODS TO SOLVE ORDINARY FIRST ORDER DIFFERENTIAL EQUATIONS

Esiquio Martín Gutiérrez Armenta
Department of Systems, Area of Computer Systems, ``Universidad Autónoma Metropolitana`` Azcapotzalco/ Cdmx, Mexico

Marco Antonio Gutiérrez Villegas
Department of Systems, Area of Computer Systems, ``Universidad Autónoma Metropolitana`` Azcapotzalco/ Cdmx, Mexico

Nicolas Domínguez Vergara
Systems Department, Applied Statistics Area and Operations Research, ``Universidad Autónoma Metropolitana`` Azcapotzalco/, Cdmx, Mexico

Israel Isaac Gutiérrez Villegas
Division of Engineering in Computer Systems, Thesis - Tecnm, /state of Mexico
Department of Administration, Mathematics and Systems Area, ``Universidad Autónoma Metropolitana`` Azcapotzalco/Cdmx, Mexico

Alfonso Jorge Quevedo Martinez
Systems Department, Applied Statistics Area and Operations Research, ``Universidad Autónoma Metropolitana`` Azcapotzalco/, Cdmx, Mexico

Josué Figueroa González
Department of Systems, Area of Computer Systems, ``Universidad Autónoma Metropolitana`` Azcapotzalco/ Cdmx, Mexico

Abstract: The objective of this article is to make a comparison of the classical methods for solving first-order ordinary differential equations. Both the analytical solution and the implementation in programming language in Dev-C++ of the fourth-order Euler, Improved Euler (Heun) and Runge-Kutta methods will be addressed. This way, it is intended to offer a complete guide for selecting the most appropriate one to solve a given problem. In this work, classical methods are used to find approximate solutions to problems with initial values. Cauchy showed that an ordinary differential equation with initial conditions has a unique solution. These equations are widely used in various areas, such as engineering, exact sciences, humanities, biology, management and medicine, among others.

Keywords: Ordinary differential equations, Analytical solution, DevC++ programming language, Euler, Improved Euler (Heun), Runge-kutta, Cauchy.

INTRODUCTION

Numerical methods emerge as a viable alternative to solve differential equations that do not admit analytical solutions. These methods provide numerical approximations to the solutions of ordinary differential equations in a given interval.

The first numerical method forFirst-order ordinary differential equations was proposed by Leonhard Euler in 1768. Kamruzzaman and Mithun Chandra Nath (2018) introduced a simplified version of Euler's method, applicable to both linear and nonlinear ordinary differential equations Kamruzzaman eat al. (2018) and Nurujjaman. (2020). Although this method is fundamental among numerical methods for ordinary differential equations, its simplicity makes it susceptible to significant propagation error, especially when a high number of partitions is used.

In order to improve the precision of Euler's method, mathematicians reformulated this method to increase its speedup and reduce the error. Among the proposed variants are the midpoint method and the modified Euler method. This article provides a concise review of the geometric interpretation of the Euler method, the midpoint method, and the modified Euler method.

In 1900, Heun (who also proposed the improved Euler method) made additional contributions, while Kutta, in 1901, fully characterized the set of order 4 RK methods and proposed the first order 5 method, presented in a published paper. that same year. A further development of this work is found in the article by Geeta eat al(2020).

Picard, in 1891, proposed a method based on the fundamental theorem of integral calculus by Youssef and Arabawy.(2007). They begin by reformulating the initial value problem for systems of differential equations.

$$y' = f(t, y), y(t_0) = y_0 (1)$$

In equation (1), the solutions converge. As shown by Hubbard eat al. (2001), the theorem states that if the function is defined for on an interval and on an interval, and satisfies the Lipschitz condition with respect to for, then there exists a unique solution of the equation (1) for each in.

$$.f(t,x)x \;\; x\epsilon[c,d]tt\epsilon[a,b]|f(t,x_1) - f(t,x_2)| \leq K|x_1 - x_2|\; tt \in [a,b], x_1x_1, x_2x_1, x_2 \in [c,d]$$

From equation (1) the method converges if it satisfies:

The theorem proven byhubbard eat. (2001). If the following conditions are met:

1. The function is defined for on an interval and on an interval. $f(t,x)x_1$, x_2x_1, $x_2 \in [c,d]$ $tt \in [a,b]$

2. The function satisfies the Lipschitz condition with respect to en. $f(t,x)x_1$, x_2x_1, $x_2 \in [c,d]$

3. For each sequence that of Euler approximations for the first-order ordinary differential equation given by with u_n: $[a,b] \to [c,d]$ $x' = f(t,x)$ $u(t_0) = c$

4. Step lengths tend to 0, and if it exists for some then there is a unique solution for $\lim_{n\to\infty} u_n(t) = ct_0 \in [a,b\,]x' = f(x,t)\ con\ u(t_0) = c$

