Time parallelism and Newton-adaptivity of the two-derivative deferred correction discontinuous Galerkin method

DOI: 10.1016/j.amc.2023.128198
Handle: http://hdl.handle.net/1942/40527

Available online at www.sciencedirect.com
ScienceDirect
Applied Mathematics and Computation 00 (2023) 1–31
Appl. Math. Comput.

Time Parallelism and Newton-Adaptivity of the Two-Derivative Deferred Correction Discontinuous Galerkin Method

Jonas Zeifang[a], Arjun Thenery Manikantan[a], Jochen Schütz[a,*]

[a] *Faculty of Sciences & Data Science Institute, Hasselt University, Agoralaan Gebouw D, BE-3590 Diepenbeek, Belgium*

Abstract

In this work, we consider a high-order discretization of compressible viscous flows allowing parallelization both in space and time. The discontinuous Galerkin spectral element method, which is well-suited for massively parallel simulations, is used for spatial discretization. The main novelty in this work is the additional demonstration of time-parallel capabilities within an implicit two-derivative timestepping procedure to further increase the parallel speedup. Temporal parallelism is made possible by a predictor-corrector-type time discretization that allows to split the associated workload onto multiple processors. We identify a homogeneous load balance with respect to the linear (GMRES) iterations on each processor as a key for parallel efficiency. To homogenize the load and to enable practical simulations, an adaptive strategy for Newton's method is introduced. It is shown that the time-parallel method provides a parallel efficiency of approx. 60-70% on 4-7 computational partitions. Moreover, the capabilities of the novel method for the simulation of large-scale problems are illustrated with a mixed temporal and spatial parallelization on more than 1000 processors.

Keywords: Implicit time stepping, Parallel-in-Time, Multiderivative schemes, Newton adaptivity

1. Introduction

In this work, we are interested in solving the compressible Navier-Stokes equations, which can be cast into flux formulation

$$\mathbf{w}_t + \nabla_x \cdot (\mathbf{F}(\mathbf{w}) - \mathbf{F}^v(\mathbf{w}, \nabla_x \mathbf{w})) = 0, \quad \text{with} \quad \mathbf{w} = \begin{pmatrix} \rho \\ \rho \mathbf{v} \\ E \end{pmatrix}, \tag{1}$$

for the unknown quantities density ρ, velocity $\mathbf{v}$ and energy E. Note that we have closed the system by defining the pressure p via the ideal gas equation of state with the isentropic coefficient $\gamma = 1.4$ and reference Mach number ε. For a precise definition of the fluxes, consult Appendix A. All occuring quantities are non-dimensionalized.

In this work, we are interested in a parallel algorithm for the temporal discretization of Eq. (1). Upon defining

$$\mathbf{R}^{(1)}(\mathbf{w}) := -\nabla_x \cdot (\mathbf{F}(\mathbf{w}) - \mathbf{F}^v(\mathbf{w}, \nabla_x \mathbf{w})), \tag{2}$$

*Corresponding author
Email addresses: jonas.zeifang@uhasselt.be (Jonas Zeifang), arjun.thenerymanikantan@uhasselt.be (Arjun Thenery Manikantan), jochen.schuetz@uhasselt.be (Jochen Schütz)

Eq. (1) can be cast as an ODE in some infinite-dimensional function space,

$$\mathbf{w}_t = \mathbf{R}^{(1)}(\mathbf{w}). \tag{3}$$

While classical timestepping methods only make use of the information of the first time derivative $\mathbf{w}_t$, the idea of two-derivative schemes is to additionally make use of the second temporal derivative. This adds an extra degree of freedom to the discretization and hence facilitates the development of storage- and runtime efficient high-order schemes. The second temporal derivative of $\mathbf{w}$ can be obtained by differentiating Eq. (1),

$$\mathbf{w}_{tt} = \mathbf{R}^{(2)}(\mathbf{w}, \mathbf{R}^{(1)}(\mathbf{w})), \tag{4}$$

where $\mathbf{R}^{(2)}$ for the Navier-Stokes equations is defined through

$$\mathbf{R}^{(2)}(\mathbf{w}, \mathbf{R}^{(1)}(\mathbf{w})) := -\nabla_x \cdot \left(\frac{\partial \mathbf{F}}{\partial \mathbf{w}} \mathbf{R}^{(1)}(\mathbf{w}) - \frac{\partial \mathbf{F}^v}{\partial \mathbf{w}} \mathbf{R}^{(1)}(\mathbf{w}) - \frac{\partial \mathbf{F}^v}{\partial \nabla_x \mathbf{w}} \nabla_x \mathbf{R}^{(1)}(\mathbf{w}) \right). \tag{5}$$

For more details on the derivation of $\mathbf{w}_{tt}$, consult [1]. In [2], a novel class of implicit two-derivative deferred correction time discretization methods has been introduced. The concept is based on a predictor-corrector formulation and can - in principle - achieve arbitrary orders. After a predictor step based on the two-derivative Taylor method, successive correction steps improve the solution towards a background two-derivative Hermite-Birkhoff Runge-Kutta method, giving rise to the name Hermite-Birkhoff predictor-corrector methods (HBPC). In [3], HBPC schemes up to order 8 have been numerically investigated. The schemes are A(α)-stable with stability angles α close to 90°, see [4]. Recently, these schemes have been combined with a high order discontinuous Galerkin spectral element spatial discretization of the Euler and Navier-Stokes equations [1].

