CREAT. MATH. INFORM.
Volume **35** (2026), No. 3,
Pages 349-358

Online version at https://creative-mathematics.cunbm.utcluj.ro/
Print Edition: ISSN 1584 - 286X; Online Edition: ISSN 1843 - 441X
DOI: https://doi.org/10.37193/CMI.2026.03.03

A direct Euler's method for solving numerically a class of explicit second order ordinary differential equations

VASILE BERINDE

ABSTRACT. A second order ordinary differential equation is commonly solved numerically by transforming it into a system of two first order differential equations and then by integrating the system by a standard numerical method. An alternative is to solve it directly, without converting it to first-order systems. In this paper we introduce a direct Euler's method which for solving numerically second order explicit ordinary differential equations with first derivative missing:

$$y'' = f(x, y),\ x \in I \subset \mathbb{R}.$$

The new method involves less computations in comparison with other numerical methods for solving second order explicit ODEs.

1. INTRODUCTION

It is well known that many important problems in science and technology are modeled by means of ordinary or partial differential equations. Therefore, to solve the original practical problem we have to find the solution of the corresponding differential equation (s). For example, in the case of an ordinary differential equation with the unknown function denoted by $y(x)$, in a few cases we can solve it analytically, in the sense that we can find an explicit expression of $y(x)$ in terms of a finite number of elementary functions of variable x, only for few equations.

On the other hand, for most of the first order or second order ordinary differential equations appearing in physical problems we cannot apply analytical methods and hence the only alternative is to solve them *numerically*. For example, in studying mechanical systems without dissipation, it is important to solve the following particular class of second order differential equations

$$y'' = f(x, y),\ x \in I \subset \mathbb{R}, \tag{1.1}$$

where the first derivative y' is missing in the right-hand side.

The numerical solution of such a second-order ordinary differential equations is accomplished mainly by using one of the next two ways.

(1) The first approach consists in transforming the second-order ordinary differential equation into a system of two first order differential equations and then to integrate the system by a standard numerical method for first order differential equations. There are many such kind of numerical methods, e.g., Euler, Runge-Kutta, Milne, Adams-Bashforth etc. see for example [1, 5, 7, 8, 9, 10].
(2) The second approach is to apply a direct technique without converting the second-order ordinary differential equation into systems of first-order differential equations. To this category belong the methods of Nyström, see for example [6], which

Received: 4.10.2024. In revised form: 2.02.2025. Accepted: 1.12.2025
2020 *Mathematics Subject Classification.* 65L05, 65L07, 65L10, 65L12.
Key words and phrases. *explicit second order ordinary differential equation, initial value problem, boundary value problem, Euler's method, Runge-Kutta, ode45, numerical method, Matlab.*

have been used to numerically approximate the solution to many initial value problems for second-order ordinary differential equations.

In this paper we introduce a third approach for solving second order explicit ordinary differential equations of the form (1.1), i.e., with the first derivative missing, which were called "special" in Henrici [6], Chapter 6. This is in fact a direct Euler's numerical method.

The new method that we shall present in this note involves less computations in comparison with other similar numerical methods used for solving second order ODEs.

2. PRELIMINARIES

For the scope of the present paper, we start by reminding two of the main approaches commonly used for solving numerically differential equations and systems of differential equations, see for example Atkinson, Han and Stewart [1], Butcher [5], Henrici [6], Pearson [8] and Shampine [9]. These are based on

(1) *numerical derivation*, when the derivative (s) involved in the differential equations are approximated by a difference scheme;
(2) *numerical integration*, when the differential equation is first converted equivalently into an integral equation and then, in order to solve it, a certain interpolation scheme is used to approximate the integral (s) involved in the integral equations.