From equation (1) the solutions converge. Hubbard eat. (2001). al, demonstrate the theorem, that if $f(t,x)$ is defined for and satisfies the Lipschitz condition if para is a sequence of Euler approximations for the first-order ordinary differential equation given by. $x' = f(x,t)$ *con* With step lengths tending to 0, and if then there exists a unique solution for. Although the theorem mentioned above is not valid for the differential equation (1) when x=0, that is, when the solution crosses the t-axis, Euler approximations can still converge under certain conditions. If theorem is not true for the differential bond (1) when x = 0, that is, when the solution crosses the axis. $x' = f(x,t)$ *con* $u(t_0) = ct$

But the Euler approximations converge if they satisfy the Lipschitz condition: if a sequence of Euler approximation (t), defined for, all satisfy for some then the theorem applies and they converge to a solution if they converge. in a single point. This is stated in the following proposition. $u_n t \in [a,b]$ $u_n(t) > \varepsilon$ ó $u_n(t) < -\varepsilon\ \varepsilon > 0$ u_n

Proposition. Let be a secession of step lengths tending to 0, and let $h_k > 0$ u_k

a secession that approximates Euler's method with step length. h_k

1. If there exists such that ac converges with c > 0, then the sequence converges for each a function a function t_0 ck = $u_k(t_0)u_k(t)t \geq t_0 - 2\sqrt{-c} = bv(t) = -\frac{(t-b)^2}{4}$

2. If there exists such that converges to c with c < 0, then the sequence converges for each a function t_0 ck = $u_k(t_0)u_k(t)t \geq t_0 - 2\sqrt{-c} = bv(t) = -\frac{(t-b)^2}{4}$

The fact depends on the fundamental equality theorem without which it is essentially impossible to prove anything about different questions. We reformulate it here with the following theorem.

Theorem.

Let it be defined for everything, and satisfy the Lipschitz condition for everything,, Suppose that they are continuous functions differentiable by parts that satisfy $f(x,t)t \in$

$$[a,b]x \in [c,d]|f(t,x_1) - f(t,x_2)| \leq K|x_1 - x_2|t \in [a,b]x_1, x_2 \in [c,d]u_1, u_2: [a,b] \to [c,d]$$

$$\left|u_1'(t) - f(t,u_1(t)\right| \leq \epsilon_1, \left|u_2'(t) - f(t,u_2(t)\right| \leq \epsilon_2$$

For all points where the functions and are differentiable and $t \in [a,b]u_1u_2$

$$|u_1(t_0) - u_2(t_0)| \leq \delta$$

For some, then. You get $\epsilon_1\epsilon_2\delta \geq 0|u_1(t_0) - u_2(t_0)| \leq \delta$

$$|u_1(t_0) - u_2(t_0)| \leq \delta e^{K|t-t_0|} + \frac{\epsilon_1+\epsilon_2}{K}\left(e^{K|t-t_0|} - 1\right)(2)$$

It can be thought that the are the errors of the slopes that measure the extent to which they do not have the appropriate slope to be the solution and as the error of the initial condition. $\varepsilon u_1u_2x' = f(x,t)\ \delta$

Then, regardless of the choice of the initial condition, successive approximations converge in some interval to the solution of problem (1). Furthermore, if it is continuous in the rectangle R, then the error of the approximate solution is estimated by the inequality: $y_n(t)[t_0, t_0 + h]f(t,y)$

$$\varepsilon_n = |y(t) - y_n(t)| \leq ML^n \frac{(t-t_0)^{n+1}}{(n+1)!},$$

$$M = \max_{\{t,y\}\in R} |f(t,y|(3)$$

METHODOLOGY

In this study, the fourth-order Euler, Improved Euler (Heun Method) and Runge-Kutta methods will be used to approximate the solution of the ordinary differential equation (1).

The same spacing will be used for each method in order to make accurate comparisons with the known analytical solution.

Euler's method is a simple and direct numerical method for approximating solutions of first-order ordinary differential equations. It was given the initial value problem by equation (4).

$$\begin{cases} \frac{dy}{dt} = f(t,y) \quad a < t < b(4) \\ y(a) = \alpha \end{cases}$$

In these methods, a partition for time is used as follows, where the other approximations will be determined from the initial condition. $\{t_0, t_1, t_2, \ldots t_n\} h > 0 t_i = a + ih, i = 1,2,..i = n - 1 t_0 = a t_n = b$

Equation (5) represents a generalization of Euler's method:

$$y_{i+1} = y_i + h\, f(t_i, y_i)(5)$$

John H. Hubbard. eat. (2001). In his article he gives some examples where the convergence of Euler's method exists where there are cases in which its convergence is not obvious or may fail. To avoid these the Lipschitz condition must be satisfied and if a is a sequence then Euler's approximation of the differential equation converges i.e. $u_n\, u(t) = \lim_{n\to\infty} u_n(t)$