A common strategy to enable large-scale simulations of discretizations of Eq. (1) is the use of spatial parallelization. It typically comes with high parallel efficiencies. However, caused by an increase of the communication to computation ratio, the spatial parallelization tends to saturate as the assigned work per processor decreases. This has been observed by various authors, see e.g. [5, 6, 7]. One remedy is to additionally consider the parallelization of the temporal domain, which requires specifically designed strategies due to the causality principle. It has been shown that combining temporal and spatial parallelization can further reduce the required wallclocktimes, see e.g. [8, 9, 10]. An overview on parallel-in-time (PinT) algorithms can be found in the review articles [11] and [12]. Further literature and information can also be found on the PinT web page [13].

One particularly attractive property of the HBPC methods is that they offer a mild time parallelism. This class of time parallel methods is sometimes classified as "parallel-across-the-method" [14] or "direct time-parallel methods" [12]. This time-parallelism is based on the idea of distributing different correction steps to different processors, and has been introduced in [15], but has also been used for the RIDC (revisionist integral deferred correction) schemes in [16, 17]. While being limited to a mild parallelization, i.e. using $O(10)$ processors at maximum, this concept offers good parallel efficiencies [17]. Also for the HBPC schemes, a good speedup in computational time has been observed when solving ODEs, see [3]. One prerequisite for a good parallel speedup of this type of parallelization is equally expensive prediction/correction steps. However, already for the ODE examples investigated in [3], a large discrepancy of the computational work of the different prediction/correction steps has been observed. This is due to the different costs of the solution of the algebraic systems of equations in the prediction/corrections steps. This non-homogeneous work distribution also transfers to the Navier-Stokes equations discretized with the discontinuous Galerkin spectral element method. We illustrate this with an introductory example that describes an advection-diffusion process of a density sine-wave, see Eq. (25) for initial conditions. The same simulation setup as described in [1, Sec. 5.2.] is used. The required number of GMRES iterations per prediction/correction step is reported in Fig. 1. One can see that especially the predictor (and, to a less extent, also the first correction step) requires significantly more computational work than the other correction steps. Therefore, in order to achieve a good parallel speedup when distributing different prediction/correction steps to different processors, one has two opportunities: try to harmonize the computational work and/or to develop a parallelization strategy that takes the different costs of the different iterates into account.

In this work, we harmonize the computational work per processor through a novel parallelization strategy where, in addition to the parallelization over the correction steps, there is also a parallelization over the stages of the predictor and the first corrector. Furthermore, to use computational resources as efficiently as possible, a novel strategy to adaptively determine the amount of Newton steps is developed. It is shown through several numerical testcases that this

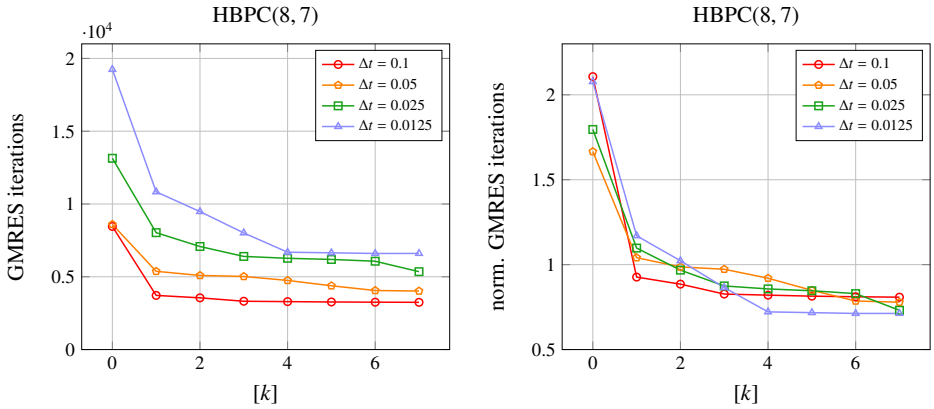

Figure 1. Cumulated (left) and normalized (right) number of GMRES iterations per prediction/correction step for the Navier-Stokes example described in [1, Sec. 5.2.] using the serial HBPC(8, 7) scheme [3] with different timestep sizes for the temporal discretization. Normalization (right) has been done with the mean number of GMRES iterations per timestep. $[k]$ denotes the number of the correction step, [0] corresponding to the predictor.

