Available online at www.sciencedirect.com

ScienceDirect

Mathematics and Computers in Simulation 141 (2017) 24–39

MATHEMATICS AND COMPUTERS IN SIMULATION

www.elsevier.com/locate/matcom

Original articles

A generalized Taylor method of order three for the solution of initial value problems in standard and infinity floating-point arithmetic[☆]

P. Amodio[a], F. Iavernaro[a], F. Mazzia[a], M.S. Mukhametzhanov[b,c], Ya.D. Sergeyev[b,c,*]

[a] *Dipartimento di Matematica, Università degli Studi di Bari, Italy*
[b] *DIMES, Università della Calabria, Italy*
[c] *Department of Software and Supercomputing Technologies, Lobachevsky State University of Nizhni Novgorod, Russia*

Received 26 November 2015; received in revised form 22 February 2016; accepted 18 March 2016
Available online 7 April 2016

Abstract

A well-known drawback of algorithms based on Taylor series formulae is that the explicit calculation of higher order derivatives formally is an over-elaborate task. To avoid the analytical computation of the successive derivatives, numeric and automatic differentiation are usually used. A recent alternative to these techniques is based on the calculation of higher derivatives by using the Infinity Computer—a new computational device allowing one to work numerically with infinities and infinitesimals. Two variants of a one-step multi-point method closely related to the classical Taylor formula of order three are considered. It is shown that the new formula is order three accurate, though requiring only the first two derivatives of $y(t)$ (rather than three if compared with the corresponding Taylor formula of order three). To get numerical evidence of the theoretical results, a few test problems are solved by means of the new methods and the obtained results are compared with the performance of Taylor methods of order up to four.
© 2016 International Association for Mathematics and Computers in Simulation (IMACS). Published by Elsevier B.V. All rights reserved.

Keywords: Ordinary differential equations; Initial value problems; Taylor methods; Numerical infinitesimals; Infinity computer

1. Introduction

The Taylor series method is one of the earliest algorithms to approximate the solution of initial value problems

$$\begin{cases} y' = f(t, y), & t \in [t_0, T], \\ y(t_0) = y_0, \end{cases} \tag{1}$$

where $f : [t_0, T] \times \mathbb{R}^n \to \mathbb{R}^n$ is assumed sufficiently differentiable.

☆ 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". 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".

* Corresponding author at: DIMES, Università della Calabria, Italy.
E-mail addresses: pierluigi.amodio@uniba.it (P. Amodio), felice.iavernaro@uniba.it (F. Iavernaro), francesca.mazzia@uniba.it (F. Mazzia), muhametzhanov.m@gmail.com (M.S. Mukhametzhanov), yaro@dimes.unical.it (Ya.D. Sergeyev).

http://dx.doi.org/10.1016/j.matcom.2016.03.007

A generalized Taylor method of order three for the solution of initial value problems in standard and infinity floating-point arithmetic[☆]

P. Amodio[a], F. Iavernaro[a], F. Mazzia[a], M.S. Mukhametzhanov[b,c], Ya.D. Sergeyev[b,c,*]

[a]*Dipartimento di Matematica, Università degli Studi di Bari, Italy*
[b]*DIMES, Università della Calabria, Italy*
[c]*Department of Software and Supercomputing Technologies Lobachevsky State University of Nizhni Novgorod, Russia*

Abstract

A well-known drawback of algorithms based on Taylor series formulae is that the explicit calculation of higher order derivatives formally is an over-elaborate task. To avoid the analytical computation of the successive derivatives, numeric and automatic differentiation are usually used. A recent alternative to these techniques is based on the calculation of higher derivatives by using the Infinity Computer – a new computational device allowing one to work numerically with infinities and infinitesimals. Two variants of a one-step multi-point method closely related to the classical Taylor formula of order three are considered. It is shown that the new formula is order three accurate, though requiring only the first two derivatives of $y(t)$ (rather than three if compared with the corresponding Taylor formula of order three). To get numerical evidence of the theoretical results, a few test problems are solved by means of the new methods and the obtained results are compared with the performance of Taylor methods of order up to four.

Keywords: Ordinary differential equations, initial value problems, Taylor methods, numerical infinitesimals, Infinity Computer

[☆]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". 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".

[*]Corresponding author

Email addresses: `pierluigi.amodio@uniba.it` (P. Amodio), `felice.iavernaro@uniba.it` (F. Iavernaro), `francesca.mazzia@uniba.it` (F. Mazzia), `muhametzhanov.m@gmail.com` (M.S. Mukhametzhanov), `yaro@dimes.unical.it` (Ya.D. Sergeyev)

1. Introduction

The Taylor series method is one of the earliest algorithms to approximate the solution of initial value problems

$$\begin{cases} y' = f(t, y), \quad t \in [t_0, T], \\ y(t_0) = y_0, \end{cases} \tag{1}$$

where $f : [t_0, T] \times \mathbb{R}^n \to \mathbb{R}^n$ is assumed sufficiently differentiable.

Newton and Euler describe this approach in their seminal works of the 18th century. Since then, many authors mention Taylor series methods and some codes have been developed for both ODEs and DAEs: within the recent literature, we mention [3, 4, 5, 15, 53, 54].

A well-known drawback of the algorithms based on Taylor series formulae is that the explicit calculation of higher order derivatives formally is an over-elaborate task, especially when the dimension n of the system is not small. To avoid the analytical computation of the successive partial derivatives involved in the truncated Taylor expansion of f, a numerical differentiation approach has been often considered (see, for example, [24, 25]). A further interest in Taylor series methods stemmed from considering automatic rather than numerical differentiation, which makes use of specific tools based on the involved elementary functions (see [29]) and allows for a speed up of the overall computation.

We report two instances for which the use of Taylor series methods has proved to be a powerful tool:

- The analysis of the stability properties of equilibria and periodic orbits of dynamical systems often requires an accurate integration of the variational equations. Within this context, high-order Taylor series methods have been successfully exploited to correctly reproduce the highly oscillatory behavior of their solutions, avoiding extremely small stepsizes during the integration procedure. In most cases the variational equations are slight modifications of the original ones, so that it is possible to formulate the integration algorithm for both systems with little added effort.

- In some physical problems [1, 2, 22] it is important to approximate the solution with a very high precision, as in the determination of normal forms of differential systems, initial conditions for periodic problems, numerical detection of periodic orbits, computation of physical constants, etc. The Taylor method, just by increasing the degree of the formulae, permits high-precision integration, provided a multi-precision library is also used.

A recent alternative to numeric and automatic differentiation is based on the calculation of higher derivatives by using the Infinity Computer which is equipped with a new numeral system (see [30, 33, 38, 32, 42, 47]) for performing numerical computations with infinite and infinitesimal quantities. The possibility to work with numerical infinitesimals allows one both to calculate the exact values of the derivatives numerically without finding the respective derivatives analytically and to work with infinitesimal stepsizes. The first attempts to use the Infinity Computer in this direction have been done in [39, 43, 45].

In order to see the place of the new approach in the historical panorama of ideas dealing with infinite and infinitesimal, see [17, 18, 19, 26, 35, 37, 49]. In particular, connections of the new approach with bijections are studied in [19] and metamathematical investigations on the theory and its non-contradictory identification can be found in [18]. The new methodology has been successfully used in such fields as numerical differentiation and optimization (see [9, 39, 56]), fractals (see [13, 14, 31, 34, 41, 48]), models for percolation and biological processes (see [13, 14, 52, 41]), hyperbolic geometry (see [20, 21]), infinite series (see [16, 35, 40, 55]), set theory, lexicographic ordering, and Turing machines (see [37, 46, 44, 49, 50, 50]), cellular automata (see [10, 11, 12]), etc.

The paper is structured as follows. Section 2 gives a brief introduction into the work with numerical infinitesimals and infinities on the Infinity Computer. Section 3 introduces the new methods. Convergence and stability analysis of the new algorithms is performed in Section 4. Some numerical illustrations are provided in Section 5. Finally, Section 6 concludes the paper.