From the first class of methods, we illustrate the forward Euler's method for the case of an IVP associated to first order ordinary differential equations. Let us consider the initial value problem (IVP) for the first order explicit differential equation:

$$\begin{cases} y' = f(x, y(x)),\ x \in I = [x_0, b] \subset \mathbb{R}, \\ y(x_0) = y_0. \end{cases} \tag{2.2}$$

Numerical methods for solving (2.2) are commonly designed to find an approximate solution $y(x)$ at a discrete set of nodes in I

$$a = x_0 < x_1 < x_2 < \cdots < x_n = b.$$

Usually, one considers the nodes to be equidistant, i.e.,

$$x_k = x_0 + kh,\ k = 0, 1, 2, \ldots, n,$$

where $h = \dfrac{b-a}{n}$. Denote

$$y_k = y(x_k),\ k = 0, 1, 2, \ldots, n.$$

The *forward Euler's method* for solving (2.2) is obtained by using the *forward difference approximation* to the derivative y' at $\overline{x}$:

$$y'(\overline{x}) \approx \frac{1}{h}\left[y(\overline{x} + h) - y(\overline{x})\right]. \tag{2.3}$$

Now, by inserting (2.3) in (2.2) for $\overline{x} := x_k$ we obtain

$$\frac{1}{h}\left[y(x_k + h) - y(x_k)\right] \approx f(x_k, y(x_k))$$

which defines the forward Euler's method:

$$y_{k+1} = y_k + hf(x_k, y_k),\ k = 0, 1, 2, \ldots, n-1. \tag{2.4}$$

Hence, by using (2.4) and starting from y_0, which is known, the forward Euler's method allows us to compute iteratively $y_1, y_2, \ldots, y_n$.

This is a typical *numerical derivation method* and has a very nice geometric interpretation, see Figure 2.

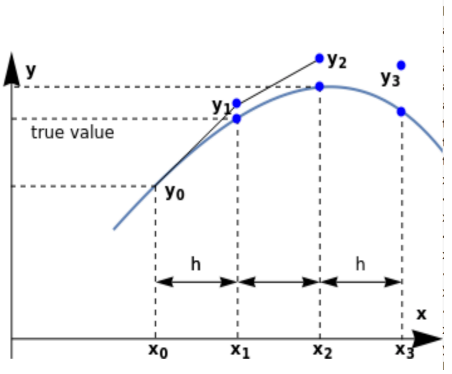

FIGURE 1. Geometric interpretation of forward Euler's method

The *numerical integration* counterpart of the forward Euler's method is obtained by integrating the equation

$$y'(t) = f(t, y(t))$$

on the subinterval $[x_k, x_{k+1}]$ to get

$$y(x_{k+1}) - y(x_k) = \int_{x_k}^{x_{k+1}} f(t, y(t)),$$

thus obtaining

$$y_{k+1} = y_k + \int_{x_k}^{x_{k+1}} f(t, y(t)), \tag{2.5}$$

which generates various numerical integration schemes by replacing the function $f(t, y(t))$, which is known, by an interpolating polynomial having the values

$$f_k = f(x_k, y_k)$$

on the set of nodes x_k, where y_k is either known (has already been computed) or is just to be computed.

Other important methods, like the Picard iteration or the method based on Taylor series could also be used in several contexts for solving first order differential equations, see Atkinson, Han and Stewart [1], Butcher [5], Henrici [6], Pearson [8] and Shampine [9].

3. A DIRECT EULER'S METHOD FOR A CLASS OF SECOND ORDER DIFFERENTIAL EQUATIONS

To open the way in defining our direct Euler's method, we first remind how one constructs the Euler's method for an initial value problem for a second order differential equation.

Consider the following initial value problem

$$\begin{cases} y''(x) = f(x, y(x), y'(x)),\ x \in I \subset \mathbb{R}, \\ y(x_0) = y_0 \\ y'(x_0) = \overline{y}_0 \end{cases} \tag{3.6}$$

where f, $x_0 \in I$, $y_0, \overline{y}_0 \in \mathbb{R}$ are known and $y \in C^2(I)$ is the unknown function.