The improved Euler method, also known as Heun's method, is a variant of the explicit Euler method that offers greater precision in solving first-order ordinary differential equations.This method is based on the following equations (6,7) where the iterative expression is obtained:

$$y^*_{i+1} = y_i + h\, f(t_i, y_i), i = 0,1,..(6)$$

$$y_{i+1} = y_i + \frac{h}{2}[\, f(t_i, y_i) + f(t_{i+1}, y^*_{i+1})](7)$$

The fourth-order Runge-Kutta iterative method is an explicit high-order numerical method for solving first-order ordinary differential equations.following iterative formulas Equations (8-12):

$$k_1 = f(t_i, y_i)(8)$$

$$k_2 = hf(t_i + \frac{h}{2}, y_i + \frac{1}{2}\, k_1)(9)$$

$$k_3 = hf\left(t_i + \frac{h}{2}, y_i + \frac{1}{2}k_2\right) \quad (10)$$

$$k_4 = hf(t_i + h, y_i + k_3)(11)$$

$$y_{i+1} = y_i + \frac{1}{6}(k_1 + 2k_2 + 2k_3 + k_4)(12)$$

APPLICATION 1

It is important to note that the choice of the most appropriate numerical method depends on the specific ordinary differential equation, the level of precision required and the available computational resources. The comparison between numerical and analytical solutions provides valuable information for making this decision.

$$\frac{dy}{dx} = y + x, y(0) = 1 \;\; y(1) = ? h = 0.05(13)$$

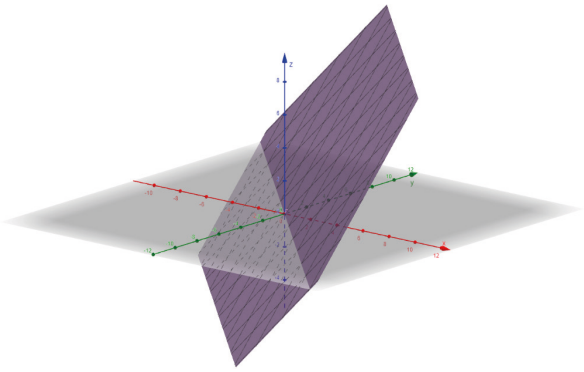

Figure1. The graph of equation (4).

The analytical solution of equation (13) is the following:

$$y(x) = 2e^x - x - 1(14)$$

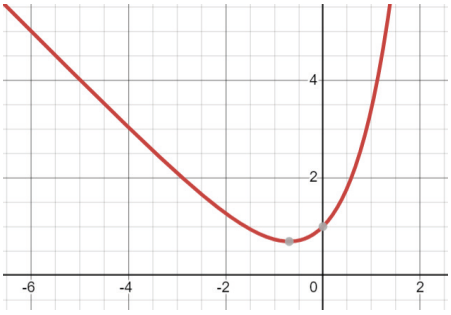

Figure 2: The graph of equation (14)

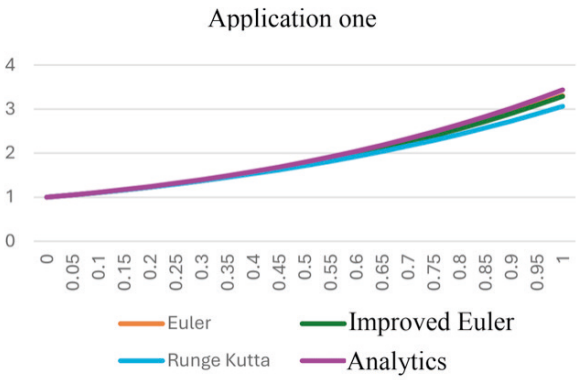

Figure 3: A comparative graphical representation between the numerical approximations and the analytical solution

APPLICATION 2

The following ordinary differential equation (ODE) will be solved using the numerical methods described above.

$$\frac{dy}{dx} = ycos(x), y(0) = 1 \;\; y(1) =? h = 0.05(15)$$

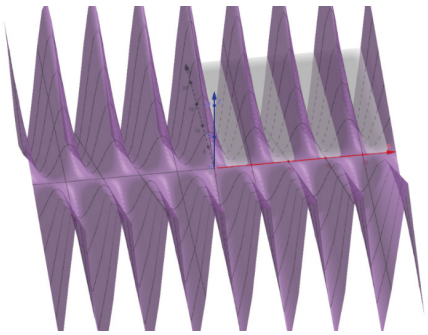

Figure 4: A graphical representation of equation (15).