leads to a work distribution that is more homogeneous and hence more efficient than the straightforward application of the scheme in [1]. Time-parallel efficiencies of 60-70% are demonstrated. Also comparisons to established ESDIRK schemes are being made. As such, the main contributions of this work can be summarized as follows:

- An adaptive strategy for the Newton procedure, including a reliable error estimator, is developed in the context of the HBPC-DGSEM-methods. The strategy is numerically investigated.
- A parallelization strategy for the HBPC-DGSEM-methods that balances the loads over the different processors more evenly is developed.
- The actual parallel speedup is thoroughly investigated numerically.

The remainder of this paper is structured as follows: In Sec. 2 the implicit two-derivative predictor-corrector time discretization method and its temporal parallelization strategy are introduced. The fully discrete scheme is summarized in Sec. 3. In order to homogenize the computational work and to enable efficient simulations, an adaptive strategy for the non-linear solver is introduced in Sec. 4. After having introduced all the ingredients of the novel method, its parallel performance and its efficiency compared to established serial methods is evaluated in Sec. 5. Finally, conclusion and outlook are given in Sec. 6.

2. Parallel-in-Time HBPC Method

2.1. The Hermite-Birkhoff Predictor-Corrector Method

The parallel-in-time algorithm described in this paper is based on the two-derivative deferred correction method introduced in [2] and [3], which relies on the approximate quantities

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

Here, $\mathcal{N}_T$ is the number of discrete time levels, c_l a Runge-Kutta-type relative timestep of an s-stage Runge-Kutta method and $k_{\max}$ denotes the number of correction steps of the underlying deferred correction procedure. The s-stage Runge-Kutta methods are given by their two-derivative Butcher tableaux consisting of typically dense matrices $B^{(1)}$, $B^{(2)} \in \mathbb{R}^{s\times s}$ and a vector $c \in \mathbb{R}^s$. They define the background Hermite-Birkhoff Runge-Kutta scheme and are given in the appendix, Eq. (B.1) and Eq. (B.3). More details can be found in [3]. The coefficients of the Butcher tableaux define a quadrature formula $\mathcal{I}_l$ of order q through

$$\mathcal{I}_l\left(\mathbf{w}^1, \ldots, \mathbf{w}^s\right) := \Delta t \sum_{j=1}^{s} B^{(1)}_{lj} \mathbf{R}^{(1)}(\mathbf{w}^j) + \Delta t^2 \sum_{j=1}^{s} B^{(2)}_{lj} \mathbf{R}^{(2)}(\mathbf{w}^j) \tag{6}$$

for every stage $1 \leq l \leq s$. Note that we have omitted the additional dependencies of $\mathbf{R}^{(1)}$ and $\mathbf{R}^{(2)}$ given by Eq. (4) and Eq. (5) for the sake of brevity.

We use the parallel-in-time HBPC method according to [3, Alg. 2] and its improvement according to [4]. The predictor requires more computational load than higher correction steps, we have therefore modified the algorithm such that it allows for a parallelization of the stages for the predictor and the first correction step. The modifications in comparison to [4] have been marked in red color, they only apply to the definition of the quadrature rule. Note that while the original algorithm in [3] offers the possibility to use an IMEX splitting, here, only the implicit part is considered.

Algorithm 1 (HBPC($q, k_{\max}$))**.** *To advance the solution to Eq. (3) in time, we compute values $\mathbf{w}^{n,[k],l}$. To account for the initial conditions $\mathbf{w}_0$, define*

$$\mathbf{w}^{-1,[k],s} := \mathbf{w}_0.$$

First, the values $\mathbf{w}^{n,[0],l}$ are filled using a straightforward second-order implicit Taylor method departing from $\mathbf{w}^{n-1,[1],s}$.

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

$$\begin{aligned} \mathbf{w}^{n,[0],1} &:= \mathbf{w}^{n-1,[1],s}, \\ \mathbf{w}^{n,[0],l} &:= \mathbf{w}^{n-1,[1],s} + c_l \Delta t \mathbf{R}^{(1)}(\mathbf{w}^{n,[0],l}) - \frac{(c_l \Delta t)^2}{2} \mathbf{R}^{(2)}(\mathbf{w}^{n,[0],l}). \end{aligned} \tag{7}$$

