UHASSELT

Parallel-in-time high-order multiderivative IMEX solvers

J. Schütz, D. Seal and J. Zeifang

This preprint has not undergone peer review or any post-submission improvements or corrections. The Version of Record of this article is published in Journal of Scientific Computing, and is available online at https://link.springer.com/article/10.1007/s10915-021-01733-3.

UHasselt Computational Mathematics Preprint
Nr. UP-21-01

Dec. 13th, 2021

Noname manuscript No.
(will be inserted by the editor)

Parallel-in-time high-order multiderivative IMEX solvers

Jochen Schütz · David C. Seal · Jonas Zeifang

Received: date / Accepted: date

Abstract In this work, we present a novel class of high-order time integrators for the numerical solution of ordinary differential equations. These integrators are of the two-derivative type, i.e., they take into account not only the first, but also the second temporal derivative of the unknown solution. While being motivated by two-derivative Runge-Kutta schemes, the methods themselves are of the predictor-corrector type, constructed in such a way that time-parallelism through pipelining is possible. This means that predictor and corrector steps can be computed simultaneously on different processors. Two variants are shown, the second-one being of Gauss-Seidel-type and hence having lower storage requirements. It turns out that this second variant is not only low-storage, but also performs better in numerical simulations. The algorithms presented can be cast as implicit two-step-two-derivative-multi-stage methods for which we present a detailed mathematical convergence analysis. Subsequently, numerical results are shown, demonstrating first the algorithms' ability to cope with very stiff equations and showing up to eighth order of accuracy. Following, the parallel-in-time capabilities are illustrated. We conclude that the class of methods is very well-suited if the time parallelization of very stiff equations is an option.

Keywords Multiderivative · IMEX · parallelism in time · time integration

J. Schütz
Faculty of Sciences and Data Science Institute, Universiteit Hasselt,
Agoralaan Gebouw D, BE-3590 Diepenbeek, Belgium
E-mail: jochen.schuetz@uhasselt.be

D. C. Seal
United States Naval Academy, Department of Mathematics,
572C Holloway Road, Annapolis, MD 21402, USA
E-mail: seal@usna.edu

J. Zeifang
Faculty of Sciences and Data Science Institute, Universiteit Hasselt,
Agoralaan Gebouw D, BE-3590 Diepenbeek, Belgium
E-mail: jonas.zeifang@uhasselt.be

1 Introduction

In this work, we consider numerical approximations of nonlinear differential equations of the form

$$\begin{aligned} w'(t) &= \Phi(w) \equiv \Phi_{\mathrm{I}}(w) + \Phi_{\mathrm{E}}(w), \quad t \in (0, T_{end}), \\ w(0) &= w_0. \end{aligned} \tag{1}$$

It is assumed that the right-hand side Φ is split into the contributions Φ_{I} ('stiff') and Φ_{E} ('non-stiff'), to account for different scales in the solution. The rationale is that the term Φ_{I} will be treated implicitly to obtain a stable method, while the term Φ_{E} will be treated explicitly, typically for efficiency reasons by reducing the complexity of the algebraic solves required to address the implicit terms per time step.

Differential equations that can be put into this form arise frequently, often resulting from the discretization of singularly perturbed partial differential equations. Some examples include problems in meteorology [26], geophysics [8], aerodynamics [11,27,50], molecular dynamics [24] and many more. The list is by no means exhaustive. Here, we assume a splitting has already been performed. Our goal is to develop a high-order implicit-explicit (IMEX) method that lends itself to time-parallelism.

At the core of this work is the novel discretization method of Eq. 1 developed in [42]. To the best of the authors' knowledge, this method is, together with the very recent publication [3], the first one to combine the IMEX (implicit/explicit, see, e.g., [5,6,9,33,39]) and the multiderivative paradigms [4,14,17,32,44,48]. In this context, the term *multiderivative* refers to the fact that higher derivatives of the unknown solution w are used in the numerical method. While more established methods (Runge-Kutta, Adams methods, backward differentiation formulas (BDF) and the like) rely on only evaluating Φ_{E} and Φ_{I}, multiderivative methods also take the temporal derivatives into account. To be more precise, we use two derivatives, i.e., also the quantities[1]

$$\dot{\Phi}_{\mathrm{I}}(w) := \Phi_{\mathrm{I}}'(w)\Phi(w), \qquad \text{and} \qquad \dot{\Phi}_{\mathrm{E}}(w) := \Phi_{\mathrm{E}}'(w)\Phi(w)$$

are taken into account; and necessarily

$$w''(t) = \dot{\Phi}(w) \equiv \dot{\Phi}_{\mathrm{I}}(w) + \dot{\Phi}_{\mathrm{E}}(w).$$

The use of multiple derivatives has the potential to increase the order of consistency without adding more stages or steps, respectively. We limit ourselves to the use of two derivatives only for the ease of presentation. The method developed in [42] has already been extended to incorporate higher derivatives [21]; and also in the context of this present work, this is possible.

There are three unique features to consider for the class of methods presented here:

[1] The dot here $(\cdot)$ refers to the time derivative d/dt, whereas the prime $(')$ refers to the Jacobian of the vector valued Φ_{I} and Φ_{E}.

- The methods we present use a **predictor-corrector** strategy for increasing the overall order of accuracy. This means that in each timestep, a predicted value (we will indicate values corresponding to the predictor with [0]), as well as several corrector steps (indicated with $[k]$, $1 \leq k \leq k_{\max}$) are computed. Here, $k_{\max}$ is a user-defined parameter that defines the maximum number of corrections in a single time step. Similiar to deferred correction (DC), or DC-type methods (cf. [13,22,38] and the references therein), the corrected steps pick up an order of accuracy each time, until some maximum accuracy is reached based on the underlying quadrature rule.
- The methods we construct are derived from two-derivative Runge-Kutta methods, however, they are modified in such a way that they can be applied to any **implicit-explicit (IMEX) splitting** $\Phi(w) = \Phi_{\mathrm{I}}(w) + \Phi_{\mathrm{E}}(w)$ of the ODE.
- The methods we produce can all leverage **parallel-in-time** implementations to enable distributed or multi-core computer architectures. To preserve the high order nature one has to add a mild dependency on the previous step, that leads to a two-step multi-stage method. Hence the method has a general linear method flavor with associated convergence theory [12], but with higher derivatives of the unknown involved.

The ability to parallelize an ODE solver in time has become more and more important in recent years with the number of processors available growing constantly. Because of the causality principle, parallelization in time poses different challenges than parallelization in space. For an overview of temporally parallel methods, we refer to Gander's work [25] and the very recent overview article by Ong and Schroder [37]. Also the website [1] is a beautiful source of information on the topic. Temporally parallel methods can be cast into different classes. The scheme we present in this work uses the concept of pipelining, i.e., the information and data flow is directed in such a way that the correction iterations can be put on different processors. We have been inspired by the RIDC (revisionist integral deferred correction) schemes [16,18,19], however, similar ideas have been around for a long time, see [36]. The essence of pipelining is that, given enough processors at one's disposal, the higher order achieved through the corrections can be obtained in roughly the same wallclock-time that it takes to compute the lower-order predictor. In our case, the predictor's convergence order is two, while the corrections can increase this order up to the order of the underlying Runge-Kutta quadrature. In our examples, this is four, six and eight, respectively, but conceptually, the order is not limited to this.

The paper is organized as follows: In Sec. 2, we describe the underlying algorithm in detail and introduce the necessary notation. Thereafter, in Sec. 3, we analyze stability and convergence of the method through writing it in a one-step fashion. The method is inherently time-parallel, which is laid out in Sec. 4. Also, two alternative approaches are shown and later compared. In Sec. 5 we show numerical results for well-known test cases. We find some improvements that one can make to the numerical algorithm described earlier; these are shown in Sec. 6. Subsequently, scaling results are presented. As usual, the last section, Sec. 7 offers conclusion and outlook.

2 Numerical algorithm

We start with setting the notation that is used in this work. For expository purposes, we work with a fixed timestep Δt, and therefore with a fixed total number of timesteps, denoted by N, we must have

$$\Delta t := \frac{T_{end}}{N}.$$

The discrete time levels are defined as

$$t^n := n\Delta t, \qquad 0 \leq n \leq N.$$

Note that the uniform timestep we assume here is not necessary in practical computations, but it simplifies the analysis. Throughout the whole publication, we assume that Φ and the split functions Φ_{I}, Φ_{E} and the solution w are sufficiently smooth to warrant picking up high-order accuracy from the numerical method. Formally, we make the following assumption:

Assumption 1 *All occuring functions Φ, Φ_I, Φ_E and their temporal derivatives $\dot{\Phi}, \dot{\Phi}_I$, $\dot{\Phi}_E$ are assumed to be smooth and Lipschitz continuous. Lipschitz constants are denoted with L_Φ, L_{Φ_I} and the like.*

In this work, we are interested in multiderivative Runge-Kutta methods. These are a relatively straightforward extension of well-known Runge-Kutta methods that include extra derivatives of the right hand side function [44, Definition 1]). To set notation, we give a full definition of a *two*-derivative Runge-Kutta method. Higher derivatives can be included by adding more terms in the expansion for each of the stages.

Definition 1 [44] Let w^n denote an approximate solution to w at point t^n, and let $w^{n,l}$, $1 \leq l \leq s$ denote the stage values to be computed. A *two-derivative Runge-Kutta method* is any method that can be cast as

$$w^{n,l} := w^n + \Delta t \sum_{j=1}^{s} B^{(1)}_{lj} \Phi(w^{n,j}) + \Delta t^2 \sum_{j=1}^{s} B^{(2)}_{lj} \dot{\Phi}(w^{n,j}), \qquad 1 \leq l \leq s,$$

and an update given by

$$w^{n+1} := w^n + \Delta t \sum_{j=1}^{s} b^{(1)}_{j} \Phi(w^{n,j}) + \Delta t^2 \sum_{j=1}^{s} b^{(2)}_{j} \dot{\Phi}(w^{n,j}),$$

where the $B^{(1)}$, $B^{(2)}$, $b^{(1)}$, and $b^{(2)}$ are the Butcher tableaux that define the scheme. We say the method is *globally stiffly accurate* if $b^{(1)}_j = B^{(1)}_{sj}$ and $b^{(2)}_j = B^{(2)}_{sj}$, i.e., the last stage defines the update:

$$w^{n+1} := w^{n,s}.$$

Note that this is a fully coupled system of (potentially nonlinear) equations whenever the tableaux are not lower triangular. In this form, Runge-Kutta methods are hence typically not suited for high-dimensional ODEs, nonetheless, they can serve as excellent background methods for creating more efficient solvers.

Remark 1 Note that the implicit-explicit flavor of our methods does not come from the underlying Runge-Kutta method, but from implicit-explicit stabilization terms which will be explained later. Similar approaches have also been pursued for implicit-explicit SDC methods, see, e.g., [35].

Of particular note are the two-derivative Runge-Kutta *collocation* methods. These methods are derived by fitting a Hermite-Birkhoff polynomial interpolant through the time instances given by $c\Delta t$ and integrating the result. In this work, we use equidistant collocation points, which implies:

– the method is of order q, with $q = 2s$ being twice the number of stages (hence, number of elements in c);
– the stage order is also q;
– the last stage is equal to the update step, which corresponds to the globally stiffly accurate or first-same-as-last property already known in standard Runge-Kutta methods [10,28,29] and will have an impact on asymptotic properties of the method [41].

Example 1 (Two-derivative Hermite-Birkhoff collocation methods) In this work, we use the following two-derivative Runge-Kutta methods:

– A fourth-order method ($q = 4$) with two stages ($s = 2$, one being fully explicit), which exactly corresponds to the method used in [42]:

$$c = \begin{pmatrix} 0 \\ 1 \end{pmatrix}, \quad B^{(1)} = \begin{pmatrix} 0 & 0 \\ \frac{1}{2} & \frac{1}{2} \end{pmatrix}, \quad B^{(2)} = \begin{pmatrix} 0 & 0 \\ \frac{1}{12} & \frac{-1}{12} \end{pmatrix}. \tag{2}$$

– A sixth-order method ($q = 6$) with three stages ($s = 3$, one being fully explicit), as also used in [43]:

$$c = \begin{pmatrix} 0 \\ \frac{1}{2} \\ 1 \end{pmatrix}, \quad B^{(1)} = \begin{pmatrix} 0 & 0 & 0 \\ \frac{101}{480} & \frac{8}{30} & \frac{55}{2400} \\ \frac{7}{30} & \frac{16}{30} & \frac{7}{30} \end{pmatrix}, \quad B^{(2)} = \begin{pmatrix} 0 & 0 & 0 \\ \frac{65}{4800} & -\frac{25}{600} & -\frac{25}{8000} \\ \frac{5}{300} & 0 & -\frac{5}{300} \end{pmatrix}. \tag{3}$$

– An eigth-order method ($q = 8$) with four stages ($s = 4$, one being fully explicit):

$$\begin{aligned} c = \begin{pmatrix} 0 \\ \frac{1}{3} \\ \frac{2}{3} \\ 1 \end{pmatrix}, \quad B^{(1)} &= \begin{pmatrix} 0 & 0 & 0 & 0 \\ \frac{6893}{54432} & \frac{313}{2016} & \frac{89}{2016} & \frac{397}{54432} \\ \frac{223}{1701} & \frac{20}{63} & \frac{13}{63} & \frac{20}{1701} \\ \frac{31}{224} & \frac{81}{224} & \frac{81}{224} & \frac{31}{224} \end{pmatrix}, \\ B^{(2)} &= \begin{pmatrix} 0 & 0 & 0 & 0 \\ \frac{1283}{272160} & -\frac{851}{30240} & -\frac{269}{30240} & -\frac{163}{272160} \\ \frac{43}{8505} & -\frac{16}{945} & -\frac{19}{945} & -\frac{8}{8505} \\ \frac{19}{3360} & -\frac{9}{1120} & \frac{9}{1120} & -\frac{19}{3360} \end{pmatrix}. \end{aligned} \tag{4}$$

We choose to use the above-mentioned class of methods with equispaced abscissa for the sake of a clearer presentation of results. Note that $c_1 = 0$ for all considered collocation methods. This will facilitate the reformulation of the novel parallel-in-time method as a "one-step equivalent", see Sec. 3. Other multiderivative Runge-Kutta methods can also be used, of course with the necessary modifications. Higher derivatives can be incorporated as an alternative option to further increase the overall order.