Equation (16) represents the analytical solution of equation (15).

$$y(x) = e^{\sin(x)}(16)$$

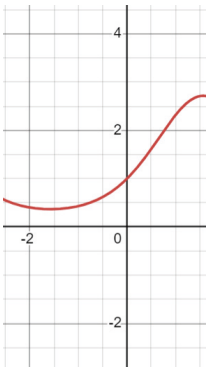

Figure 5: Presents a graphic representation of equation (16).

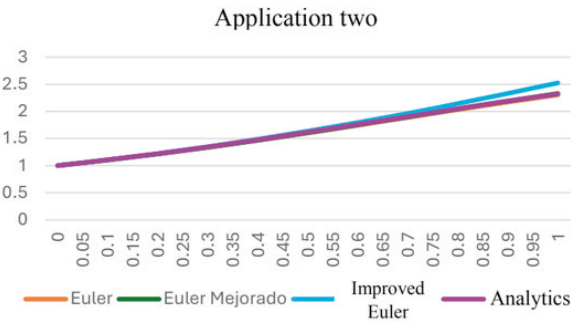

Figure 6: Presents a graphic representation of the approximations obtained together with the analytical solution

APPLICATION 3

Solve the ordinary differential equation by numerical methods and compare it with the analytical solution.

$$\frac{dy}{dx} = ysin(2x) - 20y, y(0) = 1$$

$$y(1) =? h = 0.025(17)$$

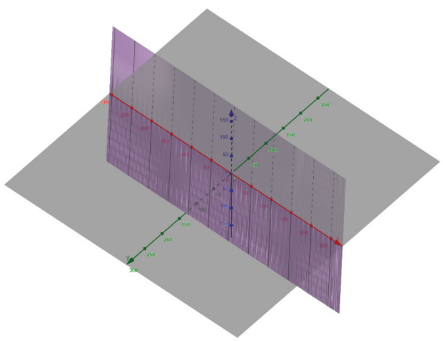

Figure 7: It shows the graph of equation (17).

Iteration	x	Euler	Improved Euler	Runge Kutta	Analytics
0	0	1	1	1	1
1	0.05	1.05	1.05125	1.05125	1.05254219
2	0.1	1,105	1.105125	1.10643906	1.11034184
3	0.15	1.16525	1.16425625	1.16576906	1.17366849
4	0.2	1.2310125	1.22890656	1.22945223	1.24280552
5	0.25	1.30256313	1.29935189	1.29771166	1.31805083
6	0.3	1.38019128	1.37588199	1.37078188	1.39971762
7	0.35	1.46420085	1.45880108	1.44890945	1.4881351
8	0.4	1.55491089	1.54842864	1.53235356	1.5836494
9	0.45	1.65265643	1.64510007	1.62138668	1.68662437
10	0.5	1.75778925	1.74916757	1.71629525	1.79744254
11	0.55	1.87067872	1.86100095	1.81738038	1.91650604
12	0.6	1.99171265	1.9809885	1.92495862	2.0442376
13	0.65	2.12129828	2.10953793	2.03936275	2.18108166
14	0.7	2.2598632	2.24707732	2.16094259	2.32750541
15	0.75	2.40785636	2.39405619	2.2900659	2.48400003
16	0.8	2.56574918	2.5509465	2.42711928	2.65108186
17	0.85	2.73403664	2.71824382	2.57250914	2.8292937
18	0.9	2.91323847	2.89646851	2.72666273	3.01920622
19	0.95	3.10390039	3.08616694	2.8900292	3.22141932
20	1	3.30659541	3.28791279	3.06308069	3.43656366

Table 1: The results obtained in each iteration of equation (13) are presented.

Iteration	x	Euler	Improved Euler	Runge Kutta	Analytics
0	0	1	1	1	1
1	0.05	1.05	1.05125	1.05125	1.0512492
2	0.1	1.10243439	1.10377805	1.10509208	1.10498683
3	0.15	1.15728073	1.15878958	1.16158312	1.16118163
4	0.2	1.21449501	1.21625025	1.22077187	1.21977856
5	0.25	1.27400931	1.2761024	1.28269766	1.28069636
6	0.3	1.33572949	1.33826264	1.34738917	1.34382524
7	0.35	1.39953304	1.40261959	1.41486316	1.40902476
8	0.4	1.4652672	1.4690318	1.48512319	1.47612195
9	0.45	1.53274722	1.53732589	1.55815836	1.54490981
10	0.5	1.60175511	1.60729505	1.63394209	1.6151463
11	0.55	1.67203873	1.67869789	1.71243102	1.68655372
12	0.6	1.74331143	1.75125787	1.7935639	1.75881885
13	0.65	1.81525228	1.82466316	1.87726066	1.8315936
14	0.7	1.88750693	1.89856728	1.96342167	1.90449653
15	0.75	1.95968917	1.97259028	2.05192704	1.9771151
16	0.8	2.03138331	2.04632083	2.14263623	2.04900865
17	0.85	2.10214723	2.11931895	2.23538785	2.11971237
18	0.9	2.17151632	2.19111968	2.32999966	2.18874191
19	0.95	2.23900813	2.26123743	2.42626886	2.25559885
20	1	2.30412779	2.32917115	2.52397272	2.31977682

Table 2: It presents the results obtained in each iteration.

