UNIVERSIDAD DE CONCEPCIÓN

Centro de Investigación en Ingeniería Matemática (CI^{2}MA)

Approximate implicit Taylor methods for ODEs

Antonio Baeza, Raimund Bürger,
María Carmen Martí, Pep Mulet,
David Zorío

PREPRINT 2018-19

SERIE DE PRE-PUBLICACIONES

Approximate implicit Taylor methods for ODEs

Antonio Baeza · Raimund Bürger · M.Carmen Martí · Pep Mulet · David Zorío

Abstract In this work an approximate version of the implicit Taylor methods for the solution of initial-value problems for systems of ordinary differential equations (ODEs) is introduced. The approach is based on the idea of a previous work on explicit Taylor methods [A. Baeza, S. Boscarino, P. Mulet, G. Russo and D. Zorío, Approximate Taylor methods for ODEs, *Computers & Fluids*, **159** (2017), 156–166] and produces a method which only requires evaluations of the function that defines the ODE and its first order derivative, in contrast with the exact version that requires as many derivatives as the order of the method indicates. The numerical solution of the implicit equation that arises from the discretization by means of Newton's method is analyzed. The resulting algorithm is simpler to implement and has better performance than its exact counterpart.

Keywords Taylor methods · implicit schemes · explicit schemes · ODE integrators · approximate formulation

Mathematics Subject Classification (2000) 65L05 · 65L06 · 65L99

Antonio Baeza
Departament de Matemàtiques, Universitat de València, Av. Vicent Andrés Estellés, s/n, E-46100 Burjassot, Spain.
E-mail: antonio.baeza@uv.es

Raimund Bürger
CI^{2}MA and Departamento de Ingeniería Matemática, Facultad de Ciencias Físicas y Matemáticas, Universidad de Concepción, Casilla 160-C, Concepción, Chile.
E-mail: : rburger@ing-mat.udec.cl

M.Carmen Martí
Departamento de Matemática y Física Aplicadas, Facultad de Ingeniería, Universidad Católica de la Santísima Concepción, Casilla 297, Concepción, Chile.
E-mail: : mmarti@ucsc.cl

Pep Mulet
Departament de Matemàtiques, Universitat de València, Av. Vicent Andrés Estellés, s/n, E-46100 Burjassot, Spain.
E-mail: mulet@uv.es

David Zorío
CI^{2}MA, Universidad de Concepción, Casilla 160-C, Concepción, Chile.
E-mail: dzorio@ci2ma.udec.cl

1 Introduction

1.1 Scope

This work is related to numerical methods for the solution of the autonomous system of ordinary differential equations (ODEs)

$$\begin{aligned} &\boldsymbol{u}'(t) = \boldsymbol{f}\big(\boldsymbol{u}(t)\big), \quad t \in (t_0, T], \\ &\boldsymbol{u}(t) = (u_1(t), \ldots, u_M(t))^{\mathrm{T}}, \quad \boldsymbol{f}(\boldsymbol{u}) = (f_1(\boldsymbol{u}), \ldots, f_M(\boldsymbol{u}))^{\mathrm{T}}, \end{aligned} \tag{1.1}$$

where derivatives of a vector of univariate scalar functions are understood componentwise, posed along with initial data

$$\boldsymbol{u}(t_0) = \boldsymbol{u}_0.$$

Taylor series methods are nowadays an established alternative for the numerical solution of problems modelled by initial value problems of ODEs. The idea is to compute approximations to the solution of the ODE for the next time instant using a Taylor polynomial of the unknown. The resulting methods are conceptually simple, as the expressions required for the iteration are exactly computable (i. e., with no error) from the equation, and the truncation error is governed by the error term of the Taylor formula, so that the order of accuracy of the global error of the method corresponds to the degree of the Taylor polynomial used. Explicit Taylor methods are obtained by computing the Taylor polynomial centered on the current time instant, while their implicit counterparts correspond to Taylor polynomials centered on a future time instant. In these cases, an auxiliary system of nonlinear equations has to be solved to compute the solution at the next time step. Implicit Taylor methods are often used in the literature to solve problems where explicit methods have strong stability restrictions, in particular stiff systems of ODE. Combinations of implicit and explicit steps have also been considered to improve stability or accuracy [5, 9].

Altogether, Taylor methods are both difficult to implement (because the computation of the terms involved in the Taylor series requires intensive symbolic calculus) and computationally expensive, especially the implicit versions, because of the need of solving the auxiliary equations.

In this work we present a simplification of the implicit Taylor method, based on the work developed by Baeza et al. in [2] for the explicit Taylor method. This approximation inherits the benefits of the explicit version in terms of easiness of implementation and performance. Moreover, the implicit character of the method requires the resolution of an auxiliary system of equations, commonly performed by means of Newton's method, which requires the computation of the system Jacobian. We show that the system that arises from the approximate version is simpler to implement and faster to solve than the one arising from the *exact* implicit Taylor method.

1.2 Related work

As we have stated, the effective computation of the derivatives in the Taylor polynomial is one of the critical points for the numerical implementation of Taylor methods, for both explicit and implicit versions, because it is an over-elaborate task, computationally

expensive and problem dependent. As a result, recently, several authors have attempted to design simplified versions of the Taylor schemes.

For instance, in [6] the authors propose an alternative based in a numerical approximation of the derivatives of f in ODEs of the form $u' = f(u)$ for the explicit Taylor method up to fourth order and in [7] for the implicit version up to fifth order. Later on, in [2], a procedure to obtain a numerical approximation of $f(u) = f \circ u$ was presented to generate arbitrarily high order Taylor schemes, inspired in an approximate Cauchy-Kovalevskaya procedure developed for the numerical solution of systems of conservation laws in [10], which simplifies the exact version presented in [8]. The method presented in [2] relies on the approximated computation of the terms that appear in the Taylor polynomials, in terms of function evaluations only, avoiding the explicit computation of the derivatives, leading to a method which is simple to implement and outperforms its exact counterpart for complex systems.

1.3 Outline of the paper.

The work is organized as follows: In Section 2 the basic facts about the exact Taylor methods are reviewed. A general procedure to generate Taylor schemes of arbitrarily high accuracy order through Faà di Bruno's formula [4] is described, as well as its corresponding approximated version presented in [2]. Section 3 is devoted to the detailed description of a novel proposal of implicit Taylor methods involving an approximate version of these schemes, following an idea akin to the explicit case [2]. An efficient way to implement the Newton iteration required to update the solution with the implicit methods, both exact and approximate, is also described. Section 4 stands for several numerical experiments in which the approximated version of the implicit Taylor methods is compared against its exact counterpart, as well as against the approximate explicit version. Finally, in Section 5 some conclusions are drawn.

2 Taylor methods

2.1 Preliminaries

The (explicit) R-th order Taylor methods are based on the expansion of the unknown function

$$\boldsymbol{u}(t+h) = \boldsymbol{u}(t) + h\boldsymbol{u}'(t) + \frac{h^2}{2}\boldsymbol{u}''(t) + \cdots + \frac{h^R}{R!}\boldsymbol{u}^{(R)}(t) + \frac{h^{R+1}}{(R+1)!}\boldsymbol{u}^{(R+1)}(\xi), \quad (2.1)$$

with ξ belonging to the open interval $I(t, t+h)$ defined by t and $t+h$, valid provided $u_1, \ldots, u_M \in \mathcal{C}^R(\bar{I}(t, t+h))$ and $\boldsymbol{u}^{(R+1)}$ is bounded in $I(t, t+h)$, where $\bar{I}(t, t+h)$ denotes the closure of $I(t, t+h)$. Consider an equally spaced set of $N+1$ points $t_n = t_0 + nh$, $0 \leq n \leq N$, $h = T/N$. Dropping the last term in (2.1) and taking $t = t_n$ one obtains the approximation

$$\boldsymbol{u}(t_n + h) = \boldsymbol{u}(t_{n+1}) \approx \boldsymbol{u}(t_n) + h\boldsymbol{u}'(t_n) + \frac{h^2}{2}\boldsymbol{u}''(t_n) + \cdots + \frac{h^R}{R!}\boldsymbol{u}^{(R)}(t_n). \quad (2.2)$$

Then (1.1) can be used to write

$$\boldsymbol{u}^{(k)}(t_n) = \big(\boldsymbol{f}(\boldsymbol{u})\big)^{(k-1)}(t_n) = \frac{\mathrm{d}^{k-1}}{\mathrm{d}t^{k-1}}\big(\boldsymbol{f}(\boldsymbol{u}(t))\big)\bigg|_{t=t_n}, \quad 1 \le k \le R. \tag{2.3}$$

Consequently, the first step to apply Taylor methods is to compute these derivatives up to an appropriate order.