As we mentioned before, this problem is commonly solved numerically by reducing the second order equation to the system of first order differential equations

$$\begin{cases} y'(x) = z(x) \\ z'(x) = f(x, y(x), z(x)) \end{cases} \tag{3.7}$$

and then applying to each equation a certain numerical method, like the forward Euler's method (2.4).

Our aim in this section is to show that one can solve (3.6) directly, without reducing it to a system of equations.

To this end, we shall use an Euler's type method, which is obtained in the following way.

We considering the second order *difference approximation* of the second derivative y'' at $\overline{x}$, that is, we use the approximation scheme

$$y''(\overline{x}) \approx \frac{y(\overline{x}+h) - 2y(\overline{x}) + y(\overline{x}-h)}{h^2},$$

which, for $\overline{x} = x_k$, yields

$$y''(x_k) \approx \frac{y(x_{k+1}) - 2y(x_k) + y(x_{k-1})}{h^2}, k = 1, 2, ..., n-1.$$

Thus, similarly to the way described above for first oder differential equations, we obtain the following 2-step Euler's type scheme for solving the IVP (3.6):

$$y(x_{k+1}) - 2y(x_k) + y(x_{k-1}) = h^2 f(x_k, y(x_k), y'(x_k)),\ k = 1, 2, ..., n-1 \tag{3.8}$$

where $y(x_0) = y_0$ and $y'(x_0) = \overline{y}_0$ are known.

If we use the notations above and also denote $f_k := f(x_k, y(x_k), y'(x_k))$, then the scheme (3.8) becomes

$$y_{k+1} - 2y_k + y_{k-1} = h^2 f_k,\ k = 1, 2, ..., n-1, \tag{3.9}$$

which is in fact a diagonal system of $n-1$ linear equations with $n-1$ unknowns $y_1, y_2, \ldots, y_n$.

This system can be solved by specific methods which take advantage of the particular form of its matrix:

$$D = \begin{bmatrix} -2 & 1 & 0 & 0 & ...0 & 0 & 0 \\ 1 & -2 & 1 & 0 & ...0 & 0 & 0 \\ . & . & . & . & ... & . & . \\ 0 & 0 & 0 & 0 & ...1 & -2 & 1 \\ 0 & 0 & 0 & 0 & ...0 & 1 & -2 \end{bmatrix}$$

and of the fact that $det(D) \neq 0$.

A particular case of the approximation scheme (3.9) is obtained when no derivatives appear in the right-hand side of , i.e., the equation is of the following form

$$y''(x) = f(x, y(x)),\ x \in I \subset \mathbb{R}, \tag{3.10}$$

Equations of this type were called *special* in Henrici [6] and occur frequently in the study of mechanical systems without dissipation.

By integrating (3.10) twice on $[x_k, x_{k+1}]$ we obtain

$$y(x_{k+1}) - y(x_k) = hy'(x_k) + \int_{x_k}^{x_{k+1}} f(x_k + h - t) f(t, y(t)) dt \tag{3.11}$$

As (3.11) involves the first derivative, $y'(x_k)$, we can similarly integrate (3.10) on $[x_{k-1}, x_k]$ (which in fact means that we replace h by $-h$ in (3.11)), to get

$$y(x_{k-1}) - y(x_k) = -hy'(x_k) + \int_{x_k}^{x_{k-1}} (x_k - h - t) f(t, y(t)) dt. \tag{3.12}$$

Now, by adding (3.11) and (3.12) we obtain

$$y(x_{k+1}) - 2y(x_k) + y(x_{k-1}) = \int_{x_k}^{x_{k+1}} (x_k + h - t) f(t, y(t)) dt + \int_{x_k}^{x_{k-1}} f(x_k - h - t) f(t, y(t)) dt. \tag{3.13}$$