Equation (18) provides the analytical solution of equation (17).

$$y(x) = \left(\frac{-3}{101}\right)\cos(2x) + \left(\frac{30}{101}\right)\sin(2x) + \left(\frac{104}{101}\right)e^{-20x}(18)$$

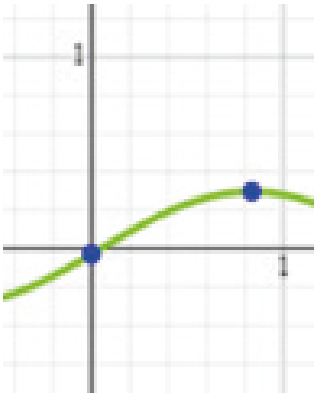

Figure 8: It shows the graph of equation (18).

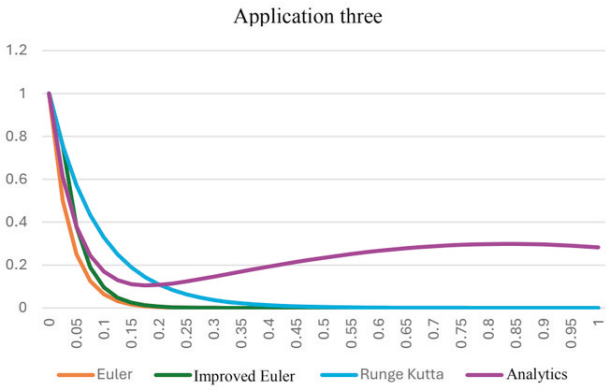

Figure 9: Graphically shows the approximations with the analytical solution

APPLICATION 4

Solve the following ordinary differential equation by the previous numerical methods and compare it with the analytical solution.

$$\frac{dy}{dx} + \frac{1}{2}y = \frac{3}{2}, y(0) = 4$$

$$y(1) = ? h = 0.025(20)$$

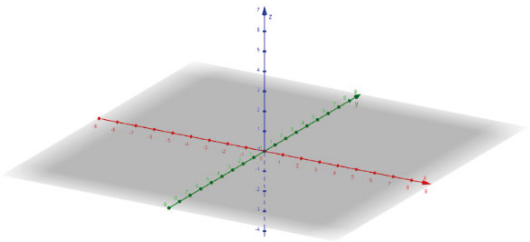

Figure 10: It shows the graph of equation (20)

The analytical solution of equation (20).

$$y(x) = 3 + e^{-\frac{x}{2}}(21)$$

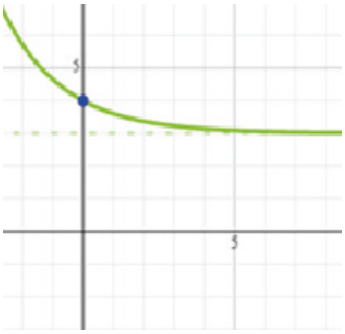

Figure 11: It shows the graph of equation (21)

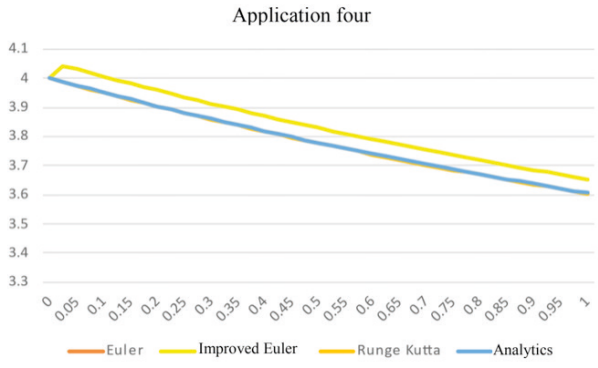

Figure 12: Graphically shows the approximations with the analytical solution (20).