2.2 Faà di Bruno's formula

The evaluation of high-order derivatives of the function $t \mapsto (\boldsymbol{f} \circ \boldsymbol{u})(t)$, which arise in (2.2), is greatly simplified by Faà di Bruno's formula, as stated in [2]. To this end, we recall that for a multi-index $\boldsymbol{s} = (s_1, \ldots, s_r) \in \mathbb{N}_0^r$, one defines $|\boldsymbol{s}| := s_1 + \cdots + s_r$ and

$$\binom{r}{\boldsymbol{s}} := \frac{r!}{s_1! s_2! \cdots s_r!}.$$

Moreover, for $r \in \mathbb{N}$ we define an index set

$$\mathcal{P}_r := \left\{ \boldsymbol{s} \in \mathbb{N}_0^r \,\middle|\, \sum_{\nu=1}^{r} \nu s_\nu = r \right\},$$

and $(\boldsymbol{D}^{\boldsymbol{s}}\boldsymbol{u})(t)$ to be a matrix of size $M \times |\boldsymbol{s}|$ whose $(s_0 + s_1 + \cdots + s_{j-1} + i)$-th column is given by

$$\big((\boldsymbol{D}^{\boldsymbol{s}}\boldsymbol{u})(t)\big)_{s_0+s_1+\cdots+s_{j-1}+i} = \frac{1}{j!}\frac{\mathrm{d}^j}{\mathrm{d}t^j}\boldsymbol{u}(t), \quad i = 1, \ldots, s_j, \quad j = 1, \ldots, r. \tag{2.4}$$

Finally, we denote by $f^{(k)} \bullet \boldsymbol{A}$ the action of the k-th order derivative tensor of f on a $M \times k$ matrix $\boldsymbol{A} = (A_{ij})$:

$$f^{(k)} \bullet \boldsymbol{A} := \sum_{i_1,\ldots,i_k=1}^{M} \frac{\partial^k f}{\partial u_{i_1} \cdots \partial u_{i_k}}(\boldsymbol{u}) A_{i_1,1} \cdots A_{i_k,k}.$$

Proposition 1 (Faà di Bruno's formula [4]) *Assume that the functions $f : \mathbb{R}^M \to \mathbb{R}$ and $\boldsymbol{u} : \mathbb{R} \to \mathbb{R}^M$ are r times continuously differentiable. Then*

$$\frac{\mathrm{d}^r}{\mathrm{d}t^r} f\big(\boldsymbol{u}(t)\big) \equiv \big(f(\boldsymbol{u})\big)^{(r)}(t) = \sum_{\boldsymbol{s} \in \mathcal{P}_r} \binom{r}{\boldsymbol{s}} \big[\big(f(\boldsymbol{u})\big)^{(|\boldsymbol{s}|)} \bullet (\boldsymbol{D}^{\boldsymbol{s}}\boldsymbol{u})\big](t). \tag{2.5}$$

Proposition 1 applies to just one scalar function f, so to obtain all components of, say, $(\boldsymbol{f}(\boldsymbol{u}))^{(k)}(t_n)$ in (2.3), we must apply (2.5) to each of the components of $\boldsymbol{f} = (f_1, \ldots, f_M)^{\mathrm{T}}$. Clearly, the matrix $\boldsymbol{D}^{\boldsymbol{s}}\boldsymbol{u}$ is the same for all these components.

2.3 Explicit Taylor methods

The derivatives $(\boldsymbol{f}(\boldsymbol{u}))^{(k-1)}$ can be evaluated by using Faà di Bruno's formula (2.5) (see [2] for more details), leading to an expression of $\boldsymbol{u}^{(k)}(t_n)$ in terms of $\boldsymbol{u}(t_n)$ and derivatives of $\boldsymbol{f}$, namely

$$\begin{aligned}\boldsymbol{u}^{(k)}(t_n) &= \boldsymbol{G}_k\Big(\boldsymbol{u}(t_n), \big(\boldsymbol{f}(\boldsymbol{u})\big)(t_n), \big(\boldsymbol{f}(\boldsymbol{u})\big)'(t_n), \ldots, \big(\boldsymbol{f}(\boldsymbol{u})\big)^{(k-1)}(t_n)\Big)\\ &= \tilde{\boldsymbol{G}}_k\big(\boldsymbol{u}(t_n)\big).\end{aligned} \tag{2.6}$$

Replacing the derivatives $\boldsymbol{u}^{(k)}(t_n)$ in (2.2) by (2.6), we obtain the expression

$$\boldsymbol{u}(t_{n+1}) \approx T_R(\boldsymbol{u}(t_n), h) = \boldsymbol{u}(t_n) + \sum_{k=1}^{R} \frac{h^k}{k!}\tilde{\boldsymbol{G}}_k\big(\boldsymbol{u}(t_n)\big). \tag{2.7}$$

The R-th order Taylor method, denoted by $T_R(\boldsymbol{u}_n, h)$, is then obtained by replacing the exact values of the solution $u(t_n)$ and $u(t_{n+1})$ by their corresponding approximations in (2.7), denoted by $\boldsymbol{u}_n$ and $\boldsymbol{u}_{n+1}$, respectively:

$$\boldsymbol{u}_{n+1} = T_R(\boldsymbol{u}_n, h) = \boldsymbol{u}_n + \sum_{k=1}^{R} \frac{h^k}{k!}\boldsymbol{u}_n^{(k)}, \quad \boldsymbol{u}_n^{(k)} := \tilde{\boldsymbol{G}}_k(\boldsymbol{u}_n). \tag{2.8}$$

From (2.1) and (2.8) we infer that the local truncation error is given by

$$\boldsymbol{E}_{\mathrm{L}} = \frac{h^{R+1}}{(R+1)!}\boldsymbol{u}^{(R+1)}(\xi),$$

so that $\boldsymbol{E}_{\mathrm{L}} = \mathcal{O}(h^{R+1})$ as long as $\boldsymbol{u}^{(R+1)}$ is bounded in $[t_0, T]$, resulting in a global error $\mathcal{O}(h^R)$.

3 Implicit Taylor methods

3.1 Exact implicit Taylor methods

Implicit Taylor methods are based on approximating $\boldsymbol{u}(t_n)$ by means of the Taylor polynomial of $\boldsymbol{u}$ centered at t_{n+1}:

$$\boldsymbol{u}(t_n) \approx T_R\big(\boldsymbol{u}(t_{n+1}), -h\big), \tag{3.1}$$

so that the value of $\boldsymbol{u}_{n+1} \approx \boldsymbol{u}(t_{n+1})$ is determined as solution of the nonlinear system of algebraic equations

$$\boldsymbol{u}_n = T_R(\boldsymbol{u}_{n+1}, -h). \tag{3.2}$$

In the easiest case, with $R = 1$, one gets the implicit Euler method. As in the case of explicit Taylor methods, the expressions of $\boldsymbol{u}^{(k)}(t_{n+1})$ that appear in (3.1) can be expressed as functions of $\boldsymbol{u}(t_{n+1})$ and the derivatives of $\boldsymbol{f}$. As an example, the second-order implicit Taylor method is given by

$$\boldsymbol{u}_n = \boldsymbol{u}_{n+1} - h\boldsymbol{f}(\boldsymbol{u}_{n+1}) + \frac{h^2}{2}\left(\frac{\partial \boldsymbol{f}}{\partial \boldsymbol{u}}(\boldsymbol{u}_{n+1})\boldsymbol{f}(\boldsymbol{u}_{n+1})\right), \tag{3.3}$$

where $\partial \boldsymbol{f}/\partial \boldsymbol{u} = (\partial f_i/\partial u_j)_{1\le i,j\le M}$ is the functional matrix of $\boldsymbol{f}(\boldsymbol{u})$. In what follows, the family of methods based on (3.2) will be referred to as *exact* implicit Taylor methods since they are based on exact expressions of the derivatives of $\boldsymbol{f}$.