With each Runge-Kutta method, there is an underlying quadrature rule connected to the solver. That is, given some function values $\Phi(w^j)$, $1 \leq j \leq s$, we define the "quadrature rule" as the multiderivative Runge-Kutta flux update, given by

$$\mathcal{I}_l(\Phi(w^1), \dots \Phi(w^s)) := \Delta t \sum_{j=1}^{s} B^{(1)}_{lj} \Phi(w^j) + \Delta t^2 \sum_{j=1}^{s} B^{(2)}_{lj} \dot{\Phi}(w^j). \tag{5}$$

These terms are precisely the terms required to produce each stage in the Runge-Kutta method. They may or may not be high-order quantities, unless further assumptions are made about the solver. In the case of the Hermite-Birkhoff-type approach, they are of higher order:

Lemma 1 *For each Hermite-Birkhoff Runge-Kutta method, there holds,*

$$\int_{t^n}^{t^n + c_l \Delta t} \Phi(w(t)) \mathrm{d}t = \mathcal{I}_l \left(\Phi(w(t^n + c_1 \Delta t)), \dots, \Phi(w(t^n + c_s \Delta t)) \right) + \mathcal{O}(\Delta t^{q+1}).$$

Proof This is due to the construction: The Hermite-Birkhoff polynomial is of order $2s - 1 = q - 1$, which gives an integral approximation of order $q + 1$. (Note that the length of the integration area is Δt.) □

We exploit precisely this quadrature to extend the method shown in [42] to higher orders and parallelism in time. Before we actually define the algorithm, let us clarify the notation used. The algorithm to be presented relies on the approximated quantities:

$$w^{n,[k],l} \approx w(t^n + c_l \Delta t), \quad 0 \leq n \leq N, \quad 0 \leq k \leq k_{\max}, \quad 1 \leq l \leq s. \tag{6}$$

Here, n is the usual discrete time level, and the l refers to the stages that are present within the Runge-Kutta method. The index k is a parameter that is associated to 'correction' steps (to be explained below). The final index $k_{\max}$ is a fixed parameter chosen by the user to describe the total number of iterations sought. For non-stiff problems, it is typically dictated by the maximum order that can be reached, while for stiff problems, it can be advantageous to use a larger $k_{\max}$ to overcome convergence issues [42], or even allow the algorithm to choose this adaptively.

Finally, we make the abbreviations

$$\Phi^{n,[k],l} := \Phi\left(w^{n,[k],l}\right), \quad \Phi^{n,[k],l}_{\mathrm{I}} := \Phi_{\mathrm{I}}\left(w^{n,[k],l}\right), \quad \Phi^{n,[k],l}_{\mathrm{E}} := \Phi_{\mathrm{E}}\left(w^{n,[k],l}\right),$$

to simplify the notation.

We are now ready to formulate the following algorithm. Given the Butcher tableaux $B^{(1)}$ and $B^{(2)}$ and the time instances c, with $c_1 = 0$, (e.g., any of the methods from Example 1), the **H**ermite-**B**irkhoff **P**redictor **C**orrector (**HBPC**) method is as follows:

Algorithm 1 (HBPC$(q, k_{\max},)$) *To advance the solution to Eq.* (1) *from time level t^n to time level t^{n+1}, fill the values $w^{n,[0],l}$ using a second-order IMEX-Taylor method:*

1. ***Predict.*** *Solve the following expression for $w^{n,[0],l}$ and each $2 \leq l \leq s$:*

$$\begin{aligned} w^{n,[0],1} &:= w^{n-1,[0],s}, \\ w^{n,[0],l} &:= w^{n-1,[0],s} + c_l \Delta t \left(\Phi_I^{n,[0],l} + \Phi_E^{n-1,[0],s} \right) \\ &\qquad + \frac{(c_l \Delta t)^2}{2} \left(\dot{\Phi}_E^{n-1,[0],s} - \dot{\Phi}_I^{n,[0],l} \right). \end{aligned} \tag{7}$$

Subsequently:

2. ***Correct.*** *Solve the following for $w^{n,[k+1],l}$, for each $2 \leq l \leq s$ and each $0 \leq k < k_{\max}$:*

$$\begin{aligned} w^{n,[k+1],1} &:= w^{n-1,[k+2],s}, \\ w^{n,[k+1],l} &:= w^{n-1,[k+2],s} + \Delta t \left(\Phi_I^{n,[k+1],l} - \Phi_I^{n,[k],l} \right) \\ &\quad - \frac{\Delta t^2}{2} \left(\dot{\Phi}_I^{n,[k+1],l} - \dot{\Phi}_I^{n,[k],l} \right) + \mathcal{I}_l(\Phi^{n,[k],0}, \ldots, \Phi^{n,[k],s}), \end{aligned} \tag{8}$$

where $\mathcal{I}_l$ is the q-th order Hermite-Birkhoff quadrature rule given in Eq. (5)*. In order to close the recursion, if $k = k_{\max} - 1$, then replace each of the $k+2$ with $k_{\max}$.*

3. ***Update.*** *In order to preserve the first-same-as-last property, we put*

$$w^{n+1} := w^{n,[k_{\max}],s}.$$

Finally, to seed initial conditions for this solver, we define

$$w^{-1,[k],s} := w_0, \qquad 0 \leq k \leq k_{\max}.$$

Note that the starting value of $k = 0$ denotes a straightforward second-order IMEX-Taylor scheme; each $k > 0$ represents a corrected version thereof, taking into account the quadrature rule obtained from the Runge-Kutta scheme. The reasoning is that with each additional correction, we see one extra order of convergence until the maximal order of the Runge-Kutta method, q, has been found.

Remark 2 The difference between this algorithm and the one found in [42] lies in the highlighted red terms:

$$\left\{ w^{n-1,[0],s}, \Phi_{\mathrm{E}}^{n-1,[0],s}, \dot{\Phi}_{\mathrm{E}}^{n-1,[0],s}, w^{n-1,[k+2],s} \right\}.$$

In the original work [42] each of these would have been evaluated at w^n. Here, we work with different iterate numbers. **This potent modification makes it possible to parallelize the method in time through pipelining without sacrificing the order of accuracy.** The concept of pipelining will be explained in Sec. 4 and a sketch of the iterate's dependencies is provided in Fig. 1. The idea that will be layed out in this publication is very similar to – and in fact inspired by – the strategy proposed in [16,18].

Remark 3 In this work, we exclusively treat $c_1 = 0$. For $c_1 \neq 0$, which might become necessary when considering Gauss-type quadrature rules, the first stage in both (7) and (8) has to be treated in the same way as the following stages.

Remark 4 Please note that Alg. 1 is not a Runge-Kutta method.

Remark 5 There is no implicitness in the Runge-Kutta quadrature $\mathcal{I}_l$. The only implicitness is in the difference of the Φ_{I} and $\dot{\Phi}_{\mathrm{I}}$. This term is used to introduce the necessary stability for stiff problems.

In the next three sections we present a thorough analysis of this solver followed by a discussion of a parallelization strategy and numerical results. In Section 6, we suggest an alternative formulation that provides a low-storage alternative and even better scaling results.

3 Convergence analysis

Due to the changes made in the algorithm in comparison to the one proposed in [42], this is no longer a one-step multiderivative time integrator, but a general linear method. It has both a multistage as well as a multistep flavour, and we need to follow the convergence analysis outlined in the seminal work by Butcher [12] to conduct our analysis. To ease the presentation, we assume that we are treating a scalar differential equation, i.e., we postulate that

$$\Phi : \mathbb{R} \to \mathbb{R}\,.$$

We start by rewriting Alg. 1 to make it more accessible to a convergence analysis. Define the vector $Y^{n+1} \in \mathbb{R}^{s\cdot(k_{\max}+1)}$ consisting of all stages and all correction steps at time instance $t^n = n\Delta t$, $n > 0$, i.e.,

$$Y^{n+1} := \left(w^{n,[0],1}, \cdots, w^{n,[0],s}, w^{n,[1],1}, \cdots, w^{n,[1],s}, \cdots, w^{n,[k_{\max}],s}\right)^T \in \mathbb{R}^{s\cdot(k_{\max}+1)}\,.$$

Note that we have made a shift in the n for a clearer exposition. For $n = 0$, we define Y^0 to be the vector with entries consisting of the solution at time $t = 0$:

$$Y^0 := \left(w_0, \cdots, w_0\right)^T \in \mathbb{R}^{s\cdot(k_{\max}+1)}\,.$$

Then, we can write the time integrator from Alg. 1 in a "one-step-fashion" as

$$\begin{aligned} Y^{n+1} = AY^n &+ \Delta t\left(B_E\Phi_{\mathrm{E}}(Y^{n+1}) + B_I\Phi_{\mathrm{I}}(Y^{n+1})\right) \\ &+ \frac{\Delta t^2}{2}\left(\tilde{B}_E\dot{\Phi}_{\mathrm{E}}(Y^{n+1}) + \tilde{B}_I\dot{\Phi}_{\mathrm{I}}(Y^{n+1})\right) \end{aligned} \tag{9}$$

for matrices A, B_I, B_E, $\tilde{B}_E$ and $\tilde{B}_I$ in $\mathbb{R}^{s\cdot(k_{\max}+1)\times s\cdot(k_{\max}+1)}$. While A is given in Eq. (10), the remaining matrices can be found in the appendix A. Note that due to $c_1 = 0$ for all considered Hermite-Birkhoff quadrature rules, no additional dependency on Y^n is obtained for the explicit contributions Φ_{E} and $\dot{\Phi}_{\mathrm{E}}$ in Eq. (9).

We use the well-known inf-norm $\|\cdot\|_\infty$ for both matrices and vectors. For the sake of readability, we write $\|\cdot\|$ instead of $\|\cdot\|_\infty$. That is, for $v \in \mathbb{R}^\kappa$, we define $\|v\| := \max_{1\le i\le\kappa} |v_i|$; for matrices $C \in \mathbb{R}^{\kappa\times\kappa}$, there holds

$$\|C\| := \sup_{v\in\mathbb{R}^\kappa} \frac{\|Cv\|}{\|v\|}.$$

Lemma 2 *The matrix A is power-bound by one, i.e., for each $n \in \mathbb{N}$, there holds*

$$\|A^n\| \le 1.$$

Proof Define the matrix $\mathfrak{E} \in \mathbb{R}^{s\times s}$ that is zero except for the last column, which is filled with ones, i.e.,

$$\mathfrak{E} = \begin{pmatrix} 0 & 0 & \cdots & 0 & 1 \\ 0 & 0 & \cdots & 0 & 1 \\ \vdots & \vdots & & \vdots & \vdots \\ 0 & 0 & \cdots & 0 & 1 \end{pmatrix}.$$

Then, A is given as the block matrix

$$A = \begin{pmatrix} \mathfrak{E} & 0 & 0 & 0 & \cdots & 0 & 0 \\ 0 & 0 & \mathfrak{E} & 0 & \cdots & 0 & 0 \\ 0 & 0 & 0 & \mathfrak{E} & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & 0 & \cdots & \mathfrak{E} & 0 \\ 0 & 0 & 0 & 0 & \cdots & 0 & \mathfrak{E} \\ 0 & 0 & 0 & 0 & \cdots & 0 & \mathfrak{E} \end{pmatrix}. \tag{10}$$

There holds $\|Av\| \le \|v\|$, which can be seen due to the fact that for each $1 \le i \le s\cdot(k_{\max}+1)$, there is a $1 \le j \le s\cdot(k_{\max}+1)$ such that $(Av)_i = v_j$. This concludes the proof. □

Corollary 1 *There exists a constant $C \in \mathbb{R}$ such that for each $j \in \mathbb{N}^{\ge 0}$, there holds*

$$\|A^j B_E\| \le C, \quad \|A^j B_I\| \le C, \quad \|A^j \tilde{B}_E\| \le C, \quad \|A^j \tilde{B}_I\| \le C.$$

Similarly to [12], we define the vector W^n as the vector of the exact solutions at time t^n, i.e.,

$$W^{n+1} := \begin{pmatrix} w(t^n + c_1\Delta t) \\ \vdots \\ w(t^n + c_s\Delta t) \\ w(t^n + c_1\Delta t) \\ \vdots \\ w(t^n + c_s\Delta t) \end{pmatrix} \in \mathbb{R}^{s\cdot(k_{\max}+1)}, \qquad n \ge 0.$$

We define $W^0 := Y^0$. For a convergence analysis, we are interested in the difference between the approximate and the exact solution,

$$Z^n := Y^n - W^n.$$

This quantity satisfies the following time-discrete equation:

$$\begin{aligned} Z^n \equiv& Y^n - W^n \\ &= AZ^{n-1} + \Delta t B_E \left(\Phi_{\mathrm{E}}(Y^n) - \Phi_{\mathrm{E}}(W^n)\right) + \Delta t B_I \left(\Phi_{\mathrm{I}}(Y^n) - \Phi_{\mathrm{I}}(W^n)\right) \\ &\quad + \frac{\Delta t^2}{2}\tilde{B}_E \left(\dot{\Phi}_{\mathrm{E}}(Y^n) - \dot{\Phi}_{\mathrm{E}}(W^n)\right) + \frac{\Delta t^2}{2}\tilde{B}_I \left(\dot{\Phi}_{\mathrm{I}}(Y^n) - \dot{\Phi}_{\mathrm{I}}(W^n)\right) \\ &\quad + E^n, \end{aligned} \tag{11}$$

with E^n the local consistency error

$$\begin{aligned} E^n :=& AW^{n-1} + \Delta t\left(B_E \Phi_{\mathrm{E}}(W^n) + B_I \Phi_{\mathrm{I}}(W^n)\right) \\ &+ \frac{\Delta t^2}{2}\left(\tilde{B}_E \dot{\Phi}_{\mathrm{E}}(W^n) + \tilde{B}_I \dot{\Phi}_{\mathrm{I}}(W^n)\right) - W^n. \end{aligned}$$

As in [42], we can prove that the local consistency error at correction level k is given by $\mathcal{O}\left(\Delta t^{\min\{2+k,q\}+1}\right)$. For $k = 0$, this is straightforward – this is second-order IMEX-Taylor. For $k > 0$, techniques as in [42] (and long known to the spectral deferred correction (SDC) community, see, e.g., [22]) can be employed.

Lemma 3 *Subdivide E^n into $k_{\max} + 1$ blocks of size s each; call the individual blocks $E^{n,[k]}$, $0 \le k \le k_{\max}$. Then, there holds*

$$E^{n,[k]} = O(\Delta t^{\min\{2+k,q\}+1}).$$

Proof The proof goes along the same lines as in [42] and is hence omitted.

With this, it is straightforward to show second-order convergence:

Theorem 1 *Given that Δt is sufficiently small, and given the assumptions on the functions Φ, Φ_I and Φ_E as done in Assumption 1, there is a constant C such that*

$$Z^n \le C\Delta t^2,$$

and hence, the method is second-order convergent.

