Numerical methods for solving initial value problems on the Infinity Computer

YA.D. SERGEYEV[1,2*], M.S. MUKHAMETZHANOV[1,2], F. MAZZIA[3], F. IAVERNARO[3], P. AMODIO[3]

[1] *DIMES, Università della Calabria, Italy*
[2] *Department of Software and Supercomputing Tecnologies, Lobachevsky State University of Nizhni Novgorod, Russia*
[3] *Dipartimento di Matematica, Università degli studi di Bari, Italy*

Received ...; In final form ...

New algorithms for the numerical solution of Ordinary Differential Equations (ODEs) with initial condition are proposed. They are designed for work on a new kind of a supercomputer – the Infinity Computer, – that is able to deal numerically with finite, infinite and infinitesimal numbers. Due to this fact, the Infinity Computer allows one to calculate the exact derivatives of functions using infinitesimal values of the stepsize. As a consequence, the new methods described in this paper are able to work with the exact values of the derivatives, instead of their approximations.

Key words: Ordinary differential equations, initial value problem, numerical infinitesimals, Infinity Computer.

1 INTRODUCTION

Numerical solutions to Ordinary Differential Equations (ODEs) are required very often in practical problems (see, e.g., [5–7, 13–15, 21]). In this paper,

* email: `yaro@dimes.unical.it`

we introduce numerical methods for solving ODEs on the Infinity Computer (see [29, 31, 32, 35]). We consider the following Initial Value Problem (IVP)

$$y'(x) = f(x, y), \;\; y(x_0) = y_0, \;\; x_0 = a, \;\; x \in [a, b]. \tag{1}$$

and we suppose that $f(x, y)$ is a "black-box" function, i.e., $f(x, y)$ is given by a computer procedure and the analytical representation of $f(x, y)$ is unknown to the person who solves (1).

There exists a huge number of numerical methods proposed to solve ODEs on conventional computers (see, e.g., [5–7, 13–16, 21, 27]). The simplest algorithm to solve the problem (1) is probably the explicit Euler Method (see, e.g., [6]). At each step it constructs a linear approximation of $y(x)$ starting from the initial point $(x_0, y(x_0))$. The $(n + 1)$th step of the Euler algorithm describes how to move from the point x_n to $x_{n+1} = x_n + h, n \geq 0$, and is executed as follows

$$y_{n+1} = y_n + hf(x_n, y_n). \tag{2}$$

The Infinity Computer is based on a new numeral system described in [29, 31, 32, 37, 41, 46] for performing computations with infinite and infinitesimal quantities. This allows one to calculate the exact values of the derivatives numerically without finding the respective derivatives analytically and to work with infinitesimal steps of integration in (2). The first attempts to use the Infinity Computer in this direction has been done in [38, 42, 44].

In order to see the place of the new approach in the historical panorama of ideas dealing with infinite and infinitesimal, see [20, 22–24, 28, 34, 36, 48]. In particular, connections of the new approach with bijections are studied in [24] and metamathematical investigations on the theory and its non-contradictory can be found in [23]. The new methodology has been successfully used in several fields. We can mention numerical differentiation and optimization (see [8, 38, 52]), models for percolation and biological processes (see [17, 18, 40, 50]), hyperbolic geometry (see [25, 26]), fractals (see [17, 18, 30, 33, 40, 47]), infinite series (see [19, 34, 39, 51]), the first Hilbert problem, lexicographic ordering, and Turing machines (see [36, 43, 45, 48, 49]), cellular automata (see [9–11]), etc.

The numeral system proposed in [32, 37, 41] is based on an infinite unit of measure expressed by the numeral ① called *grossone* and introduced as the number of elements of the set $\mathbb{N}$ of natural numbers (a clear difference with non-standard analysis can be seen immediately since non-standard infinite numbers are not connected to concrete infinite sets and do not belong to $\mathbb{N}$). Other symbols dealing with infinities and infinitesimals (∞, Cantor's ω,

$\aleph_0, \aleph_1, ...,$ etc.) are not used together with ①. Similarly, when the positional numeral system and the numeral 0 expressing zero had been introduced, symbols V, X, and other symbols from the Roman numeral system had not been involved.

The numeral ① allows one to construct different numerals expressing different infinities and infinitesimals and to execute numerical computations with all of them. As a result, in occasions requiring infinities and infinitesimals indeterminate forms and various kind of divergence are not present when one works with any (finite, infinite, or infinitesimal) numbers expressible in the new numeral system and it becomes possible to execute arithmetical operations with a variety of different infinities and infinitesimals. For example, for ① and $①^{4.5}$ (that are examples of infinities) and $①^{-1}$ and $①^{-4.5}$ (that are examples of infinitesimals) it follows

$$\begin{gathered}
0 \cdot ① = ① \cdot 0 = 0, \quad ① - ① = 0, \quad \frac{①}{①} = 1, \quad ①^0 = 1, \quad 1^{①} = 1, \quad 0^{①} = 0, \\
0 \cdot ①^{-1} = ①^{-1} \cdot 0 = 0, \qquad ①^{4.5} > ①^1 > 1 > ①^{-1} > ①^{-4.5} > 0, \\
①^{-1} - ①^{-1} = 0, \qquad \frac{①^{-1}}{①^{-1}} = 1, \quad \frac{6 + ①^{-4.5}}{①^{-4.5}} = 6①^{4.5} + 1, \quad (①^{-1})^0 = 1, \\
① \cdot ①^{-1} = 1, \quad ① \cdot ①^{-4.5} = ①^{-3.5}, \quad \frac{①^{4.5} + 61①}{①} = ①^{3.5} + 61, \\
\frac{①^{4.5}}{①^{-4.5}} = ①^9, \quad (①^{4.5})^0 = 1, \quad ①^{4.5} \cdot ①^{-1} = ①^{3.5}, \quad ①^{4.5} \cdot ①^{-4.5} = 1.
\end{gathered} \tag{3}$$

It can be seen in (3) that $①^0 = 1$, therefore, a finite number a can be represented in the new numeral system simply as $a①^0 = a$, where the numeral a itself can be written down by any convenient numeral system used to express finite numbers. The simplest infinitesimal numbers are represented by numerals having only negative finite powers of ① (e.g., $40.17①^{-13.26}+87.32①^{-25.7}$, see also examples above). Notice that all infinitesimals are not equal to zero. In particular, $\frac{1}{①} > 0$ because it is a result of division of two positive numbers.

In Section 2, methods using infinitesimals are described briefly. In particular, Section 2.1 presents the main idea regarding the usage of infinitesimals for approximating derivative and Section 2.2 recall the Taylor methods solving ODEs and the core Method 1.0 from [42]. Sections 2.3–2.5 present new methods evolving the Method 1.0. Section 2.6 contains detailed results of numerical experiments in a compact form presenting a comparison with the Runge-Kutta methods on a class of test functions taken from the literature.

Optimal in certain sense parameters for the new methods are calculated in the Appendix, which also contains a description of test problems used in the numerical experiments. Finally, Section 3 concludes the paper.

2 METHODS USING NUMERICAL INFINITESIMALS FOR SOLVING ODES

2.1 Approximation of the derivative

Many numerical methods for solving ODEs require the computation of the derivative of the unknown function $y(x)$ at some specific points. In particular, this is the case with methods based on Taylor expansion.