3.2 Approximate Implicit Taylor methods

Let us briefly review approximate explicit Taylor methods as described in [2]. These methods are based on computing approximations of the derivatives in (2.2) by means of finite differences, so that $\boldsymbol{u}^{(k)}(t_n)$ is replaced by an approximation

$$\boldsymbol{v}_{h,n}^{(k)} = \boldsymbol{u}^{(k)}(t_n) + \mathcal{O}(h^{R-k+1}), \quad k = 2, \dots, R,$$

resulting in an R-th order accurate method

$$\boldsymbol{v}_{h,n+1} = \boldsymbol{v}_{h,n} + \sum_{k=1}^{R} \frac{h^k}{k!} \boldsymbol{v}_{h,n}^{(k)},$$

where the approximations $v_{h,n}^{(k)}$ are computed as follows:

$$\begin{aligned}
\boldsymbol{v}_{h,n}^{(0)} &= \boldsymbol{u}_n, \\
\boldsymbol{v}_{h,n}^{(1)} &= \boldsymbol{f}(\boldsymbol{u}_n), \\
\boldsymbol{v}_{h,n}^{(k+1)} &= \Delta_h^{k,\lceil \frac{R-k}{2} \rceil} \boldsymbol{f}\big(\boldsymbol{P}_n^k(h)\big), \quad k = 1, \dots, R-1,
\end{aligned}$$

where $\boldsymbol{P}^k(\rho)$ is the M-component vector given by

$$P_n^k(\rho) = \sum_{l=0}^{k} \frac{\boldsymbol{v}_{h,n}^{(l)}}{l!} \rho^l, \quad n = 1, \dots, N,$$

and $\Delta_h^{p,q}$ is the centered finite-difference operator that approximates p-th order derivatives to order $2q$ on a grid with spacing h, i.e., the one that satisfies

$$\Delta_h^{p,q}(y) = y^{(p)} + \mathcal{O}(h^{2q}),$$

for a sufficiently differentiable function y. (The operator $\Delta_h^{p,q}$ is understood as acting on each component of $\boldsymbol{f}(\boldsymbol{P}_n^k(h))$ separately.)

There exist constants $\beta_j^{k,R}$ so that for some integers $\gamma_{k,R}$, we can write (see [10])

$$\boldsymbol{v}_{h,n}^{(k+1)} = h^{-k} \sum_{j=-\gamma_{k,R}}^{\gamma_{k,R}} \beta_j^{k,R} \boldsymbol{f}\left(\sum_{l=0}^{k} \frac{(jh)^l}{l!} \boldsymbol{v}_{h,n}^{(l)} \right). \tag{3.4}$$

Using these approximations of the derivatives, and with the notation of the previous sections, one obtains the approximate explicit Taylor method

$$\boldsymbol{u}_{n+1} = \tilde{T}_R(\boldsymbol{u}_n, h). \tag{3.5}$$

For instance, the second-order approximate Taylor method is based on the approximation

$$\boldsymbol{u}^{(2)}(t_n) = \big(\boldsymbol{f}(\boldsymbol{u})\big)'(t_n) \approx \frac{1}{2h}\Big(\boldsymbol{f}\big(\boldsymbol{u}(t_n) + h\boldsymbol{f}(\boldsymbol{u}(t_n))\big) - \boldsymbol{f}\big(\boldsymbol{u}(t_n) - h\boldsymbol{f}(u(t_n))\big)\Big),$$

hence the method can be written as

$$\boldsymbol{u}_{n+1} = \boldsymbol{u}_n + h\boldsymbol{f}(\boldsymbol{u}_n) + \frac{h}{4}\Big(\boldsymbol{f}\big(\boldsymbol{u}_n + h\boldsymbol{f}(u_n)\big) - \boldsymbol{f}\big(\boldsymbol{u}_n - h\boldsymbol{f}(u_n)\big)\Big),$$

i.e.,

$$\tilde{T}_2(\boldsymbol{u}_n, h) = \boldsymbol{u}_n + h\boldsymbol{f}(\boldsymbol{u}_n) + \frac{h}{4}\Big(\boldsymbol{f}\big(\boldsymbol{u}_n + h\boldsymbol{f}(u_n)\big) - \boldsymbol{f}\big(\boldsymbol{u}_n - h\boldsymbol{f}(u_n)\big)\Big).$$

The new methods advanced in this contribution, namely *approximate implicit* Taylor methods are obtained by replacing h by $-h$ and interchanging $\boldsymbol{u}_n$ and $\boldsymbol{u}_{n+1}$ in (3.5):

$$\boldsymbol{u}_n = \tilde{T}_R(\boldsymbol{u}_{n+1}, -h) := \tilde{T}_R(\boldsymbol{u}_{n+1}, -h).$$

For the case of second order, it follows that the implicit second-order approximate Taylor method is

$$\begin{aligned}\boldsymbol{u}_n &= \tilde{T}_2(\boldsymbol{u}_{n+1}, -h) \\ &= \boldsymbol{u}_{n+1} - hf(\boldsymbol{u}_{n+1}) - \frac{h}{4}\Big(\boldsymbol{f}\big(\boldsymbol{u}_{n+1} - h\boldsymbol{f}(\boldsymbol{u}_{n+1})\big) - \boldsymbol{f}\big(\boldsymbol{u}_{n+1} + h\boldsymbol{f}(\boldsymbol{u}_{n+1})\big)\Big).\end{aligned} \tag{3.6}$$

3.3 Linear stability

The linear stability of a numerical scheme for initial value problems of ordinary differential equations is usually examined by applying it to the scalar linear equation

$$u' = \lambda u, \quad \lambda \in \mathbb{C}, \quad \operatorname{Re}\lambda < 0. \tag{3.7}$$

For the sake of completeness, we consider the non-homogeneous linear ODE

$$u' = \lambda u + g(t), \quad \lambda \in \mathbb{C},$$

with g sufficiently smooth. For the solution u of the ODE, we can establish by induction on k that

$$u^{(k)} = \lambda^k u + \sum_{j=0}^{k-1} \lambda^{k-j-1} g^{(j)}(t),$$

so the explicit Taylor method reads in this case as

$$\begin{aligned}u_{n+1} &= \sum_{k=0}^{R} \frac{h^k}{k!}\left(\lambda^k u_n + \sum_{j=0}^{k-1} \lambda^{k-j-1} g^{(j)}(t_n)\right) \\ &= u_n \sum_{k=0}^{R} \frac{(h\lambda)^k}{k!} + \sum_{j=0}^{R-1} \frac{g^{(j)}(t_n)}{\lambda^{j+1}} \sum_{k=j+1}^{R} \frac{(h\lambda)^k}{k!} \\ &= Q_R(h\lambda)u_n + \sum_{j=0}^{R-1} \frac{g^{(j)}(t_n)}{\lambda^{j+1}}\big(Q_R(h\lambda) - Q_j(h\lambda)\big),\end{aligned}$$

where

$$Q_j(x) = \sum_{k=0}^{j} \frac{x^k}{k!}.$$

The implicit Taylor method is obtained by interchanging the roles of n and $n+1$ and reads as

$$u_n = Q_R(-h\lambda)u_{n+1} + \sum_{j=0}^{R-1} \frac{g^{(j)}(t_{n+1})}{\lambda^{j+1}} \big(Q_R(-h\lambda) - Q_j(-h\lambda)\big),$$
$$u_{n+1} = \frac{1}{Q_R(-h\lambda)}u_n - \sum_{j=0}^{R-1} \frac{g^{(j)}(t_{n+1})}{\lambda^{j+1}} \left(1 - \frac{Q_j(-h\lambda)}{Q_R(-h\lambda)}\right). \tag{3.8}$$

In particular, for $g = 0$, the explicit and implicit Taylor methods of order R are given by

$$u_{n+1} = Q_R(h\lambda)u_n \tag{3.9}$$

and

$$u_{n+1} = \frac{1}{Q_R(-h\lambda)}u_n.$$

The exact Taylor method of order R is stable provided that $|Q_R(h\lambda)| < 1$. Since $\operatorname{Re}\lambda < 0$, this condition us usually satisfied on a bounded domain only (as can be inferred from $R = 1$, in which case (3.9) is the explicit Euler method). On the other hand, the exact implicit Taylor method is stable for those values of $z = h\lambda$ that satisfy

$$\begin{aligned} z \in \mathcal{S} :=& \big\{ z \in \mathbb{C} \mid \operatorname{Re} z < 0, |Q_R(-z)|^{-1} < 1 \big\} \\ =& \big\{ z \in \mathbb{C} \mid \operatorname{Re} z < 0, |Q_R(-z)| > 1 \big\}. \end{aligned}$$

As for its exact counterpart, in [3] it is shown that the approximate explicit Taylor method applied to (3.7) is

$$\tilde{T}_R(u_n, h) = Q_R(h\lambda)u_n,$$

thus the implicit version is

$$\tilde{T}_R(u_n, -h) = Q_R(-h\lambda)^{-1}u_n,$$

and therefore both methods have the same stability region as their corresponding exact versions, in particular the approximate implicit Taylor method is absolutely stable whenever $\lambda < 0$.