2. ***Correct.*** *Next, the corrected values $\mathbf{w}^{n,[k],l}$ for $1 \leq k \leq k_{\max}$ are computed through solving for each $2 \leq l \leq s$ and each $1 \leq k \leq k_{\max}$:*

$$\begin{aligned} \mathbf{w}^{n,[k],1} &:= \mathbf{w}^{n-1,[k+1],s}, \\ \mathbf{w}^{n,[k],l} &:= \mathbf{w}^{n-1,[k+1],s} + \theta_1 \Delta t \left(\mathbf{R}^{(1)}(\mathbf{w}^{n,[k],l}) - \mathbf{R}^{(1)}(\mathbf{w}^{n,[k-1],l}) \right) - \theta_2 \frac{\Delta t^2}{2} \left(\mathbf{R}^{(2)}(\mathbf{w}^{n,[k],l}) - \mathbf{R}^{(2)}(\mathbf{w}^{n,[k-1],l}) \right) + \mathcal{I}_l, \end{aligned} \tag{8}$$

with

$$\begin{aligned} \mathcal{I}_l &:= \mathcal{I}_l\left(\mathbf{w}^{n,[0],1}, \ldots, \mathbf{w}^{n,[0],s}\right), && \text{for} \quad k = 1, \\ \mathcal{I}_l &:= \mathcal{I}_l\left(\mathbf{w}^{n,[k],1}, \ldots, \mathbf{w}^{n,[k],l-1}, \mathbf{w}^{n,[k-1],l}, \ldots, \mathbf{w}^{n,[k-1],s}\right), && \text{for} \quad k > 1. \end{aligned} \tag{9}$$

$\mathcal{I}_l(\cdot)$ denotes the q-th order Hermite-Birkhoff quadrature rule given in Eq. (6). If $k = k_{\max}$, then the $k+1$ superscripts in Eq. (8) 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*

$$\mathbf{w}^{n+1} := \mathbf{w}^{n,[k_{\max}],s}. \tag{10}$$

The coefficients $\theta = (\theta_1, \theta_2)$ are obtained by an optimization of the stability region, see [4]. For Alg. 1 with the Butcher tables given in Eq. (B.1) and Eq. (B.3) we find

$$\theta = (0.296, 0.0531) \quad \text{and} \quad \theta = (0.259, 0.0288) \tag{11}$$

for the sixth and the eighth order quadrature rules, respectively. The resulting methods are $A(\alpha)$-stable with the stability angles $\alpha > 89.81°$ (HBPC($6, k_{\max}$)) and $\alpha > 88.66°$ (HBPC($8, k_{\max}$)).

Remark 1. *Please note that for efficiency considerations, we only treat background schemes with an explicit first stage. Furthermore, the last stage corresponds to collocation point $c_s = 1$. Strictly speaking, this is not necessary; the update step (10) has then to be modified accordingly.*

Remark 2. *For $k > 1$, we use a Gauß-Seidel type procedure in the quadrature formula (9). Obviously, this could be done for $k = 1$ as well. However, the way we have formulated it in Alg. 1 makes it possible to parallelize over the stages for $k = 0$ and $k = 1$. This concept was not present in the original work [3]. The parallelization concept will be described in the next section.*

2.2. Parallelization of the HBPC Method

The structure of Alg. 1 allows to distribute the predictor and the correction steps on multiple processors, see [3]. The underlying basic idea of pipelining has been introduced in [15] and has also been used by the RIDC schemes [16, 17, 18].

Although the main ingredients of the parallelization of Alg. 1 have been already introduced in [3], we summarize them here and describe the differences of the present algorithm. The keys to parallelize Alg. 1 are:

- The stages of the prediction step at time instance n, i.e. $\mathbf{w}^{n,[0],l}$ only depend on the single value $\mathbf{w}^{n-1,[1],s}$ of the previous timestep. Hence, the different stages of the predictor can be calculated independently of each other.
- As the quadrature rule for the first correction step, i.e. $\mathbf{w}^{n,[1],l}$, only depends on values of the predictor, the different stages of the first correction step can also be calculated independently of each other.
- For $1 \leq k < k_{\max}$ the $[k]$-th correction step at time instance n, i.e. $\mathbf{w}^{n,[k],l}$, depends on the $[k-1]$-th iterate at the same time level n, as well as on the $[k+1]$-th correction step at the previous time step, $\mathbf{w}^{n-1,[k+1],s}$, see Eq. (8).
- The last correction step $[k_{\max}]$, i.e. $\mathbf{w}^{n,[k_{\max}],l}$ for $1 \leq l \leq s$, depends only on the $[k_{\max}-1]$-th iterate at the same time level n, as well as on the last correction iterate of the previous time level, $\mathbf{w}^{n-1,[k_{\max}],s}$.