Let us denote by $y_i^{(j)}$ an estimate of the j-th derivative of the solution $y(x)$ at the point x_i. It has been shown in [42] that in order to calculate the j-th derivative at the point x_i j infinitesimals steps from the point x_i using the Euler formula with $h = ①^{-1}$ should be executed as follows

$$y_{i1} = y_i + ①^{-1} f(x_i, y_i), \;\; y_{i2} = y_{i1} + ①^{-1} f(x_i + ①^{-1}, y_{i1}), \;\ldots$$

$$y_{ik} = y_{ik-1} + ①^{-1} f(x_i + (k-1)①^{-1}, y_{ik-1}).$$

Then, since approximations of the derivatives can be obtained by the forward differences $\Delta_h^j,\ 1 \le j \le k$, with $h = ①^{-1}$ as follows

$$\Delta^k_{①^{-1}} = \sum_{j=0}^{k} (-1)^j \binom{j}{k} y_{x_i+(k-j)①^{-1}}, \tag{4}$$

where $\binom{j}{k}$ is a binomial coefficient, we have

$$y^{(k)}(x_i) \approx \frac{\Delta^k_{①^{-1}}}{①^{-k}} + O(①^{-1}), \tag{5}$$

i.e., the value $\frac{\Delta^k_{①^{-1}}}{①^{-k}}$ is finite and it gives us the *exact* derivative $y^{(k)}(x_i)$. For a more detailed description of the computation of derivatives see [42].

Usually derivatives are approximated by using automatic or numerical differentiation. Automatic differentiation makes use of specific tools based on the involved elementary functions (see [4]) and allows one to speed up the computation and does not suffers from typical problems of numerical differentiation when finite precision arithmetic is used. Numerical differentiation, especially when higher order derivative are necessary, suffer of numerical instability and is not able to reach a higher precision. Finite difference formulas are, in fact, ill-conditioned, moreover the cancellation error produces a value of zero if h is small. We observe that the use of the Infinity Computer makes finite difference formulas exact when the stepsize is equal to $①^{-1}$. This feature, besides preventing the above mentioned ill-conditioning phenomenon, is particularly appealing in the case where the function to be differentiated is a “black box”.

2.2 The main idea and the core Method 1.0

The main idea of the usage of numerical infinitesimals and a core method for solving the IVP (1) on the Infinity Computer has been proposed in [42].

Since we are able to compute values of k derivatives of the function $y(x)$ at a generic point x_0, we can estimate $y(x)$ in a neighborhood of the point x_0 by its Taylor expansion

$$y(x) \approx \hat{y}(x) = y_0 + \sum_{i=1}^{k} \frac{y^{(i)}(x_0)}{i!}(x - x_0)^i. \tag{6}$$

In cases where the radius of convergence of the Taylor expansion $\hat{y}(x)$ from (6) covers the whole interval $[a, b]$ of our interest, then there is no necessity to execute several steps with a finite value of h. In fact, thanks to the exact derivatives calculated numerically on the Infinity Computer the function $y(x)$ can be approximated in the neighborhood of the initial point x_0 by its Taylor expansion $\hat{y}(x)$ with an order k depending on the desired accuracy and then $\hat{y}(x)$ can be evaluated at any point $x \in [a, b]$. This method is called *Taylor for the Infinity Computer (TIC)* hereinafter. This method does not assume the execution of several iterations with the step h.

Table 1 presents results of numerical experiments executed on a class of 12 test functions taken from the literature and described in Table 7 of the Appendix. The method TIC is compared over the interval $[0, 0.2]$ with the Runge-Kutta method of the fourth order (RK4) with the integration step $h = 0.04$, i.e., to obtain an approximation at the point $x = 0.2$ the method RK4 executes 5 steps and 20 evaluations of the function $f(x, y)$ from (1). After the results for RK4 had been obtained, the TIC method was applied to each of 12 problems. The method TIC stopped when the accuracy ε_TIC at the point $x = 0.2$ was better than the accuracy ε_RK4 of the method RK4. The last column, N_TIC, in Table 1 presents the number of evaluations of $f(x, y)$ executed by the TIC to reach the accuracy ε_TIC. In other words, it shows the number of infinitesimal steps executed by the TIC that is equal to the number of exact derivatives calculated by this method. The respective solutions y_RK4 and y_TIC are also shown in the table. For the considered

	y_RK4	ε_RK4	y_TIC	ε_TIC	N_TIC
1	0.837462	-8.62538e-009	0.837462	-5.91687e-009	6
2	1.242806	8.11157e-009	1.242806	4.19151e-009	6
3	1.221403	4.12685e-009	1.221403	2.13248e-009	6
4	1.221403	3.89834e-008	1.221403	2.13248e-009	6
5	1.491817	1.27726e-007	1.491817	1.13693e-008	7
6	0.135416	-5.96529e-004	0.135379	-3.24420e-004	10
7	36.154673	-8.16405e-005	36.149608	5.84540e-005	9
8	35.968459	-8.18293e-005	35.963409	5.85817e-005	9
9	1.239230	-5.78803e-009	1.239230	-4.08211e-009	10
10	0.781397	-1.76949e-009	0.781397	7.94128e-011	7
11	1.153846	8.98577e-009	1.153846	4.09600e-009	11
12	0.472441	2.95775e-010	0.472441	-1.60782e-010	10

TABLE 1
Results of a comparison of the Taylor for the Infinity Computer method with the Runge-Kutta method of the fourth order that executes 20 evaluations of $f(x, y)$ to reach the accuracy ε_RK4

problem the TIC method executes fewer evaluations of $f(x, y)$, in comparison with the Runge-Kutta method. The Taylor method with automatic differentiation, provides, if applicable, the same solution given by TIC, we observe, however that, if the evaluation of $f(x, y)$ involves k elementary functions (*; =; ln; exp; sin; cos; . . .) then the computational complexity of the evaluation of the first $n - 1$ derivatives is $kn^2 + O(n)$. For the TIC the computational cost is exactly n, considering that the arithmetic operation are executed on the Infinity Computer Arithmetic.

Suppose now that the interval $[a, b]$ of our interest is wider than the radius of convergence of the Taylor expansion, or that the rate of convergence is so slow as to make the method unsuitable for the problem at hand. Then finite values of the integration step h should be used together with infinitesimal ones. So, we consider a mesh of $n + 1$ points x_i where

$$x_0 = a,\ x_{i+1} = x_i + h,\ 0 \leq i \leq n - 1,\ x_n = b,$$

where h is a finite integration step.

Let us consider the Method 1.0 from [42] being our core algorithm that is used hereinafter for further developments. At each iteration it calculates k derivatives (in [42] the particular case $k = 2$ has been considered) at a point x_i using infinitesimal steps and then executes a finite step of the length $h = (b - a)/n$ to the point x_{i+1}. So, at each iteration it applies the TIC

	y_RK4	ε_RK4	y_1.0	ε_1.0	N_1.0
1	0.735759	-2.20568e-008	0.735759	-1.51306e-008	30
2	3.436564	3.26429e-008	3.436564	1.68677e-008	30
3	2.718282	2.06343e-008	2.718282	1.06624e-008	30
4	2.718281	3.02546e-007	2.718282	1.65499e-008	30
5	7.388579	6.38533e-007	7.388584	5.66017e-008	35
6	0.000046	-2.98620e-003	0.000045	-1.62315e-003	50
7	20.026862	-1.22480e-006	20.026819	8.76400e-007	45
8	18.474354	-1.34674e-006	18.474311	9.47222e-007	45
9	2.732051	-7.46806e-009	2.732051	-8.00658e-010	50
10	-0.301169	6.85909e-008	-0.301169	-3.02846e-010	35
11	1.000000	3.82195e-008	1.000000	1.37934e-009	55
12	0.571429	7.69103e-009	0.571429	-2.01651e-011	50

TABLE 2
Results of a comparison of the Method 1.0 with the Runge-Kutta method of the fourth order that executes 100 evaluations of $f(x, y)$ to reach the accuracy ε_RK4

```
for $i = 0; i < n; i = i + 1$
    $y(x, x_i) = y_i + \sum_{j=1}^{k} \frac{y_i^{(j)}}{j!}(x - x_i)^j$,
    $y_{i+1} = y(x_{i+1}, x_i)$
endfor
```

FIGURE 1
Method 1.0 from [42]

method. The Method 1.0 uses the initial values $x_0 = a$, $y_0 = y(x_0)$, from (1) and its detailed description is presented in Figure 1.