3.4 Newton iteration

The computation of $\boldsymbol{u}_{n+1}$ for given $\boldsymbol{u}_n$ using an implicit method requires the solution of an auxiliary equation

$$\boldsymbol{F}(\boldsymbol{u}_{n+1}) = \boldsymbol{0}, \tag{3.10}$$

which is often approximated by means of Newton's method. In this section, we address the computation of the elements required for Newton's method for both the exact and approximate implicit Taylor method.

3.4.1 Exact implicit Taylor method

As an example, let us consider the scalar nonlinear problem

$$u' = u + u^2 \Rightarrow u'' = (1+2u)u' = (1+2u)(u+u^2).$$

The second-order exact implicit Taylor method can be written as

$$u_n = u_{n+1} - h(u_{n+1} + u_{n+1}^2) + \frac{h^2}{2}(1+2u_{n+1})(u_{n+1}+u_{n+1}^2),$$

and, therefore it requires the solution of the following cubic equation:

$$F(u_{n+1}) := u_{n+1} - h(u_{n+1} + u_{n+1}^2) + \frac{h^2}{2}(1+2u_{n+1})(u_{n+1}+u_{n+1}^2) - u_n = 0. \quad (3.11)$$

In the general case, the solution of (3.10) by means of Newton's method requires the computation of the derivative $F'(u_{n+1})$. In the case of (3.11) this is an easy task, but in general the resulting iteration can become complicated. An option is to introduce

$$\boldsymbol{z}_k \approx \boldsymbol{u}_{n+1}^{(k)}, \quad k = 0, \ldots, R,$$

and use Faà di Bruno's formula (2.5) to get the system

$$\begin{aligned}
\boldsymbol{u}_n &= \boldsymbol{z}_0 - h\boldsymbol{z}_1 + \cdots + (-1)^R \frac{h^R}{R!}\boldsymbol{z}_R, \\
\boldsymbol{z}_1 &= \boldsymbol{f}(\boldsymbol{z}_0), \\
\boldsymbol{z}_{n+1} &= \sum_{\boldsymbol{s}\in\mathcal{P}_r} \binom{r}{\boldsymbol{s}} \begin{pmatrix} f_1^{(|\boldsymbol{s}|)}(\boldsymbol{z}_0) \bullet \tilde{\boldsymbol{D}}^{\boldsymbol{s}}\boldsymbol{z} \\ \vdots \\ f_N^{(|\boldsymbol{s}|)}(\boldsymbol{z}_0) \bullet \tilde{\boldsymbol{D}}^{\boldsymbol{s}}\boldsymbol{z} \end{pmatrix}, \quad r = 1, \ldots, R-1,
\end{aligned}$$

(see Section 2.2 for the definitions of $\mathcal{P}_r$, $\boldsymbol{s}$ and $\binom{r}{\boldsymbol{s}}$), where the definition of $\tilde{\boldsymbol{D}}^{\boldsymbol{s}}\boldsymbol{z}$ mimics that of $\boldsymbol{D}^{\boldsymbol{s}}\boldsymbol{z}$ in (2.4), by taking into account that $\boldsymbol{z}_k \approx \boldsymbol{u}^{(k)}(t)$, namely

$$\left(\tilde{\boldsymbol{D}}^{\boldsymbol{s}}\boldsymbol{z}\right)_{s_0+s_1+\cdots+s_{j-1}+i} = \frac{1}{j!}\boldsymbol{z}_j, \quad i = 1, \ldots, s_j, \quad j = 1, \ldots, r.$$

These equations can be differentiated systematically. For instance, for the case of one scalar equation, $M = 1$, one gets

$$\begin{aligned}
&\partial_{z_0}\left(\sum_{\boldsymbol{s}\in\mathcal{P}_r}\binom{r}{\boldsymbol{s}} f^{(|\boldsymbol{s}|)}(z_0)\boldsymbol{D}^{\boldsymbol{s}}z\right) = \sum_{\boldsymbol{s}\in\mathcal{P}_r}\binom{r}{\boldsymbol{s}} f^{(|\boldsymbol{s}|+1)}(z_0)\left(\frac{z_1}{1!}\right)^{s_1}\cdots\left(\frac{z_r}{r!}\right)^{s_r}, \\
&\partial_{z_j}\left(\sum_{\boldsymbol{s}\in\mathcal{P}_r}\binom{r}{\boldsymbol{s}} f^{(|\boldsymbol{s}|)}(z_0)\boldsymbol{D}^{\boldsymbol{s}}z\right) \\
&= \sum_{\boldsymbol{s}\in\mathcal{P}_r}\frac{s_j}{j!}\binom{r}{\boldsymbol{s}} f^{(|\boldsymbol{s}|+1)}(z_0)\left(\frac{z_1}{1!}\right)^{s_1}\cdots\left(\frac{z_j}{j!}\right)^{s_j-1}\cdots\left(\frac{z_r}{r!}\right)^{s_r}.
\end{aligned}$$

For this scalar case and the second-order implicit Taylor method (3.3) the system to be solved is

$$\begin{aligned} 0 &= z_0 - hz_1 + \frac{h^2}{2} z_2 - u_n, \\ 0 &= f(z_0) - z_1, \\ 0 &= f'(z_0) z_1 - z_2. \end{aligned} \tag{3.12}$$

If we write this as $\boldsymbol{F}(z_0, z_1, z_2) = \mathbf{0}$ where the components of $\boldsymbol{F} = (F_1, F_2, F_3)$ are given by the right-hand side of (3.12), then the corresponding Jacobian is given by

$$\mathcal{J}_{\boldsymbol{F}}(z_0, z_1, z_2) = \begin{bmatrix} 1 & -h & h^2/2 \\ f'(z_0) & -1 & 0 \\ f''(z_0) z_1 & f'(z_0) & -1 \end{bmatrix}. \tag{3.13}$$

Depending on the expression of f the Jacobian matrix may become highly complex, even for low values of R. It is clear that, for higher order methods, the system to be solved will be more complicated. For instance, for $R = 4$, it reads as:

$$\begin{aligned} 0 &= z_0 - hz_1 + \frac{h^2}{2} z_2 - \frac{h^3}{6} z_3 + \frac{h^4}{24} z_4 - u_n, \\ 0 &= f(z_0) - z_1, \\ 0 &= f'(z_0) z_1 - z_2, \\ 0 &= f''(z_0) z_1^2 + f'(z_0) z_2 - z_3, \\ 0 &= f'''(z_0) z_1^3 + 3 f''(z_0) z_1 z_2 + z_3 f'(z_0) - z_4, \end{aligned}$$

which results in the following expression for the Jacobian matrix:

$$\mathcal{J}_{\boldsymbol{F}}(z_0, \ldots, z_4) = \begin{bmatrix} 1 & -h & \frac{h^2}{2} & \frac{h^3}{6} & \frac{h^4}{24} \\ f'(z_0) & -1 & 0 & 0 & 0 \\ f''(z_0) z_1 & f'(z_0) & -1 & 0 & 0 \\ f'''(z_0) z_1^2 + f''(z_0) z_2 & 2 f''(z_0) z_1 & f'(z_0) & -1 & 0 \\ f^{(4)}(z_0) z_1^3 + 3 f'''(z_0) z_1 z_2 + f''(z_0) z_3 & 3 f'''(z_0) z_1 + 3 f''(z_0) z_2 & 3 f''(z_0) z_1 & f'(z_0) & -1 \end{bmatrix}. \tag{3.14}$$

Note that the submatrix composed by the first three rows and columns of (3.14) is exactly (3.13). It is easy to check that the Jacobian matrix corresponding to $R = 3$ is the submatrix of (3.14) composed by its first four rows and columns.