Proof Using (11), the fact that $\|A\| \le 1$, and some straightforward Lipschitz-inequalities, one can show that

$$\|Z^n\| \le \|Z^{n-1}\| + \tilde{C}\Delta t\|Z^n\| + \|E^n\|$$

for some constant $\tilde{C}$. Given the smallness condition $\tilde{C}\Delta t \le \alpha < 1$, one can deduce the inequality

$$\|Z^n\| \le (1 + C\Delta t)\|Z^{n-1}\| + (1 + C\Delta t)\|E^n\|,$$

where C is a constant chosen in such a way that $\frac{1}{1-\tilde{C}\Delta t} \le 1 + C\Delta t$. From this and the fact that $E^n \le C\Delta t^3$ (note again: generic constants C are subject to change), one can deduce via recursion and standard estimates that

$$\begin{aligned} \|Z^n\| &\le (1 + C\Delta t)\|Z^{n-1}\| + (1 + C\Delta t)\|E^n\| \le \dots \\ &\le (1 + C\Delta t)^n\|Z^0\| + C\Delta t^3 \frac{(1 + C\Delta t)^n - 1}{C\Delta t} \\ &\le (1 + C\Delta t)^n\|Z^0\| + C\Delta t^2. \end{aligned}$$

With $Z^0 = 0$, this yields the result. □

Second order convergence is of course not completely what we expect from Alg. 1. In the following, we explore the nature of the higher orders in some more detail.

Remark 6 The structure of the matrix A as given in (10) is very interesting: it "pushes elements in vectors downwards". To facilitate the understanding of the following analysis, we show different exponents of A. Obviously, $A^0 = \mathrm{Id}$, the identiy matrix, and A^1 is given in (10). There holds, always given that $k_{\max}$ is large enough:

$$A^2 = \begin{pmatrix} \mathfrak{E} & 0 & 0 & 0 & 0 & \cdots & 0 & 0 & 0 \\ 0 & 0 & 0 & \mathfrak{E} & 0 & \cdots & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & \mathfrak{E} & \cdots & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & 0 & 0 & \cdots & \mathfrak{E} & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & \cdots & 0 & \mathfrak{E} & 0 \\ 0 & 0 & 0 & 0 & 0 & \cdots & 0 & 0 & \mathfrak{E} \\ 0 & 0 & 0 & 0 & 0 & \cdots & 0 & 0 & \mathfrak{E} \\ 0 & 0 & 0 & 0 & 0 & \cdots & 0 & 0 & \mathfrak{E} \end{pmatrix}, \quad A^3 = \begin{pmatrix} \mathfrak{E} & 0 & 0 & 0 & 0 & \cdots & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & \mathfrak{E} & \cdots & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & \cdots & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \\ 0 & 0 & 0 & 0 & 0 & \cdots & \mathfrak{E} & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & \cdots & 0 & \mathfrak{E} & 0 \\ 0 & 0 & 0 & 0 & 0 & \cdots & 0 & 0 & \mathfrak{E} \\ 0 & 0 & 0 & 0 & 0 & \cdots & 0 & 0 & \mathfrak{E} \\ 0 & 0 & 0 & 0 & 0 & \cdots & 0 & 0 & \mathfrak{E} \\ 0 & 0 & 0 & 0 & 0 & \cdots & 0 & 0 & \mathfrak{E} \end{pmatrix}, \quad \ldots$$

Eventually, there is a limit matrix A^∞, given as

$$A^\infty = \begin{pmatrix} \mathfrak{E} & 0 & 0 & 0 & 0 & \cdots & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & \cdots & 0 & 0 & \mathfrak{E} \\ 0 & 0 & 0 & 0 & 0 & \cdots & 0 & 0 & \mathfrak{E} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & 0 & 0 & \cdots & 0 & 0 & \mathfrak{E} \\ 0 & 0 & 0 & 0 & 0 & \cdots & 0 & 0 & \mathfrak{E} \\ 0 & 0 & 0 & 0 & 0 & \cdots & 0 & 0 & \mathfrak{E} \\ 0 & 0 & 0 & 0 & 0 & \cdots & 0 & 0 & \mathfrak{E} \\ 0 & 0 & 0 & 0 & 0 & \cdots & 0 & 0 & \mathfrak{E} \\ 0 & 0 & 0 & 0 & 0 & \cdots & 0 & 0 & \mathfrak{E} \end{pmatrix}.$$

More formally, we have $A^j = A^\infty$ for all $j \geq k_{\max} - 1$. This structure is important, because it significantly simplifies the convergence analysis. Essentially, it states that most contributions to the overall error come from higher corrections. Intuitively, this is desirable, because higher corrections are supposed to have smaller error contributions, at least asymptotically.

To investigate convergence properties in more detail, Z^n is computed explicitly through the following lemma:

Lemma 4 *The recursion for Z^n given in (11) can be explicitly written as:*

$$
\begin{aligned}
Z^n =& \Delta t \sum_{j=0}^{n-1} A^j B_E \left(\Phi_E(Y^{n-j}) - \Phi_E(W^{n-j})\right) \\
&+ \Delta t \sum_{j=0}^{n-1} A^j B_I \left(\Phi_I(Y^{n-j}) - \Phi_I(W^{n-j})\right) \\
&+ \frac{\Delta t^2}{2} \sum_{j=0}^{n-1} A^j \tilde{B}_E \left(\dot{\Phi}_E(Y^{n-j}) - \dot{\Phi}_E(W^{n-j})\right) \\
&+ \frac{\Delta t^2}{2} \sum_{j=0}^{n-1} A^j \tilde{B}_I \left(\dot{\Phi}_I(Y^{n-j}) - \dot{\Phi}_I(W^{n-j})\right) \\
&+ \sum_{j=0}^{n-1} A^j E^{n-j}.
\end{aligned}
\tag{12}
$$

Proof This can be easily proved using (11) and an induction over n. Note that one has to take into account that $Z^0 = 0$ by definition. □

Lemma 5 *Subdivide Z^n into $k_{\max} + 1$ blocks of size s each, similarly to La. 3. Define ε_k^n, $0 \leq k \leq k_{\max}$ to be the infinity-norm of the $k-$th block. For $n = 0$, there is no error, and hence*

$$\varepsilon_k^0 = 0, \qquad 0 \leq k \leq k_{\max}.$$

Furthermore, for $k = 0$, the underlying consistency analysis is trivial, and hence ε_0^n can be bound by a positive constant C times Δt^2, i.e.,

$$\varepsilon_0^n \leq C \Delta t^2, \qquad 1 \leq n \leq N.$$

The constant C depends of course on the analytical solution, the fluxes Φ, the initial conditions w_0 and the like, but it does not depend on Δt or n. Furthermore, there is a constant C such that there holds for $1 \leq k < k_{\max}$ and $1 \leq n \leq N$:

$$\varepsilon_k^n \leq C\Delta t \left(\varepsilon_k^n + \varepsilon_{k-1}^n\right) + C\Delta t \sum_{j=1}^{n-1} \sum_{h=k}^{k_{\max}} \varepsilon_h^{n-j} + C\Delta t^{\min\{2+k,q\}}, \tag{13}$$

$$\varepsilon_{k_{\max}}^n \leq C\Delta t \sum_{j=0}^{n-1} \left(\varepsilon_{k_{\max}-1}^{n-j} + \varepsilon_{k_{\max}}^{n-j}\right) + C\Delta t^{\min\{2+k_{\max},q\}}. \tag{14}$$

Remark 7 Before proving this lemma, we point out a couple of salient features:

- Note that $\varepsilon_{k_{\max}}^n$ has a global (i.e., over *all* n) dependence on $\varepsilon_{k_{\max}-1}^n$, while all the other ε_k^n do not have this kind of dependence on ε_{k-1}^n.
- We have chosen to take the same constant C for both the expression of ε_0^n as well as for all the terms in the inequality for ε_k^n. This can be done without loss of generality, as we take the largest of all these constants.

Proof To prove (13), we consider the analytical expression of Z^n given in (12). Let us first treat the last term in (12), $\sum_{j=0}^{n-1} A^j E^{n-j}$. Due to La. 3, we know that there is a constant C such that $E^{n-j,[k]} \leq C\Delta t^{\min\{2+k,q\}+1}$. Furthermore, due to Cor. 1, there holds

$$\left(\sum_{j=0}^{n-1} A^j E^{n-j}\right)^{[k]} \leq NC\Delta t^{\min\{2+k,q\}+1} \leq T_{end} C\Delta t^{\min\{2+k,q\}}.$$

By $(\cdot)^{[k]}$, we denote the $k-$th block of a vector, again, $0 \leq k \leq k_{\max}$. We have used the fact that $\Delta t = \frac{T_{end}}{N}$ and that A^j "pushes elements downwards," so there will be no Δt contributions of lower orders than the ones given here. This explains why we carefully chose the red terms in Alg. 1 the way we did.

In a similar way, we treat the other terms. They can all be handled alike, so we only consider the term

$$\begin{aligned}&\Delta t \sum_{j=0}^{n-1} A^j B_I \left(\Phi_{\mathrm{I}}(Y^{n-j}) - \Phi_{\mathrm{I}}(W^{n-j})\right)\\ &= \Delta t B_I \left(\Phi_{\mathrm{I}}(Y^n) - \Phi_{\mathrm{I}}(W^n)\right) + \Delta t \sum_{j=1}^{n-1} A^j B_I \left(\Phi_{\mathrm{I}}(Y^{n-j}) - \Phi_{\mathrm{I}}(W^{n-j})\right).\end{aligned}$$

Note that the first term couples errors at correction level $k-1$ to those of k. Estimated, this gives, independent of whether $k = k_{\max}$ or not, a contribution of

$$C\Delta t \left(\varepsilon_k^n + \varepsilon_{k-1}^n\right).$$

For $k < k_{\max}$, due to the peculiar structure of A^j, $A^j B_I$ remains a block-matrix; on the $k-$th block-row, only two blocks are filled.[2] The block-columns thereof are j and $j+1$, with $j \geq k$. A generous estimate yields the contribution

$$C\Delta t \sum_{j=1}^{n-1} \sum_{h=k}^{k_{\max}} \varepsilon_h^{n-j}.$$

(In fact, for $j \geq k_{\max} - 1$, the block columns are $k_{\max} - 1$ and $k_{\max}$, so sharper estimates could be used.) For $k = k_{\max}$, the non-zero block-columns are $k_{\max} - 1$ and $k_{\max}$. This then gives the contribution

$$C\Delta t \sum_{j=1}^{n-1} \left(\varepsilon_{k_{\max}-1}^{n-j} + \varepsilon_{k_{\max}}^{n-j}\right).$$

This yields the desired result. □

We do not have a proof for the following statement, which is why we formulate it as a proposition.

[2] Let us clarify what we mean by block-rows and block-columns: The matrices we are dealing with here are of size $s \cdot (k_{\max}+1) \times s \cdot (k_{\max}+1)$. They can hence be subdivided into $(k_{\max}+1)^2$ blocks of size $s \times s$. According to the notation in Alg. 1, we begin counting by $k = 0$. Block-row k means hence in the big matrix the rows from $k(s+1)+1$ to $(k+1)(s+1)$.

Proposition 1 *From the results in La. 5, we conjecture that the method is convergent with the $k-$dependent orders. That is, we postulate that each of the correction steps satisfy*

$$\varepsilon_k^n = \mathcal{O}\left(\Delta t^{\min\{2+k,q\}}\right), \qquad k < k_{\max},$$

and that the final correction step satisfies

$$\varepsilon_{k_{\max}}^n = \mathcal{O}\left(\Delta t^{\min\{1+k_{\max},q\}}\right), \qquad k = k_{\max}.$$

Remark 8 Please note that the last correction step does not increase the order. This is different to the behavior of the algorithm in [42]; it is a consequence of the fact that we have modified the values at time level t^n to make the algorithm ready for parallelism.

Remark 9 Although we only present numerical results that support Proposition 1 – and we only know from Thm. 1 that the method is at-least second order convergent – we can very well see the underlying structure from Eqs. (13)-(14): For $k < k_{\max}$, ε_k^n essentially possesses the contributions $\Delta t \varepsilon_{k-1}^n$, $\Delta t \sum_{j=1}^{n-1} \sum_{h=k}^{k_{\max}} \varepsilon_h^{n-j}$ and $\Delta t^{\min\{2+k,q\}}$. (We have neglected the constants here.) If we assume that for increasing k, the order in Δt of ε_k^n would also increase or at least not decrease, the contribution $\Delta t \sum_{j=1}^{n-1} \sum_{h=k}^{k_{\max}} \varepsilon_h^{n-j}$ scales as $\mathcal{O}(\varepsilon_k^n)$. Furthermore, if the order between ε_k^n and ε_{k-1}^n differs by one, then $\Delta t \varepsilon_{k-1}^n$ scales again as ε_k^n. The final constant term $\Delta t^{\min\{2+k,q\}}$ then gives the overall order.

Remark 10 In [42], we have investigated asymptotic properties of the related fourth-order method. Under standard assumptions on the ODE and the corresponding IMEX splitting, we have obtained asymptotic consistency. The essential ingredient here was the term

$$\Delta t \left(\Phi_{\mathrm{I}}^{n,[k+1],l} - \Phi_{\mathrm{I}}^{n,[k],l}\right) - \frac{\Delta t^2}{2}\left(\dot{\Phi}_{\mathrm{I}}^{n,[k+1],l} - \dot{\Phi}_{\mathrm{I}}^{n,[k],l}\right),$$

which is still present in this current work. Therefore, also the method presented here is asymptotically consistent; this can be shown in an analogous manner to [42, Proof of La. 4]. To keep the focus of this work, we do not go more into detail here.

4 Parallel implementation

The method presented in Alg. 1 has been carefully designed in such a way that it can be implemented in a parallel setting. The ideas we use are inspired by the ones shown in [18] – and in fact, the much older ideas presented in [36] – and are roughly based on the fact that dependencies are chosen in such a way that pipelining becomes possible. We propose a different pipelining than the one presented in [18] to reduce processor idle times. The following observations are key to being able to parallelize this solver:

– The predictor, i.e., the values $w^{n,[0],l}$ that correspond to zeroth iteration number $k = 0$, do *not* depend on the values of any other correction step. They only depend on the final value $w^{n-1,[0],s}$ obtained by the *predictor* in the previous time step.
– For $1 \leq k < k_{\max}$, the $(k)-$th corrector step, i.e., the values $w^{n,[k],l}$, depend on the $(k-1)$-st corrector step at time level n, as well as on the next correction at the previous time step, $w^{n-1,[k+1],s}$.
– The $(k_{\max})$-th corrector step, $w^{n,[k_{\max}],l}$ depends only on the $(k_{\max}-1)$-st corrector step at time level n as well as the final correction at the previous time level, $w^{n-1,[k_{\max}],s}$.