Table 2 presents results for the Method 1.0 extending numerical experiments described in Table 1 from the interval $[0, 0.2]$ to the interval $[0, 1]$. Thus, the Method 1.0 executed five finite steps during which the method TIC was applied five times at the points $x_i = a + ih$, where $a = 0,\ h = 0.2$. The Method 1.0 is compared with the method RK4 with the same integration step as before ($h = 0.04$). At each interval $[x_i, x_{i+1}]$ the method TIC executed N_TIC evaluations of the function $f(x, y)$, where N_TIC was taken from sixth column of the Table 1 for each test function. It can be seen from Table 2 that the accuracy of the Method 1.0 is better than the accuracy of the Runge-Kutta method of the fourth order and the Method 1.0 executes fewer

	Method 1.0		Method RK2		Method 1.2	
#	y_n	ε_n	y_n	ε_n	y_n	ε_n
1	0.74148	-7.77538e-003	0.74148	-7.77538e-003	0.73262	4.26152e-003
2	3.40542	9.06351e-003	3.40542	9.06351e-003	3.42709	2.75755e-003
3	2.70271	5.72923e-003	2.70271	5.72923e-003	2.71354	1.74310e-003
4	2.69451	8.74561e-003	2.65824	2.20893e-002	2.70459	5.03795e-003
5	7.10043	3.89998e-002	7.10041	3.90024e-002	7.24952	1.88217e-002
6	1.00000	-2.20255e+004	1.00000	-2.20255e+004	-2.33333	5.13961e+004
7	31.63147	-5.79454e-001	31.63147	-5.79454e-001	-55.88025	3.79027e+000
8	30.04452	-6.26285e-001	30.05380	-6.26787e-001	-57.20706	4.09657e+000
9	2.74018	-2.97481e-003	2.73309	-3.80229e-004	2.72931	1.00341e-003
10	-0.30737	-2.05896e-002	-0.29889	7.56958e-003	-0.29849	8.89270e-003
11	0.99078	9.21515e-003	0.99824	1.76122e-003	1.00396	-3.96140e-003
12	0.57150	-1.33171e-004	0.57099	7.59569e-004	0.57166	-4.02684e-004

TABLE 3
For all the methods taken into consideration resulting values y_n at the point $x = 1$ and the respective relative error $\varepsilon_n = \frac{y(1)-y_n}{y(1)}$ are reported, where $y(1)$ is the exact solution

evaluations of $f(x, y)$ in comparison with the RK4 method.

2.3 Methods 1.1 and 1.2

Let us describe now a new algorithm called *Method 1.2* hereinafter. It is a generalization of the Method 1.1 from [42]. The main idea is to use derivatives calculated at the point x_{i+1} in order to return to the point x_i and to construct at this point a correction leading to a new approximation y^c_{i+1} that is better than the original value y_{i+1} provided by the Method 1.0.

The Method 1.2 works as follows. First, initial values are chosen in the same way as in the Method 1.0 and values $y_i,\ i = 1, ..., n$, and functions $y(x, x_i)$ are calculated as in the Method 1.0. Then for each $i = 1, ..., n-1$, the backward functions $\overline{y}_i(x)$ using forward differences from (5) are computed as follows

$$\overline{y}_i(x) = y(x_i, x_{i-1}) + \sum_{j=1}^{k} \frac{y^{(j)}(x_i, x_i)}{j!}(x - x_i)^j. \qquad (7)$$

Note that for $i = n$ the backward differences and the points $x_n - ①^{-1}, x_n - 2①^{-1}, ..., x_n - k①^{-1}$ should be used (see Corollary 1 in [42]) to calculate the derivatives $y^{(j)}(x_n, x_n)$. After that, the function $r_i(x)$ is defined as follows:

$$\begin{aligned} r_i(x) &= y_{i-1} + [p_0 y_{i-1} - (1-p_0)\overline{y}_i(x_{i-1})] + \\ &+ \textstyle\sum_{j=1}^{k} \frac{1}{j!}[p_j y^{(j)}(x_{i-1}, x_{i-1}) + (1-p_j)\overline{y}_i^{(j)}(x_{i-1})](x - x_{i-1})^j, \end{aligned} \quad (8)$$

where the weights $p_j \in [0,1],\ j = 0, ..., k$, are parameters of the Method 1.2. So, the global correction c_i can be obtained following the rule

$$c_i = c(x_i) = c(x_{i-1}) + r_i(x_i) - y(x_i, x_{i-1}),\ \ i = 1, ..., n, \quad (9)$$

with $c_0 = 0$. As a result, the desired corrected value y_i^c can be computed

$$y_n^c = y(x_n, x_{n-1}) + c(x_n). \quad (10)$$

The choice of the parameters of the Method 1.2 can be done using different criteria. For instance, the simplest choice for $k = 2$, $p_0 = p_1 = p_2 = 0.5$ gives us the Method 1.1 from [42]. If we apply the method to the standard test equation $y' = \lambda y$ we could choose the parameters by imposing that the method should become equivalent to the Taylor series method of the highest possible order. For example, for $k = 2$, it is shown in the Appendix that the choice of parameters $p_0 = 0, p_1 = 5/6, p_2 = 0.5$ makes the method equivalent to the Taylor series method with $k = 4$. This means that for the test equations the order increases from two to four. Results of numerical experiments executed with this choice of the parameters, compared with the method 1.0 used with $k = 2$ and with the Runge-Kutta method of second order on the same class of test functions from the Appendix are shown in Table 3.

As can be seen from Table 3, the introduced correction has allowed us to improve the results on some problems with respect to the Method 1.0. Again, as it was with the Method 1.0, among the Runge-Kutta methods a natural competitor for the Method 1.2 with $k = 2$ is the Runge-Kutta method of the second order since both methods execute $f(x, y)$ two times at each iteration. Then, the behavior of the Method 1.2 is comparable with that of the Runge-Kutta method of the second order on the considered test problems.

2.4 Method 1.3

Another possible way to approximate the solution to the problem (1) is introduced in the Method 1.3 described below. The main two differences between the Method 1.2 and the Method 1.3 are the following:

```
for $i = 1; i \leq n; i = i+1$
    $y(x, x_{i-1}) = y_{i-1} + \sum_{j=1}^{k} \frac{y_{i-1}^{(j)}}{j!}(x - x_{i-1})^j,$
    $\widetilde{y}_i(x) = y(x_i, x_{i-1}) + \sum_{j=1}^{k} \frac{\tilde{y}_i^{(j)}}{j!}(x - x_i)^j,$
    $\widetilde{r}_i(x) = y_{i-1} + [p_0 y_{i-1} - (1-p_0)\widetilde{y}_i(x_{i-1})]+$
        $+ \sum_{j=1}^{k} \frac{1}{j!}[p_j y^{(j)}(x_{i-1}, x_{i-1}) + (1-p_j)\widetilde{y}_i^{(j)}(x_{i-1})](x - x_{i-1})^j,$
    $y_i = \widetilde{r}_i(x_i)$
endfor
```

FIGURE 2
Method 1.3

i.) The Method 1.2 executes n iterations of the Method 1.0 to calculate the approximated values of $y_i,\ i = 1, ..., n$, and then these values are corrected by the backward function $\overline{y}_i(x)$ from (7) and the mixed function $r_i(x)$ from (8). The Method 1.3 at each subinterval $[x_{i-1}, x_i]$ executes the evaluation of the approximated value y_i and then immediately evaluates the backward function $\widetilde{y}_i(x)$ and the mixed function $\widetilde{r}_i(x)$ before moving to the next interval $[x_i, x_{i+1}]$.

ii.) The Method 1.2 in the formula (7) of $\overline{y}_i(x)$ uses for the forward function $y(x, x_i)$ at each point $(x_i, y(x_i, x_{i-1}))$ old values of derivatives calculated at points (x_i, y_i) before the correction. However, the values y_i and $y(x_i, x_{i-1})$ can be different and, as a consequence, the respective derivatives can be also different. Thus, the Method 1.3 after the correction and before moving to the next interval $[x_i, x_{i+1}]$ calculates the exact derivatives at each point $(x_i, y(x_i, x_{i-1}))$.