3.4.2 Approximate implicit Taylor method

For simplicity, let us start with the second-order approximate implicit Taylor method (3.6) for the scalar case $M = 1$. Similarly to the exact case we introduce the unknowns $z_0 = u_{n+1}$, $z_1 = f(u_{n+1})$ and

$$z_2 = \frac{1}{2}\Big(f\big(u_{n+1} - h f(u_{n+1})\big) - f\big(u_{n+1} + h f(u_{n+1})\big)\Big),$$

and one gets the system of equations

$$\begin{aligned}
0 &= z_0 - hz_1 - \frac{h}{2}z_2 - u_n,\\
0 &= f(z_0) - z_1,\\
0 &= \frac{1}{2}f(z_0 - hz_1) - \frac{1}{2}f(z_0 + hz_1) - z_2,
\end{aligned}$$

so that its solution gives the terms that appear in (3.6). To apply Newton's method to this system, the Jacobian of the system is required, and it is now given by

$$\mathcal{J}_{\boldsymbol{F}}(z_0, z_1, z_2) = \begin{bmatrix} 1 & -h & -\frac{h}{2} \\ f'(z_0) & -1 & 0 \\ \frac{1}{2}\big(f'(z_0 - hz_1) & -\frac{h}{2}\big(f'(z_0 - hz_1) & -1 \\ -f'(z_0 + hz_1)\big) & +f'(z_0 + hz_1)\big) & \end{bmatrix}.$$

3.5 General case

Let us now consider the general case of a system of M scalar ordinary differential equations. From (3.4), the approximate implicit R-th order Taylor method can be written as

$$\boldsymbol{u}_n = \tilde{T}_R(\boldsymbol{u}_{n+1}, -h) = \sum_{k=0}^{R} \frac{(-h)^k}{k!}\boldsymbol{v}_{-h,n}^{(k)}, \tag{3.15}$$

$$\boldsymbol{v}_{-h,n}^{(k+1)} = (-h)^{-k} \sum_{j=-\gamma_{k,R}}^{\gamma_{k,R}} \beta_j^{k,R} \boldsymbol{f}\left(\sum_{l=0}^{k} \frac{j^l(-h)^l}{l!}\boldsymbol{v}_{-h,n}^{(l)}\right). \tag{3.16}$$

Let us denote $\boldsymbol{z}_k = (-h)^{k-1}\boldsymbol{v}_{-h,n}^{k}$, so that (3.16) for $k-1$ reads as:

$$\boldsymbol{z}_k = \sum_{j=-\gamma_{k-1,R}}^{\gamma_{k-1,R}} \beta_j^{k-1,R} \boldsymbol{f}\left(z_0 - h\sum_{l=1}^{k-1} \frac{j^l}{l!}\boldsymbol{z}_l\right)$$

and (3.15) as

$$\boldsymbol{u}_n = \boldsymbol{z}_0 - h\sum_{k=1}^{R} \frac{1}{k!}\boldsymbol{z}_k.$$

Define the function $\boldsymbol{F} = (\boldsymbol{F}_0, \boldsymbol{F}_1, \ldots, \boldsymbol{F}_M)^{\mathrm{T}}: \quad \mathbb{R}^{(R+1)M} \to \mathbb{R}^{(R+1)M}$ by

$$\begin{aligned}
\boldsymbol{F}_0 &= \boldsymbol{z}_0 - h\sum_{k=1}^{R} \frac{1}{k!}\boldsymbol{z}_k - \boldsymbol{u}_n,\\
\boldsymbol{F}_k &= \sum_{j=-\gamma_{k-1,R}}^{\gamma_{k-1,R}} \beta_j^{k-1,R} \boldsymbol{f}\left(\boldsymbol{z}_0 - h\sum_{l=1}^{k-1} \frac{j^l}{l!}\boldsymbol{z}_l\right) - \boldsymbol{z}_k, \quad k = 1, \ldots, R.
\end{aligned}$$

To solve $\boldsymbol{F}(\boldsymbol{z}) = \boldsymbol{0}$ by Newton's method, we compute the Jacobian of $\boldsymbol{F}$ as the block matrix

$$\mathcal{J}_{\boldsymbol{F}}(\boldsymbol{z}) = \big(\boldsymbol{F}_{i,j}(\boldsymbol{z})\big)_{0\le i,j\le R}, \quad \text{where} \quad \boldsymbol{F}_{i,j}(\boldsymbol{z}) = \frac{\partial \boldsymbol{F}_i}{\partial \boldsymbol{z}_j}(\boldsymbol{z}) \in \mathbb{R}^{M\times M}.$$

If $\boldsymbol{I}_M$ denotes the $M \times M$ identity matrix, we get

$$\begin{aligned}
&\boldsymbol{F}_{0,0} = \boldsymbol{I}_M,\\
&\boldsymbol{F}_{0,l} = -\frac{h}{l!}\boldsymbol{I}_M, \quad l = 1,\dots,R,\\
&\boldsymbol{F}_{k,0} = \sum_{j=-\gamma_{k-1,R}}^{\gamma_{k-1,R}} \beta_j^{k-1,R} \boldsymbol{f}'\left(\boldsymbol{z}_0 - h\sum_{l=1}^{k-1}\frac{j^l}{l!}\boldsymbol{z}_l\right), \quad k = 1,\dots,R,\\
&\boldsymbol{F}_{k,l} = -h\sum_{j=-\gamma_{k-1,R}}^{\gamma_{k-1,R}} \beta_j^{k-1,R} \boldsymbol{f}'\left(\boldsymbol{z}_0 - h\sum_{m=1}^{k-1}\frac{j^m}{m!}\boldsymbol{z}_m\right)\frac{j^l}{l!}, \quad \begin{cases} l = 1,\dots,k-1,\\ k = 1,\dots,R,\end{cases}\\
&\boldsymbol{F}_{k,k} = -\boldsymbol{I}_M, \quad k = 1,\dots,R,\\
&\boldsymbol{F}_{k,l} = \boldsymbol{0}, \quad l = k+1\dots,R, \quad k = 1,\dots,R.
\end{aligned}$$

Setting $\boldsymbol{\delta}^{(\nu)} = \boldsymbol{z}^{(\nu+1)} - \boldsymbol{z}^{(\nu)}$, we may write an iteration of Newton's method as

$$\mathcal{J}_{\boldsymbol{F}}(\boldsymbol{z}^{(\nu)})\boldsymbol{\delta}^{(\nu)} = -\boldsymbol{F}(\boldsymbol{z}^{(\nu)}).$$

In block form and dropping ν, we get

$$\begin{bmatrix} \boldsymbol{F}_{0,0} & \boldsymbol{F}_{0,1} & \cdots & \boldsymbol{F}_{0,R}\\ \boldsymbol{F}_{1,0} & \boldsymbol{F}_{1,1} & \cdots & \boldsymbol{F}_{1,R}\\ \vdots & \vdots & & \vdots\\ \boldsymbol{F}_{R,0} & \boldsymbol{F}_{R,1} & \cdots & \boldsymbol{F}_{R,R}\end{bmatrix}\begin{pmatrix}\boldsymbol{\delta}_0\\ \boldsymbol{\delta}_1\\ \vdots\\ \boldsymbol{\delta}_R\end{pmatrix} = -\begin{pmatrix}\boldsymbol{F}_0\\ \boldsymbol{F}_1\\ \vdots\\ \boldsymbol{F}_R\end{pmatrix},$$

which we write in compact form as

$$\begin{bmatrix} \boldsymbol{F}_{0,0} & \boldsymbol{F}_{0,1:R}\\ \boldsymbol{F}_{1:R,0} & \boldsymbol{F}_{1:R,1:R}\end{bmatrix}\begin{pmatrix}\boldsymbol{\delta}_0\\ \boldsymbol{\delta}_{1:R}\end{pmatrix} = -\begin{pmatrix}\boldsymbol{F}_0\\ \boldsymbol{F}_{1:R}\end{pmatrix}. \tag{3.17}$$