The dependencies described above are visualized in Fig. 2 at the example of the sixth-order method. On the y-axis the different correction levels $0 \leq k \leq k_{\max}$ are illustrated, while on the x-axis, the different time levels $n, n+1, \ldots$ are indicated. A full circle at position (n,k) corresponds to the computation of all stages of $\mathbf{w}^{n,[k],l}$, $l = 2, \ldots, s$. (Calculation of the first stage $l = 1$ is trivial, see Eq. (8).) Splitted circles indicate computations of only one specific stage $l > 1$ of $\mathbf{w}^{n,[k],l}$. Note that the sixth order quadrature rule has two implicit stages; we hence split the circle in two semi-circles[1]. Numbers inside (semi-)circles indicate when the corresponding calculations can be performed: (semi-) circles with the same number can be computed at the same time in parallel, while those with a higher number have to wait for those with a lower number to finish. The main difference of the parallelization strategy performed here and the one described in [3, Alg. 2] is that we exploit the independence of different stages for the prediction and the first correction step. This is inspired by the observation that the calculation of the predictor and the first correction step is typically more expensive than the remaining correction steps, see [3]. This is also true for the PDE discretization considered in this work, see Fig. 1. The adaptive Newton strategy, which is described later in Sec. 4, will sharpen this observation, see Fig. 5. From Fig. 2 one can see that if one groups the correction iterates $[k]$ and $[k+1]$, one obtains consecutively numbered circles on all processors. For the predictor and the first correction step this is done in an analogous way, i.e. the predictor and corrector of one specific stage $l > 1$ are grouped together. The processor boundaries resulting from this grouping are visualized with dashed lines in Fig. 2.

Finally, one can see that each processor contains consecutively numbered circles, i.e. if communication is instantaneous and all calculations indicated with a (semi-)circle are equally expensive, there is no processor idle time.

Remark 3. *The underlying assumption behind this is that* solving for one stage of the predictor or the first corrector has the same cost as solving for all stages of one of the following correction steps ($k > 1$). *While this is of course not true in a mathematically rigorous way, our numerical experience, see also Fig. 1, indicates that this assumption is reasonable.*

Hence, the total amount of work packages per timestep is $2(s-1) + k_{\max} - 1$, where $2(s-1)$ work packages stem from the predictor and the first corrector, and $k_{\max} - 1$ work packages are due to the following correction steps. Note again that we have directly assumed that the calculation of the first stage is trivial. For the evaluation of the temporal parallelization's maximum achievable speedup, one additionally has to find the relation between the total amount of work packages and the work packages on a single processor where the initial startup phase is taken into account. While the amount of work packages on one single processor is $2N_T$, the startup phase takes $k_{\max} - 1$ work packages until the processor with index #0 can start. Under those assumptions the maximum achievable speedup can be calculated by

$$\frac{N_T(2(s-1)+k_{\max}-1)}{2N_T + k_{\max} - 1} \to \frac{k_{\max}+1}{2} + s - 2, \quad N_T \to \infty. \tag{12}$$

[1] It should be three semi-circles for the eighth-order method.