These dependencies are illustrated in more detail on the left hand side of Fig. 1. On the $x-$axis, the time instances $n, n+1, \ldots$, are indicated, while on the $y-$axis, the correction iterates k and the predictor $(k = 0)$ are sketched. The circles at position (n, k) correspond to the computation of $w^{n,[k],l}$ for all $1 \leq l \leq s$. Circles with the same number on it can be computed in parallel, while those with a higher number have to wait for those with a lower number to finish. The arrows indicate direct dependencies, required for the calculation of $w^{n,[k],l}$. Given the dependencies, it also makes sense to always group the correction iterates $2k$ and $2k+1$ on the same process, since one would otherwise create processor idle time. In addition, in order to use computational resources as efficiently as possible, we also do this for the predictor, so the predictor $(k = 0)$ is grouped together with the first correction iterate $k = 1$, which would strictly speaking not be necessary. Red arrows then imply communication over processes. Note that communication is always unidirectional. For comparison, the dependencies of the original serial algorithm from [42] are visualized in the center of Fig. 1. One can clearly see that without the modifications proposed in Alg. 1, the predictor and all correction steps depend on the $(k_{\max})$-th iteration at time level $n-1$.

The High-Order Parallel (HO-Parallel) version is designed to deliver parallel speedup while retaining high-order accuracy. This is the method described in Alg. 1, but here we describe one such grouping of processors that yields a parallel solver. Theoretically, this speedup can be estimated as follows: Denote the number of timesteps again by N. The unparallelized algorithm has to perform $N \cdot (k_{\max}+1)$ operation blocks (the computation of $w^{n,[k],l}$ for given n, k and for all $1 \leq l \leq s$). Each process in the parallelized version has to perform $2N$ operations, and the last process has to wait $k_{\max}-1$ cycles before it can start. This gives, neglecting all other sources of imbalance and overhead such as communication and the like, a maximum theoretical speedup of

$$\frac{N \cdot (k_{\max}+1)}{2N + k_{\max} - 1} \to \frac{k_{\max}+1}{2}, \qquad N \to \infty. \tag{15}$$

Please note that this is the speedup in comparison to letting the algorithm run in its unparallelized version, it is not the theoretical speedup in comparison to the underlying Runge-Kutta method, as is done, e.g., in [23]. This speedup is harder to determine, in particular in the IMEX setting that we follow here, because it depends on the convergence of the correction iterates as well as on the (nonlinear) algebraic solves, which can differ tremendously from the fully implicit Runge-Kutta method and the semi-implicit correction iterates used here. Despite these objections, such an investigation is presented in Sec. 6.2.2 for an exemplary setting.

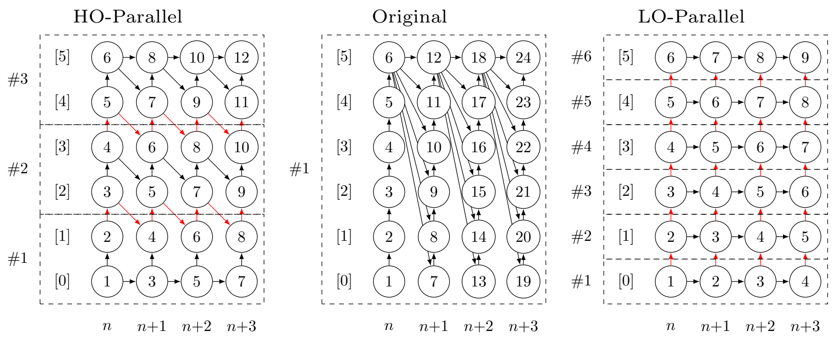

Fig. 1 Detailed view on the dependencies of the parallelization flavors. Here we include an illustrative example with a total of four time steps and $k_{\max} = 5$ total iterations. On the $x-$axis, time instances n, $n+1, \ldots$ are indicated, while on the $y-$axis, the correction iterates k and the predictor $(k = 0)$ are sketched. The circles at position (n, k) correspond to the computation of $w^{n,[k],l}$ for all $1 \leq l \leq s$. Circles with the same number on it can be computed in parallel, while those with a higher number have to wait for those with a lower number to finish. The arrows indicate direct dependencies required for the calculation of $w^{n,[k],l}$, with red arrows indicating that communication between processors is needed. Left: Higher order parallel-in-time MD scheme. Middle: Original serial algorithm proposed in [42]. Right: Low order parallel-in-time MD scheme, enabling the use of more processors.

Grouping the processors has the obvious disadvantage that it reduces the parallelization capabilities, and therefore if attaining parallel speed is of chief concern, we suggest the Low-Order Parallel (LO-Parallel) method described in the right frame of Fig. 1. In short, the basic idea is to dedicate an entire process to each correction step and let each correction run completely independent from the other corrections insofar as they need to wait for the previous correction to finish in a queue like fashion. That is, the **LO-Parallel** version changes the red terms in Alg. 1 from $w^{n-1,[k+2],s}$ to $w^{n-1,[k+1],s}$:

$$\begin{aligned} w^{n,[k+1],1} &:= w^{n-1,[k+1],s}, \\ w^{n,[k+1],l} &:= w^{n-1,[k+1],s} \\ &\quad + \Delta t \left(\Phi_{\mathrm{I}}^{n,[k+1],l} - \Phi_{\mathrm{I}}^{n,[k],l}\right) - \frac{\Delta t^2}{2}\left(\dot{\Phi}_{\mathrm{I}}^{n,[k+1],l} - \dot{\Phi}_{\mathrm{I}}^{n,[k],l}\right) \\ &\quad + \mathcal{I}_l(\Phi^{n,[k],0}, \Phi^{n,[k],1}, \ldots, \Phi^{n,[k],s}). \end{aligned} \tag{16}$$

This means that on correction iterate $k+1$, we only take the previous $(k)-$th correction iterate into account, and not as in the novel algorithm, the $(k+2)$-nd iterate. Provided we lag each of the corrections sufficiently through a startup procedure, then each of the correctors can follow up and clean up the previous iterate at the same cost of the predictor. In comparison to the serial evaluation of Alg. 1, the speedup would be

$$\frac{N \cdot (k_{\max} + 1)}{N + (k_{\max} + 1)} \to k_{\max} + 1, \quad N \to \infty \tag{17}$$

under ideal circumstances.

Remark 11 Unfortunately, this latter modification reduces the overall method to a second-order method, and therefore we do not recommend this LO-Parallel strategy unless parallelization is of utmost importance. The reason for this order reduction is that the predictor – being second-order accurate – computes numerical results completely independently of the higher-order corrections. The higher-order corrections though rely on these globally second-order accurate values only and can hence not improve them. Intermediate synchronizing after $\tilde{N}$ timesteps could make the method high-order in $\varDelta T := \tilde{N}\varDelta t$ at the cost of a reduced parallelism.

5 Numerical results

In this section, we first experimentally validate the theoretical findings of Sec. 3 with sample scalar ODE test cases. Here, we show that the desired orders of accuracy are reached after the predictor and each of the correction steps. We then increase the complexity of our test cases to systems and stiff problems. There we study the influence of choosing one of the algorithms presented in Fig. 1. It is shown that the modifications of the original algorithm from [42] described in Alg. 1 only have a small influence on the obtained solution. Finally, it is shown that the proposed method converges to the limiting Runge-Kutta method given in Def. 1 even for very stiff problems.

5.1 ODE for Convergence Testing

We start by considering a scalar model problem

$$w'(t) = -w^{-\frac{5}{2}}, \qquad w_0 = 1, \tag{18}$$

to validate the order of convergence of the introduced methods. The analytical solution of this ODE is $w(t) = \left(-\frac{7}{2}t + w_0^{7/2}\right)^{2/7}$; the error is evaluated at $T_{end} = 0.25$. For this test problem we introduce an artificial IMEX splitting via

$$\varPhi_{\mathrm{E}}(w) = -\alpha w^{-\frac{5}{2}}, \qquad \varPhi_{\mathrm{I}}(w) = -(1-\alpha)w^{-\frac{5}{2}},$$

for $\alpha = 0.2$.

Fig. 2 shows the convergence of the fourth-, sixth- and eighth-order schemes after all correction steps. The figure shows that the predictor step is second order accurate and the correction steps increase the order of accuracy successively. With $k_{\max} = 3$ fourth order of accuracy is achievable and one can note that the last two correction steps are always of the same order. Increasing $k_{\max}$ shows that with each correction step the scheme picks up one order of accuracy - until the maximum order defined by the quadrature rule is reached. Hence, the maximum achievable order is $\min(k_{\max} + 1, \text{accuracy of quadrature rule})$, see also Prop. 1.

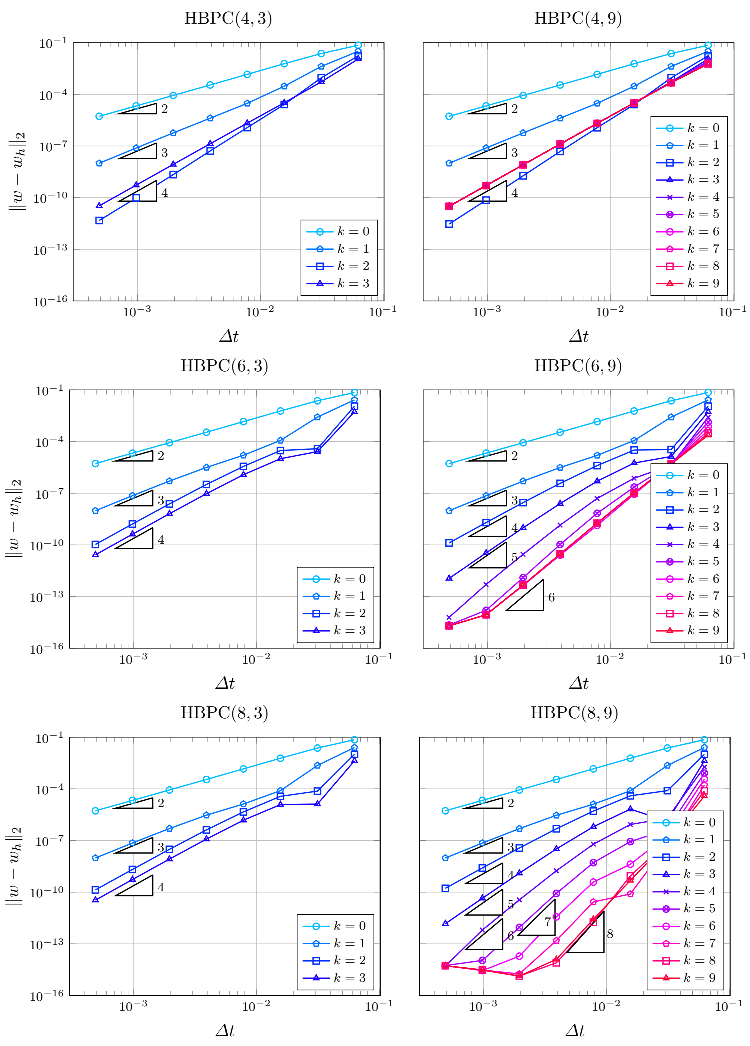

Fig. 2 Error of the iterates of the parallel IMEX-MD schemes for the simple ODE problem $w' = -w^{-\frac{5}{2}}$, see Eq. (18). The error is computed at $T_{end} = 0.25$. Different Runge-Kutta tables and $k_{\max} = 3$ (left) and $k_{\max} = 9$ (right) are used.

5.2 Pareschi-Russo Problem

Next, we use the model problem introduced by Pareschi and Russo (PR) [39] to additionally consider the ability of treating stiff ODEs with the novel method. The system of ODEs is given by

$$w_1'(t) = -w_2, \qquad w_2'(t) = w_1 + \frac{\sin(w_1) - w_2}{\varepsilon}, \qquad w_0 = \left(\frac{\pi}{2}, 1\right), \tag{19}$$

and is computed until $T_{\text{end}} = 5$. To account for the stiff behavior, similar as it is done in [39], we split the problem into a non-stiff, explicitly treated part Φ_{E} and

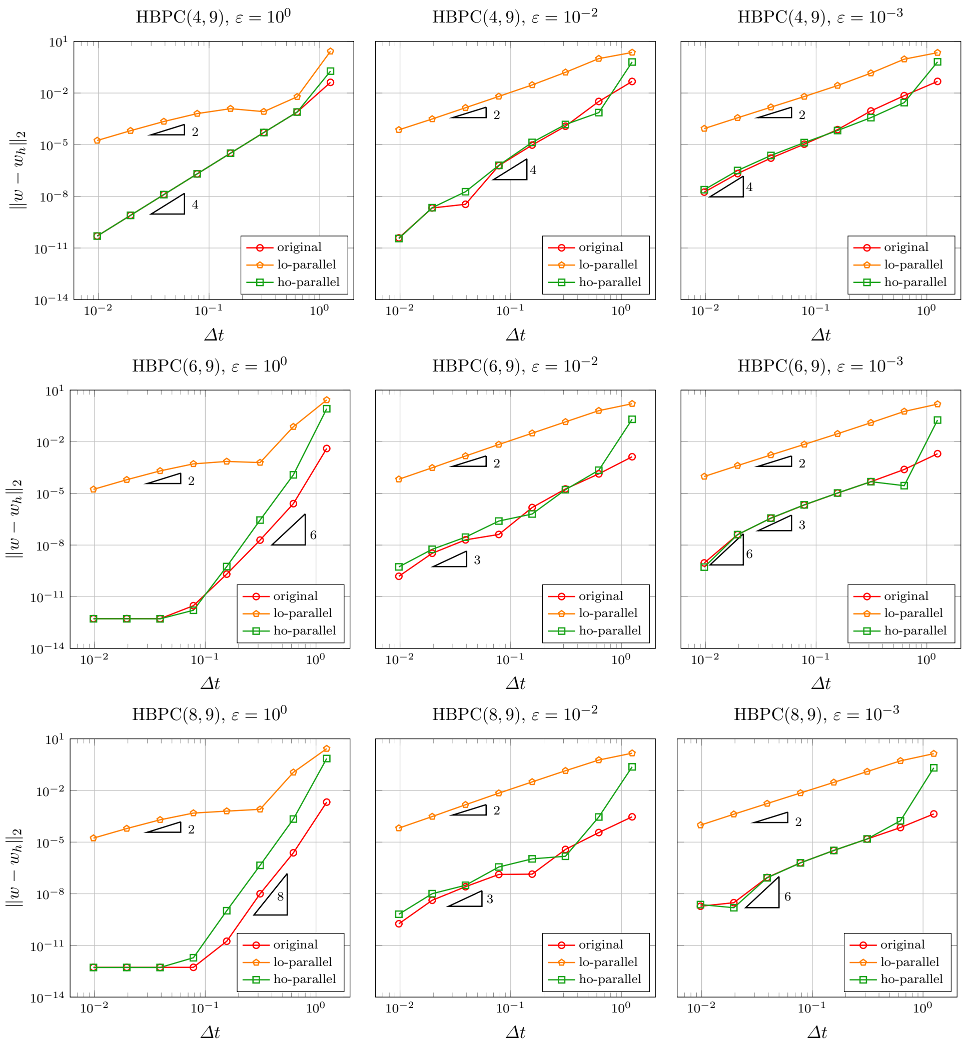

Fig. 3 Error for Pareschi-Russo IMEX problem [39], see Eq. (19), at $T_{end} = 5$ with 4^{th} (top), 6^{th} (middle) and 8^{th} (bottom) order IMEX-MD scheme with $k_{\max} = 9$ with different stiffness parameters $\varepsilon \in \{10^0, 10^{-2}, 10^{-3}\}$.

a stiff, implicitly treated part Φ_{I},

$$\Phi_{\mathrm{E}} = \begin{pmatrix} -w_2 \\ w_1 \end{pmatrix}, \qquad \Phi_{\mathrm{I}} = \frac{1}{\varepsilon}\begin{pmatrix} 0 \\ \sin(w_1) - w_2 \end{pmatrix}.$$

We use this test problem to investigate the modifications of Alg. 1 compared to the original one in [42] and the straight-forward, but low-order parallelization variant. For an overview on the three different algorithms see Fig. 1.