The sum of the two integrals in the right-hand side can be transformed to

$$\int_{x_k}^{x_{k+1}} (x_k + h - t) f(t, y(t)) dt + \int_{x_k}^{x_{k-1}} (x_k - h - t) f(t, y(t)) dt =$$

$$\int_{x_k}^{x_{k+1}} (x_k + h - t)[f(t, y(t)) + f(2x_k - k, y(2x_k - k))] dt$$

and thus by (3.12) we obtain the 2-step numerical scheme

$$y(x_{k+1}) - 2y(x_k) + y(x_{k-1}) = \int_{x_k}^{x_{k+1}} (x_k + h - t)[f(t, y(t)) + f(2x_k - k, y(2x_k - k))] dt, \tag{3.14}$$

whose left-hand side coincides with that of the numerical derivation scheme (3.8).

Remark 3.1. *Replacing* $F(t) = (x_k + h - t)[f(t, y(t)) + f(2x_k - k, y(2x_k - k))]$ *by an interpolation polynomial of degree* p *on the points* $x_k, x_{k-1}, \ldots, x_p$, *by* (3.14) *we can obtain various* numerical integration *schemes.*

But our aim in this paper is to derive a *numerical derivation* scheme, starting from the two point method (3.8) in the case the first derivative is missing, that is,

$$y(x_{k+1}) - 2y(x_k) + y(x_{k-1}) = h^2 f(x_k, y(x_k)),\ k = 1, 2, ..., n-1 \tag{3.15}$$

Having in mind the initial conditions in the IVP (3.6), i.e.,

$$\begin{cases} y(x_0) = y_0 \\ y'(x_0) = \overline{y}_0 \end{cases} \tag{3.16}$$

we can compute y_1 by using the idea in the Euler's method.

Since we know $y'(x_0)$, we can write the equation of the tangent to the graph of $y(x)$ at the point (x_0, y_0):

$$y - y_0 = y'(x_0) \cdot (x - x_0) \Leftrightarrow y - y_0 = \overline{y_0} \cdot (x - x_0)$$

which we intersect to the line $x = x_1$ and thus we obtain, see Figure 2:

$$y_1 = y_0 + \overline{y_0} \cdot (x_1 - x_0) = y_0 + h\overline{y_0},\ \text{as } x_1 - x_0 = h. \tag{3.17}$$

Now, with y_0 known and y_1 already computed by using (3.17), we get from the first equation in the system (3.9)

$$y_2 - 2y_1 + y_0 = h^2 f_1,$$

from which we obtain y_2, and then with y_1 and y_2 thus computed, from the second equation in the system (3.9) we obtain y_3,..., from the last equation in the system (3.9) we obtain y_n, and thus we are done.

Remark 3.2. *We note that our scheme has only two moments where we introduce errors: first, when we approximate the second derivative, and secondly, when we compute y_1, which is not the exact value of $y(x_1)$. Therefore, the error analysis of this scheme is quite easy to perform.*

We end this section by stressing on the fact that it is possible to improve the accuracy when computing $y(x_1)$ by the Euler's scheme (3.17) described above, which is actually deduced by truncating the Taylor's series.

Instead we shall use the non truncated Taylor's series in the form to compute y_1:

$$y_1 = y_0 + h\overline{y_0} + h^2 f(x_0, y_0). \tag{3.18}$$

The truncation error in the approximation scheme (3.17) is just the additional term $h^2 f(x_0, y_0)$ in (3.18).

4. Applying the direct Euler's scheme to two point boundary value problems

Finally, let us note that a similar scheme to the one presented for IVPs cannot be constructed completely for the two point boundary value problem

$$\begin{cases} y''(x) = f(x, y(x)),\ x \in [a, b], \\ y(a) = A \\ y(b) = B, \end{cases} \tag{4.19}$$

because in this case we do not have information on the first order derivative y' at $x_0 = a$.