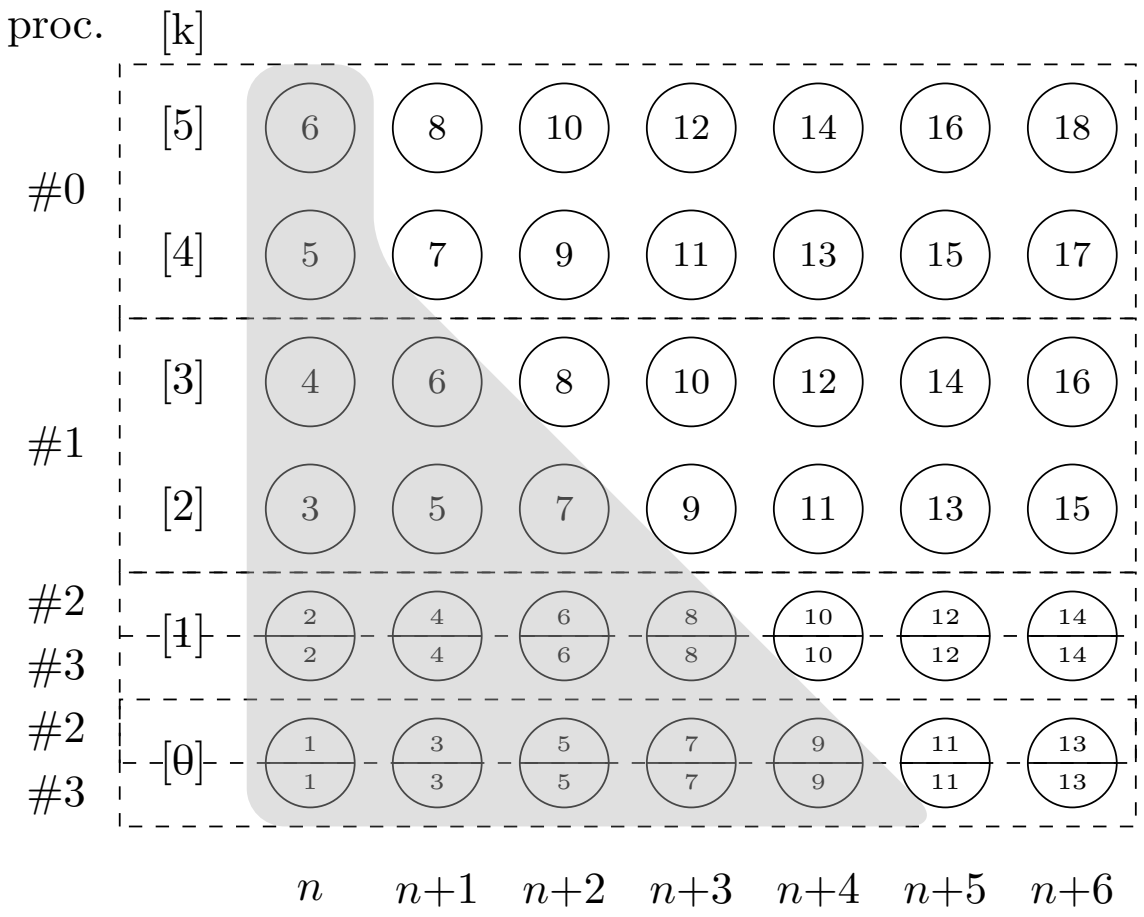

Figure 2. Schematic overview on parallelization strategy of HBPC(6, 5) method. The parallelization is according to [3], with the only difference that for $k = 0$ and $k = 1$, parallelization is also done over the stages, which is indicated by semi-circles. Note that the sixth order quadrature rule has two implicit stages; we hence split the circle in two semi-circles. At the left, the processor index #i and the current iterate $[k]$ are indicated. On the x-axis, time instances $n, n+1, \ldots$ are visualized. Numbers inside (semi-)circles indicate when the corresponding calculations can be performed. The gray-shaded area highlights solution steps where no adaptive Newton strategy can be performed, see Remark 11.

3. Fully Discrete Method

3.1. Two-Derivative Discontinuous Galerkin Method

After having introduced the temporal discretization procedure, a spatial discretization of $\mathbf{R}^{(1)}$ and $\mathbf{R}^{(2)}$ is needed. In [19] it has been shown that a careful discretization of the second derivative operator $\mathbf{R}^{(2)}$ is required to retain the stability properties of the ODE integrator as it is desirable for a method-of-lines approach. This idea from [19] has been formulated for a Discontinuous Galerkin Spectral Element Method (DGSEM [20]) discretization of nonlinear equations in [1]. Here, we will only very briefly recall this discretization for a purely hyperbolic PDE and refer the reader to [1] and the references therein for more details. The DGSEM is based on the weak formulation of Eq. (1),

$$\sum_{e=1}^{\mathcal{N}_E} \left(\mathbf{w}_t, \phi\right)_{\Omega_e} - \left(\mathbf{F}(\mathbf{w}), \nabla_x \phi\right)_{\Omega_e} + \left\langle \mathbf{F}^*(\mathbf{w}^L, \mathbf{w}^R)\cdot\mathbf{n}, \phi \right\rangle_{\partial\Omega_e} = 0, \qquad \forall \phi \in \Pi_{\mathcal{N}_p}, \tag{13}$$

where the function space $\Pi_{\mathcal{N}_p}$ of the test functions ϕ is the tensor-product of the one-dimensional Lagrange polynomials ℓ, each of degree $\mathcal{N}_p$. The domain Ω is split into $\mathcal{N}_E$ non-overlapping hexahedral (3d) or quadrangular (2d) elements. The integration over an element $\Omega_e \in \Omega$ is denoted by the scalar product $(\cdot,\cdot)$ and integration over the cell edges $\partial\Omega_e$ is denoted by $\langle\cdot,\cdot\rangle$. For the evaluation of the surface integral, the flux is substituted by a numerical flux $\mathbf{F}^*$, depending on the values of both adjacent elements of the edge ($\mathbf{w}^L$ and $\mathbf{w}^R$) and the outward pointing normal vector $\mathbf{n}$ of the current element. The numerical flux is chosen to be a global Lax-Friedrichs, see [1, Eq. (13)]. Using DGSEM techniques on (13), see [21], yields the discrete operator $\mathbf{R}_h^{(1)}(\mathbf{w}_h)$ as an approximation to $\mathbf{R}^{(1)}(\mathbf{w})$.