In order to investigate the different behaviors for stiff and non-stiff problems, ε in Eq. (19) is varied from $\varepsilon = 1$ to $\varepsilon = 10^{-3}$. The comparison is done for all three quadrature rules described in Eq. (2)–(4). A fixed $k_{\max} = 9$ is used. In

Fig. 3 the results of the calculations are summarized. One can clearly see that the parallel low-order scheme is always second order accurate, regardless of the equipped quadrature rule and stiffness of the problem (orange lines). Additionally, the error of the parallel-low order scheme is always higher than with the other schemes. Considering the high-order parallel and the original serial scheme, the figure shows that the modifications introduced in Alg. 1 compared to the original algorithm only slightly influence the obtained error and have no effect on the achieved order (compare red and green lines). Nevertheless, some differences between Alg. 1 and the original algorithm are present, mostly showing up for large timesteps and non-stiff problems. Here, the original algorithm gives slightly better results. This is most probably due to the very few timesteps used for large Δt (the coarsest resolution uses only four timesteps). Differently to the original algorithm, the information of the last corrector step has not yet reached the first corrector step in those settings, see Fig. 1 for illustration. This reduces the accuracy of the algorithm if only a few timesteps are performed. Moreover, in the original algorithm a better w^n is used for all steps $k \neq k_{\max}$. Naturally, this has a favorable influence on the solution.

One can see that while for the fourth order scheme no order reduction can be observed for stiff problems, HBPC$(6, k_{\max})$ and HBPC$(8, k_{\max})$ show order reduction for an increasing stiffness. In [42] it has been shown that increasing the number of correction steps can cure this problem. Therefore, we perform a convergence study with respect to the number of correction steps $k_{\max}$ for the stiffest setting $\varepsilon = 10^{-3}$ with HBPC$(6, k_{\max})$ and HBPC$(8, k_{\max})$. Fig. 4 shows how the increase of corrector steps improves the accuracy of the obtained results. It is shown numerically that for an increasing $k_{\max}$, the predictor-corrector scheme described in Alg. 1 converges towards the limit method, i.e. the fully coupled two-derivative Runge-Kutta method, see Def. 1.

Summarizing, the presented simulations of the PR problem show that the modifications of the predictor-corrector scheme from [42] described in Alg. 1 have only a slight influence on the solution quality. Using the straightforward possibility to parallelize the scheme (Fig. 1 right) is clearly inferior to the original method and Alg. 1 even for stiff problems. Moreover, for the high order methods, it gets obvious that for stiff problems it can be beneficial to increase the number of corrector steps $k_{\max}$.

5.3 Van-der-Pol Equation

The van-der-Pol equation (vdP) allows us to study the convergence properties for very stiff problems in more detail. It is given by

$$w_1'(t) = w_2, \qquad w_2'(t) = \frac{(1-w_1^2)w_2 - w_1}{\varepsilon}, \qquad w_0 = \left(2, \frac{-2}{3} + \frac{10}{81}\varepsilon\right), \tag{20}$$

with $T_{\mathrm{end}} = 0.5$. Again, we split the equation into a non-stiff explicitly treated part Φ_{E} and a stiff, implicitly treated part Φ_{I} via

$$\Phi_{\mathrm{E}} = \begin{pmatrix} w_2 \\ 0 \end{pmatrix}, \qquad \Phi_{\mathrm{I}} = \frac{1}{\varepsilon}\begin{pmatrix} 0 \\ (1-w_1^2)w_2 - w_1 \end{pmatrix}.$$

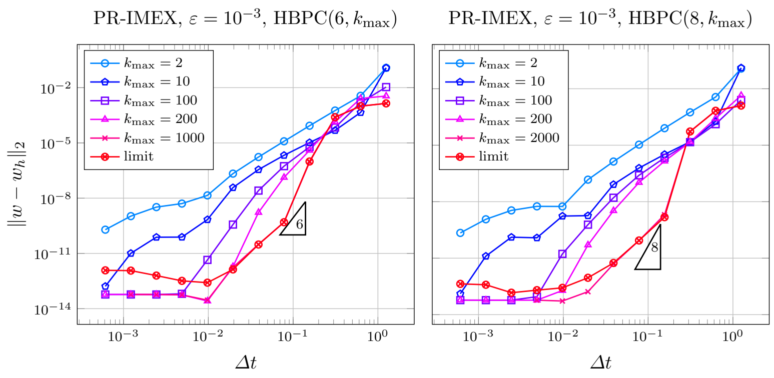

Fig. 4 Influence of $k_{\max}$ iterations on error of PR-IMEX problem with $\varepsilon = 10^{-3}$ for Alg. 1 with 6^{th} order (left) and 8^{th} order (right) quadrature formula. Limit denotes the corresponding sixth- or eigth-order Runge-Kutta method as given in Def. 1.

We use this test problem to study the convergence properties with a typical number of correction steps one would use for practical applications for stiff and non-stiff problems. The calculations are performed on one Intel skylake node with 36 cores. We compare calculations with using all processors on the node ($k_{\max} = 71$), to half the number of the processors on the node ($k_{\max} = 35$). We choose the minimum number of iterations to obtain the desired order for non-stiff problems, ($k_{\max} = 3$, $k_{\max} = 5$ and $k_{\max} = 7$, respectively for the 4^{th}, 6^{th} and 8^{th} order method). Similar to the previously performed evaluations with the PR test problem, we compare the obtained results with $k_{\max} \to \infty$ and the limiting method from Def. 1.

In order to compute the limiting method as $k_{\max} \to \infty$, we compare the numerical errors w.r.t. the exact solution of simulations with $k_{\max}$ and $k_{\max}/2$. If the relative difference between those two errors differs by more than 1%, $k_{\max}$ is increased by a factor of two and the calculation is repeated. That means, repeat the simulation with $2k_{\max}$ if

$$\frac{\left|(\|w - w_h\|_2)_{k_{\max}} - (\|w - w_h\|_2|)_{k_{\max}/2}\right|}{(\|w - w_h\|_2)_{k_{\max}}} > 0.01,$$

else use the solution with corresponding $k_{\max}$ as the "converged" solution. With this investigation we can experimentally show that our method converges towards the limiting Runge-Kutta method, i.e. the algorithm does not diverge.

Remark 12 Note that a similar strategy can be pursued to obtain an adaptation criterion for the IMEX-MD scheme. Since the exact solution is not known for general problems, one could think about evaluating the change of the numerical solution after each iteration. If the iterates do not differ more than a user-defined threshold, no further correction steps are performed. Also timestep adaptation could be taken into account. We leave the detailed analysis of an adaptation of Alg. 1 in this direction to future investigations.

In Fig. 5 the results of the simulation are summarized for the 6^{th}-order method. For an increasing stiffness, order reduction can be observed. This can, at least partially, be cured by increasing the number of correction steps; perfect convergence

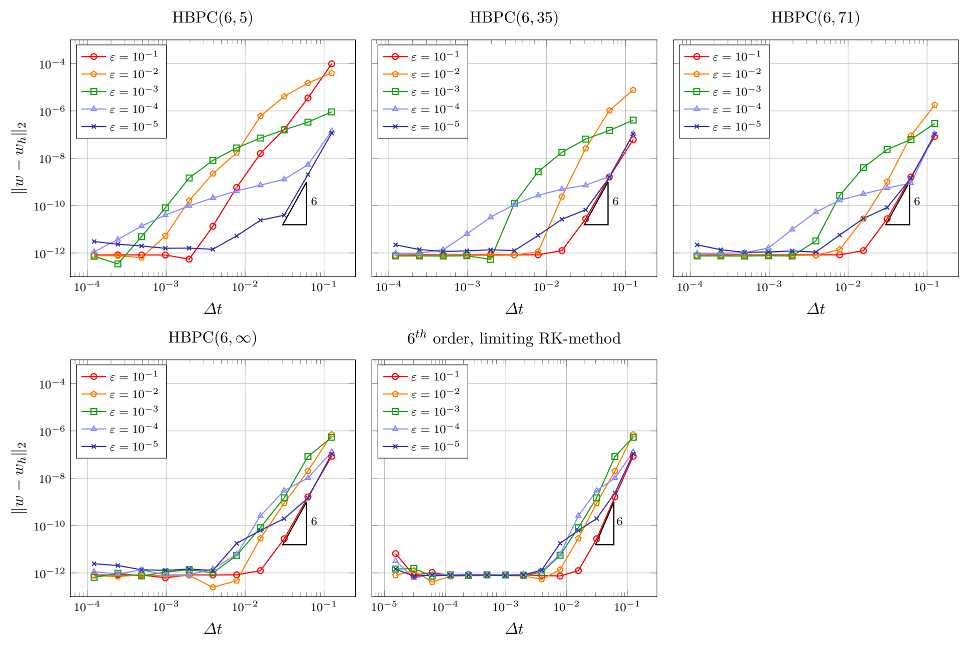

Fig. 5 Error for van-der-Pol equation at $T_{end} = 0.5$ of simulations with HBPC$(6, k_{\max})$. Orders 4 and 8 are omitted for brevity, but show similar behavior. In each image, we increase the level of stiffness of the problem by shrinking ε, and in each frame we consider the impact of increasing the total number of correction steps $k_{\max}$. The last picture (second on the second line) shows the results of solving the fully coupled limiting Runge-Kutta method, cf. Def. 1. The first image on the second line shows the numerically determined limit if $k_{\max} \to \infty$. This is in excellent agreement with the underlying Runge-Kutta method in the right frame, as we expect to be the case when the iterates converge.

cannot be observed. This is consistent with the limiting method and one can see that with $k_{\max} \to \infty$ the limiting method is reached. The results also extend to the 4^{th}- and 8^{th}-order scheme (not shown here for the sake of brevity): small values of $k_{\max}$ produce order reduction for some regimes. The effect can be mitigated by increasing the number of iterates with $k_{\max} \to \infty$. For the 4^{th}-order limiting-scheme, order reduction is completely ruled out, which is not the case for 6^{th} and 8^{th}-order schemes.

6 Improved algorithm and scaling results

Given the flexibility of the solver, there are several opportunities to enhance the proposed Alg. 1. In this subsection, we present one such variation that improves the accuracy and scalability of the solver by making a total of two modifications:

– **Improved predictor.** The predictor in Alg. 1 is completely independent of the correction steps, as the used initial condition is given by the previously computed predictor results. A minor modification would be to use the more accurate result from the first (or higher) correction from the previous time step. This is advantageous whenever the main computational load lies on the predictor. This could happen, e.g., if the number of Newton iterations for the predictor is higher than for the corrector, which is often observed in practice. Moreover, the predictor could, in principle, be leveraged as the cornerstone for adaptive timestepping, which is crucial for many large-scale applications. Based on our observations, we find we can accomplish this and have a predictor that is overall third-order accurate in time, which reduces the total number of corrections required.

– **Improved quadrature.** The $l-$th stage of the $(k+1)$-st corrector with $l > 1$ uses a quadrature rule $\mathcal{I}_l$ in Alg. 1 that is still fully computed with values from correction step k, although for $\iota < l$, the values $w^{n,[k+1],\iota}$ are readily available. If we consider a Gauß-Seidel type approach, similar to the one found in [20], we change this so that in $\mathcal{I}_l$, we replace $w^{n,[k],\iota}$ by $w^{n,[k+1],\iota}$ for $\iota < l$. Here, we show that this not only leads to a more accurate scheme, but has also lower storage requirements.

We formalize the proposed changes to Alg. 1 as its own algorithm.

Algorithm 2 (HBPC*$(q, k_{\max})$) *To advance the solution to Eq. (1) in time, we compute values $w^{n,[k],l}$. The meaning of the parameters n, k and l is explained in Eq. (6) and thereafter. To account for the initial conditions, define*

$$w^{-1,[k],s} := w_0.$$

First, the values $w^{n,[0],l}$ are filled using a straightforward second-order IMEX-Taylor method with an improved initial condition.

1. ***Predict.*** *Solve the following expression for $w^{n,[0],l}$ and each $2 \leq l \leq s$:*

$$\begin{aligned} w^{n,[0],1} :=& w^{n-1,[1],s}, \\ w^{n,[0],l} :=& w^{n-1,[1],s} \\ &+ c_l \Delta t \left(\Phi_I^{n,[0],l} + \Phi_E^{n-1,[1],s} \right) + \frac{(c_l \Delta t)^2}{2} \left(\dot{\Phi}_E^{n-1,[1],s} - \dot{\Phi}_I^{n,[0],l} \right). \end{aligned} \tag{21}$$

   The modified terms ($w^{n-1,[1],s}, \Phi_E^{n-1,[1],s}$, and $\dot{\Phi}_E^{n-1,[1],s}$, marked in blue) simply use one higher iterate than the one found in Alg. 1. This makes the predictor third-order accurate due to its improved local truncation error.