Instead, one can try a kind of shooting method, by taking an arbitrary value

$$y'(a) = m$$

and then proceeding similarly. Let us consider the same set of equidistant nodes in I

$$a = x_0 < x_1 < x_2 < \cdots < x_n = b,$$

that is,

$$x_k = x_0 + kh,\ k = 0, 1, 2, \ldots, n,$$

with $h = \dfrac{b-a}{n}$. We denote

$$y_k = y(x_k),\ k = 0, 1, 2, \ldots, n$$

and use the second order *difference approximation* for the second derivative y'' at $\overline{x}$, that is,

$$y''(\overline{x}) \approx \frac{y(\overline{x}+h) - 2y(\overline{x}) + y(\overline{x}-h)}{h^2}.$$

For $\overline{x} = x_k$, we get

$$y''(x_k) \approx \frac{y(x_{k+1}) - 2y(x_k) + y(x_{k-1})}{h^2}, k = 1, 2, ..., n-1$$

and thus one obtains the scheme

$$y(x_{k+1}) - 2y(x_k) + y(x_{k-1}) = h^2 f(x_k, y(x_k)),\ k = 1, 2, ..., n-1.$$

where we know only $y_0 = y(a) = A$. Using the notations above, we have:

$$y_{k+1} - 2y_k + y_{k-1} = h^2 f(x_k, y_k),\ k = 1, 2, ..., n-1. \tag{4.20}$$

In order to obtain y_1, we use a "shooting type" method: $y'(x_0) = y'(a) = m$. We write the equation of the tangent to the graph of $y(x)$ at the point (a, A):

$$y - y(a) = m \cdot (x - a) \Leftrightarrow y - A = \overline{y_0} \cdot (x - a)$$

which we intersect to the line $x = x_1$ and thus one obtains:

$$y_1 = A + m \cdot (x_1 - a) = A + hm, \text{ as } x_1 - x_0 = h.$$

Now, with y_0 known and y_1 computed in this way, from the first equation in the system (3.9), we get

$$y_2 - 2y_1 + y_0 = h^2 f_1,$$

from which we obtain y_2, and then with y_1 and y_2 thus computed, from the second in the system (4.20) we obtain y_3,..., from the last equation in the system (4.20) we obtain pe y_n.

So, one can repeat the method above until we get a reasonable small value for $|y_n - B|$.

Remark 4.3. *The accuracy of the method* (4.20) *will depend - as in the case of the classical shooting method - on how close is the computed value y_n to the prescribed value $B = y(b)$.*

5. Numerical examples

We illustrate our new method by means of the following examples of IVPs for some special second order differential equations. The differential equations in Examples 5.1 and 5.2 are taken from Pearson [8], see Exercise 10.34 and Exercise 10.35, pages 482-483, where they are associated to some two point boundary value conditions.

Example 5.1. *Solve the equation $y'' = x + y$ with the initial conditions $y(0) = 0$, $y'(0) = \dfrac{4e}{e^2 - 1} - 1$ on the interval $[0, 5]$.*

Solution. We considered this case because we know the exact analytical solution,

$$y(x) = \frac{\sinh(x)}{\sinh(1)} - 1,$$

and so we can compare it to the numerical solutions obtained by the proposed scheme.

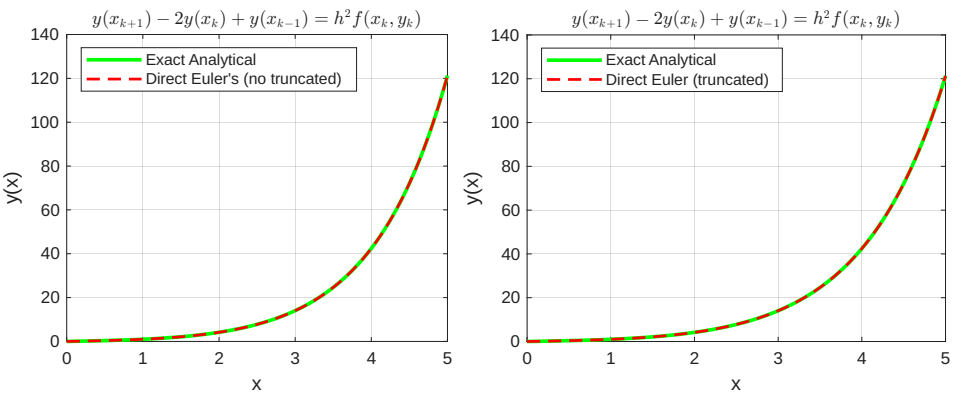

FIGURE 2. Graph of exact solution *versus* approximate solution obtained by direct Euler's scheme (not truncated, left and truncated, right)