The second derivative operator $\mathbf{R}^{(2)}$ is defined through the artificial quantity

$$\boldsymbol{\sigma} := \mathbf{R}^{(1)}(\mathbf{w}) \equiv \mathbf{w}_t. \tag{14}$$

In [1] a DGSEM discretization of the second temporal derivative has been proposed via the weak formulation

$$\sum_{e=1}^{\mathcal{N}_E} \left(\mathbf{w}_{tt}, \phi\right)_{\Omega_e} - \left(\frac{\partial \mathbf{F}(\mathbf{w})}{\partial \mathbf{w}}\boldsymbol{\sigma}, \nabla_x\phi\right)_{\Omega_e} + \left\langle \frac{\partial \mathbf{F}^*(\mathbf{w}^L,\mathbf{w}^R)}{\partial \mathbf{w}^L}\boldsymbol{\sigma}^L\cdot\mathbf{n} + \frac{\partial \mathbf{F}^*(\mathbf{w}^L,\mathbf{w}^R)}{\partial \mathbf{w}^R}\boldsymbol{\sigma}^R\cdot\mathbf{n}, \phi \right\rangle_{\partial\Omega_e} = 0, \qquad \forall \phi \in \Pi_{\mathcal{N}_p}. \tag{15}$$

Note that the discretization of the second derivative operator is similar to the first derivative operator except for the flux which has to be substituted by $\partial\mathbf{F}(\mathbf{w})/\partial\mathbf{w}\cdot\boldsymbol{\sigma}$ (compare Eq. (13) and Eq. (15)). In analogy to the first derivative operator we obtain the discrete operator $\mathbf{R}_h^{(2)}(\mathbf{w}_h, \boldsymbol{\sigma}_h)$ for the second temporal derivative.

Considering the Navier-Stokes equations, see Eq. (1), second order spatial derivatives occur by the introduction of the viscous flux $\mathbf{F}^v(\mathbf{w}, \nabla_x\mathbf{w})$. They are discretized by following the BR2 lifting approach [22]. A detailed description of how the Navier-Stokes equations can be handled with the two-derivative DGSEM can be found in [1].

3.2. Solving for the Stage Values

In Sec. 2.1 and Sec. 3.1 we have introduced the temporal and the spatial discretization, respectively. Bringing both together, one has to solve for the stages $l > 1$ of the predictor and the correction steps in Eq. (7) and Eq. (8). The resulting non-linear system to be solved is very similar for the predictor and the corrector (see also [1, Sec. 3.2.1.]). Due to the non-linearity of the considered systems of equations, one has to use some non-linear solution procedure. As it is common for time-dependent PDE discretizations, we use Newton's method for that purpose.

We start by casting the predictor and corrector step (Eq. (7) and Eq. (8)) for the current timestep n, iterate k and stage l into a uniform formulation. Due to the introduction of the quantity $\boldsymbol{\sigma}_h^{n,[k],l} := \mathbf{R}_h^{(1)}(\mathbf{w}_h^{n,[k],l})$, we have to extend the state vector to consist of the discrete $\mathbf{w}_h^{n,[k],l}$ and $\boldsymbol{\sigma}_h^{n,[k],l}$, i.e. we introduce $\boldsymbol{X}^{[k]} := \left(\mathbf{X}_w^{[k]}, \mathbf{X}_\sigma^{[k]}\right)^T := \left(\mathbf{w}_h^{n,[k],l}, \boldsymbol{\sigma}_h^{n,[k],l}\right)^T$. The non-linear equation to be solved can then be written as

$$\boldsymbol{X}^{[k]} \stackrel{!}{=} \begin{pmatrix}\mathbf{w}_{\text{old}}\\ 0\end{pmatrix} + \begin{pmatrix}\Phi\left(\boldsymbol{X}^{[k]}, \boldsymbol{X}^{[k-1],1:s}\right)\\ \mathbf{R}_h^{(1)}(\mathbf{w}_h^{n,[k],l})\end{pmatrix} =: \boldsymbol{W}_{\text{old}} + \bar{\Phi}\left(\boldsymbol{X}^{[k]}, \boldsymbol{X}^{[k-1],1:s}\right), \tag{16}$$

with $\mathbf{w}_{\text{old}} := \mathbf{w}_h^{n-1,[k+1],s}$ for $k < k_{\max}$ and $\mathbf{w}_{\text{old}} := \mathbf{w}_h^{n-1,[k_{\max}],s}$ for $k = k_{\max}$. For the sake of notation, we use the abbreviation $\boldsymbol{X}^{[k-1],1:s} := \left(\boldsymbol{X}^{[k-1],1}, \boldsymbol{X}^{[k-1],2}, \dots, \boldsymbol{X}^{[k-1],s}\right)$. For the predictor, Φ is given by

$$\Phi\left(\boldsymbol{X}^{[k]}, \boldsymbol{X}^{[k-1],1:s}\right) := c_l \Delta t \mathbf{R}_h^{(1)}(\mathbf{w}_h^{n,[k],l}) - \frac{(c_l\Delta t)^2}{2}\mathbf{R}_h^{(2)}(\mathbf{w}_h^{n,[k],l}, \boldsymbol{\sigma}_h^{n,[k],l}), \tag{17}$$