We stress that the action played by the Infinity Computer only concerns the accurate evaluation of the derivatives appearing in the expression of the methods. As a matter of fact, the variant of the standard Taylor methods we are going to introduce may be efficiently implemented in both standard and infinity floating-point arithmetic.

2. Numerical infinitesimals and infinities

In our everyday activities with finite numbers the *same* finite numerals[1] are used for *different* purposes (e.g., the same numeral 9 can be used to express the number of elements of a set, to indicate the position of an element in a sequence, and to execute practical computations). In contrast, when we face the necessity to work with infinities or infinitesimals, the situation changes drastically. In fact, in this case *different* numerals are used to work with infinities and infinitesimals in *different* situations. To illustrate this fact it is sufficient to mention that we use the symbol ∞ in standard analysis, ω for working with ordinals, $\aleph_0, \aleph_1, ...$ for dealing with cardinalities.

Many codes and theories dealing with infinite and infinitesimal quantities have a symbolic (not numerical) character. For instance, many versions of non-standard analysis (see [28]) are symbolic, since they have no numeral systems to express their numbers by a finite number of symbols (the finiteness of the number of symbols is necessary for organizing numerical computations). Namely, if we consider a finite n, any numeral used to express finite quantities and consisting

[1]There exists an important distinction between *numbers* and *numerals*. A *numeral* is a symbol (or a group of symbols) that represents a *number*. A *number* is a concept that a *numeral* expresses. The same number can be represented by different numerals. For example, the symbols '9', 'nine', 'IIIIIIIII', and'IX', are different numerals, but they all represent the same number.

of a finite number of symbols may be associated with it, such as, for example, $n = 72$, or $n = 30$. In contrast, if we consider a non-standard infinite m, then it is not clear which numerals can be used to assign a concrete value to m. Analogously, in non-standard analysis, if we consider an infinitesimal h then it is not clear which numerals consisting of a finite number of symbols can be used to assign a concrete value to h and to write $h = ...$ In fact, very often in non-standard analysis texts, a *generic* infinitesimal h is used and it is considered as a symbol, i.e., only symbolic computations can be done with it. Approaches of this kind leave unclear such issues, e.g., whether the infinite $1/h$ is integer or not or whether $1/h$ is the number of elements of an infinite set.

In order to allow one to execute *numerical* computations with different infinities and infinitesimals and to use the same numerals in all the situations (as it happens with numerals expressing finite quantities), a new computational methodology and the respective numeral system have been developed in [30, 33, 38]. This numeral system avoids indeterminate forms and situations similar to $\infty + 1 = \infty$ and $\infty - 1 = \infty$ providing results ensuring that if a is a numeral written in this numeral system then for any a (i.e., a can be finite, infinite, or infinitesimal) it follows $a + 1 > a$ and $a - 1 < a$.

The numeral system is based on a new infinite unit of measure expressed by the numeral ①, called *grossone*, that is introduced as the number of elements of the set of natural numbers. Concurrently with the introduction of ① in the mathematical language all other symbols (like ∞, Cantor's ω, $\aleph_0, \aleph_1, ...$, etc.) traditionally used to deal with infinities and infinitesimals are excluded from the language because ① and other numbers constructed with its help not only can be used instead of all of them but can be used with a higher accuracy. Analogously, when zero and the positional numeral system had been introduced in Europe, Roman numerals I, V, X, etc. had not been involved and new symbols 0, 1, 2, etc. have been used to express numbers. The new element – zero expressed by the numeral 0 – had been introduced by describing its properties in the form of axioms. Analogously, ① is introduced by describing its properties postulated by the Infinite Unit Axiom added to axioms for real numbers (see [33, 38] for a detailed discussion).

Let us see now how, thanks to the introduction of ① in place of the usual symbol ∞, one can write down different numerals expressing different infinities and infinitesimals and to execute computations with all of them. Indeterminate forms are not present and, for example, the following relations hold for infinite numbers ①, $①^{2.7}$ and $①^{-1}$, $①^{-2.7}$ (that are infinitesimals), as for any other (finite, infinite, or infinitesimal) number expressible in the new numeral system

$$0 \cdot ① = ① \cdot 0 = 0, \quad ① - ① = 0, \quad \frac{①}{①} = 1, \quad ①^0 = 1, \quad 1^{①} = 1, \quad 0^{①} = 0, \qquad (2)$$

$$0 \cdot ①^{-1} = ①^{-1} \cdot 0 = 0, \qquad ①^{-1} > 0, \qquad ①^{-2.7} > 0, \qquad ①^{-1} - ①^{-1} = 0,$$

$$\frac{①^{-1}}{①^{-1}} = 1, \qquad (①^{-1})^0 = 1, \qquad ① \cdot ①^{-1} = 1, \qquad ① \cdot ①^{-2} = ①^{-1},$$

$$\frac{①^{-2.7}}{①^{-2.7}} = 1, \quad \frac{①^{2.7}}{①} = ①^{1.7}, \quad \frac{①^{-1}}{①^{-2}} = ①, \quad ①^{2.7} \cdot ①^{-2.7} = 1.$$

The introduction of the numeral ① allows us to represent infinite and infinitesimal numbers in a unique framework and to work with all of them numerically on the Infinity Computer (see the patent [36]). For this purpose a numeral system similar to traditional positional numeral systems was introduced in [30, 33]. To construct a number C in the numeral positional system with base ①, we subdivide C into groups corresponding to powers of ①:

$$C = c_{p_m}①^{p_m} + \ldots + c_{p_1}①^{p_1} + c_{p_0}①^{p_0} + c_{p_{-1}}①^{p_{-1}} + \ldots + c_{p_{-k}}①^{p_{-k}}. \qquad (3)$$

Then, the record

$$C = c_{p_m}①^{p_m} \ldots c_{p_1}①^{p_1} c_{p_0}①^{p_0} c_{p_{-1}}①^{p_{-1}} \ldots c_{p_{-k}}①^{p_{-k}} \qquad (4)$$

represents the number C, where all numerals $c_i \neq 0$, they belong to a traditional numeral system and are called *grossdigits*. They express finite positive or negative numbers and show how many corresponding units $①^{p_i}$ should be added or subtracted in order to form the number C. Note that in order to have a possibility to store C in the computer memory, values k and m should be finite.

Numbers p_i in (4) are sorted in the decreasing order with $p_0 = 0$

$$p_m > p_{m-1} > \ldots > p_1 > p_0 > p_{-1} > \ldots p_{-(k-1)} > p_{-k}.$$

They are called *grosspowers* and they themselves can be written in the form (4). In the record (4), we write $①^{p_i}$ explicitly because in the new numeral positional system the number i in general is not equal to the grosspower p_i. This gives the possibility to write down numerals without indicating grossdigits equal to zero.

The term having $p_0 = 0$ represents the finite part of C since $c_0①^0 = c_0$ (see (2)). Terms having finite positive grosspowers represent the simplest infinite parts of C. Analogously, terms having negative finite grosspowers represent the simplest infinitesimal parts of C. For instance, the number $①^{-1} = \frac{1}{①}$ mentioned above is infinitesimal. Note that all infinitesimals are not equal to zero. In particular, $\frac{1}{①} > 0$ since it is a result of division of two positive numbers.

A number represented by a numeral in the form (4) is called *purely finite* if it has neither infinite nor infinitesimals parts. For instance, 14 is purely finite and $14 + 5.3①^{-1.5}$ is not. All grossdigits c_i are supposed to be purely finite. Purely finite numbers are used on traditional computers and for obvious reasons have a special importance for applications. All of the numbers introduced above can be grosspowers, as well, giving thus a possibility to have various combinations of quantities and to construct terms having a more complex structure.

Many numerical methods for solving ODEs require the computation of the derivatives of an unknown function $y(t)$ at some specific points. In particular, this is the case with methods based on Taylor expansion. Let us see how the usage of infinitesimals on the Infinity Computer allows one to calculate exact derivatives of $y(t)$ numerically.