2. ***Correct.*** *Next, the correction steps take place to fill the values of $w^{n,[k],l}$ for $1 \leq k \leq k_{\max}$. Solve the following for $w^{n,[k+1],l}$, for each $2 \leq l \leq s$ and each $0 \leq k < k_{\max}$:*

$$\begin{aligned} w^{n,[k+1],1} :=& w^{n-1,[k+2],s}, \\ w^{n,[k+1],l} :=& w^{n-1,[k+2],s} \\ &+ \Delta t \left(\Phi_I^{n,[k+1],l} - \Phi_I^{n,[k],l} \right) - \frac{\Delta t^2}{2} \left(\dot{\Phi}_I^{n,[k+1],l} - \dot{\Phi}_I^{n,[k],l} \right) \\ &+ \mathcal{I}_l(\Phi^{n,[k+1],0}, \ldots, \Phi^{n,[k+1],l-1}, \Phi^{n,[k],l}, \ldots, \Phi^{n,[k],s}), \end{aligned} \tag{22}$$

   where $\mathcal{I}_l$ is the q-th order Hermite-Birkhoff quadrature rule given in Eq. (5). The red term(s) ($w^{n-1,[k+2],s}$) are the same modified values we implemented *in Alg. 1. The blue terms $\Phi^{n,[k+1],0}, \ldots, \Phi^{n,[k+1],l-1}$ are evaluated at iterate $k+1$ instead of k. This produces smaller errors and requires less storage as terms involving $\Phi^{n,[k],\iota<l}$ can be overwritten. Again, if $k = k_{\max} - 1$, then the $k+2$ in the red terms are replaced by $k_{\max}$ in order to close the recursion.*

3. ***Update.*** *In order to retain a first-same-as-last property, we update the solution with*

$$w^{n+1} := w^{n,[k_{\max}],s}.$$

The parallelization strategy suggested in Fig. 1 remains largely unaltered due to the suggested grouping. Given that the first thread operates on both the $k = 0$ and the $k = 1$ iterates, the predictor simply draws information from the $k = 1$ iterate instead of the $k = 0$ iterate. The corrector also remains unaltered - in fact there is slightly looser coupling given that the quadrature rule uses information that is being updated.

In the following, we explore the impact of the further modifications to the algorithm numerically. We find that in most cases the modified algorithm behaves better than the standard one. Scaling results are therefore given on the basis of Alg. 2 only.

6.1 Influence of the modifications

We study the influence of the two modifications with previously considered Pareschi-Russo problem (Eq. (19)) with $\varepsilon = 1$. In Fig. 6 the results with Alg. 1 and the modified Alg. 2 are compared side-by-side. Although the modification in the predictor and the corrector are presented simultaneously, we carry out a separated analysis of both changes in order to highlight the individual influence.

We make the following observations concerning the additional modifications:

- **Modification of the predictor.** The modification of the predictor has the drawback that the predictor loses its independence from the correction steps as it now uses the solution of the first corrector step as the solution at t^n. In Fig. 6 we see the influence of modifying the predictor step: the predictor's order of accuracy is increased from second to third order. This results in an additional order of the following correction steps, of course only until the maximum achievable order is reached. For the 6^{th}- and 8^{th}-order method we additionally observe improvements in the error found after the last iteration.
- **Modification of the corrector.** The primary advantage in the modification in the corrector is that less overall storage is required for the solver. Simply put, as the stage values of the previous iteration are replaced with the stage values from the current iteration $k+1$, no additional storage array for the old stage values is required. Moreover, potentially better values are used for the quadrature rule. We call this modification "Gauß-Seidel style" as "better" information is used as soon as it is available [40]. Note that similar ideas have been pursued for an integral deferred correction method in [20].

In Fig. 6 we can also observe the influence of using the Gauß-Seidel style corrector by comparing the iterates to each other. The improved quadrature significantly decreased the error. This effect is especially visible for large timesteps since each

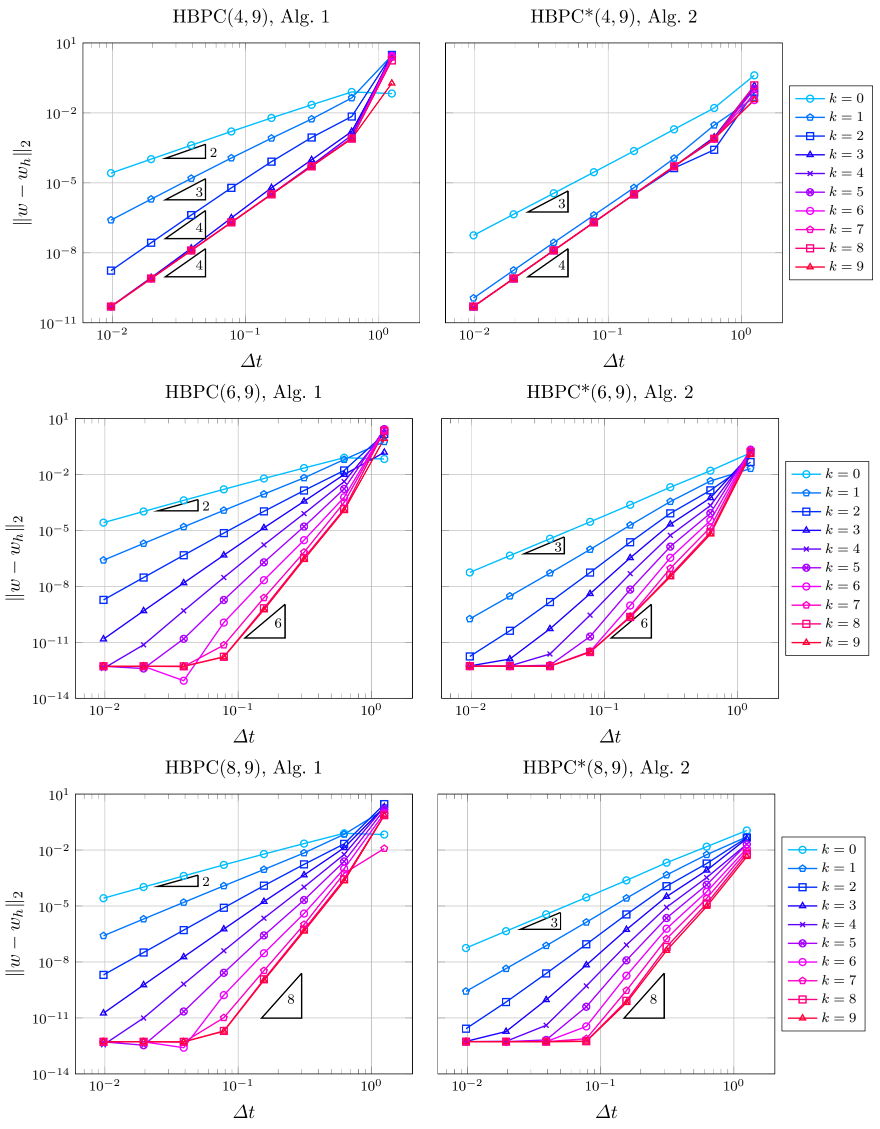

Fig. 6 Error for Pareschi-Russo IMEX problem [39] with $\varepsilon = 1$ at $T_{end} = 5$ with our IMEX-MD schemes. We use a fixed $k_{\max} = 9$ for all the simulations. We display the impact of increasing the iterate number k and the fact that each additional correction improves the overall order of accuracy by one, save the final one. We compare side-by-side the analytically investigated Alg. 1 (left column) and the modified Alg. 2 (right column). The modifications made for Alg. 2 increase the order of the predictor and therefore the following correction steps by one until the maximum order of convergence is reached. Moreover, the errors especially for large timesteps are significantly reduced with Alg. 2.

correction step improves the solution with $\mathcal{O}(\Delta t)$ (of course only until the maximum order of convergence is reached). Although not shown here, this effect is even more pronounced for stiff problems. Moreover, the required amount of Newton iterations is significantly reduced for such cases that require large timesteps. This could certainly be beneficial for multiscale problems. Hence, using values of the predictor that are "one correction step better" as soon as they are available facilitates the solution of the non-linear problem and at the same time allows for more accurate results.

6.2 Parallel performance

We use the improved Alg. 2 to test the parallel performance of the proposed method. In order to do this, we run Alg. 2 in both serial and parallel settings. The method is implemented in `MATLAB` using the parallel computing toolbox [34]. Calculations are done on one Intel skylake node with 2 Xeon Gold 6140 CPUs@2.3 GHz, with 18 cores each, provided by the Vlaams Supercomputing Centrum (VSC).

6.2.1 Parallel performance w.r.t. serial execution of HBPC($q, k_{\max}$)*

We start by considering the serial and parallel execution of the HBPC*($q, k_{\max}$) scheme for both the van-der-Pol and Pareschi-Russo problem. The HBPC*($q, k_{\max}$) scheme necessitates the solution of a non-linear system of equations

$$F(\tilde{w}) := f(\tilde{w}) - \text{rhs} = 0,$$

with

$$\tilde{w} = w^{n,[0],l}, \quad f(\tilde{w}) = w^{n,[0],l} - c_l \Delta t \left(\Phi_{\mathrm{I}}^{n,[0],l}\right) + \frac{(c_l \Delta t)^2}{2} \left(\dot{\Phi}_{\mathrm{I}}^{n,[0],l}\right),$$

$$\text{rhs} = w^{n-1,[1],s} + c_l \Delta t \left(\Phi_{\mathrm{E}}^{n-1,[1],s}\right) + \frac{(c_l \Delta t)^2}{2} \left(\dot{\Phi}_{\mathrm{E}}^{n-1,[1],s}\right),$$

for the predictor and

$$\tilde{w} = w^{n,[k+1],l}, \quad f(\tilde{w}) = w^{n,[k+1],l} - \Delta t \left(\Phi_{\mathrm{I}}^{n,[k+1],l}\right) + \frac{\Delta t^2}{2} \left(\dot{\Phi}_{\mathrm{I}}^{n,[k+1],l}\right),$$

$$\text{rhs} = w^{n-1,[k+2],s} - \Delta t \left(\Phi_{\mathrm{I}}^{n,[k],l}\right) + \frac{\Delta t^2}{2} \left(\dot{\Phi}_{\mathrm{I}}^{n,[k],l}\right) + \mathcal{I}_l,$$

for the corrector. We choose to solve this using a damped Newton's method with starting point $\tilde{w}_0 = w^{n-1,[1],s}$ for the predictor and $\tilde{w}_0 = w^{n-1,[k+2],s}$ for the corrector. We define a relative convergence criterion of

$$\frac{\|F(\tilde{w})\|_2}{\|F(\tilde{w}_0)\|_2} \leq \varepsilon_{\text{Newton}} = 10^{-6}$$

and an absolute convergence criterion of $\|F(\tilde{w})\|_2 \leq \varepsilon'_{\text{Newton}} = 10^{-14}$. A maximum of 1000 iterations is allowed. Starting with a damping factor of 1, the damping factor is halved if one Newton step's residual exceeds 0.9 times the previous Newton step's residual. If none of the convergence criteria is met after 1000 iterations, we mark the solver as converged. Typically, when this happens it is then due to machine accuracy, and not to a lack of convergence for Newton's method.

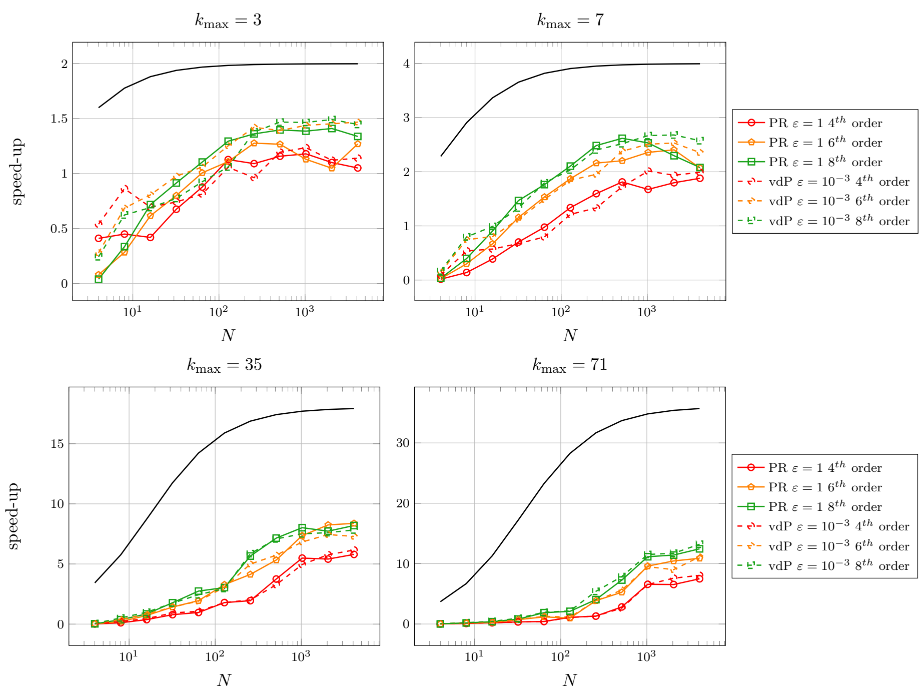

Fig. 7 Speed-up over number of timesteps for different $k_{\max}$ with non-stiff Pareschi-Russo problem (solid lines) and stiff van-der-Pol equation (dashed lines) for HBPC*$(4, k_{\max})$ (red), HBPC*$(6, k_{\max})$ (orange) and HBPC*$(8, k_{\max})$ (green) with the more efficient Alg. 2. Black line indicates theoretically achievable speed-up $\frac{N \cdot (k_{\max}+1)}{2N + k_{\max} - 1}$.

Remark 13 Obviously, choosing a fixed tolerance for Newton's method is not the most efficient way. One can think of an adaptive criterion which differs for predictor and corrector. We leave this investigation for future work.

The hierarchical structure of the method allows us to improve the iterative scheme by replacing the initial $\tilde{w}$ with $w^{n,[k],l}$ for the correction steps. This improves the starting point of Newton's method and can reduce the amount of Newton iterations. We could also think about replacing $\tilde{w}_0$ with $w^{n,[k],l}$. This would have an influence on the results presented below. Nevertheless, we observe that these changes do not significantly alter the reported results. Hence, we can only conclude that the proposed method is quite robust against such algorithmic variants.