APPLICATION 5

Solve the following ordinary differential equation by the previous numerical methods and compare it with the analytical solution.

$$\frac{dy}{dx} = tan(x), y(0) = 0\ y(0.5\pi) = ?$$

$$h = 0.0785398(22)$$

Iteration	x	Euler	Improved Euler	Runge Kutta	Analytics
0	0	4	4	4	4
1	0.025	3.9875	4.04359375	3.9875	3.9875778
2	0.05	3.97515625	4.03101758	3.97515625	3.97530991
3	0.075	3.9629668	4.01859861	3.9629668	3.96319442
4	0.1	3.95092971	4.00633488	3.95092971	3.95122942
5	0.125	3.93904309	3.99422444	3.93904309	3.93941306
6	0.15	3.92730505	3.98226538	3.92730505	3.92774349
7	0.175	3.91571374	3.97045582	3.91571374	3.91621887
8	0.2	3.90426732	3.95879387	3.90426732	3.90483742
9	0.225	3.89296398	3.9472777	3.89296398	3.89359735
10	0.25	3.88180193	3.93590547	3.88180193	3.8824969
11	0.275	3.8707794	3.92467541	3.8707794	3.87153435
12	0.3	3.85989466	3.91358571	3.85989466	3.86070798
13	0.325	3.84914598	3.90263464	3.84914598	3.85001609
14	0.35	3.83853165	3.89182046	3.83853165	3.83945702
15	0.375	3.82805001	3.88114145	3.82805001	3.82902912
16	0.4	3.81769938	3.87059594	3.81769938	3.81873075
17	0.425	3.80747814	3.86018224	3.80747814	3.80856032
18	0.45	3.79738466	3.84989871	3.79738466	3.79851622
19	0.475	3.78741735	3.83974372	3.78741735	3.78859689
20	0.5	3.77757464	3.82971568	3.77757464	3.77880078
21	0.525	3.76785495	3.81981298	3.76785495	3.76912636
22	0.55	3.75825677	3.81003407	3.75825677	3.75957212
23	0.575	3.74877856	3.80037739	3.74877856	3.75013657
24	0.6	3.73941883	3.79084143	3.73941883	3.74081822
25	0.625	3.73017609	3.78142466	3.73017609	3.73161563
26	0.65	3.72104889	3.7721256	3.72104889	3.72252735
27	0.675	3.71203578	3.76294278	3.71203578	3.71355197
28	0.7	3.70313533	3.75387475	3.70313533	3.70468809
29	0.725	3.69434614	3.74492006	3.69434614	3.69593431
30	0.75	3.68566681	3.73607731	3.68566681	3.68728928
31	0.775	3.67709598	3.72734509	3.67709598	3.67875163
32	0.8	3.66863228	3.71872203	3.66863228	3.67032005
33	0.825	3.66027437	3.71020675	3.66027437	3.6619932
34	0.85	3.65202094	3.70179792	3.65202094	3.65376979
35	0.875	3.64387068	3.6934942	3.64387068	3.64564853
36	0.9	3.6358223	3.68529427	3.6358223	3.63762815
37	0.925	3.62787452	3.67719684	3.62787452	3.62970741
38	0.95	3.62002609	3.66920063	3.62002609	3.62188506
39	0.975	3.61227576	3.66130437	3.61227576	3.61415988
40	1	3.60462232	3.65350682	3.60462232	3.60653066

Table 3: It shows the results in each iteration using equation (17).

Iteration	x	Euler	Improved Euler	Runge Kutta	Analytics
0	0	4	4	4	4
1	0.025	3.9875	4.04359375	3.9875	3.9875778
2	0.05	3.97515625	4.03101758	3.97515625	3.97530991
3	0.075	3.9629668	4.01859861	3.9629668	3.96319442
4	0.1	3.95092971	4.00633488	3.95092971	3.95122942
5	0.125	3.93904309	3.99422444	3.93904309	3.93941306
6	0.15	3.92730505	3.98226538	3.92730505	3.92774349
7	0.175	3.91571374	3.97045582	3.91571374	3.91621887
8	0.2	3.90426732	3.95879387	3.90426732	3.90483742
9	0.225	3.89296398	3.9472777	3.89296398	3.89359735
10	0.25	3.88180193	3.93590547	3.88180193	3.8824969
11	0.275	3.8707794	3.92467541	3.8707794	3.87153435
12	0.3	3.85989466	3.91358571	3.85989466	3.86070798
13	0.325	3.84914598	3.90263464	3.84914598	3.85001609
14	0.35	3.83853165	3.89182046	3.83853165	3.83945702
15	0.375	3.82805001	3.88114145	3.82805001	3.82902912
16	0.4	3.81769938	3.87059594	3.81769938	3.81873075
17	0.425	3.80747814	3.86018224	3.80747814	3.80856032
18	0.45	3.79738466	3.84989871	3.79738466	3.79851622
19	0.475	3.78741735	3.83974372	3.78741735	3.78859689
20	0.5	3.77757464	3.82971568	3.77757464	3.77880078
21	0.525	3.76785495	3.81981298	3.76785495	3.76912636
22	0.55	3.75825677	3.81003407	3.75825677	3.75957212
23	0.575	3.74877856	3.80037739	3.74877856	3.75013657
24	0.6	3.73941883	3.79084143	3.73941883	3.74081822
25	0.625	3.73017609	3.78142466	3.73017609	3.73161563
26	0.65	3.72104889	3.7721256	3.72104889	3.72252735
27	0.675	3.71203578	3.76294278	3.71203578	3.71355197
28	0.7	3.70313533	3.75387475	3.70313533	3.70468809
29	0.725	3.69434614	3.74492006	3.69434614	3.69593431
30	0.75	3.68566681	3.73607731	3.68566681	3.68728928
31	0.775	3.67709598	3.72734509	3.67709598	3.67875163
32	0.8	3.66863228	3.71872203	3.66863228	3.67032005
33	0.825	3.66027437	3.71020675	3.66027437	3.6619932
34	0.85	3.65202094	3.70179792	3.65202094	3.65376979
35	0.875	3.64387068	3.6934942	3.64387068	3.64564853
36	0.9	3.6358223	3.68529427	3.6358223	3.63762815
37	0.925	3.62787452	3.67719684	3.62787452	3.62970741
38	0.95	3.62002609	3.66920063	3.62002609	3.62188506
39	0.975	3.61227576	3.66130437	3.61227576	3.61415988
40	1	3.60462232	3.65350682	3.60462232	3.60653066