Let us denote by $y^{(k)}(t_i)$ the k-th derivative of the solution $y(t)$ at the point t_i and suppose that $f(t, y)$ assumes purely finite values at purely finite t and y. It has been shown in [43] that, in order to calculate the k-th derivative at the point t_i, k infinitesimals steps from the point t_i using the Euler formula with $h = ①^{-1}$ should be executed as follows

$$y_{i,1} = y_i + ①^{-1} f(t_i, y_i), \quad y_{i,2} = y_{i,1} + ①^{-1} f(t_i + ①^{-1}, y_{i,1}), \ \ldots$$
$$y_{i,k} = y_{i,k-1} + ①^{-1} f(t_i + (k-1)①^{-1}, y_{i,k-1}).$$

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

$$\Delta^k_{①^{-1}} = \sum_{j=0}^{k} (-1)^j \binom{k}{j} y_{i,k-j}, \tag{5}$$

where $y_{i,0} = y_i$. We obtain

$$y^{(k)}(t_i) = \frac{\Delta^k_{①^{-1}}}{①^{-k}} + O(①^{-1}). \tag{6}$$

Since the error of the approximation is $O(①^{-1})$, the finite part of the value $\frac{\Delta^k_{①^{-1}}}{①^{-k}}$ gives us the *exact* derivative $y^{(k)}(t_i)$. For a more detailed description of the numerical computation of exact derivatives on the Infinity Computer see [39, 43].

3. Definition of the new method

The method we are interested in is a one-step multi-point method of the form $(w_1, y_1) = \Phi_h(w_0, y_0)$ closely related to the classical Taylor formula of order three. Here h stands for the integration stepsize and w_1 and y_1 are approximations to $y(t_1)$, with $t_1 = t_0 + h$. More specifically, the extra-point w_1 is to be meant as a preliminary low order approximation to $y(t_1)$ which is then exploited to derive the more accurate approximation y_1 (see Section 3.2 for more details).

As will be shown in the next section, the new formula is order three accurate, though requiring only the first two derivatives of $y(t)$ (rather than three if compared with the corresponding Taylor formula of order three).

We begin with introducing an *auxiliary* method, denoted by $\hat{y}_1 = \widehat{\Phi}_h(y_0)$, which will prove very helpful to properly define method Φ_h as well as to study its convergence and stability properties. Methods $\widehat{\Phi}_h$ and Φ_h first appeared in [51] where they are referred to as methods 1.3 and 1.4, respectively. For sake of simplicity, but without loss of generality, in the sequel we assume that problem (1) is scalar and autonomous.[2] For later use, we list the shape of the first four

[2]Section 5 also includes results pertaining to the non-autonomous and vector cases (see also Remark 1). Notice that we avoid the computation of high-order mixed derivatives on the Infinity Computer, which will be the object of a future research.

derivatives of the solution $y(t)$ of (1) evaluated at t_0, in terms of the function f and its derivatives:

$$\begin{aligned}
y'(t_0) &= f(y_0),\\
y''(t_0) &= f'(y_0)f(y_0),\\
y'''(t_0) &= f''(y_0)(f(y_0))^2 + (f'(y_0))^2 f(y_0),\\
y^{(iv)}(t_0) &= f'''(y_0)(f(y_0))^3 + 4f''(y_0)f'(y_0)(f(y_0))^2 + (f'(y_0))^3 f(y_0).
\end{aligned} \tag{7}$$

On the Infinity Computer, the derivatives appearing in (7) are evaluated with the aid of formulae (5)-(6). In standard arithmetic they are provided analytically, even though we also illustrate the effects of approximating them by suitable divided differences (see Section 5).

The following result regards the computational cost to compute the Taylor coefficients.

Theorem 1 ([3, 27])**.** *If the evaluation of $f(y)$ involves r elementary functions, the computational complexity of the evaluation of $f(y)$, $f'(y)$, $\ldots f^{(s-1)}(y)$ is*

$$C = rs^2 + O(s). \tag{8}$$

The efficiency of Taylor methods as compared with classical Runge–Kutta methods has been discussed in [3]. For a numerical approach to the computation of the coefficients in Taylor expansions see [25].

3.1. Definition of method $\hat{y}_1 = \widehat{\Phi}_h(y_0)$

We first consider the standard Taylor formula of order two to obtain an initial guess, say v_1, of $y(t_1)$ (see (7)):

$$v_1 = y_0 + hy'(t_0) + \frac{h^2}{2}y''(t_0) \equiv y_0 + hf(y_0) + \frac{h^2}{2}f'(y_0)f(y_0). \tag{9}$$

We use again the same formula to obtain an approximation to the solution of (1), denoted by $p_2(t)$, in a neighborhood of t_1:

$$p_2(t) = v_1 + f(v_1)(t-t_1) + \frac{1}{2}f'(v_1)f(v_1)(t-t_1)^2. \tag{10}$$

Rather than advancing the solution in time, we exploit the information brought by $p_2(t)$ to improve the accuracy of the numerical solution at time t_1. To this end, we first recast $p_2(t)$ as a polynomial expanded around t_0

$$\begin{aligned}
p_2(t) &= v_1 + f(v_1)(t-t_0+t_0-t_1) + \frac{1}{2}f'(v_1)f(v_1)(t-t_0+t_0-t_1)^2\\
&= v_1 - hf(v_1) + \frac{h^2}{2}f'(v_1)f(v_1) + (1-hf'(v_1))\,f(v_1)(t-t_0)\\
&\quad + \frac{1}{2}f'(v_1)f(v_1)(t-t_0)^2,
\end{aligned} \tag{11}$$

and then we blend the coefficients of the polynomial $p_2(t)$ with the corresponding ones in the classical Taylor formula to form a new second degree polynomial $q_2(t)$, namely

$$\begin{aligned} q_2(t) \quad &= \quad \alpha_0 y_0 + (1-\alpha_0)\big(v_1 - hf(v_1) + \frac{h^2}{2} f'(v_1)f(v_1)\big) \\ &\quad +\big(\alpha_1 f(y_0) + (1-\alpha_1)\,(1 - hf'(v_1))\,f(v_1)\big)(t-t_0) \\ &\quad +\frac{1}{2}\big(\alpha_2 f'(y_0)f(y_0) + (1-\alpha_2)f'(v_1)f(v_1)\big)(t-t_0)^2, \end{aligned} \tag{12}$$

which will be used to advance the solution, by setting $\hat{y}_1 = \widehat{\Phi}_h(y_0) = q_2(t_1)$. The parameters α_i, $i = 0, 1, 2$, will be selected in order to improve the convergence and stability properties of the standard second order Taylor formula, as is discussed in Section 4. The use of convex combinations in (12) comes from imposing the consistency conditions up to order two. In other words, order two is achieved independently of the choice of the parameters α_i. Furthermore, without loss of generality, we assume $\alpha_0 = 1$ since it can be shown that the value of α_0 does not alter the shape of the resulting method. The following scheme then summarizes the implementation details of the method to construct the numerical approximation $\hat{y}_k \simeq y(t_k)$, with $t_k = t_0 + kh$.

```
$\hat{y}_0 = y_0$
$h = (T - t_0)/n$
for    $k = 1, \dots, n$
        $v_k = \hat{y}_{k-1} + hf(\hat{y}_{k-1}) + \frac{h^2}{2} f'(\hat{y}_{k-1})f(\hat{y}_{k-1})$
        $\hat{y}_k = \hat{y}_{k-1} + h\big(\alpha_1 f(\hat{y}_{k-1}) + (1-\alpha_1)\,(1 - hf'(v_k))\,f(v_k)\big)$
              $+ \frac{h^2}{2}\big(\alpha_2 f'(\hat{y}_{k-1})f(\hat{y}_{k-1}) + (1-\alpha_2)f'(v_k)f(v_k)\big)$          (13)
end
```

3.2. Definition of method $(w_1, y_1) = \Phi_h(w_0, y_0)$

To reduce the computational effort per step associated with the implementation of the method defined by $\widehat{\Phi}_h$, we consider a variant consisting in approximating the values of $f(\hat{y}_{k-1})$ and $f'(\hat{y}_{k-1})$ appearing in algorithm (13) by means of suitable known quantities available for free. More precisely, we assume that the very first step is performed by method $\widehat{\Phi}_h$, and we set $w_1 = v_1$ and $y_1 = \hat{y}_1 = \widehat{\Phi}_h(y_0)$.