The results regarding speed-up are reported in Fig. 7 for an increasing number of correction steps $k_{\max}$ and hence also an increasing number of processors ranging from #procs $= 2$ ($k_{\max} = 3$) to #procs $= 36$ ($k_{\max} = 71$). We choose a non-stiff (Pareschi-Russo with $\varepsilon = 1$) and a stiff (van-der-Pol with $\varepsilon = 10^{-3}$) problem to see if this has an influence on the parallel performance. All calculations are done with the 4^{th}, 6^{th} and 8^{th} order method. The figure shows that the achieved speed-ups are quite similar for both considered problems, but differently for the three different quadrature rules: with HBPC*$(8, k_{\max})$ the highest speed-up can be achieved. This is most probably caused by the more 'processor-local' work due to

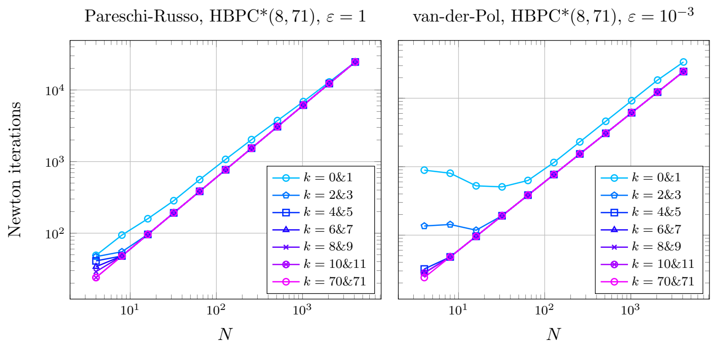

Fig. 8 Required amount of Newton iterations for different correction steps (grouped by the processor on which they are performed) and two selected test setups (left: Pareschi-Russo with $\varepsilon = 1$; right: van-der-Pol with $\varepsilon = 10^{-3}$) over the number of timesteps.

the more stages compared to the other considered schemes. As the communication introduces some overhead, the scaling properties of the algorithm are increased by performing more processor-local operations for an almost similar amount of communication. This is a typical observation made when parallelizing numerical methods. It has, e.g., also been observed for high-order discontinuous Galerkin methods where a better parallel performance can be achieved if higher order ansatz polynomials are chosen as they increase the amount of processor-local work [30]. As the parallel performance results are almost the same for the stiff and non-stiff problem, we anticipate that the obtained performance gain is transferable to other problems.

One can see that if too few timesteps are used, no speed-up is achieved and a deceleration of the simulation is obtained. This can be caused by the overhead introduced by establishing the communication. Nevertheless, if more timesteps are used, a significant speed-up can be achieved with the proposed parallelization strategy. Considering for example the case with 36 processors ($k_{\max} = 71$), one can achieve a speed-up of up to a factor of ≈ 13 for HBPC*$(8, k_{\max})$, ≈ 11 for HBPC*$(6, k_{\max})$ and up to ≈ 8 for HBPC*$(4, k_{\max})$. These values are in the same range as reported in the review paper by Ong and Schroder [37]. In terms of their paper, our method belongs to the class of direct time-parallel methods.

The maximum theoretical speed-up is not achieved for all cases. There can be two reasons for this: First, the overhead introduced by the communication slows down the computations. Second, the processor-local work is distributed unevenly. The first point is strongly influenced by the current implementation and architecture and is therefore beyond the scope of this work. To gain more insight into the amount of processor-local work we consider the amount of Newton iterations performed by each processor.

In Fig. 8 the Newton iterations on the first 6 and the last processor are shown for the HBPC*$(8, k_{\max})$ scheme for the non-stiff Pareschi-Russo problem and the stiff van-der-Pol equation. It is shown that the first processor which is responsible for the predictor and the first corrector step requires significantly more iterations than the other processors. This trend is even enhanced for the considered stiff problem. To cure this imbalance one can either think about a different distribution of the processors than presented in Fig. 1, or a termination criterion for Newton's method depending on the current index of the correction step.

6.2.2 Parallel performance w.r.t. limiting method

The novel HBPC*$(q, k_{\max})$ can be seen as a relaxation method towards the Hermite-Birkhoff Runge-Kutta method, see Def. 1. Hence, another interesting investigation is the evaluation of the efficiency of the parallel HBPC*$(q, k_{\max})$ algorithm in comparison to the limiting Hermite-Birkhoff Runge-Kutta method.

The establishment of a fair comparison is not as straightforward as in the previous case for several reasons. First, the limiting method is a fully implicit algorithm whereas the HBPC*$(q, k_{\max})$ is an IMEX scheme which achieves different errors for different $k_{\max}$. If one chooses a fully implicit test case for the comparison, one could improve the predictor of the HBPC*$(q, k_{\max})$ method by the use of a fourth order scheme with almost no additional costs as it has been done in [49]. Moreover, as one solves another non-linear system, the tolerances of Newton's method do not correspond to each other. Lastly, solving PDEs, it might even be impossible to set up the large matrix for the limiting method due to, amongst others, limitations on storage, efficiency or conditioning, see, e.g., the recent publications [45,46]. Therefore, one would choose a different method with a smaller linear system such as the HBPC method.

To give insight into the performance of the schemes in a practical application, we evaluate the required computational time w.r.t. the achieved accuracy. As the problem size has a large impact on the performance of both methods, we use the non-linear heat conduction

$$w_t = (\kappa(w) \cdot w_x)_x, \quad \text{with} \quad \kappa(w) := 1 + w^2, \quad \text{and} \quad w_0 = 5\sin(x),$$

for this investigation. A fourth order central finite difference is used for the discretization of the spatial derivatives on the domain $\Omega = [0, 2\pi]$. With this setting, the influence of the system size can be easily investigated by varying the number of spatial discretization points $X \in \{50, 100, 200\}$. The temporal discretization is then done fully implicitly with the serial HBPC*$(8, k_{\max})$, the parallel HBPC*$(8, k_{\max})$ and the limiting HBRK8 scheme. We choose $T_{\text{end}} = 5$, set the number of timesteps to $N \in \{250, 500, 1000\}$ and use the same relative and absolute Newton tolerances for all calculations ($\varepsilon_{\text{Newton}} = 10^{-10}$, $\varepsilon'_{\text{Newton}} = 10^{-12}$). The reference solution is obtained with `MATLAB`'s `ode15s` solver with very fine tolerances.

In Fig. 9, we report the required wallclocktimes and resulting l_2-errors for the different simulations. Note that we have chosen $k_{\max} \in \{1, 7, 15, 35, 71\}$ for the HBPC* methods. One can see that for most cases, the serial HBPC* method requires more computational time than the limiting method, which is not really surprising as one has to solve significantly more systems of equations in serial. One can clearly see that using the parallel-in-time algorithm reduces the required wallclocktime such that it can outperform the limiting Hermite-Birkhoff Runge-Kutta method in terms of efficiency. That means that the parallel HBPC* method requires less wallclocktime than the limiting HBRK scheme to achieve a comparable error. This behavior is enhanced as the system size increases.

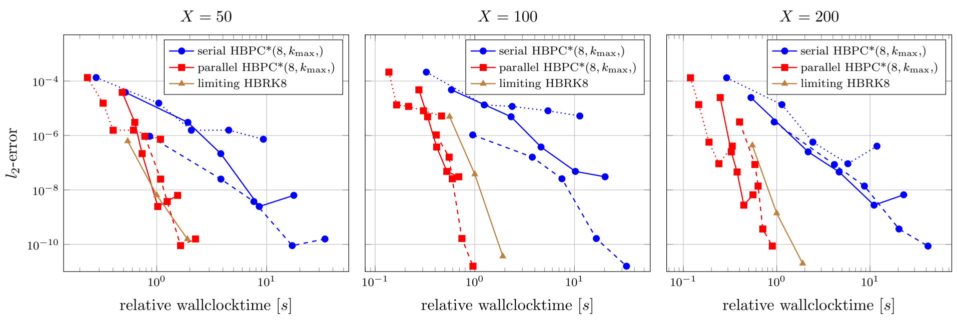

Fig. 9 Required relative wallclocktime and resulting errors for serial HBPC*$(8, k_{\max},)$ and parallel HBPC*$(8, k_{\max},)$ each with $k_{\max} \in \{1, 7, 15, 35, 71\}$ for the non-linear heat conduction problem. Dotted lines correspond to $N = 250$, solid lines to $N = 500$ and dashed lines correspond to $N = 1000$, where N denotes the number of timesteps. The brown dot shows the result obtained with the limiting HBRK8 scheme with $N \in \{250, 500, 1000\}$. Wallclocktimes are normalized such that for each plot, the relative wallclocktime of HBRK8 with $N = 500$ equals one. The system size is increased with $X \in \{50, 100, 200\}$ from left to right, where X denotes the number of spatial discretization points.

6.3 Arenstorf Orbit

Finally, we use the Arenstorf orbit problem [28,38] for further illustration of the proposed method's capabilities. The Arenstorf orbit problem describes a three body problem, where the movement of a light object is influenced by two heavy objects. This can, for example, be a satellite being influenced by two planets. The problem is actually a second-order differential equation, formulated as a system of first-order ODEs:

$$w' = \begin{pmatrix} w_3 \\ w_4 \\ w_1 + 2w_4 - \mu' \frac{w_1+\mu}{D_1} - \mu \frac{w_1-\mu'}{D_2} \\ w_2 - 2w_3 - \mu' \frac{w_2}{D_1} - \mu \frac{w_2}{D_2} \end{pmatrix} .$$

Here, we have defined

$$D_1 := \left((w_1+\mu)^2 + w_2^2\right)^{\frac{3}{2}}, \quad D_2 := \left((w_1-\mu')^2 + w_2^2\right)^{\frac{3}{2}},$$
$$\mu := 0.012277471, \quad \mu' := 1-\mu,$$

and the initial conditions

$$w_0 = (0.994,\ 0,\ 0,\ -2.001585106379)^T .$$

Although there is no multi-scale character of the problem, we artificially split the equation into an explicit and an implicit part. For that purpose, all parts of the equation which are divided by D_1 or D_2 are simply treated implicitly, the remaining parts are treated explicitly.

The solution is a closed orbit with a period of 17.065216560159. Similar to [38], we choose 10^5 equidistant timesteps to simulate one period of the problem. We choose the HBPC*$(8, k_{\max})$ method with Alg. 2 for our calculations, which results in an error after one period of $\|w - w_0\|_2 = 1.7818 \cdot 10^{-9}$ for $k_{\max} = 71$.

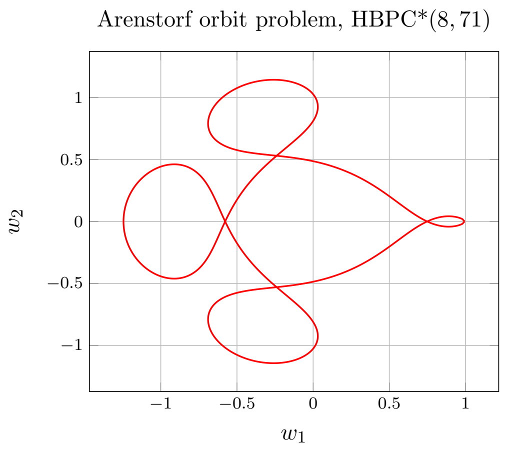

Fig. 10 Solution of the Arenstorf orbit problem for the two spatial variables. 10^5 timesteps of the 8th order IMEX-MD scheme described in Alg. 2 are used to simulate one period.

In Fig. 10 the solution of w_1 and w_2 for the Arenstorf orbit problem is shown. One can see that after one period, the initial condition is reached with good accuracy and a periodic orbit is obtained. With $k_{\max} = 71$, and hence choosing 36 processors, the speed-up is ≈ 14.

To obtain a comparison in a "real-world" setting we consider the scenario of having a serial code and one performs 7 correction steps, what is a reasonable $k_{\max}$ for the 8th order method. Having now the possibility to use a parallel scheme on one compute node with 36 processors, one might use $k_{\max} = 71$. Comparing those two computing times, one still obtains a speed-up of ≈ 1.5 and at the same time a potentially better solution.

As a very last example, we consider the Arenstorf orbit problem with relatively few timesteps, $N = 5{,}000$. (Note that in this work, we only use timesteps that are uniformly spaced. The Arenstorf orbit is an example of a difficult problem that asks for adaptive timestepping, which is beyond the scope of this work.) We use the 8th order scheme and $k_{\max} = 7$. In Fig. 11 we plot the orbits for the predictor and the last correction step, once for Alg. 1 (left) and once for Alg. 2 (right). All other parameters are exactly the same. It is clear that at least the predictor of Alg. 1 is rather useless here, and also the last correction value is not a closed orbit. On the other hand, the modifications done in Alg. 2 lead to a better solution, where both predictor and the last correction step are, in the 'eyeball-norm', closed orbits. This once again shows the superiority of Alg. 2 over its more straightforward counterpart.

7 Conclusions and outlook

In this work, we have developed a novel class of IMEX solvers for the numerical treatment of ODEs that operate on multiple derivatives of the ODE's flux function. The algorithm has been specifically designed to be

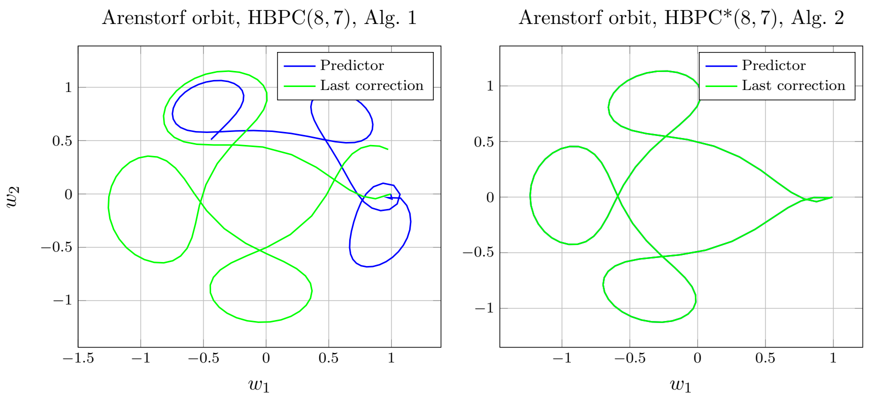

Fig. 11 Solution of the Arenstorf orbit problem for the two spatial variables. A total of $N = 5{,}000$ timesteps of the 8th order IMEX-MD scheme described in Alg. 1 (left) and in Alg. 2 (right) are used to simulate until final time $T = 17.065216560159$, which should correspond to one period. The improved algorithm produces something that is close to what is expected to be a closed orbit, whereas the first algorithm is far from the exact solution, with the predictor being wildly off.

– of high-order (we showed results until order eight, higher orders are easily achievable by increasing the number of collocation points or the number of derivatives used),
– and parallelizable in time.

There are two flavors of the method, one (Alg. 1) is a softer modification to our previous result [42] that is more amenable to analysis, and the second one (Alg. 2) is more storage efficient and has better scaling results. We have shown numerical results demonstrating the behavior of the algorithms. In the light of [37], the scaling results seem to be very much in line with other state-of-the-art methods.