Let us denote now as $y_{i-1}^{(j)}$ the approximation of the j-th derivative using the rule (5) calculated at the point (x_{i-1}, y_{i-1}) and as $\tilde{y}_i^{(j)}$ the approximation of the j-th derivative again using the rule (5) but at the point $(x_i, y(x_i, x_{i-1}))$. Notice that for $i = 1, ..., n-1$, the forward differences are used and the backward ones (see Corollary 1 from [42]) are applied for $i = n$. Taking into consideration that the initial values are the same as in the previous cases and the values $p_j, j = 0, ..., k$, are parameters of the Method 1.3 having the same meaning as in the Method 1.2, the Method 1.3 is described in Figure 2.

Results of numerical experiments for the Method 1.3 with the value $k = 2$ and the same parameters used in the Method 1.2 are given in Table 4 for the same test problems. As can be seen from this table, the attained accuracy of

	Method 1.3		Method RK2		Method RK3		Method RK4	
#	y_n	ε_n	y_n	ε_n	y_n	ε_n	y_n	ε_n
1	0.73577	-1.57578e-005	0.74148	-7.77538e-003	0.73547	3.91315e-004	0.73577	-1.57578e-005
2	3.43650	1.78619e-005	3.40542	9.06351e-003	3.43502	4.49549e-004	3.43650	1.78619e-005
3	2.71825	1.12909e-005	2.70271	5.72923e-003	2.71751	2.84169e-004	2.71825	1.12909e-005
4	2.71718	4.03706e-004	2.65824	2.20893e-002	2.71351	1.75595e-003	2.71787	1.52387e-004
5	7.38632	3.06560e-004	7.10041	3.90024e-002	7.35996	3.87457e-003	7.38632	3.06113e-004
6	0.00412	-8.96439e+001	1.00000	-2.20255e+004	-0.00412	9.16439e+001	0.00412	-8.96439e+001
7	20.11564	-4.43440e-003	31.63147	-5.79454e-001	20.00000	1.34005e-003	20.11564	-4.43440e-003
8	18.56287	-4.79261e-003	30.05380	-6.26787e-001	18.44666	1.49775e-003	18.56337	-4.81998e-003
9	2.73185	7.36503e-005	2.73309	-3.80229e-004	2.73178	9.74546e-005	2.73207	-5.35061e-006
10	-0.30091	8.73137e-004	-0.29889	7.56958e-003	-0.30105	3.94699e-004	-0.30116	4.13631e-005
11	1.00100	-1.00013e-003	0.99824	1.76122e-003	1.00093	-9.33299e-004	0.99997	3.18508e-005
12	0.57176	-5.73749e-004	0.57099	7.59569e-004	0.57164	-3.64397e-004	0.57143	5.73049e-006

TABLE 4
For all the methods taken into consideration resulting values y_n at the point $x = 1$ and the respective relative error $\varepsilon_n = \frac{y(1)-y_n}{y(1)}$ are reported, where $y(1)$ is the exact solution. The Method 1.3 uses the following parameters: $p_0 = 0, p_1 = 5/6, p_2 = 0.5$

the Method 1.3 is better with respect to the Runge-Kutta method of second order for all test problems. This is due to the choice of the parameters, that make the methods of order at least three. A formal discussion related to the convergence properties of this method is presented in [2], where it has been proved that the order of convergence of this method is 3. The behavior of the Method 1.3 is comparable with respect to the Runge-Kutta method of third order on all the test problems. Finally, we observe that similar results are obtained with respect to the Runge-Kutta method of fourth order for linear problems. It should be mentioned that the obtained improvement has its price. In fact, the Method 1.3 executes $2kn$ evaluations of the function $f(x, y)$ from (1) whereas the Method 1.2 performs just $kn + k$ evaluations of $f(x, y)$.

2.5 Method 1.4

The main idea of this method is to avoid calculation of the derivatives at the close points (x_i, y_i) and $(x_i, y(x_i, x_{i-1}))$. Since these points are close, the difference between y_i and $y(x_i, x_{i-1})$ can be relatively small. Thus, instead of recalculating derivatives at the points (x_i, y_i) as it is done in the Method 1.3, the values of the derivatives calculated at the points $(x_i, y(x_i, x_{i-1}))$ can be used also at the points (x_i, y_i). This is the main difference between the Methods 1.3 and 1.4.

Let us again denote the approximation of the j-th derivative using infinitesimals starting from the point (x_i, y_i) as $y_i^{(j)}$ and the approximation of the j-

	Method 1.4		Method RK2		Method RK3		Method RK4	
#	y_n	ε_n	y_n	ε_n	y_n	ε_n	y_n	ε_n
1	0.73495	1.09797e-003	0.74148	-7.77538e-003	0.73547	3.91315e-004	0.73577	-1.57578e-005
2	3.43265	1.13895e-003	3.40542	9.06351e-003	3.43502	4.49549e-004	3.43650	1.78619e-005
3	2.71632	7.19955e-004	2.70271	5.72923e-003	2.71751	2.84169e-004	2.71825	1.12909e-005
4	2.71142	2.52314e-003	2.65824	2.20893e-002	2.71351	1.75595e-003	2.71787	1.52387e-004
5	7.32003	9.27820e-003	7.10041	3.90024e-002	7.35996	3.87457e-003	7.38632	3.06113e-004
6	0.03704	-8.14795e+002	1.00000	-2.20255e+004	-0.00412	9.16439e+001	0.00412	-8.96439e+001
7	23.46140	-1.71498e-001	31.63147	-5.79454e-001	20.00000	1.34005e-003	20.11564	-4.43440e-003
8	21.89863	-1.85355e-001	30.05380	-6.26787e-001	18.44666	1.49775e-003	18.56337	-4.81998e-003
9	2.73104	3.68986e-004	2.73309	-3.80229e-004	2.73178	9.74546e-005	2.73207	-5.35061e-006
10	-0.30030	2.87314e-003	-0.29889	7.56958e-003	-0.30105	3.94699e-004	-0.30116	4.13631e-005
11	1.00311	-3.10616e-003	0.99824	1.76122e-003	1.00093	-9.33299e-004	0.99997	3.18508e-005
12	0.57188	-7.83660e-004	0.57099	7.59569e-004	0.57164	-3.64397e-004	0.57143	5.73049e-006

TABLE 5
For all the methods taken into consideration resulting values y_n at the point $x = 1$ and the respective relative error $\varepsilon_n = \frac{y(1)-y_n}{y(1)}$ are reported, where $y(1)$ is the exact solution. The Method 1.4 uses the following parameters: $p_0 = 0, p_1 = 5/6, p_2 = 0.5$

```
for $i = 1; i \le n; i = i + 1$
    $\widehat{y}_i(x) = y(x_i, x_{i-1}) + \sum_{j=1}^{k} \frac{\widehat{y}_i^{(j)}}{j!}(x - x_i)^j,$
    $r_i(x) = y_{i-1} + [p_0 y_{i-1} - (1 - p_0)\widehat{y}_i(x_{i-1})] +$
        $+ \sum_{j=1}^{k} \frac{1}{j!}[p_j y^{(j)}(x_{i-1}, x_{i-1}) + (1 - p_j)\widehat{y}_i^{(j)}(x_{i-1})](x - x_{i-1})^j,$
    $y_i = r_i(x_i)$
    $y(x, x_i) = y_i + \sum_{j=1}^{k} \frac{\widehat{y}_i^{(j)}}{j!} x^j,$
endfor
```

FIGURE 3
Method 1.4

th derivative using infinitesimals but starting from the point $(x_i, y(x_i, x_{i-1}))$ as $\widehat{y}_i^{(j)}$ for $i = 1, ..., n-1$, (for $i = n$ the backward approximation, see Corollary 1 from [42], is used). The initial values are the same as above and $p_j, j = 0, ..., k$ are parameters of the method. Then the Method 1.4 is described in Figure 3.