At the second step, we avoid the evaluations of $f(y_1)$ and $f'(y_1)$, as required by (9), and instead approximate them by $f(w_1)$ and $f'(w_1)$ respectively, which we inherit from the previous step. More in general, to compute the subsequent approximations $y_k = \Phi_h(y_{k-1})$, $k = 2, 3, \dots$, we replace the quantities $f(y_{k-1})$ and $f'(y_{k-1})$, required by algorithm (13), by $f(w_{k-1})$ and $f'(w_{k-1})$ respectively. The implementation details of the method defined by Φ_h are summarized below.

```
$h = (T - t_0)/n$
$w_1 = y_0 + hf(y_0) + \frac{h^2}{2} f'(y_0)f(y_0)$
$y_1 = y_0 + h\big(\alpha_1 f(y_0) + (1-\alpha_1)\,(1 - hf'(w_1))\,f(w_1)\big)$
$\qquad + \frac{h^2}{2}\big(\alpha_2 f'(y_0)f(y_0) + (1-\alpha_2)f'(w_1)f(w_1)\big)$
for   $k = 2, \dots, n$                                                    (14)
    $w_k = y_{k-1} + hf(w_{k-1}) + \frac{h^2}{2} f'(w_{k-1})f(w_{k-1})$
    $y_k = y_{k-1} + h\big(\alpha_1 f(w_{k-1}) + (1-\alpha_1)\,(1 - hf'(w_k))\,f(w_k)\big)$
    $\qquad + \frac{h^2}{2}\big(\alpha_2 f'(w_{k-1})f(w_{k-1}) + (1-\alpha_2)f'(w_k)f(w_k)\big)$
end
```

Remark 1. The shape of Algorithms (13) and (14) does not change for vector-valued functions. In such a case, $f'(y)$ denotes the Jacobian matrix of $f(y)$. We also recall that a non-autonomous system $y' = f(t, y)$, with $y \in \mathbb{R}^n$, may be always recast as an autonomous system of the form $z' = F(z)$, with $z \in \mathbb{R}^{n+1}$, by setting:

$$z = \begin{pmatrix} y \\ t \end{pmatrix} \qquad \text{and} \quad F(z) = \begin{pmatrix} f(y) \\ 1 \end{pmatrix}.$$

Modulo this transformation, the two algorithms above may also handle the non-autonomous case.

4. Convergence and stability analysis

The analysis of the integrator described in algorithm (14) will be carried out by interpreting Φ_h as a perturbation of $\widehat{\Phi}_h$. For this reason, we begin with stating some preliminary results pertaining to this latter formula.

Theorem 2. *If the coefficients α_1 and α_2 satisfy*

$$\alpha_1 - \alpha_2 = \frac{1}{3} \tag{15}$$

the method $\widehat{\Phi}_h$ has order $p = 3$. In addition, if the coefficients α_1 and α_2 are selected as

$$\alpha_1 = \frac{5}{6}, \quad \alpha_2 = \frac{1}{2}, \tag{16}$$

(thus also satisfying (15)), $\widehat{\Phi}_h$ becomes the standard fourth-order Taylor formula when applied to the linear problem $y' = \lambda y$, $\lambda \in \mathbb{C}$, where $\mathbb{C}$ denotes the set of complex numbers.

Proof. The local truncation error associated with the method $\widehat{\Phi}_h$ at the first step is

$$\tau(h) = y(t_0+h) - \widehat{\Phi}_h(y_0) = \sum_{k\geq 0} \frac{y^{(k)}(t_0)}{k!}h^k - \widehat{\Phi}_h(y_0), \tag{17}$$

assuming f analytical. Thus, to estimate $\tau(h)$ we need to expand $\widehat{\Phi}_h(y_0)$ in powers of h. With reference to (13) with $k=1$, we consider the expansions

$$\begin{aligned}
f(v_1) &= f\big(y_0 + hf(y_0) + \tfrac{h^2}{2}f'(y_0)f(y_0)\big) \\
&= f(y_0) + f'(y_0)\big(hf(y_0) + \tfrac{h^2}{2}f'(y_0)f(y_0)\big) \\
&\quad + \tfrac{f''(y_0)}{2}\big(hf(y_0) + \tfrac{h^2}{2}f'(y_0)f(y_0)\big)^2 + O(h^3) \\
&= f(y_0) + hf(y_0)f'(y_0) \\
&\quad + \tfrac{h^2}{2}\big[f(y_0)(f'(y_0))^2 + (f(y_0))^2 f''(y_0)\big] + O(h^3),
\end{aligned}$$

$$\begin{aligned}
f'(v_1) &= f'\big(y_0 + hf(y_0) + \tfrac{h^2}{2}f'(y_0)f(y_0)\big) \\
&= f'(y_0) + hf(y_0)f''(y_0) \\
&\quad + \tfrac{h^2}{2}\big[f(y_0)f'(y_0)f''(y_0) + (f(y_0))^2 f'''(y_0)\big] + O(h^3).
\end{aligned}$$

Plugging them into the equation in (13) defining $\hat{y}_1$ yields

$$\begin{aligned}
\widehat{\Phi}_h(y_0) &= \hat{y}_1 = y_0 + f(y_0)h + \frac{1}{2}f(y_0)f'(y_0)h^2 \\
&\quad + \frac{\alpha_1-\alpha_2}{2}f(y_0)\big((f'(y_0))^2 + f(y_0)f''(y_0)\big)h^3 + O(h^4).
\end{aligned} \tag{18}$$

Inserting (18) into (17) and taking into account relations (7) we deduce that the method defined by $\widehat{\Phi}_h$ has order three if condition (15) is fulfilled.[3]

In the specific case where the problem is linear, namely $f(y) = \lambda y$ and hence $f^{(k)}(y) = \lambda^k y$, a direct computation based upon the previous argument shows that

$$\tau(h) = \big(\frac{1}{3!} - \frac{\alpha_1}{2} + \frac{\alpha_2}{2}\big)(h\lambda)^3 y_0 + \big(\frac{1}{4!} + \frac{1}{4} - \frac{\alpha_1}{2} + \frac{\alpha_2}{4}\big)(h\lambda)^4 y_0 + \sum_{k\geq 5}\frac{y^{(k)}(t_0)}{k!}h^k,$$

and order four is achieved by imposing

$$\begin{cases} \dfrac{\alpha_1}{2} - \dfrac{\alpha_2}{2} = \dfrac{1}{3!}, \\[2mm] \dfrac{\alpha_1}{2} - \dfrac{\alpha_2}{4} = \dfrac{1}{4!} + \dfrac{1}{4}, \end{cases}$$

which has solution (16). □

[3]One may check that it is not possible to achieve order four under the assumption that the coefficients α_i are independent of the problem at hand.

More in general, method $\widehat{\Phi}_h$ has order four when applied to autonomous linear problems $y' = Ay + b$. Consequently, an increase of order is also experienced numerically for nonlinear problems when the dynamics takes place in a neighborhood of an equilibrium point where the Lyapunov first approximation theorem holds true (see Problem 2 in Section 5).

The linear stability analysis amounts to study the (global) asymptotic behavior of the sequence $\hat{y}_n = \widehat{\Phi}_h(\hat{y}_{n-1})$ when the method is applied to the well-known linear test equation $y' = \lambda y$, with $\lambda \in \mathbb{C}$. In such an event, as has been shown in Theorem 2, the method $\widehat{\Phi}_h$ is equivalent to the fourth order Taylor formula and, as a direct consequence, we can state the following result.