Since $\boldsymbol{F}_{1:R,1:R}$ is blockwise lower triangular with the diagonal blocks given by $-\boldsymbol{I}_M$, this matrix is invertible and we deduce that

$$\boldsymbol{\delta}_{1:R} = -\boldsymbol{F}_{1:R,1:R}^{-1}(\boldsymbol{F}_{1:R} + \boldsymbol{F}_{1:R,0}\boldsymbol{\delta}_0),$$

which, when inserted into the first equation of (3.17), yields

$$\boldsymbol{\delta}_0 = -\big(\boldsymbol{F}_{0,0} - \boldsymbol{F}_{0,1:R}\boldsymbol{F}_{1:R,1:R}^{-1}\boldsymbol{F}_{1:R,0}\big)^{-1}\big(\boldsymbol{F}_0 - \boldsymbol{F}_{0,1:R}\boldsymbol{F}_{1:R,1:R}^{-1}\boldsymbol{F}_{1:R}\big).$$

If we denote

$$\boldsymbol{A} := \boldsymbol{F}_{1:R,1:R}^{-1}\boldsymbol{F}_{1:R}, \quad \boldsymbol{B} := \boldsymbol{F}_{1:R,1:R}^{-1}\boldsymbol{F}_{1:R,0},$$

then we can write

$$\boldsymbol{\delta}_0 = -\big(\boldsymbol{F}_{0,0} - \boldsymbol{F}_{0,1:R}\boldsymbol{B}\big)^{-1}\big(\boldsymbol{F}_0 - \boldsymbol{F}_{0,1:R}\boldsymbol{A}\big), \quad \boldsymbol{\delta}_{1:R} = -\big(\boldsymbol{A} + \boldsymbol{B}\boldsymbol{\delta}_0\big).$$

Therefore, the system can be solved efficiently as long as $\boldsymbol{F}_{0,0} - \boldsymbol{F}_{0,1:R}\boldsymbol{B}$ is invertible. Recall that the Newton iteration only requires the computation of $\boldsymbol{f}$ and $\boldsymbol{f}'$, in contrast with the exact version, which requires the computation of all the derivatives of f up to order R.

4 Numerical experiments

4.1 Preliminaries

In this section, we analyze the performance of the approximate implicit Taylor (AIT) methods described on this paper. We first compare the AIT methods with their *exact* counterparts, IT methods, of the same order. We have only compared the AIT with the IT methods for scalar equations since the implementation for systems of the IT methods is extremely involved. For linear scalar equations the implementation for any order is performed using (3.8). For nonlinear scalar equations, the implementation of IT methods is complex enough for only having implemented them up to order 4.

We compare these methods in terms of error and numerical order, using some scalar problems. We then raise two initial-value problems for systems of equations. For those problems, we compare the AIT methods with approximate explicit Taylor (AET) methods of the same order [2], so as to stress the superior stability of the implicit method. In all the numerical examples we show the numerical errors, computed with 1-norm, and the order of the numerical method, computed by

$$o(N) = \log_2 \left(|e(N)/e(N/2)| \right),$$

with $e(N)$ standing for the numerical error for N time steps.

4.2 Examples 1 and 2: scalar equations

In Example 1 we consider the linear equation

$$u' = -5u + 5\sin(2t) + 2\cos(2t), \quad u(0) = 0, \tag{4.1}$$

with exact solution $u(t) = \sin(2t)$. The results for IT and AIT methods for $T = 1$ and orders $R \in \{2, 3, 4, 5, 6\}$ are collected in Table 1, where it can be seen that with both methods, the expected orders of convergence are recovered in all cases. Comparing with the IT methods, we see that the approximate version attains the expected order faster than the exact version, but produces a slightly bigger error for coarse resolutions. This fact is possibly due to the simplicity of the equation under consideration, which produces a local truncation error smaller than the error corresponding to the approximation of derivatives performed in the AIT method and hinders the correct order of accuracy for the exact method whenever the step size is not small enough.

In Example 2 is we consider the more involved problem

$$u' = \log\left(\frac{u + u^3 + u^5}{1 + u^2 + u^4 + u^6}\right), \quad u(0) = 1, \tag{4.2}$$

and compute its solution up to $T = 1$ for orders $R \in \{2, 3, 4\}$. The solution computed by the AIT method with $R = 4$ and a resolution of 20000 points is taken as reference solution. We can see in Table 2 that the errors for both methods are similar and the numerical order converges to the expected values in each case. In Figure 1 we compare the errors obtained by each method with respect to the CPU time required to run the algorithm. It can be seen that the performance is increasingly favorable to the approximate method as the order increases, as expected.

$R = 2$				
N	$e(N)$ IT	$o(N)$ IT	$e(N)$ AIT	$o(N)$ AIT
10	2.62e-02	—	4.99e-02	—
20	9.15e-03	1.52	1.38e-02	1.85
40	2.86e-03	1.68	3.63e-03	1.93
80	8.15e-04	1.81	9.29e-04	1.97
160	2.19e-04	1.89	2.35e-04	1.98
320	5.70e-05	1.94	5.90e-05	1.99
640	1.45e-05	1.97	1.48e-05	2.00
$R = 3$				
10	1.30e-03	—	3.37e-02	—
20	2.88e-04	2.17	6.21e-03	2.44
40	4.43e-05	2.70	9.52e-04	2.71
80	5.84e-06	2.92	1.31e-04	2.86
160	7.37e-07	2.99	1.71e-05	2.94
320	9.19e-08	3.00	2.18e-06	2.97
640	1.15e-08	3.00	2.76e-07	2.99
$R = 4$				
10	7.55e-04	—	7.84e-03	—
20	9.43e-05	3.00	4.81e-04	4.03
40	8.42e-06	3.48	2.58e-05	4.22
80	6.27e-07	3.75	1.39e-06	4.22
160	4.26e-08	3.88	7.84e-08	4.15
320	2.78e-09	3.94	4.61e-09	4.09
640	1.77e-10	3.97	2.79e-10	4.05
$R = 5$				
10	4.32e-05	—	4.10e-03	—
20	2.59e-06	4.06	1.50e-04	4.77
40	9.73e-08	4.73	4.87e-06	4.94
80	3.14e-09	4.95	1.54e-07	4.98
160	9.75e-11	5.01	4.86e-09	4.99
320	3.02e-12	5.01	1.53e-10	4.99
640	9.41e-14	5.00	4.78e-12	5.00
$R = 6$				
10	1.59e-05	—	1.06e-03	—
20	5.51e-07	4.85	1.35e-05	6.29
40	1.25e-08	5.46	1.56e-07	6.43
80	2.33e-10	5.75	1.88e-09	6.38
160	3.96e-12	5.88	2.45e-11	6.26
320	6.47e-14	5.94	3.43e-13	6.16
640	9.99e-16	6.02	5.11e-15	6.07

Table 1 Example 1 (linear scalar problem (4.1)): numerical errors and orders for IT and AIT methods.