Table 4: It shows the results in each iteration of equation (20).

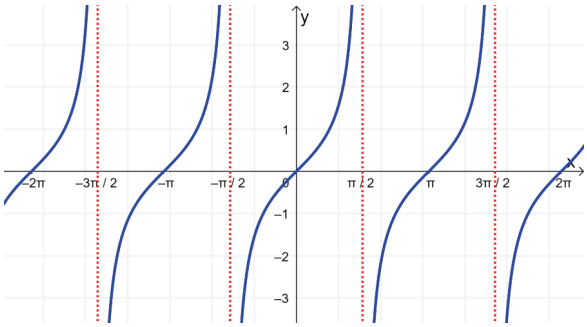

Figure 13: It shows the graph of equation (22) (https://www.neurochispas.com/wiki/grafica-de-la-tangente/#2-grafica-de-la-funcion-tangente-basica)

Analytical solution of equation (22)

$$y(x) = e^{\sec(x)} (23)$$

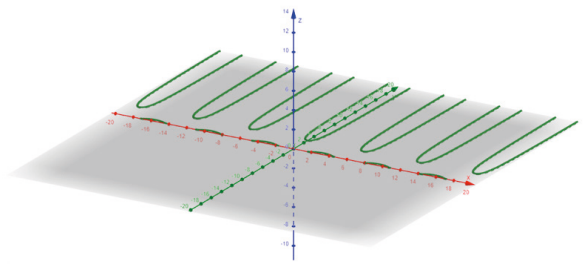

Figure 14: It shows the graph of the analytical solution of equation (23)

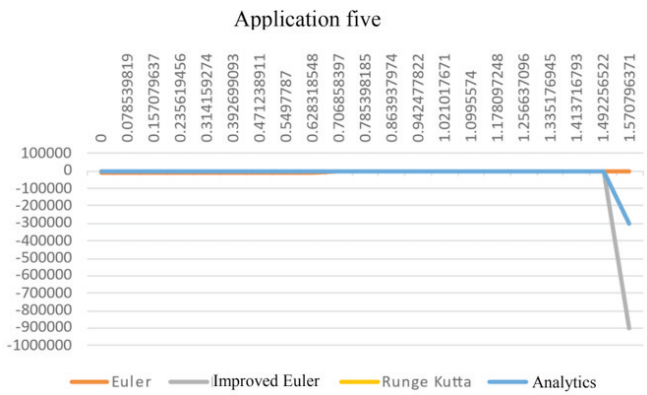

Figure 15: The graph shows the approximations obtained using numerical methods and the analytical solution (20).

In their article Zhang Lijuan and Guan Tiannye. (2018) use the local and global truncation errors which are shown in table 6.

method	Local truncation error	Global truncation error
Euler	EITHER () h^2	$0\ (h)$
Runge-Kuta second order improved Euler or Heun method	$0\ (h^3)$	EITHER () h^2
anoints-Kuta fourth order	$0\ (h^5)$	$0\ (h^4)$

Table 6: It shows the local and global truncation errors for different numerical methods used.

RESULTS AND ANALYSIS

It is important to mention that, if the step size is too large, the program will not be able to capture certain values where a significant alteration in the solution must be observed.

This is because the approximation grows too quickly, outpacing the computer's representation capacity. In this case, the program could indicate an "overflow" at the upper and lower extremes, or report that the value is outside the allowed range.