Because the analytical solution is dominated by the hyper-exponential function

$$\sinh(x) \approx \frac{1}{2} e^x,$$

for values of x approaching, e.g., $x = 50$, the value of $y(x)$ will escalate to an "astronomical" value 4.4117×10^{21}.

This is why, for our numerical tests, we worked on the "sufficiently small" interval $[0, 5]$, with $h = 0.01$ and $n = 500$.

We used Matlab to compute and graph the approximate solutions obtained by the proposed direct Euler's scheme, first by using (3.17) and then (3.18) to compute y_1.

As it can be seen from the Figure 2, for the equation in Example 5.1, our method works well as there is no significant difference between the approximate solution, $y(5) = 121.277505$ (in both cases, truncated and not truncated) and the analytical exact solution, $y(5) = 121.281714$, with the absolute error 0.004209 and relative error 3.4704×10^{-6}.

Example 5.2. *Solve the Duffing type equation without damping or external forcing* $y'' = y^3 - y$ *with the initial conditions* $y(0) = 0$, $y'(0) = \dfrac{1}{\sqrt{2}}$.

Solution. In this case the analytical solution is also known, $y(x) = \tanh\left(\dfrac{x}{\sqrt{2}}\right)$. In contrast to Example 1, because the solution is a hyperbolic tangent function, the trajectory starts at 0 and smoothly asymptotically approaches the horizontal line 1 as $x \to \infty$.

In this case $f(x, y) = y^4 - y$ and because the partial derivative $\dfrac{\partial f}{\partial y} = 3y^2 - 1$ approaches 2 as $y \to 1$, in order to maintain numerical stability and avoid amplification of truncation errors, the step size should satisfy the condition $h < \sqrt{2}$.

We used Matlab with $h = 0.01$ and $n = 500$ to compute and graph the approximate solutions obtained by the proposed direct Euler's scheme, first by using (3.17) (truncated) and then by using (3.18) (not truncated) to compute y_1.

Similar to the case of the equation in Example 5.1, there is no significant difference between the approximate solution ($y(5) = 0.998919$, in both cases) and the analytical exact solution ($y(5) = 0.998303$), as it can be seen from the graphs.

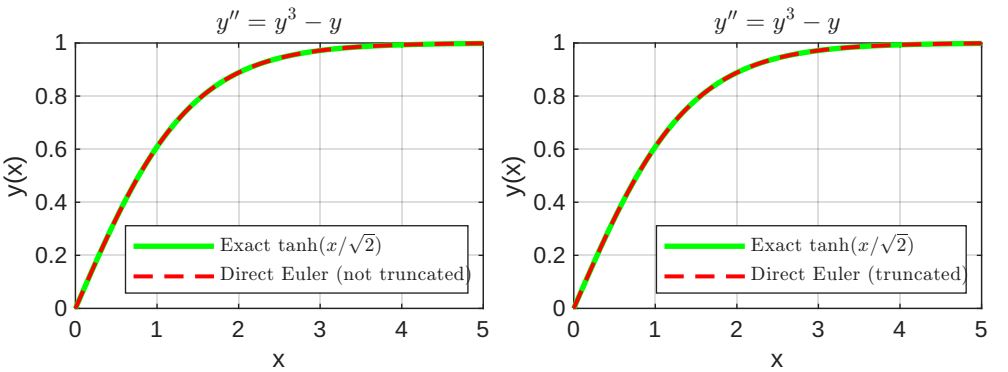

FIGURE 3. Graph of analytical solution *versus* approximate solution obtained by direct Euler's scheme (not truncated, left and truncated, right)