There are multiple important open areas that need to be addressed with future work. The most accessible problem to consider would be to increase the total number of derivatives used, or work on different integration points c. Of notable interest would be to explore collocation solvers described in Ex. 1 constructed from the so-called Gauß-Turan-type [47] quadrature rules. These solvers would ideally offer low-storage alternatives to the present solvers. Next, it would be interesting to explore versions of this solver that make use of arbitrary multiderivative Runge-Kutta methods, not of the collocation variety. In addition, it is imperative that we further explore implementations for PDEs. To date, multiderivative methods have largely been sidelined, even though they can outperform lauded and highly optimized `MATLAB` routines such as the builtin `ode15s` (cf. [2] for one such case study). One reason for this lack of broader interest is arguably the difficult computation of the second (and third, ...) derivatives in practical applications. For some ODEs stemming from a semi-discrete PDE, this can be done rather elegantly [31], but for many, it is a tedious task, which requires leveraging Lax-Wendroff type time discretizations. We therefore suggest exploring the possibilities of using finite-difference approximations, as done in [7,15,51] in the context of explicit Taylor methods. It is not yet clear how this interacts with the stability properties of the overall method, in particular for highly stiff problems, as these type of discretizations require fully discrete stability analyses, but is certainly worth looking into. Furthermore, the extension and testing of this time-stepping option to the classical IMEX application areas remains to be explored.

Declarations

Funding This study was initiated during a research stay of D.C. Seal at the University of Hasselt, which was supported by the Special Research Fund (BOF) of Hasselt University. Additional funding came from the Office of Naval Research, grant numbers N0001419WX01523, N0001420WX00219, and N0001421WX01360. J. Zeifang was funded by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) through the project GRK 2160/1 "Droplet Interaction Technologies" and through the project no. 457811052. The HPC-resources and services used in this work were provided by the VSC (Flemish Supercomputer Center), funded by the Research Foundation - Flanders (FWO) and the Flemish Government.

Conflicts of interest The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Availability of data and material The datasets generated and/or analyzed during the current study are available from the corresponding author on reasonable request (jochen.schuetz@uhasselt.be).

Code availability The code used to generate the results in this work is available upon reasonable request from the corresponding author (jochen.schuetz@uhasselt.be).

A Matrices of "one-step equivalent"

In the following, the matrices required for the formulation of the "one-step equivalent" of Alg. 1 in Eq. (9) are given. We start by defining the matrices $\mathfrak{I}$, $\mathfrak{C}^{(1)}$, $\mathfrak{C}^{(2)} \in \mathbb{R}^{s\times s}$ with

$$\mathfrak{I} := \begin{pmatrix} 0 & 0 & 0 & \cdots & 0 \\ 0 & 1 & 0 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & 1 \end{pmatrix}, \quad \mathfrak{C}^{(1)} := \begin{pmatrix} 0 & 0 & \cdots & 0 \\ c_2 & 0 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ c_l & 0 & \cdots & 0 \end{pmatrix}, \quad \text{and } \mathfrak{C}^{(2)} := \begin{pmatrix} 0 & 0 & \cdots & 0 \\ c_2^2 & 0 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ c_l^2 & 0 & \cdots & 0 \end{pmatrix}.$$

With this, the matrices B_I and $\tilde{B}_I$ for the stiff contribution are given by

$$B_I = \begin{pmatrix} \operatorname{diag}(c) & 0 & 0 & \cdots & \cdots & 0 \\ B^{(1)} - \mathfrak{I} & \mathfrak{I} & 0 & \cdots & \cdots & 0 \\ 0 & B^{(1)} - \mathfrak{I} & \mathfrak{I} & \cdots & \cdots & 0 \\ \vdots & \ddots & \ddots & \ddots & \ddots & \vdots \\ 0 & \cdots & \cdots & \cdots & B^{(1)} - \mathfrak{I} & \mathfrak{I} \end{pmatrix},$$

$$\tilde{B}_I = \begin{pmatrix} -\operatorname{diag}(c^2) & 0 & 0 & \cdots & \cdots & 0 \\ 2B^{(2)} + \mathfrak{I} & -\mathfrak{I} & 0 & \cdots & \cdots & 0 \\ 0 & 2B^{(2)} + \mathfrak{I} & -\mathfrak{I} & \cdots & \cdots & 0 \\ \vdots & \ddots & \ddots & \ddots & \ddots & \vdots \\ 0 & \cdots & \cdots & \cdots & 2B^{(2)} + \mathfrak{I} & -\mathfrak{I} \end{pmatrix}.$$

Note that c, $B^{(1)}$ and $B^{(2)}$ are defined by the chosen Hermite-Birkhoff quadrature rule, see Eqs. (2)-(4). The matrices B_E and $\tilde{B}_E$ of the non-stiff contribution can be found as

$$B_E = \begin{pmatrix} \mathfrak{C}^{(1)} & 0 & 0 & \cdots & \cdots & 0 \\ B^{(1)} & 0 & 0 & \cdots & \cdots & 0 \\ 0 & B^{(1)} & 0 & \cdots & \cdots & 0 \\ \vdots & \ddots & \ddots & \ddots & \ddots & \vdots \\ 0 & \cdots & \cdots & \cdots & B^{(1)} & 0 \end{pmatrix}, \quad \tilde{B}_E = \begin{pmatrix} \mathfrak{C}^{(2)} & 0 & 0 & \cdots & \cdots & 0 \\ 2B^{(2)} & 0 & 0 & \cdots & \cdots & 0 \\ 0 & 2B^{(2)} & 0 & \cdots & \cdots & 0 \\ \vdots & \ddots & \ddots & \ddots & \ddots & \vdots \\ 0 & \cdots & \cdots & \cdots & 2B^{(2)} & 0 \end{pmatrix}.$$

References

1. Parallel-in-Time. URL https://parallel-in-time.org
2. Abdi, A., Conte, D.: Implementation of second derivative general linear methods. Calcolo **57**(3), Paper No. 20, 29 (2020)
3. Abdi, A., Hojjati, G., Sharifi, M.: Implicit-explicit second derivative diagonally implicit multistage integration methods. Computational & Applied Mathematics **39**(3), Paper No. 228, 15 (2020)
4. Aiguobasimwin, I.B., Okuonghae, R.I.: A class of two-derivative two-step Runge-Kutta methods for non-stiff ODEs. Journal of Applied Mathematics **Art. ID 2459809, 9** (2019)
5. Ascher, U.M., Ruuth, S., Spiteri, R.: Implicit-explicit Runge-Kutta methods for time-dependent partial differential equations. Applied Numerical Mathematics **25**, 151–167 (1997)
6. Ascher, U.M., Ruuth, S., Wetton, B.: Implicit-Explicit methods for time-dependent partial differential equations. SIAM Journal on Numerical Analysis **32**, 797–823 (1995)
7. Baeza, A., Boscarino, S., Mulet, P., Russo, G., Zorío, D.: Reprint of: Approximate Taylor methods for ODEs. Computers & Fluids **169**, 87 – 97 (2018). Recent progress in nonlinear numerical methods for time-dependent flow & transport problems
8. Bispen, G., Arun, K.R., Lukáčová-Medvid'ová, M., Noelle, S.: IMEX large time step finite volume methods for low Froude number shallow water flows. Communications in Computational Physics **16**, 307–347 (2014)
9. Boscarino, S.: On an accurate third order implicit-explicit Runge-Kutta method for stiff problems. Applied Numerical Mathematics **59**, 1515–1528 (2009)
10. Boscarino, S., Pareschi, L.: On the asymptotic properties of IMEX Runge–Kutta schemes for hyperbolic balance laws. Journal of Computational and Applied Mathematics **316**, 60 – 73 (2017)
11. Boscarino, S., Qiu, J.M., Russo, G., Xiong, T.: A high order semi-implicit IMEX WENO scheme for the all-Mach isentropic Euler system. Journal of Computational Physics **392**, 594–618 (2019)
12. Butcher, J.C.: On the convergence of numerical solutions to ordinary differential equations. Mathematics of Computation **20**, 1–10 (1966)
13. Causley, M.F., Seal, D.C.: On the convergence of spectral deferred correction methods. Communications in Applied Mathematics and Computational Science **14**(1), 33–64 (2019)
14. Chan, R., Tsai, A.: On explicit two-derivative Runge-Kutta methods. Numerical Algorithms **53**, 171–194 (2010)
15. Chouchoulis, J., Schütz, J., Zeifang, J.: Jacobian-free explicit multiderivative Runge-Kutta methods for hyperbolic conservation laws. arXiv preprint arXiv:2107.06633 (2021)
16. Christlieb, A., Ong, B.: Implicit parallel time integrators. Journal of Scientific Computing **49**(2), 167–179 (2011)
17. Christlieb, A.J., Gottlieb, S., Grant, Z.J., Seal, D.C.: Explicit strong stability preserving multistage two-derivative time-stepping schemes. Journal of Scientific Computing **68**, 914–942 (2016)
18. Christlieb, A.J., Macdonald, C.B., Ong, B.W.: Parallel high-order integrators. SIAM Journal on Scientific Computing **32**(2), 818–835 (2010)
19. Christlieb, A.J., Macdonald, C.B., Ong, B.W., Spiteri, R.J.: Revisionist integral deferred correction with adaptive step-size control. Communications in Applied Mathematics and Computational Science **10**(1), 1–25 (2015)

20. Crockatt, M.M., Christlieb, A.J.: Low-storage integral deferred correction methods for scientific computing. SIAM Journal on Scientific Computing **40**(5), A2883–A2904 (2018)
21. Dittmann, A.J.: High-order multiderivative IMEX schemes. Applied Numerical Mathematics **160**, 205 – 216 (2021)
22. Dutt, A., Greengard, L., Rokhlin, V.: Spectral deferred correction methods for ordinary differential equations. BIT. Numerical Mathematics **40**(2), 241–266 (2000)
23. Emmett, M., Minion, M.L.: Toward an efficient parallel in time method for partial differential equations. Communications in Applied Mathematics and Computational Science **7**(1), 105–132 (2012)
24. Filbet, F., Jin, S.: A class of asymptotic-preserving schemes for kinetic equations and related problems with stiff sources. Journal of Computational Physics **229**(20), 7625–7648 (2010)
25. Gander, M.J.: 50 years of time parallel time integration. In: Multiple shooting and time domain decomposition methods, *Contributions in Mathematical and Computational Sciences*, vol. 9, pp. 69–113. Springer, Cham (2015)
26. Giraldo, F., Restelli, M., Läuter, M.: Semi-implicit formulations of the Navier-Stokes equations: Application to nonhydrostatic atmospheric modeling. SIAM Journal on Scientific Computing **32**(6), 3394–3425 (2010)
27. Haack, J., Jin, S., Liu, J.G.: An all-speed asymptotic-preserving method for the isentropic Euler and Navier-Stokes equations. Communications in Computational Physics **12**, 955–980 (2012)
28. Hairer, E., Norsett, S.P., Wanner, G.: Solving ordinary differential equations I. Springer Series in Computational Mathematics (1987)
29. Hairer, E., Wanner, G.: Solving ordinary differential equations II. Springer Series in Computational Mathematics (1991)
30. Hindenlang, F., Gassner, G., Altmann, C., Beck, A., Staudenmaier, M., Munz, C.D.: Explicit discontinuous Galerkin methods for unsteady problems. Computers & Fluids **61**, 86–93 (2012)
31. Jaust, A., Schütz, J., Seal, D.C.: Implicit multistage two-derivative discontinuous Galerkin schemes for viscous conservation laws. Journal of Scientific Computing **69**, 866–891 (2016)
32. Kastlunger, K., Wanner, G.: On Turan type implicit Runge-Kutta methods. Computing **9**, 317–325 (1972)
33. Kennedy, C.A., Carpenter, M.H.: Additive Runge-Kutta schemes for convection-diffusion-reaction equations. Applied Numerical Mathematics **44**, 139–181 (2003)
34. MathWorks: Parallel Computing Toolbox: Users Guide. URL `https://nl.mathworks.com/help/parallel-computing/`
35. Minion, M.: Semi-implicit spectral deferred correction methods for ordinary differential equations. Communications in Mathematical Sciences **1**(3), 471–500 (2003)
36. Miranker, W.L., Liniger, W.: Parallel methods for the numerical integration of ordinary differential equations. Mathematics of Computation **21**, 303–320 (1967)
37. Ong, B.W., Schroder, J.B.: Applications of time parallelization. Computing and Visualization in Science **23**(1-4), 11 (2020)
38. Ong, B.W., Spiteri, R.J.: Deferred correction methods for ordinary differential equations. Journal of Scientific Computing **83**(3), Paper No. 60, 29 (2020)
39. Pareschi, L., Russo, G.: Implicit-explicit Runge-Kutta schemes for stiff systems of differential equations. Recent Trends in Numerical Analysis **3**, 269–289 (2000)
40. Saad, Y.: Iterative Methods for Sparse Linear Systems, 2nd edn. Society for Industrial and Applied Mathematics Philadelphia (2003)
41. Schütz, J., Kaiser, K.: A new stable splitting for singularly perturbed ODEs. Applied Numerical Mathematics **107**, 18–33 (2016)
42. Schütz, J., Seal, D.: An asymptotic preserving semi-implicit multiderivative solver. Applied Numerical Mathematics **160**, 84–101 (2021)
43. Schütz, J., Seal, D., Jaust, A.: Implicit multiderivative collocation solvers for linear partial differential equations with discontinuous Galerkin spatial discretizations. Journal of Scientific Computing **73**, 1145–1163 (2017)
44. Seal, D., Güçlü, Y., Christlieb, A.: High-order multiderivative time integrators for hyperbolic conservation laws. Journal of Scientific Computing **60**, 101–140 (2014)
45. Southworth, B.S., Krzysik, O., Pazner, W.: Fast parallel solution of fully implicit Runge-Kutta and discontinuous Galerkin in time for numerical PDEs, part II: nonlinearities and DAEs. arXiv preprint arXiv:2101.01776 (2021)

46. Southworth, B.S., Krzysik, O., Pazner, W., Sterck, H.D.: Fast solution of fully implicit Runge-Kutta and discontinuous Galerkin in time for numerical PDEs, part I: the linear setting. arXiv preprint arXiv:2101.00512 (2021)
47. Stroud, A.H., Stancu, D.D.: Quadrature formulas with multiple Gaussian nodes. SIAM Journal on Numerical Analysis **2**, 129–143 (1965)
48. Ökten Turacı, M., Öziş, T.: Derivation of three-derivative Runge-Kutta methods. Numerical Algorithms **74**(1), 247–265 (2017)
49. Zeifang, J., Schütz, J.: Two-derivative deferred correction time discretization for the discontinuous Galerkin method. arXiv preprint arXiv:2109.04804 (2021)
50. Zeifang, J., Schütz, J., Kaiser, K., Beck, A., Lukáčová-Medvid'ová, M., Noelle, S.: A novel full-Euler low Mach number IMEX splitting. Communications in Computational Physics **27**, 292–320 (2020)
51. Zorío, D., Baeza, A., Mulet, P.: An approximate Lax–Wendroff-type procedure for high order accurate schemes for hyperbolic conservation laws. Journal of Scientific Computing **71**, 246–273 (2017)

UHasselt Computational Mathematics Preprint Series

www.uhasselt.be/cmat

All rights reserved.