Corollary 1. *Method $\widehat{\Phi}_h$ share the same linear stability properties of the fourth-order Taylor method.*

More specifically, setting $q = h\lambda$, we have $\hat{y}_n = \widehat{R}(q)^n y_0$, where

$$\widehat{R}(q) = \sum_{k=0}^{4} \frac{q^k}{k!}$$

is the stability function. We recall that the region of absolute stability of a generic method providing a sequence y_n when applied to the linear test equation, is defined as

$$\mathcal{D} = \{q \in \mathbb{C} : \, y_n \to 0, \text{ as } n \to \infty\}$$

In our case, we see that $q \in \mathcal{D} \Leftrightarrow |\widehat{R}(q)| < 1$.

We now move to the study of the method corresponding to the map Φ_h, introduced in Section 3.2. In particular, we will take advantage of the results previously obtained for the method defined by $\widehat{\Phi}_h$, by regarding Φ_h as a perturbation of $\widehat{\Phi}_h$.

Lemma 1. *Under the assumption* (15), *the sequences* $(v_k, \hat{y}_k)$ *and* (w_k, y_k) *defined in algoritms* (13) *and* (14) *respectively, are related as*

$$w_k = v_k + O(h^4), \qquad y_k = \hat{y}_k + O(h^4), \quad \textit{with } k = 0, 1, \ldots, N, \tag{19}$$

where N is a positive constant integer, independent of h.

Proof. We use an induction argument on the index k. For $k = 1$, (19) is obviously true since, by definition, (13) and (14) provide the same approximations ($w_1 = v_1$ and $y_1 = \hat{y}_1$). Assume that property (19) holds true for $k - 1$. From the proof of Theorem 2 we deduce that

$$\begin{aligned} \hat{y}_k \;\; = \;\; & \hat{y}_{k-1} + hf(\hat{y}_{k-1}) + \frac{h^2}{2} f(\hat{y}_{k-1}) f'(\hat{y}_{k-1}) \\ & + \frac{h^3}{3!}\big(f(\hat{y}_{k-1})(f'(\hat{y}_{k-1}))^2 + (f(\hat{y}_{k-1}))^2 f''(\hat{y}_{k-1})\big) + O(h^4), \end{aligned}$$

thus, comparing with the definition of v_k in (13), we conclude that

$$\hat{y}_k - v_k = O(h^3), \qquad \text{for any } \; k = 0, 1, \ldots. \tag{20}$$

Exploiting the induction hypothesis and (20), we finally get

$$\begin{aligned}
w_k &= y_{k-1} + f(w_{k-1})h + f(w_{k-1})f'(w_{k-1})\frac{h^2}{2} \\
&= \hat{y}_{k-1} + O(h^4) + f(v_{k-1} + O(h^4))h \\
&\quad + f(v_{k-1} + O(h^4))f'(v_{k-1} + O(h^4))\frac{h^2}{2} \\
&= \hat{y}_{k-1} + f(v_{k-1})h + f(v_{k-1})f'(v_{k-1})\frac{h^2}{2} + O(h^4) \\
&= v_k + O(h^4),
\end{aligned}$$

and analogously

$$\begin{aligned}
y_k &= y_{k-1} + \big(\alpha_1 f(w_{k-1}) + (1-\alpha_1)\left(1 - hf'(w_k)\right) f(w_k)\big)\big)h \\
&\quad + \big(\alpha_2 f'(w_{k-1})f(w_{k-1}) + (1-\alpha_2)f'(w_k)f(w_k)\big)\frac{h^2}{2} \\
&= \hat{y}_{k-1} + O(h^4) + \big[\alpha_1 f(v_{k-1} + O(h^4)) \\
&\qquad\qquad + (1-\alpha_1)\left(1 - hf'(v_k + O(h^4))\right) f(v_k + O(h^4)))\big]h \\
&\quad + \big[\alpha_2 f'(v_{k-1} + O(h^4))f(v_{k-1} + O(h^4)) \\
&\qquad + (1-\alpha_2)f'(v_k + O(h^4))f(v_k + O(h^4))\big]\frac{h^2}{2} \\
&= \hat{y}_{k-1} + \big(\alpha_1 f(v_{k-1}) + (1-\alpha_1)\left(1 - hf'(v_k)\right) f(v_k)\big)\big)h \\
&\quad + \big(\alpha_2 f'(v_{k-1})f(v_{k-1}) + (1-\alpha_2)f'(v_k)f(v_k)\big)\frac{h^2}{2} + O(h^4) \\
&= \hat{y}_{k-1} + \big(\alpha_1 f(\hat{y}_{k-1} + O(h^3)) + (1-\alpha_1)\left(1 - hf'(v_k)\right) f(v_k)\big)\big)h \\
&\quad + \big[\alpha_2 f'(\hat{y}_{k-1} + O(h^3))f(\hat{y}_{k-1} + O(h^3)) \\
&\qquad + (1-\alpha_2)f'(v_k)f(v_k)\big]\frac{h^2}{2} + O(h^4) \\
&= \hat{y}_{k-1} + \big(\alpha_1 f(\hat{y}_{k-1}) + (1-\alpha_1)\left(1 - hf'(v_k)\right) f(v_k)\big)\big)h \\
&\quad + \big(\alpha_2 f'(\hat{y}_{k-1})f(\hat{y}_{k-1}) + (1-\alpha_2)f'(v_k)f(v_k)\big)\frac{h^2}{2} + O(h^4) \\
&= \hat{y}_k + O(h^4).
\end{aligned}$$

This completes the proof. □

The following result then comes from standard computation, by letting now the integer N in (19) to increase as h decreases, under the constraint that $t = t_0 + Nh$ is a fixed time in the integration interval $[t_0, t_f]$.

Theorem 3. *The method defined by the map Φ_h and described in algorithm (14) has order of convergence $p = 3$, that is*

$$|y_N - y(t)| = O(h^3), \qquad \textit{with} \;\; h = \frac{t - t_0}{N}. \tag{21}$$

We observe that a result analogous to (21) applies to the error $|y_N - \hat{y}_N|$ and consequently, unlike $\widehat{\Phi}_h$, the method Φ_h does not increase its order when applied to linear problems.

Concerning the linear stability analysis, the application of the method defined by Φ_h to the test equation $y' = \lambda y$ yields

$$\begin{aligned}
w_k &= y_{k-1} + q w_{k-1} + \frac{1}{2} q^2 w_{k-1} \\
y_k &= y_{k-1} + \alpha_1 q w_{k-1} + (1-\alpha_1) q (1-q) w_k + \frac{1}{2}\alpha_2 q^2 w_{k-1} + \frac{1}{2}(1-\alpha_2) q^2 w_k,
\end{aligned}$$

or, in matrix form,

$$\begin{pmatrix} 1 & 0 \\ (1-\alpha_1)q(1-q) - \frac{1}{2}(1-\alpha_2)q^2 & 1 \end{pmatrix} \begin{pmatrix} w_k \\ y_k \end{pmatrix} = \begin{pmatrix} q + \frac{1}{2}q^2 & 1 \\ \alpha_1 q + \frac{1}{2}\alpha_2 q^2 & 1 \end{pmatrix} \begin{pmatrix} w_{k-1} \\ y_{k-1} \end{pmatrix}.$$

Inverting the matrix at the left-hand side, we arrive at

$$z_k = R(q) z_{k-1},$$

with $z_k = (w_k, y_k)^\top$, and the real matrix $R(q) = \big(r_{ij}(q)\big)$ defined as

$$\begin{aligned}
r_{11}(q) &= \frac{q^2}{2} + q, \\
r_{12}(q) &= 1, \\
r_{21}(q) &= (\frac{\alpha_1}{2} + \frac{\alpha_2}{4} - \frac{3}{4})q^4 + (\frac{\alpha_1}{2} + \frac{\alpha_2}{2} - 1)q^3 + (\frac{\alpha_2}{2} - \alpha_1 + 1)q^2 + \alpha_1 q, \\
r_{22}(q) &= (\alpha_1 + \frac{\alpha_2}{2} - \frac{3}{2})q^2 + (1-\alpha_1)q + 1.
\end{aligned} \tag{22}$$

Denoting by $\lambda_1(q)$ and $\lambda_2(q)$ the two eigenvalues of $R(q)$, it turns out that the absolute stability region of the method Φ_h is given by

$$\mathcal{D} \;=\; \left\{ q \in \mathbb{C} : \max_{i=1,2} |\lambda_i(q)| < 1 \right\}.$$

Figure 1 displays the absolute stability regions related to methods $\widehat{\Phi}_h$ (i.e., for linear problems, the fourth order Taylor method), and Φ_h for the choice of parameters as in (16). The Taylor formula of order two has also been considered for comparison purposes.