The results of the experiments on the same class of test functions are given in Table 5. For the Method 1.4 the value $k = 2$ and the same optimal parameters used for the Method 1.3.

	Method 1.0		Method 1.2		Method 1.3		Method 1.4	
#	y_n	ε_n	y_n	ε_n	y_n	ε_n	y_n	ε_n
1	0.74148	-7.77538e-003	0.73262	4.26152e-003	0.73577	-1.57578e-005	0.73495	1.09797e-003
2	3.40542	9.06351e-003	3.42709	2.75755e-003	3.43650	1.78619e-005	3.43265	1.13895e-003
3	2.70271	5.72923e-003	2.71354	1.74310e-003	2.71825	1.12909e-005	2.71632	7.19955e-004
4	2.69451	8.74561e-003	2.70459	5.03795e-003	2.71718	4.03706e-004	2.71142	2.52314e-003
5	7.10043	3.89998e-002	7.24952	1.88217e-002	7.38632	3.06560e-004	7.32003	9.27820e-003
6	1.00000	-2.20255e+004	-2.33333	5.13961e+004	0.00412	-8.96439e+001	0.03704	-8.14795e+002
7	31.63147	-5.79454e-001	-55.88025	3.79027e+000	20.11564	-4.43440e-003	23.46140	-1.71498e-001
8	30.04452	-6.26285e-001	-57.20706	4.09657e+000	18.56287	-4.79261e-003	21.89863	-1.85355e-001
9	2.74018	-2.97481e-003	2.72931	1.00341e-003	2.73185	7.36503e-005	2.73104	3.68986e-004
10	-0.30737	-2.05896e-002	-0.29849	8.89270e-003	-0.30091	8.73137e-004	-0.30030	2.87314e-003
11	0.99078	9.21515e-003	1.00396	-3.96140e-003	1.00100	-1.00013e-003	1.00311	-3.10616e-003
12	0.57150	-1.33171e-004	0.57166	-4.02684e-004	0.57176	-5.73749e-004	0.57188	-7.83660e-004

TABLE 6
Comparison of the Methods 1.0–1.4

As can be seen from Table 5, the attained accuracy of the Method 1.4 is better than the accuracy of the Runge-Kutta method of the second order for many test problems. The Method 1.4 executes less evaluations of the function $f(x, y)$ than the Method 1.3. Namely, it works doing the same number of evaluations of $f(x, y)$ as the Method 1.2, i.e., $kn + k$. The accuracy of the Method 1.4 is worse with respect to the Runge-Kutta methods of higher orders in all test problems. The reason is that the Method 1.4 does not use the exact derivatives at points (x_i, y_i) as the Method 1.3 does and the difference between the derivatives at the points (x_i, y_i) and $(x_i, y(x_i, x_{i-1}))$ causes errors. However, we can observe that the Method 1.4 executes the number of evaluations of the function $f(x, y)$ that is similar to the Runge-Kutta method of the second order and at the same time the accuracy of the Method 1.4 is better with respect to RK2. A detailed description of the convergence and stability properties of the Methods 1.3 and 1.4 is presented in [2], in the next section we report some experiments that show the potentiality of numerical methods using derivatives computed on the Infinity Computer.

2.6 A detailed description of results of numerical experiments

In Section 2.2 we have already seen the nice performance of the Method 1.0 with large values of k. In this subsection we analyze this method with k up to three in order to compare its behavior with the Methods 1.3 and 1.4 and the Runge-Kutta Methods of order 2 and 3.

Figures 4–7 show the behavior of the error for all the twelve considered test

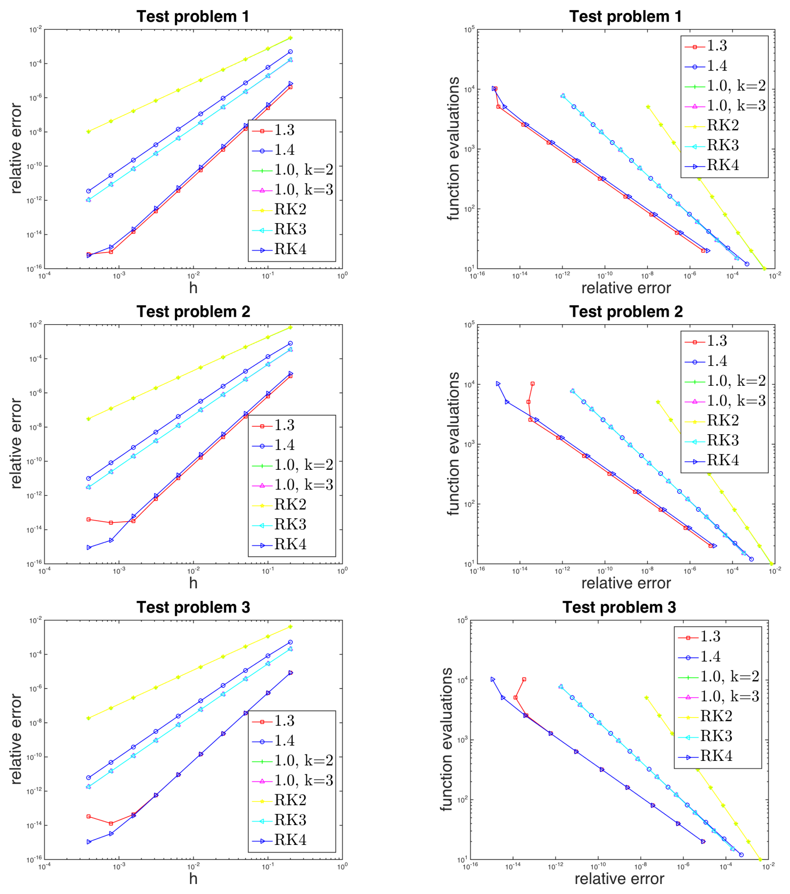

FIGURE 4
Relative error versus stepsize and function evaluation versus relative error for problems 1,2 and 3.

problems, changing the stepsize and the computational cost using the number of function evaluations (observe that the latter are performed in the Infinity Computer Arithmetic). From the pictures it could be seen that the behavior of the Method 1.3 is similar to the RK4 for linear problems, while for nonlinear ones, the order 3 of the method is experienced and the behavior is very close to the one of the Method 1.4 and of the Method 1.0 with $k = 3$. The Method 1.4 has order 3 for all the test problems and for the nonlinear tests requires a