4.3 Examples 3 and 4: systems of ODEs

We consider now two problems modelled by systems of ODEs, used in [1] to test stability properties and accuracy. Example 3 is a stiff nonlinear problem given by

$$\begin{cases} y' = -1002y + 1000z^2, & y(0) = 1, \\ z' = y - z(1+z), \quad t > 0; & z(0) = 1, \end{cases} \tag{4.3}$$

known as Kaps problem, with exact solution given by

$$y(t) = \mathrm{e}^{-2t}, \quad z(t) = \mathrm{e}^{-t},$$

$R=2$				
N	$e(N)$ IT	$o(N)$ IT	$e(N)$ AIT	$o(N)$ AIT
10	1.21e-03	—	1.23e-03	—
20	2.90e-04	2.06	2.93e-04	2.13
40	7.09e-05	2.03	7.12e-05	2.07
80	1.75e-05	2.02	1.76e-05	2.04
160	4.36e-06	2.01	4.36e-06	2.02
320	1.09e-06	2.00	1.09e-06	2.01
640	2.71e-07	2.00	2.71e-07	2.00
1280	6.77e-08	2.00	6.78e-08	2.00
2560	1.69e-08	2.00	1.69e-08	2.00
$R=3$				
10	7.52e-05	—	5.35e-05	—
20	8.75e-06	3.10	5.95e-06	3.34
40	1.05e-06	3.05	7.00e-07	3.17
80	1.29e-07	3.03	8.49e-08	3.09
160	1.60e-08	3.01	1.04e-08	3.04
320	1.99e-09	3.01	1.30e-09	3.02
640	2.49e-10	3.00	1.61e-10	3.01
1280	3.11e-11	3.00	2.01e-11	3.01
2560	3.88e-12	3.00	2.51e-12	3.00
$R=4$				
10	5.78e-06	—	4.93e-06	—
20	3.30e-07	4.13	2.44e-07	4.76
40	1.97e-08	4.07	1.36e-08	4.34
80	1.20e-09	4.03	8.00e-10	4.17
160	7.43e-11	4.02	4.86e-11	4.08
320	4.62e-12	4.01	3.00e-12	4.04
640	2.87e-13	4.01	1.88e-13	4.02
1280	2.33e-14	3.62	1.47e-14	4.00
2560	5.66e-15	2.04	3.77e-15	3.68

Table 2 Example 2 (nonlinear scalar problem (4.2)): numerical errors and orders for IT and AIT methods.

which is independent of the stiffness parameter, $k = -1000$ in this case. We compare the solution at $T = 5$ for the approximate implicit (AIT) and approximate explicit (AET) methods of the same order. Both schemes recover the expected order, the implicit one achieving it at early stages, see Tables 3 and 4. Note that the explicit scheme does not attain good results in terms of accuracy, unless meshes with more than 2000 points are used. The implicit scheme achieves the same error level as the explicit one with meshes with approximately 4 times less nodes.

Finally, in Example 4 we consider the system of ODEs

$$\begin{cases} x' = -21x + 19y - 20z, & x(0) = 1, \\ y' = 19x - 21y + 20z, & y(0) = 0, \\ z' = 40x - 40y - 40z, \quad t > 0; & z(0) = -1, \end{cases} \tag{4.4}$$

also taken from [1] and whose exact solution is given by

$$\begin{cases} x(t) = \dfrac{1}{2}\left(\mathrm{e}^{-2t} + \mathrm{e}^{-40t}(\cos(40t) + \sin(40t))\right), \\ y(t) = \dfrac{1}{2}\left(\mathrm{e}^{-2t} - \mathrm{e}^{-40t}(\cos(40t) + \sin(40t))\right), \\ z(t) = -\,\mathrm{e}^{-40t}(\cos(40t) - \sin(40t)). \end{cases}$$

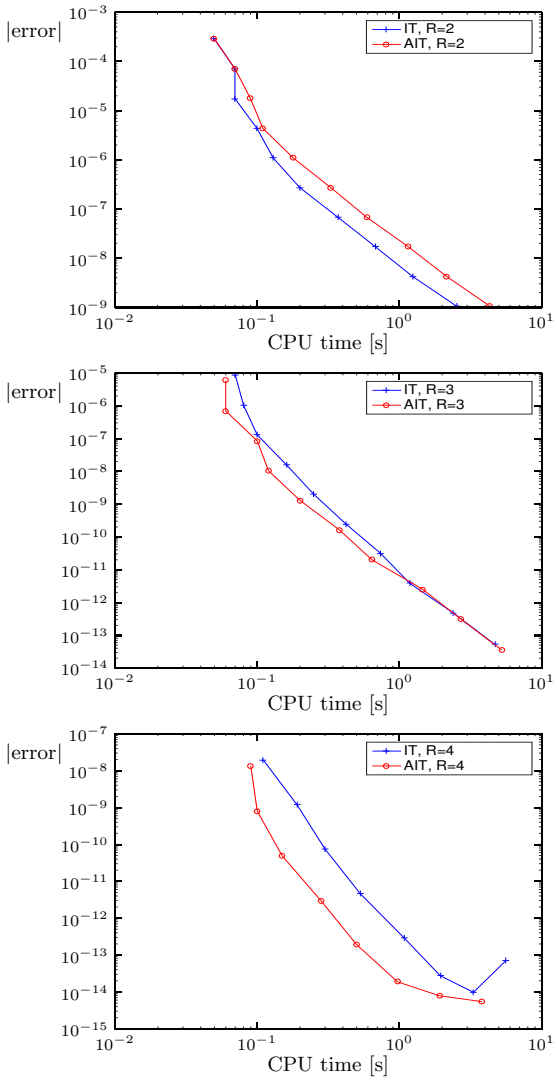

Fig. 1 Example 2 (nonlinear scalar problem (4.2)): performance of the IT and the AIT methods.

As in the previous example, the explicit scheme needs more nodes to achieve the same error level as the implicit scheme. For instance, the explicit scheme needs about 64 times more nodes, $N = 320$, to obtain the same errors that the implicit scheme attains with $N = 5$, see results in Tables 5 and 6, proving that the use of the implicit scheme is more appropriate when dealing with stiff problems.

	$R=2$		$R=3$		$R=4$		$R=5$		$R=6$	
N	$e(N)$	$o(N)$	$e(N)$	$o(N)$	$e(N)$	$o(N)$	$e(N)$	$o(N)$	$e(N)$	$o(N)$
5	3.56e-03	—	6.88e-04	—	1.26e-04	—	2.00e-05	—	2.66e-06	—
10	1.06e-03	1.74	1.21e-04	2.50	1.17e-05	3.42	9.50e-07	4.39	6.46e-08	5.36
20	3.02e-04	1.81	1.82e-05	2.72	9.05e-07	3.70	3.67e-08	4.69	1.26e-09	5.67
40	8.15e-05	1.89	2.52e-06	2.85	6.28e-08	3.84	1.27e-09	4.84	2.20e-11	5.83
80	2.12e-05	1.93	3.31e-07	2.92	4.13e-09	3.92	4.21e-11	4.92	3.64e-13	5.91
160	5.43e-06	1.96	4.24e-08	2.96	2.65e-10	3.96	1.35e-12	4.96	5.86e-15	5.95
320	1.37e-06	1.98	5.37e-09	2.98	1.68e-11	3.98	4.28e-14	4.97	1.00e-16	5.86
640	3.45e-07	1.99	6.76e-10	2.99	1.05e-12	3.98	1.34e-15	4.99	7.05e-18	3.82

Table 3 Example 3 (stiff nonlinear problem (4.3)): numerical errors and orders for AIT method.

	$R=2$		$R=3$		$R=4$		$R=5$		$R=6$	
N	$e(N)$	$o(N)$	$e(N)$	$o(N)$	$e(N)$	$o(N)$	$e(N)$	$o(N)$	$e(N)$	$o(N)$
1280	NaN	—	NaN	—	NaN	—	NaN	—	62.7	—
2560	1.03e-07	NaN	4.60e-11	NaN	1.07e-13	NaN	1.97e-16	NaN	3.03e-19	67.49
5120	2.34e-08	2.13	5.75e-12	3.00	5.65e-15	4.24	5.98e-18	5.04	3.96e-21	6.26
10240	5.84e-09	2.00	7.18e-13	3.00	3.33e-16	4.09	1.70e-19	5.14	5.48e-23	6.18
20480	1.46e-09	2.00	8.97e-14	3.00	2.00e-17	4.05	5.00e-21	5.09	7.98e-25	6.10
40960	3.64e-10	2.00	1.12e-14	3.00	1.22e-18	4.03	1.51e-22	5.05	1.20e-26	6.05

Table 4 Example 3 (stiff nonlinear problem (4.3)): numerical errors and orders for AET method.