Causes of overflow

- Excessive step size: Too large a step size can cause the approximation to stray too far from the true value, generating significant numerical errors.
- Functions with rapid growth: If the function being evaluated has exponential or similar growth, a large step size may be insufficient to capture abrupt changes in the solution.
- Loss of precision: Overflow causes a significant loss of precision in the solution, making the results inaccurate or even useless.
- Erratic behavior: The program may exhibit erratic behavior, such as generating meaningless values or stopping execution prematurely.
- Failures in execution: In extreme cases, overflow can cause a complete failure of the program

Iteration	X	Euler	Improved Euler	Runge Kutta	Analytics
0	0	0	0	0	0
1	0.0785398	0	0.003091	0.003087	0.003087
2	0.1570796	0.0061812	0.012401	0.012388	0.012388
3	0.2356195	0.0186207	0.028049	0.028019	0.028019
4	0.3141593	0.0374764	0.050236	0.050182	0.050182
5	0.3926991	0.0629956	0.079262	0.079174	0.079174
6	0.4712389	0.0955278	0.115537	0.115404	0.115404
7	0.5497787	0.1355459	0.159611	0.159418	0.159418
8	0.6283185	0.1836751	0.212206	0.211935	0.211935
9	0.7068584	0.2407377	0.274277	0.273903	0.273903
10	0.7853982	0.307817	0.347087	0.346574	0.346574
11	0.863938	0.3863568	0.432336	0.431633	0.431632
12	0.9424778	0.4783151	0.532366	0.531394	0.531394
13	1.0210177	0.5864159	0.650499	0.649134	0.649133
14	1.0995574	0.7145813	0.791653	0.789681	0.789679
15	1.1780972	0.8687243	0.96353	0.96055	0.960547
16	1.2566371	1.0583363	1.179197	1.174367	1.174359
17	1.3351769	1.3000569	1.463628	1.454832	1.454808
18	1.4137168	1.6271989	1.875139	1.855233	1.855119
19	1.4922565	2.1230803	2.622052	2.546591	2.545178
20	1.5707964	3.1210234	-898387	-299459.5	-300000

Table 5: It shows the results in each iteration of equation (22).

RESULTS AND ANALYSIS

It is important to mention that if the step size is **too large, the program may not take certain values at which a significant alteration in the solution must be observed**. This is because the approximation grows too quickly, outpacing the computer's representation capacity. In this case, the program will indicate an "overflow" at the upper and lower extremes, or will report that the value is outside the allowed range.

CONCLUSIONS

The Runge-Kutta method is consolidated as a fundamental tool to address problems of ordinary differential equations with initial conditions, offering approximate solutions with high reliability.

Its ability to generate accurate and stable results makes it an attractive option for a wide range of applications. Application 5 in Figure 5 clearly exemplifies the superiority of the Runge-Kutta method. As the complexity of the problem increases, as seen in Figure 6, when the slopes grow rapidly, these methods provide erroneous approximations, in general these methods fail.

To further strengthen the approach, it is proposed to implement an additional constraint:append to the algorithm that evaluates two slopes $y'=f(t,y)$, given one has for a depending on the problem taking two consecutive iterations. If the difference between these slopes exceeds an established threshold, the algorithm must take corrective measures, such as reducing the step size or adjusting the approximation strategy. If the methods are increased too much, a good approximation will not be obtained. $y_1,y_2|f(t,y_2)\text{-}(t,y_1)|\leq\varepsilon\varepsilon>0)$

This constraint would allow the methods to adapt to situations where slopes experience drastic changes, guaranteeing the reliability and accuracy of the solutions even in complex scenarios.

REFERENCES

• Arora, G., Joshi, V., & Garki, I. S. (2020). Developments in Runge–Kutta Method to Solve Ordinary Differential Equations https://www.researchgate.net/publication/340027249_Developments_in_Runge-Kutta_Method_to_Solve_Ordinary_Differential_Equations.

• Hubbard, J. H., Habre, S. S., and West, B. (2001). The Convergence of and Euler Approximation of an Initial Value Problem Is Not Always. Mathematical Association of America, 108(4), 326-335.

• Kamruzzaman, Md., & Nath, M. C. (2018). A Comparative Study on Numerical Solution of Initial Value Problem by Using Euler's Method and Ruge-Kutta Methodo. Journal of Computer and Mathematical Sciences, 9, 493-500.

• Nurujjaman, Md. (2020). Enhanced Euler's Method to Solve First Order Ordinary Differential Equations with Better Accuracy. Journal of Engineering Mathematics & Statistics, 4(1), MANTECH Publicationes.

• Youssef, I. K., & El-Arabawy, H. A. (2007). Picard iteration algorithm combined with Gauss–Seidel technique for initial value problems. Applied Mathematics and Computation, 190(1), 345-355.

• Zhang, L., & Guan, T. (2018). Comparison of several Numerical Algorithms for Solving Ordinary Differential Equation Initial Value Problem. Advances in Computer Science Research, 78, ATLASNTIS PRESS.