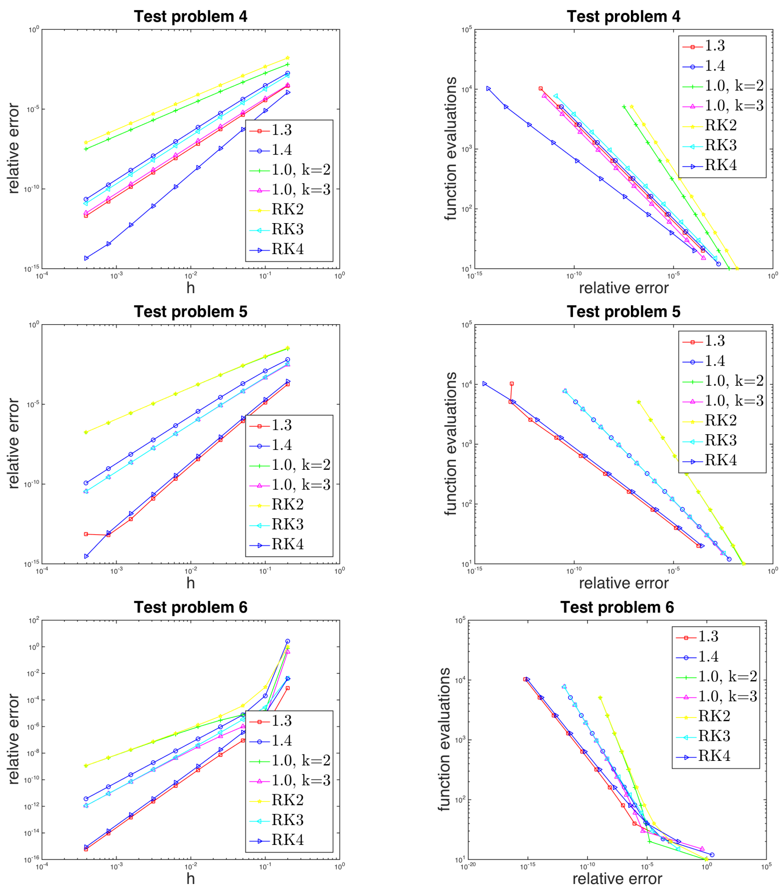

FIGURE 5
Relative error versus stepsize and function evaluation versus relative error for problems 4,5 and 6.

smaller computational effort to reach the same precision of the Method 1.3. The main potentiality of the numerical schemes using derivatives is that the use of the Infinity Computer allows one to compute the derivatives without error by a linear combinations of the computed infinitesimal values. This means that computing the exact derivatives does not give any computational problem to the method. We are aware that the Infinity Computer Arithmetic requires a computational effort that is higher than the one required by standard

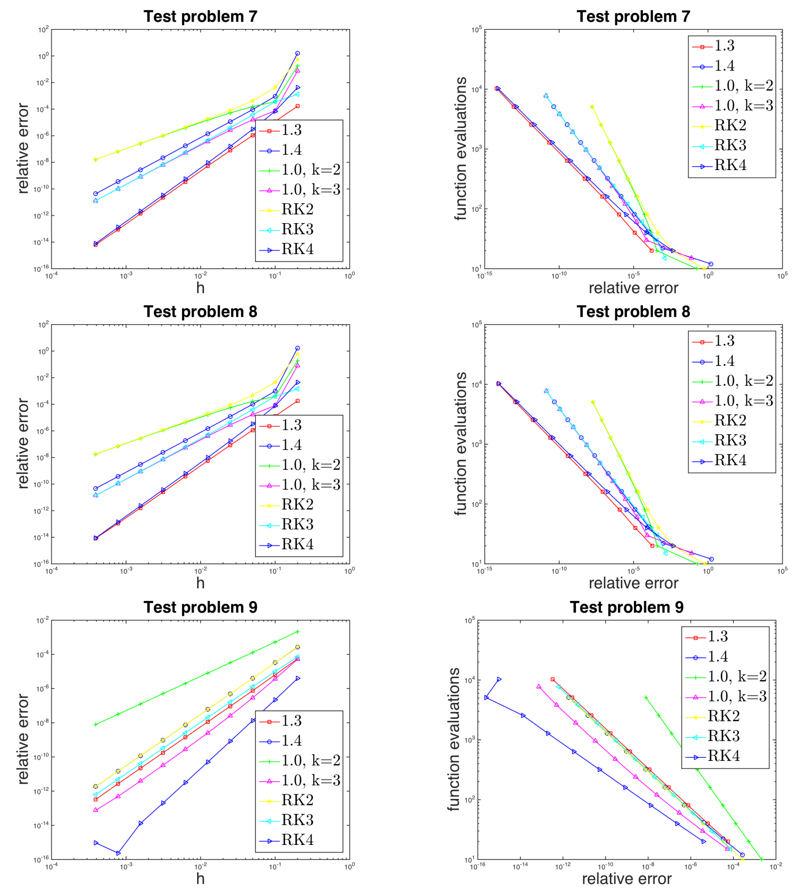

FIGURE 6
Relative error versus stepsize and function evaluation versus relative error for problems 7,8 and 9.

one, but for a computer based on this arithmetic all the complexity effort is hidden to the user, who needs only to use, in the arithmetic operations, the new numeral ①.

3 A BRIEF CONCLUSION

This paper introduces several numerical methods for solving the initial value problem on the Infinity Computer able to execute numerical computations

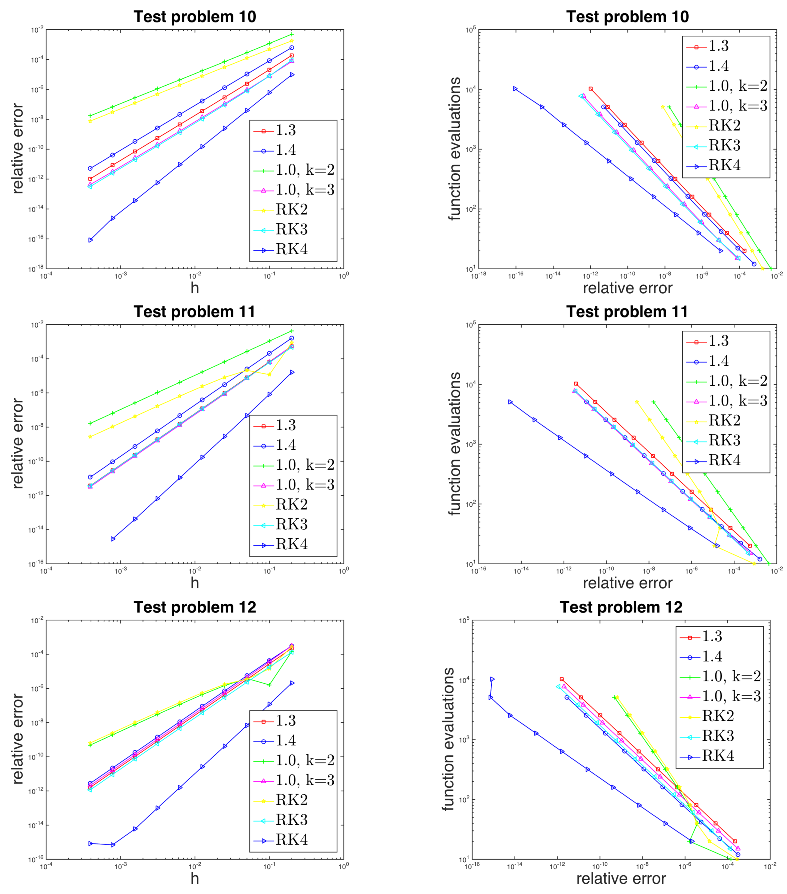

FIGURE 7
Relative error versus stepsize and function evaluation versus relative error for problems 10,11,12.

with infinities and infinitesimals. The new methods can mix finite and infinitesimal steps in their work opening so new possibilities for constructing numerical algorithms.

The obtained results seem to be very promising and show that one of the the most significant advantages of the Infinity Computer applied for solving ODEs consists of the possibility to calculate the exact values of the derivatives of the solution. This becomes possible thanks to a smart usage of infinitesimal values of the integration step h.

Numerical experiments executed on a class of problems taken from the literature show that the possibility to use exact values of derivatives of the solution allows the authors to introduce new numerical schemes that can be competitive with respect to Runge-Kutta methods widely used in practice.

4 ACKNOWLEDGEMENTS

The authors thank Olga Rubchenkova for useful discussions and executing some preliminary numerical experiments.