5. Numerical illustrations

To get numerical evidence of the theoretical results presented above, we solve a few test problems by means of the methods defined at (13) and (14) and compare their performance with Taylor methods of order up to four. Such comparisons are carried out by evaluating the error in the numerical approximations at the end of the time integration interval t_f. As a reference solution against which to measure the obtained accuracy, we use the theoretical solution of the problem when available, or a very accurate numerical solution obtained in

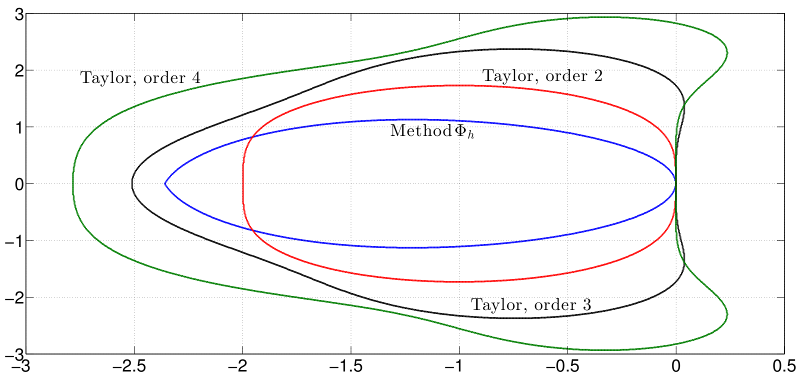


Figure 1: Absolute stability region of method Φ_h compared with those corresponding to the Taylor methods of order 2, 3, and 4.

Matlab with the aid of a solver in the ODE suite. For a m-dimensional problem, we use the following mixed-type error

$$E = \max_{1\le i\le m} \frac{|y^{(i)}(t_f) - y_N^{(i)}|}{1+|y^{(i)}(t_f)|}, \tag{23}$$

where $y^{(i)}(t_f)$ denotes the ith component of the reference solution evaluated at time t_f and $y_N^{(i)}$ is the corresponding numerical approximation ($t_f = t_0 + Nh$).

Both the Infinity Computer and analytic differentiation lead to an accurate approximation of the derivatives, thus yielding equivalent results. Consequently, we do not provide comparisons between these two different implementation procedures.

5.1. Problem 1

Consider the scalar (non-autonomous) initial value problem

$$\begin{cases} y' = \dfrac{\cos(\pi t)}{1+y}, \quad t \in [0,\pi], \\ y(0) = 0, \end{cases} \tag{24}$$

admitting solution

$$y(t) = \sqrt{\frac{2}{\pi}\sin(\pi t) + 1} - 1.$$

We solve problem (24) for decreasing values of the stepsize h

$$h_n = \frac{h_0}{2^n}, \quad \text{with} \quad h_0 = \frac{\pi}{20}, \quad \text{and} \quad n = 0,1,2,\dots,9,$$

and compare the numerical approximations at the end of the time interval with the exact one, according to formula (23). Figure 2 summarizes the obtained results. As is expected, the errors produced by the new methods Φ_h and $\widehat{\Phi}_h$ decay

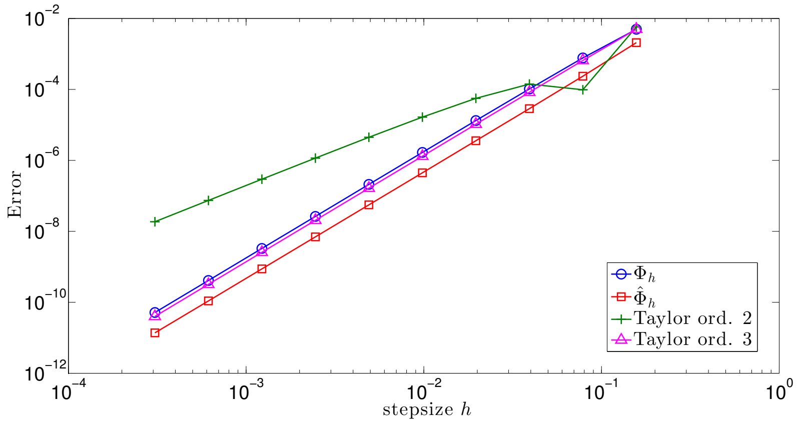


Figure 2: Problem 1. Errors versus stepsize.

with order three with respect to the stepsize. Though avoiding the computation of $y''(t)$, the performance of both methods is analogous to the third-order Taylor formula. It is worth noting that the implementation of method Φ_h requires precisely the same computational cost as the second-order Taylor formula, but achieving much better results (see Figure 2).

As was emphasized in the introduction, if the two methods are implemented on the Infinity Computer, the user may avoid to provide the analytic expression of the derivative $f'(y)$, which is instead obtained with the aid of the first order difference formula

$$f'(y) \approx \frac{f(y+\delta) - f(y)}{\delta}, \tag{25}$$

by setting $\delta = ①^{-1}$. Of course, in case of vector or non-autonomous systems, a formula equivalent to (25) is used to evaluate the first partial derivatives of the Jacobian matrix (see Remark 1). We recall that this choice yields the exact value of $f'(y)$ up to machine precision. Therefore, as a further experiment, it makes sense to see how a numerical approximation of the derivatives corresponding to decreasing values of the parameter δ, now taken as a positive real number in the standard arithmetic, may affect the overall behavior of the integrator, as compared to the choice $\delta = ①^{-1}$, also considering that for all these choices, including the latter one, the computational effort remains unchanged.

In Figure 3, we compare the performance of method Φ_h, for $\delta = 10^{-5}$, 10^{-7}, 10^{-9}, 10^{-11}, 10^{-13}, and $\delta = ①^{-1}$. Inside the range of the prescribed stepsizes, we see that choosing a value of δ not sufficiently small results in an eventual loss of convergence rate. As an example, for $\delta = 10^{-5}$ this happens when h becomes as small as 10^{-3} (solid line with asterisks in the picture). This simply means that, for $h < 10^{-3}$, the derivatives are not accurately computed and a smaller value of δ is then required to recover the order three convergence rate. Improvements are in fact obtained by reducing the value of δ to 10^{-7} and 10^{-9}.

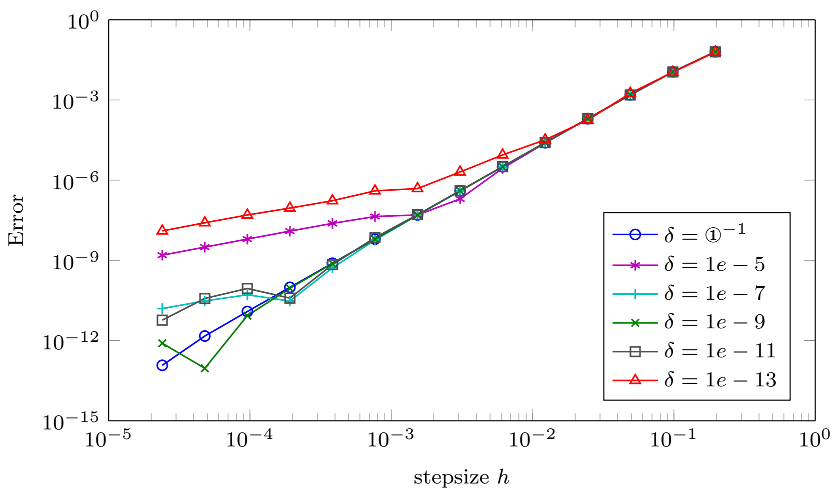


Figure 3: Problem 1. Errors generated by method Φ_h for different choices of the parameter δ to approximate the required derivatives of f, according to formula (25).