Remark 5.4. *For the differential equations considered in Examples 5.1 and 5.2 we were able to compare the numerical solutions obtained by our direct Euler's method with the corresponding analytical solutions.*

In order to illustrate the power of our method, in the next example one considers a differential equation to which we do not know its analytical solution. In that case, we shall compare the

numerical solution obtained by our direct Euler's method to the approximate solution obtained by the standard Matlab solver ode45.

Example 5.3. *Solve the equation* $y'' = y^7 - y^2$ *with the initial conditions* $y(0) = 0$, $y'(0) = \sqrt{2}$.

Solution.

Because of the presence of the highly aggressive polynomial growth term y^7 in the expression of $f(x, y)$, the differential equation $y'' = y^7 - y^2$ exhibits a **finite-time blow-up** (singularity). Due to the initial slope $y'(0) = \sqrt{2}$, the solution increases dramatically and shoots toward $+\infty$ near the point $\overline{x} \approx 1.261$. Therefore, in order to capture accurately the behavior of the solution before numeric overflow, we should work on the safe evaluation window $[0, 1.2]$ with a very fine step size.

The numerical experiments presented in Figure 4, were obtained usingd Matlab.

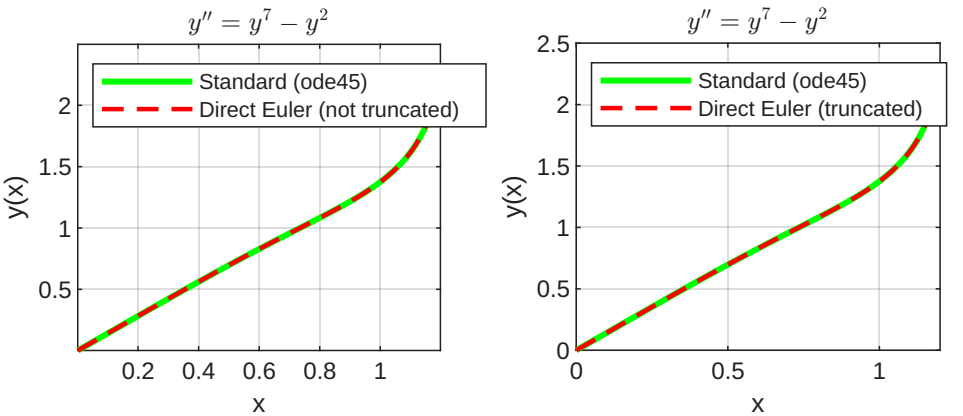

FIGURE 4. Graph of standard approximate solution (ode45) *versus* approximate solution obtained by direct Euler's scheme

The numerical experiments show that there is no significant difference between the approximate solutions obtained by the direct Euler's scheme and the standard solver ode45.

6. CONCLUSIONS

(1) Our main aim in this note was to introduce a direct Euler's method for solving numerically IVPs associated to the class of second order differential equations of the form

$$y'' = f(x, y),\ x \in I \subset \mathbb{R}, \tag{6.21}$$

which are called "special" differential equations in Henrici [6], Chapter 6.

(2) Commonly, such kind of problems, with f of the general form $f(x, y, y')$, are solved indirectly, by transforming the second order differential equations (6.21) into a system of two first order differential equations, and then apply a standard numerical solver, e.g., ode45, to the equivalent system (3.7).