Research of Ya.D. Sergeyev and M.S. Mukhametzhanov was supported by the Russian Science Foundation, project No.15-11-30022 “Global optimization, supercomputing computations, and applications”.

Research of P. Amodio, F. Iavernaro and F. Mazzia was supported by the Università degli studi di Bari, project “Equazioni di Evoluzione: analisi qualitativa e metodi numerici”.

REFERENCES

[1] R.A. Adams. (2003). *Single variable calculus*. Pearson Education Canada, Ontario, 5 edition.

[2] P. Amodio, F. Iavernaro, F. Mazzia, M.S. Mukhametzhanov, and Ya.D. Sergeyev. (2015). A generalized Taylor method of order three for the solution of initial value problems in standard and infinity floating-point arithmetic. submitted.

[3] P. Amodio and D. Trigiante. (1993). *Elementi di calcolo numerico (in Italian)*. Pitagora Editrice Bologna.

[4] R. Barrio. (2005). Performance of the Taylor series method for odes/daes. *Applied Mathematics and Computation*, 163(2):525–545. cited By 47.

[5] L. Brugnano and D. Trigiante. (1998). *Solving Differential Problems by Multistep Initial and Boundary Value Methods*. Gordon and Breach Science Publ., Amsterdam.

[6] J.C. Butcher. (2003). *Numerical methods for ordinary differential equations*. John Wiley & Sons, Chichester, 2 edition.

[7] J.R. Cash and S. Considine. (1992). An mebdf code for stiff initial value problems. *ACM Trans. Math. Software*, 18(2):142–155.

[8] S. De Cosmis and R. De Leone. (2012). The use of grossone in mathematical programming and operations research. *Applied Mathematics and Computation*, 218(16):8029–8038.

[9] L. D'Alotto. (2012). Cellular automata using infinite computations. *Applied Mathematics and Computation*, 218(16):8077–8082.

[10] L. D'Alotto. (2013). A classification of two-dimensional cellular automata using infinite computations. *Indian Journal of Mathematics*, 55:143–158.

[11] L. D'Alotto. (2015). A classification of one-dimensional cellular automata using infinite computations. *Applied Mathematics and Computation*, 255:15–24.

[12] W. Gander, M.J. Gander, and F. Kwok. (900, 2014). *Scientific Computing. An introduction using Maple and Matlab*. Springer.

[13] D.F. Griffiths and D.J. Higham. (271, 2010). *Numerical methods for Ordinary differential equations. Initial Value Problems*. Springer Undergraduate Mathematics Series.

[14] E. Hairer, S. P. Nørsett, and G. Wanner. (1993). *Solving ordinary differential equations I: Nonstiff problems*. Springer-Verlag, New York, 2 edition.

[15] P. Henrici. (1997). *Applied and computational complex analysis*, volume 1. John Wiley & Sons, Chichester.

[16] F. Iavernaro and F. Mazzia. (1998). Solving ordinary differential equations by generalized adams methods: properties and implementation techniques. *Appl. Numer. Math.*, 28(2–4):107–126.

[17] D.I. Iudin, Ya.D. Sergeyev, and M. Hayakawa. (2012). Interpretation of percolation in terms of infinity computations. *Applied Mathematics and Computation*, 218(16):8099–8111.

[18] D.I. Iudin, Ya.D. Sergeyev, and M. Hayakawa. (2015). Infinity computations in cellular automaton forest-fire model. *Communications in Nonlinear Science and Numerical Simulation*, 20(3):861–870.

[19] V. Kanovei and V. Lyubetsky. (2015). Grossone approach to Hutton and Euler transforms. *Applied Mathematics and Computation*, 255:36–43.

[20] L.H. Kauffman. (2015). Infinite computations and the generic finite. *Applied Mathematics and Computation*, 255:25–35.

[21] J.D. Lambert. (1991). *Numerical Methods for Ordinary Differential Systems: The Initial Value Problem*. John Wiley & Sons, Chichester.

[22] G. Lolli. (2012). Infinitesimals and infinites in the history of mathematics: A brief survey. *Applied Mathematics and Computation*, 218(16):7979–7988.

[23] G. Lolli. (2015). Metamathematical investigations on the theory of grossone. *Applied Mathematics and Computation*, 255:3–14.

[24] M. Margenstern. (2011). Using grossone to count the number of elements of infinite sets and the connection with bijections. *p-Adic Numbers, Ultrametric Analysis and Applications*, 3(3):196–204.

[25] M. Margenstern. (2012). An application of grossone to the study of a family of tilings of the hyperbolic plane. *Applied Mathematics and Computation*, 218(16):8005–8018.

[26] M. Margenstern. (2015). Fibonacci words, hyperbolic tilings and grossone. *Communications in Nonlinear Science and Numerical Simulation*, 21(1–3):3–11.

[27] F. Mazzia and A.M. Nagy. (2015). A new mesh selection strategy with stiffness detection for explicit Runge-Kutta methods. *Applied Mathematics and Computation*, 255:125–134.

[28] F. Montagna, G. Simi, and A. Sorbi. (2015). Taking the Pirahã seriously. *Communications in Nonlinear Science and Numerical Simulation*, 21(1–3):52–69.

[29] Ya.D. Sergeyev. (2003, 2nd ed. 2013). *Arithmetic of Infinity*. Edizioni Orizzonti Meridionali, CS.

[30] Ya.D. Sergeyev. (2007). Blinking fractals and their quantitative analysis using infinite and infinitesimal numbers. *Chaos, Solitons & Fractals*, 33(1):50–75.

[31] Ya.D. Sergeyev. (2007). Infinity computer and calculus. In Simos T.E., Psihoyios G., and Tsitouras Ch., editors, *AIP Proc. of the 5th International Conference on Numerical Analysis and Applied Mathematics*, volume 936, pages 23–26. Melville, New York.

[32] Ya.D. Sergeyev. (2008). A new applied approach for executing computations with infinite and infinitesimal quantities. *Informatica*, 19(4):567–596.

[33] Ya.D. Sergeyev. (2009). Evaluating the exact infinitesimal values of area of Sierpinski's carpet and volume of Menger's sponge. *Chaos, Solitons & Fractals*, 42(5):3042–3046.

[34] Ya.D. Sergeyev. (2009). Numerical point of view on Calculus for functions assuming finite, infinite, and infinitesimal values over finite, infinite, and infinitesimal domains. *Nonlinear Analysis Series A: Theory, Methods & Applications*, 71(12):e1688–e1707.

[35] Ya.D. Sergeyev. (2010). *Computer system for storing infinite, infinitesimal, and finite quantities and executing arithmetical operations with them*. USA patent 7,860,914.

[36] Ya.D. Sergeyev. (2010). Counting systems and the First Hilbert problem. *Nonlinear Analysis Series A: Theory, Methods & Applications*, 72(3-4):1701–1708.

[37] Ya.D. Sergeyev. (2010). Lagrange Lecture: Methodology of numerical computations with infinities and infinitesimals. *Rendiconti del Seminario Matematico dell'Università e del Politecnico di Torino*, 68(2):95–113.

[38] Ya.D. Sergeyev. (2011). Higher order numerical differentiation on the infinity computer. *Optimization Letters*, 5(4):575–585.

[39] Ya.D. Sergeyev. (2011). On accuracy of mathematical languages used to deal with the Riemann zeta function and the Dirichlet eta function. *p-Adic Numbers, Ultrametric Analysis and Applications*, 3(2):129–148.

[40] Ya.D. Sergeyev. (2011). Using blinking fractals for mathematical modelling of processes of growth in biological systems. *Informatica*, 22(4):559–576.

[41] Ya.D. Sergeyev. (2013). Numerical computations with infinite and infinitesimal numbers: Theory and applications. In Sorokin A. and Pardalos P.M., editors, *Dynamics of Information Systems: Algorithmic Approaches*, pages 1–66. Springer, New York.