Contrary to what is expected, a further reduction of the discretization step δ in (25) results in a loss of efficiency: for $\delta = 10^{-13}$ the performance of the method is poorer than for the largest value considered. The reason is that approximating the derivative by means of formulae such as (25) may lead to an ill-conditioned problem for small values of the parameter δ due to cancelation issues related to the difference at numerator. Table 1 reports the errors in the approximation of $y'' = f'(y)f(y)$ in the last computed point, obtained by replacing $f'(w_n)$ with the corresponding first order difference formula (see (7) and (14)). A loss of significant digits is experienced starting from $\delta = 10^{-9}$, independently of the stepsize h used. This unpleasant outcome is an effect of the use of finite arithmetic, and is responsible for the eventual order reduction phenomenon inferred from Figure 3 and discussed above.

This is not the case when the same computation is carried out on the Infinity Computer. In fact, while the computer representation of $x \pm \delta$, with x a floating point number and $\delta \in \mathbb{R}$, produces an error, the quantities $x \pm ①$ are precisely represented and thus do not give rise to any digit cancelation phenomenon. For completeness, we also report, in Figure 4, the result obtained by approximating the first derivative by means of the second order difference formula

$$f'(y) \approx \frac{f(y+\delta) - f(y-\delta)}{2\delta}, \tag{26}$$

which is more frequently used by codes when a numerical evaluation of the Jacobian matrix $f'(y)$ is required.[4] We can see that an analogous reduction

[4]For $\delta = ①^{-1}$ we continue to use formula (25), which is the one implemented on the Infinity Computer.

h \ δ	10^{-5}	10^{-7}	10^{-9}	10^{-11}	10^{-13}
$\pi/2^4$	2.36e-05	2.37e-07	2.61e-08	1.00e-05	1.56e-03
$\pi/2^5$	3.86e-05	3.87e-07	1.92e-08	2.57e-06	1.57e-03
$\pi/2^6$	5.39e-05	5.35e-07	1.19e-07	3.87e-05	2.33e-03
$\pi/2^7$	6.77e-05	6.80e-07	3.68e-07	3.64e-05	1.80e-03
$\pi/2^8$	5.99e-05	6.01e-07	4.17e-07	1.86e-05	2.96e-03
$\pi/2^9$	5.65e-05	5.58e-07	3.35e-07	4.28e-05	7.18e-03
$\pi/2^{10}$	5.49e-05	5.43e-07	4.55e-07	4.44e-05	5.76e-03
$\pi/2^{11}$	5.41e-05	5.35e-07	2.93e-07	3.59e-05	1.66e-04
$\pi/2^{12}$	5.37e-05	5.38e-07	4.63e-07	2.05e-05	4.60e-04
$\pi/2^{13}$	5.35e-05	5.37e-07	5.23e-07	3.45e-05	2.46e-03
$\pi/2^{14}$	5.35e-05	5.28e-07	2.14e-07	4.62e-05	6.34e-03
$\pi/2^{15}$	5.34e-05	5.30e-07	1.50e-07	5.17e-05	7.17e-03
$\pi/2^{16}$	5.34e-05	5.28e-07	3.69e-07	5.51e-05	5.84e-03
$\pi/2^{17}$	5.34e-05	5.27e-07	3.22e-07	3.87e-05	2.70e-03

Table 1: Errors in the approximation of y'' in the last computed point, generated by replacing f' with the corresponding first order difference formula (25).

of order takes place in this case as well. This means that the loss of accuracy cannot be prevented by improving the accuracy of the discretization formula, but is an unavoidable outcome of the standard floating-point arithmetic.

5.2. Problem 2

In Theorem 2 we have shown the special feature of method $\widehat{\Phi}_h$ to become a fourth order formula, actually the fourth order Taylor formula, when applied to a linear problem. One may argue that the performance of the method may benefit from this property even when applied to nonlinear systems whose dynamics takes place in a neighborhood of an equilibrium point. To illustrate this aspect, we consider the dynamics of a pendulum under influence of gravity. Using Lagrangian mechanics, its motion can be described by the dimensionless nonlinear equation

$$\begin{cases} y_1' = y_2, \\ y_2' = -\sin y_1, \end{cases} \tag{27}$$

where y_1 is the angle that the pendulum forms with its stable rest position, and y_2 is the angular velocity. In case of small amplitude oscillations around the equilibrium point $(y_1, y_2) = (0, 0)$, the problem may be well described by its linearized version, namely the so called *harmonic oscillator* $y_1'' + y_1 = 0$, obtained through the approximation $\sin y_1 \approx y_1$. We compare the behavior of methods $\widehat{\Phi}_h$ and Φ_h with Taylor methods of order 2, 3 and 4 for the following decreasing values of stepsize:

$$h_n = \frac{h_0}{2^n}, \quad \text{with } h_0 = \frac{\pi}{5}, \quad \text{and } n = 0, 1, 2, \ldots, 7,$$

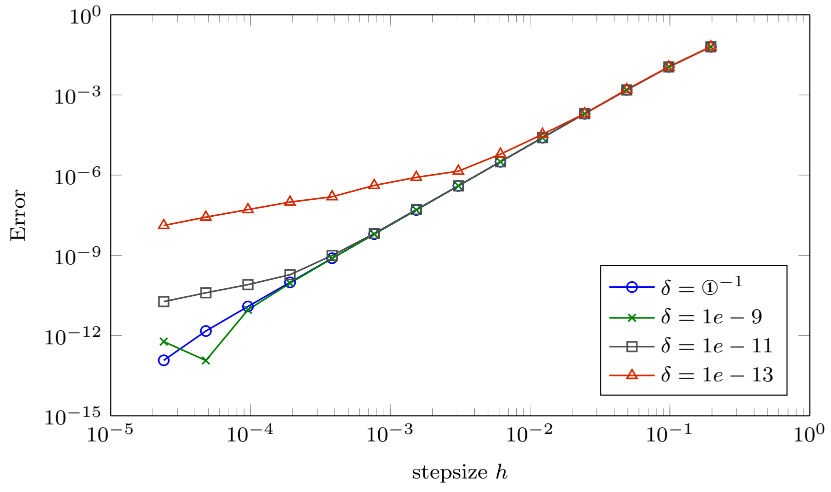


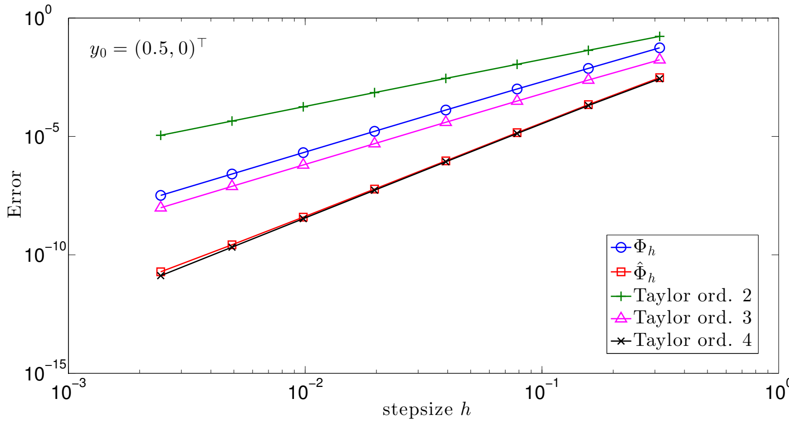
Figure 4: Problem 1. Errors generated by method Φ_h for different choices of the parameter δ to approximate the required derivatives of f, according to formula (26).