(3) Our approach is based on the fact that, due to the absence of the first derivative in the right-hand term of the differential eqiuation (6.21), and knowing $y_0 = y(x_0)$ and $\overline{y_0} = y'(x_0)$, we are able to compute $y_1 = y(x_1)$ similarly to classical Euler's scheme and then to proceed with the complete iterative solution of the second order difference scheme

$$y_{k+1} - 2y_k + y_{k-1} = h^2 f(x_k, y_k),\ k = 1, 2, ..., n-1.$$

(4) We presented in Section 3 all details of our new numerical method, by computing y_1 in two different ways, that is, by means of the truncated formula

$$y_1 = y_0 + h\overline{y_0}$$

and also by using the Taylor's expansion (not truncated) formula

$$y_1 = y_0 + h\overline{y_0} + h^2 f(x_0, y_0).$$

(5) In order to illustrate the value of the new method, in Section 5 we reported some of the numerical experiments performed in Matlab for the differential equations in Examples 5.1-5.3.

(6) For the differential equations in Examples 5.1 and 5.2, whose analytical solutions are known, we compared the approximate solutions obtained by the new numerical schemes with the exact analytical solution. In all cases, the approximation is very good, with the absolute error 0.004209 and relative error 3.4704×10^{-6} in the case of Example 5.1.

(7) In the case of Example 5.3, because the analytical solution cannot be computed, we compared the numerical solutions obtained by our direct Euler's method to the approximate solution obtained by the standard Matlab solver ode45. There are no significant differences between the two numerical methods.

(8) All the above conclusions indicate that, for solving second order differential equations of the form (6.21), we can use the direct Euler's method presented here, which is easily implementable and does not require a software algebra package.

(9) We note that the computational complexity of the numerical methods presented in this paper to solve the entire trajectory over a grid of size N is $O(N)$ (linear time).

(10) We end this paper by suggesting readers to apply the direct Euler's scheme for solving two point boundary value problems, as described in Section 4.

REFERENCES

[1] Atkinson, K. E.; Han, W.; Stewart, D. *Numerical solution of ordinary differential equations*. Pure and Applied Mathematics (Hoboken). John Wiley & Sons, Inc., Hoboken, NJ, 2009.
[2] Berinde, V.; Petracovici, B. *Ecuaţii diferenţiale / Differential equations*, Universitatea din Baia Mare, 1992
[3] Berinde, V.; Horvat-Marc, A., *Ecuaţii diferenţiale şi cu derivate parţiale / Ordinary and Partial Differential Equations*, Editura CUB PRESS 22, Baia Mare, 2006.
[4] Berinde, V.; Horvat-Marc, A., Chira M. *Ecuaţii diferenţiale şi cu derivate parţiale / Ordinary and Partial Differential Equations*, Second Edition, Editura CUB PRESS 22, Baia Mare, 2016.
[5] Butcher, J. C. *Numerical methods for ordinary differential equations*. Third edition. With a foreword by J. M. Sanz-Serna. John Wiley & Sons, Ltd., Chichester, 2016.
[6] Henrici, P. *Discrete variable methods in ordinary differential equations*. John Wiley & Sons, Inc., New York-London, 1962.
[7] Jain, M. K. *Numerical solution of differential equations*. Second edition. A Halsted Press Book. John Wiley & Sons, Inc., New York, 1984.
[8] Pearson, C. E. *Numerical methods in engineering and science*. Van Nostrand Reinhold Co., New York, 1986.
[9] Shampine, L F. *Numerical solution of ordinary differential equations*. Chapman & Hall, New York, 1994.
[10] Shampine, L. F.; Gladwell, I.; Thompson, S. *Solving ODEs with MATLAB*. Cambridge University Press, Cambridge, 2003.

DEPARTMENT OF MATHEMATICS AND COMPUTER SCIENCE
NORTH UNIVERSITY CENTRE AT BAIA MARE
TECHNICAL UNIVERSITY OF CLUJ-NAPOCA
VICTORIEI 76, 430072 BAIA MARE ROMANIA
Email address: vasile.berinde@mi.utcluj.ro

ACADEMY OF ROMANIAN SCIENTISTS, 3 ILFOV, 050044, BUCHAREST, ROMANIA
Email address: vasile.berinde@gmail.com