[42] Ya.D. Sergeyev. (2013). Solving ordinary differential equations by working with infinitesimals numerically on the infinity computer. *Applied Mathematics and Computation*, 219(22):10668–10681.

[43] Ya.D. Sergeyev. (2015). Computations with grossone-based infinities. In Calude C.S. and Dinneen M.J., editors, *Unconventional Computation and Natural Computation: Proc. of the 14th International Conference UCNC 2015*, volume LNCS 9252, pages 89–106. Springer, New York.

[44] Ya.D. Sergeyev. (2015). Numerical infinitesimals for solving ODEs given as a black-box. In Simos T.E. and Tsitouras Ch., editors, *AIP Proc. of the International Conference on Numerical Analysis and Applied Mathematics 2014 (ICNAAM-2014)*, volume 1648, page 150018. Melville, New York.

[45] Ya.D. Sergeyev. (2015). The olympic medals ranks, lexicographic ordering, and numerical infinities. *The Mathematical Intelligencer*, 37(2):4–8.

[46] Ya.D. Sergeyev. (2015). Un semplice modo per trattare le grandezze infinite ed infinitesime. *Matematica nella Società e nella Cultura: Rivista della Unione Matematica Italiana*, 8(1):111–147.

[47] Ya.D. Sergeyev. (2016). The exact (up to infinitesimals) infinite perimeter of the Koch snowflake and its finite area. *Communications in Nonlinear Science and Numerical Simulation*, 31(1–3):21–29.

[48] Ya.D. Sergeyev and A. Garro. (2010). Observability of Turing machines: A refinement of the theory of computation. *Informatica*, 21(3):425–454.

[49] Ya.D. Sergeyev and A. Garro. (2013). Single-tape and multi-tape Turing machines through the lens of the Grossone methodology. *Journal of Supercomputing*, 65(2):645–663.

[50] M.C. Vita, S. De Bartolo, C. Fallico, and M. Veltri. (2012). Usage of infinitesimals in the Menger's Sponge model of porosity. *Applied Mathematics and Computation*, 218(16):8187–8196.

[51] A.A. Zhigljavsky. (2012). Computing sums of conditionally convergent and divergent series using the concept of grossone. *Applied Mathematics and Computation*, 218(16):8064–8076.

[52] A. Žilinskas. (2012). On strong homogeneity of two global optimization algorithms based on statistical models of multimodal objective functions. *Applied Mathematics and Computation*, 218(16):8131–8136.

APPENDIX

The choice of parameters for the methods 1.2–1.4

We propose a possible way to choose parameters $p_0, p_1, \dots p_k$ for the Methods 1.2–1.4 using $k = 2$ derivatives calculated thanks to k infinitesimal steps of the explicit Euler method.

Let us consider the following IVP:

$$y' = \lambda y, \;\; y(x_0) = y_0. \tag{11}$$

Thus, it follows $y^{(k)} = \lambda^k y$ for $k = 1, 2, ...$ and the Taylor series for the function $y(x)$ is

$$y(x, x_0) = y_0 + \lambda y_0(x - x_0) + \frac{1}{2}\lambda^2 y_0(x - x_0)^2 + O(|x - x_0|^3). \tag{12}$$

Let us use designations $h = x_1 - x_0$ and $q = \lambda h$ and consider the first three summands from the Taylor series to approximate the value y_1

$$y_1 = y_0 + qy_0 + \frac{1}{2}q^2 y_0. \tag{13}$$

The function $\overline{y}(x)$ can be then defined as follows

$$\overline{y}(x) = y_1 + \lambda y_1(x - x_1) + \frac{1}{2}\lambda^2 y_1(x - x_1)^2 + o(|x - x_1|^2) =$$

$$= y_1 + \lambda y_1(x - x_0 + x_0 - x_1) + \frac{1}{2}\lambda^2 y_1(x - x_0 + x_0 - x_1)^2 +$$

$$+ o(|x - x_1|^2) = (y_1 - qy_1 + \frac{1}{2}q^2 y_1) +$$

$$+ (\lambda y_1 - h\lambda^2 y_1)(x - x_0) + \frac{1}{2}\lambda^2 y_1(x - x_0)^2 + o(|x - x_1|^2).$$

As a result, for the function $r_1(x)$ from (8) with the parameter $p_0 = 0$ we can write

$$r_1(x) = y_0 + [p_1\lambda y_0 + (1-p_1)\lambda y_1(1-q)](x-x_0)+$$
$$[\frac{p_2\lambda^2}{2}y_0 + \frac{(1-p_2)\lambda^2}{2}y_1](x-x_0)^2.$$

This allows us to calculate the value of the function $r_1(x)$ at the point x_1 as follows

$$r_1 = y_0 + p_1 q y_0 + (1-p_1)qy_1(1-q)+$$
$$\frac{p_2 q^2}{2}y_0 + \frac{(1-p_2)q^2}{2}y_1 =$$
$$= y_0 + p_1 q y_0 + (1-p_1)q(y_0 + qy_0 + \frac{1}{2}q^2 y_0)(1-q)+$$
$$\frac{p_2 q^2}{2}y_0 + \frac{(1-p_2)q^2}{2}(y_0 + qy_0 + \frac{1}{2}q^2 y_0) =$$
$$= y_0 + qy_0 + \frac{q^2 y_0}{2} + (\frac{-1+p_1}{2} - \frac{1-p_2}{2})q^3 y_0 + (\frac{-1+p_1}{2} - \frac{1-p_2}{4})q^4 y_0.$$

Now we can choose parameters p_1 and p_2 in such a way that

$$r_1 = y_0 + \sum_{i=1}^{4} \frac{1}{i!} q^i y_0.$$

To do this it is necessary to solve the following system of linear equations

$$\begin{cases} \frac{p_1}{2} - \frac{p_2}{2} = \frac{1}{3!} \\ \frac{p_1}{2} - \frac{p_2}{4} = \frac{1}{4!} + \frac{1}{4} \end{cases} \tag{14}$$

By solving this system we get the desired values of the parameters

$$p_1^* = \frac{5}{6},\ p_2^* = 0.5. \tag{15}$$

#	ODE	y_0	y_n	solution	source
1	$y' = x - y$	1	0.735759	$y(x) = x - 1 + 2e^{-x}$	[1]
2	$y' = x + y$	1	3.436564	$y(x) = 2e^{x} - x - 1$	[1]
3	$y' = y$	1	2.718282	$y(x) = e^{x}$	[6]
4	$y' = 2y - e^{x}$	1	2.718282	$y(x) = e^{x}$	[6]
5	$y' = 2y(1 - 0.00001y)$	1	7.388584	$y(x) = 100000e^{2x}/(100000 + e^{2x} - 1)$	[3]
6	$y' = -10y$	1	0.000045	$y(x) = e^{-10x}$	[12]
7	$y' = -8(y - 20)$	100	20.026837	$y(x) = 80e^{-8x} + 20$	[13]
8	$y' = -8(y - 15e^{-x/8} - 5)$	100	18.474329	$y(x) = \frac{1675}{21}e^{-8x} + \frac{320}{21}e^{-x/8} + 5$	[13]
9	$y' = \frac{y+x}{y-x}$	1	2.732051	$y(x) = x + \sqrt{1 + 2x^2}$	[6]
10	$y' = -y \cdot tan(x) - \frac{1}{cos(x)}$	1	-0.301169	$y(x) = \cos(x) - \sin(x)$	[6]
11	$y' = \frac{y-2xy^2}{1+x}$	1	1	$y(x) = \frac{1+x}{1+x^2}$	[6]
12	$y' = \frac{y-2xy^2}{1+x}$	0.4	0.571429	$y(x) = \frac{1+x}{2.5+x^2}$	[6]

TABLE 7
Test problems