We use $[t_0, t_f] = [0, 2\pi]$ as integration interval and the two initial conditions $y_0 = (0.5, 0)^\top$ and $y_0 = (1, 0)^\top$ to simulate the system under the circumstance that the nonlinear part of (27) may be neglected or not.

Figure 5 summarizes the results. The picture at the top shows the error (23) versus the stepsize, when the pendulum undertakes mild oscillations so that its dynamic is essentially equivalent to that of a harmonic oscillator. We see that, in accord with Theorem 2, the behavior of method $\widehat{\Phi}_h$ is precisely the same as that of the fourth-order Taylor method, while Φ_h displays order three and yields errors similar to the corresponding Taylor formula. In the bottom picture in Figure 5 we show the results obtained after doubling the amplitude of oscillations of the pendulum. We see that, in this case, the nonlinear part of the vector field is no longer negligible even though the benefits of the order four do not completely evaporate. In fact, for large stepsizes, the behavior of method $\widehat{\Phi}_h$ and the Taylor method of order four is again quite similar and both methods produce comparable errors. As the stepsize decreases, the accuracy curve associated with method Φ_h departs from the order four slope and eventually reveals an error reduction rate typical of a third-order method.

6. Conclusions

We have introduced two one-step multi-point methods, based on the Taylor expansion, for the numerical solution of initial value problems. Both convergence and linear stability analysis suggest that the new formulae are competitive with respect to the corresponding Taylor formulae. In particular, both integrators are order three accurate, though requiring only the first two derivatives of $y(t)$.



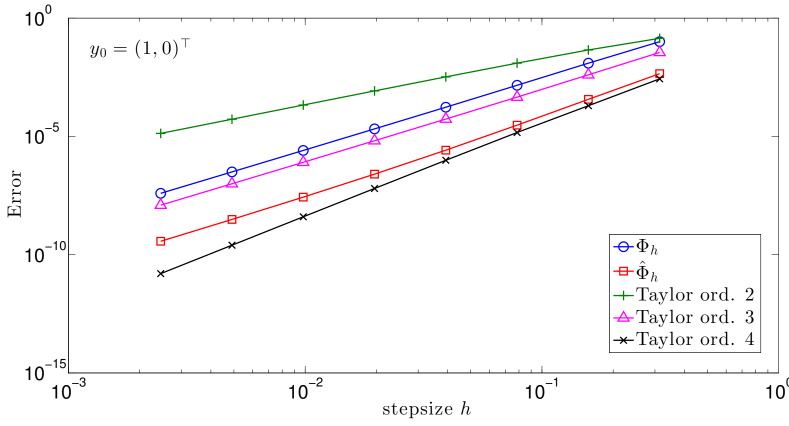


Figure 5: Problem 2. Errors versus stepsize.

The Infinity Computer, thanks to its ability of handling numerically both infinities and infinitesimals, has been proposed to compute the required derivatives as an alternative tool with respect to numeric and automatic differentiation. The numerical experiments show the good performance of the new methods as compared with the Taylor formula of order three. Furthermore, their implementation on the Infinity Computer avoids possible conditioning issues associated with the use of divided differences during the computation of the derivatives.

The presented approach leaves room to a number of possible investigations. For example, one can choose one or more free coefficients to confer some specific geometric properties on the resulting methods. This possibility has already been researched in the recent past (see, for example, [7, 8, 6]) and could make the formulae appropriate for solving conservative problems, such as Hamiltonian systems. As a further example, it is possible to associate with the proposed methods a stiffness detection strategy with a suitable mesh selection strategy, as suggested in [23] for explicit Runge-Kutta methods. These aspects will be faced in a future research.

References

[1] P. Amodio, L. Brugnano, and F. Iavernaro. Energy-conserving methods for Hamiltonian boundary value problems and applications in astrodynamics. *Adv. Comput. Math.*, 41:881–905, 2015.

[2] P. Amodio, C.J. Budd, O. Koch, G. Settanni, and E. Weinmüller. Asymptotical computations for a model of flow in saturated porous media. *Appl. Math. Comput.*, 237:155–167, 2014.

[3] R. Barrio. Performance of the Taylor series method for ODEs/DAEs. *Applied Mathematics and Computation*, 163(2):525–545, 2005.

[4] R. Barrio, F. Blesa, and M. Lara. VSVO formulation of the Taylor method for the numerical solution of ODEs. *Computers & Math. Applic.*, 50:93–111, 2005.

[5] R. Barrio, M. Rodríguez, A. Abad, and F. Blesa. Breaking the limits: The Taylor series method. *Appl. Math. Comput.*, 217:7940–7954, 2011.

[6] L. Brugnano and F. Iavernaro. *Line Integral Methods for Conservative Problems.* Monographs and Research Notes in Mathematics. Chapman and Hall/CRC, Boca Raton, FL, 2016.

[7] L. Brugnano and F. Iavernaro. Line integral methods which preserve all invariants of conservative problems. *J. Comput. Appl. Math.*, 236:3905–3919, 2012.

[8] L. Brugnano, F. Iavernaro, and D. Trigiante. Energy and QUadratic Invariants Preserving integrators based upon Gauss collocation formulae. *SIAM J. Numer. Anal.*, 50(6):2897–2916, 2012.

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

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

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

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

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

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

[15] Á. Jorba and M. Zou. A software package for the numerical integration of ODEs by means of high-order Taylor methods. *Experiment. Math.*, 14(1):99–117, 2005.

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

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

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

[19] M. Margenstern. 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, 2011.

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

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

[22] R. Martínez and C. Simó. Simultaneous binary collisions in the planar four-body problem. *Nonlinearity*, 12(4):903–930, 1999.

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

[24] E. Miletics and G. Molnárka. Taylor series method with numerical derivatives for initial value problems. *J. Comput. Meth. Sci. Eng.*, 4(1–2):105–114, 2004.

[25] E. Miletics and G. Molnárka. Implicit extension of Taylor series method with numerical derivatives for initial value problems. *Comput. Math. Appl.*, 50(7):1167–1177, 2005.

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

[27] R.A. Moore. *Interval Analysis.* Prentice-Hall, Englewood Cliffs, NJ, 1966.

[28] A. Robinson. *Non-standard Analysis.* Princeton Univ. Press, Princeton, 1996.

[29] D. Estévez Schwarz and R. Lamour. Projector based integration of DAEs with the Taylor series method using automatic differentiation. *J. Comput. Math Appl.*, 262:62–72, 2014.

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

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

[32] Ya.D. Sergeyev. 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, 2007.

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

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

[35] Ya.D. Sergeyev. 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, 2009.

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

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

[38] Ya.D. Sergeyev. 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, 2010.

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

[40] Ya.D. Sergeyev. 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, 2011.

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

[42] Ya.D. Sergeyev. 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, 2013.

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

[44] Ya.D. Sergeyev. 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, 2015.

[45] Ya.D. Sergeyev. 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, 2015.

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

[47] Ya.D. Sergeyev. 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, 2015.

[48] Ya.D. Sergeyev. 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, 2016.

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

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

[51] Ya.D. Sergeyev, M.S. Mukhametzhanov, F. Mazzia, F. Iavernaro, and P. Amodio. Numerical methods for solving initial value problems on the Infinity Computer. *International Journal of Unconventional Computing*, in press.

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

[53] V. Šátek, J. Kunovský, and A. Szöllös. Explicit and implicit Taylor series solutions of stiff systems. In F. Breitenecker and I. Troch, editors, *MathMod Vienna 2012 - 7th Vienna Conference on Mathematical Modelling*, 2012.

[54] S. Yalçinbaş and M. Sezer. A Taylor collocation method for the approximate solution of general linear Fredholm–Volterra integro-difference equations with mixed argument. *Appl. Math. Comput.*, 175:675–690, 2006.

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

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