	$R=2$		$R=3$		$R=4$		$R=5$		$R=6$	
N	$e(N)$	$o(N)$	$e(N)$	$o(N)$	$e(N)$	$o(N)$	$e(N)$	$o(N)$	$e(N)$	$o(N)$
5	2.74e-04	—	5.27e-05	—	1.40e-05	—	3.95e-06	—	1.04e-06	—
10	5.94e-05	2.20	9.59e-06	2.45	1.69e-06	3.05	2.70e-07	3.86	3.78e-08	4.78
20	1.52e-05	1.96	1.62e-06	2.56	1.56e-07	3.43	1.28e-08	4.39	9.10e-10	5.37
40	4.10e-06	1.89	2.42e-07	2.73	1.20e-08	3.70	4.97e-10	4.69	1.76e-11	5.68
80	1.08e-06	1.91	3.34e-08	2.86	8.32e-10	3.85	1.72e-11	4.84	3.08e-13	5.84
160	2.82e-07	1.94	4.39e-09	2.92	5.48e-11	3.92	5.69e-13	4.92	5.08e-15	5.92
320	7.22e-08	1.96	5.63e-10	2.96	3.51e-12	3.96	1.82e-14	4.96	8.15e-17	5.96
640	1.82e-08	1.98	7.12e-11	2.98	2.22e-13	3.98	5.79e-16	4.98	1.30e-18	5.96

Table 5 Example 4 (stiff linear problem (4.4)): numerical errors and orders for AIT method.

5 Conclusions

In this work we have reviewed the implicit Taylor methods for ODEs and showed how they can systematically be implemented, although at the cost of differentiating the function in the ODE up to the order of the method. Using the same strategy that led to approximate explicit Taylor methods for ODEs, that only require function evaluations, we propose approximate implicit Taylor methods, whose only requirement is the knowledge of function derivatives to build the Jacobian of auxiliary systems of nonlinear equations, to be solved by Newton's method. We show that the novel approach introduced in this work outperforms the exact version in terms of performance, except for low order accuracy.

	$R=2$		$R=3$		$R=4$		$R=5$		$R=6$	
N	$e(N)$	$o(N)$	$e(N)$	$o(N)$	$e(N)$	$o(N)$	$e(N)$	$o(N)$	$e(N)$	$o(N)$
10	3.39e24	—	1.24e34	—	3.29e42	—	9.71e49	—	4.39e56	—
20	3.93e37	-43.40	4.38e50	-54.97	1.43e61	-61.91	5.80e69	-65.69	6.93e76	-67.10
40	4.11e50	-43.25	4.96e63	-43.37	1.01e72	-36.05	3.26e76	-22.42	9.30e77	-3.75
80	2.99e46	13.75	1.16e46	58.57	9.17e37	113.1	4.03e19	189.0	1.60e00	258.3
160	3.37e-03	162.6	3.41e-03	161.2	7.99e-04	136.4	2.66e-04	77.01	6.34e-05	14.62
320	7.07e-04	2.25	2.03e-04	4.07	3.46e-05	4.53	5.26e-06	5.66	6.60e-07	6.59
640	1.67e-04	2.08	1.95e-05	3.38	1.73e-06	4.32	1.29e-07	5.35	8.13e-09	6.34
1280	3.88e-05	2.10	2.15e-06	3.19	9.61e-08	4.17	3.52e-09	5.19	1.12e-10	6.18

Table 6 Example 4 (stiff linear problem (4.4)): numerical errors and orders for AET method.

Acknowledgements A.B. and P.M. are supported by Spanish MINECO grant MTM2017-83942-P. R.B. is supported by Fondecyt project 1170473; Fondef project ID15I10291; and CRHIAM, Proyecto Conicyt Fondap 15130015. In addition, R.B. and M.C.M. are supported by BASAL project PFB 03, CMM, Universidad de Chile and CI^{2}MA, Universidad de Concepción. D.Z. is supported by Conicyt Fondecyt/ Postdoctorado/ 3170077. PM is supported by Conicyt (Chile), project PAI-MEC, folio 80150006.

References

1. Akinfenwa, O.A., Jator, S.N., Yao, N.M.: Continuous block backward differentiation formula for solving stiff ordinary differential equations, Comput. Math. Appl., 65, 996–1005 (2013)
2. Baeza, A., Boscarino, S., Mulet, P., Russo, G. and Zorío, D.: Approximate Taylor methods for ODEs, Comput. Fluids, 159, 156–166 (2017)
3. Baeza, A., Boscarino, S., Mulet, P., Russo, G. and Zorío, D.: On the stability of approximate Taylor methods for ODE and the relationship with Runge-Kutta schemes, Preprint, arXiv:1804.03627v1
4. Faà di Bruno, F.: Sullo sviluppo delle Funzioni, Annali di Scienze Matematiche e Fisiche, 6, 479–480 (1855)
5. Kirlinger, G., Corliss, G. F.: On implicit Taylor series methods for stiff ODEs, Argonne National Laboratory technical report ANL/CP-74795 (1991)
6. Miletics, E., Molnárka, G.: Taylor series method with numerical derivatives for numerical solution of ODE initial value problems, J. Comput. Methods Sci. Eng., 4, 105–114 (2004)
7. Miletics, E., Molnárka, G.: Implicit extension of Taylor series method with numerical derivatives for initial value problems, Comput. Math. Appl., 50(7), 1167–1177 (2005)
8. Qiu, J., Shu, C.W.: Finite difference WENO schemes with Lax-Wendroff-type time discretizations, SIAM J. Sci. Comput., 24(6), 2185–2198 (2003)
9. Scott, J. R.: Solving ODE initial value problems with implicit Taylor series methods, NASA technical memorandum TM-2000-209400 (2000)
10. Zorío, D., Baeza, A., Mulet, P.: An approximate Lax-Wendroff-type procedure for high order accurate schemes for hyperbolic conservation laws, J. Sci. Comput., 71(1), 246–273 (2017)

Centro de Investigación en Ingeniería Matemática (CI^{2}MA)

PRE-PUBLICACIONES 2018

2018-08 Gabriel N. Gatica, Bryan Gomez-Vargas, Ricardo Ruiz-Baier: *Formulation and analysis of fully-mixed methods for stress-assisted diffusion problems*

2018-09 Jay Gopalakrishnan, Manuel Solano, Felipe Vargas: *Dispersion analysis of HDG methods*

2018-10 Franco Fagnola, Carlos M. Mora: *Bifurcation analysis of a mean field laser equation*

2018-11 David Mora, Iván Velásquez: *A virtual element method for the transmission eigenvalue problem*

2018-12 Alfredo Bermúdez, Bibiana López-Rodríguez, Rodolfo Rodríguez, Pilar Salgado: *Numerical solution of a transient three-dimensional eddy current model with moving conductors*

2018-13 Raimund Bürger, Enrique D. Fernández Nieto, Victor Osores: *A dynamic multilayer shallow water model for polydisperse sedimentation*

2018-14 Antonio Baeza, Raimund Bürger, Pep Mulet, David Zorío: *Weno reconstructions of unconditionally optimal high order*

2018-15 Veronica Anaya, Mostafa Bendahmane, David Mora, Mauricio Sepúlveda: *A virtual element method for a nonlocal FitzHugh-Nagumo model of cardiac electrophysiology*

2018-16 Tomás Barrios, Rommel Bustinza: *An a priori error analysis for discontinuous Lagrangian finite elements applied to nonconforming dual mixed formulations: Poisson and Stokes problems*

2018-17 Raimund Bürger, Gerardo Chowell, Elvis Gavilán, Pep Mulet, Luis M. Villada: *Numerical solution of a spatio-temporal predator-prey model with infected prey*

2018-18 Javier A. Almonacid, Gabriel N. Gatica, Ricardo Oyarzúa, Ricardo Ruiz-Baier: *A new mixed finite element method for the n-dimensional Boussinesq problem with temperature-dependent viscosity*

2018-19 Antonio Baeza, Raimund Bürger, María Carmen Martí, Pep Mulet, David Zorío: *Approximate implicit Taylor methods for ODEs*

Para obtener copias de las Pre-Publicaciones, escribir o llamar a: Director, Centro de Investigación en Ingeniería Matemática, Universidad de Concepción, Casilla 160-C, Concepción, Chile, Tel.: 41-2661324, o bien, visitar la página web del centro: http://www.ci2ma.udec.cl







Centro de Investigación en
Ingeniería Matemática (CI²MA)
Universidad de Concepción

Casilla 160-C, Concepción, Chile
Tel.: 56-41-2661324/2661554/2661316
http://www.ci2ma.udec.cl