and for the first corrector step by

$$\begin{aligned}\Phi\left(\boldsymbol{X}^{[k]}, \boldsymbol{X}^{[k-1],1:s}\right) := \theta_1 \Delta t \mathbf{R}_h^{(1)}(\mathbf{w}_h^{n,[k],l}) - \frac{\theta_2\Delta t^2}{2}\mathbf{R}_h^{(2)}(\mathbf{w}_h^{n,[k],l}, \boldsymbol{\sigma}_h^{n,[k],l}) \\ - \theta_1 \Delta t \mathbf{R}_h^{(1)}(\mathbf{w}_h^{n,[k-1],l}) + \frac{\theta_2\Delta t^2}{2}\mathbf{R}_h^{(2)}(\mathbf{w}_h^{n,[k-1],l}, \mathbf{R}_h^{(1)}(\mathbf{w}_h^{n,[k-1],l})) + \mathcal{I}_l\left(\mathbf{w}_h^{n,[k-1],1:s}\right).\end{aligned} \tag{18}$$

Remark 4. *For the ease of presentation, we did not distinguish between the treatment of the quadrature rule $\mathcal{I}_l$ for $k = 1$ and $k > 1$, see (9). The treatment for $k > 1$ results in slightly different arguments of the quadrature formula and hence additional arguments in Φ. The modifications are straightforward and do not change the proposed arguments here, yet they make the notation more clumsy.*

We use Newton's method to solve equations of type (16), in this particular case given by:

1. For $\mathtt{r} = 1, \dots$ solve

$$\begin{aligned}\left(\mathrm{Id} - \frac{\partial\bar{\Phi}\left(\boldsymbol{X}_{\mathtt{r}-1}^{[k]}, \boldsymbol{X}_{\mathtt{r}'}^{[k-1],1:s}\right)}{\partial \boldsymbol{X}^{[k]}}\right)\Delta \boldsymbol{X}_{\mathtt{r}} &= \boldsymbol{W}_{\text{old}} + \bar{\Phi}\left(\boldsymbol{X}_{\mathtt{r}-1}^{[k]}, \boldsymbol{X}_{\mathtt{r}'}^{[k-1],1:s}\right) - \boldsymbol{X}_{\mathtt{r}-1}^{[k]} \\ \boldsymbol{X}_{\mathtt{r}}^{[k]} &= \boldsymbol{X}_{\mathtt{r}-1}^{[k]} + \Delta \boldsymbol{X}_{\mathtt{r}}.\end{aligned} \tag{19}$$

2. If the convergence criterion is met, set

$$\mathbf{w}_h^{n,[k],l} := \boldsymbol{X}_{\mathtt{r},w}^{[k]} \quad \text{and} \quad \boldsymbol{\sigma}_h^{n,[k],l} := \mathbf{R}_h^{(1)}(\mathbf{w}_h^{n,[k],l}). \tag{20}$$

Note that we have indicated that solutions from the previous correction step, i.e. $\boldsymbol{X}^{[k-1],1:s}$, are obtained via Newton's method terminated at some finite Newton iterate $\mathtt{r}'$, which can be different for different stages and different k. To initialize the iterative procedure, some initial guess

$$\boldsymbol{X}_0^{[k]} \equiv \left(\mathbf{w}_{h,0}^{n,[k],l}, \mathbf{R}_h^{(1)}\left(\mathbf{w}_{h,0}^{n,[k],l}\right)\right)^T$$

has to be specified.

Remark 5. *If not stated otherwise, we choose an explicit second order Taylor step to obtain the initial guess for the predictor. For the correction step $[k]$, the corresponding stage value of the previous iterate $[k-1]$ is used, i.e.*

$$\begin{aligned}\mathbf{w}_{h,0}^{n,[0],l} &= \mathbf{w}_{old} + c_l\Delta t \mathbf{R}_h^{(1)}(\mathbf{w}_{old}) + (c_l\Delta t)^2 \mathbf{R}_h^{(2)}(\mathbf{w}_{old}, \mathbf{R}_h^{(1)}(\mathbf{w}_{old})) && \text{for } l = 2, \dots, s. \\ \mathbf{w}_{h,0}^{n,[k],l} &= \mathbf{w}_h^{n,[k-1],l} && \text{for } l = 2, \dots, s, \text{ and } k = 1, \dots, k_{\max}.\end{aligned} \tag{21}$$

We have observed that using a second order explicit Taylor step to obtain an initial guess for the predictor is superior to performing an explicit first order step. However, using a third order Taylor step did not give noticeable advantages. Please note that the effectiveness of using the second order step remains problem- and timestep-dependent.

Remark 6. *Please note that at the end of the Newton algorithm, we define $\boldsymbol{\sigma}_h^{n,[k],l} := \mathbf{R}_h^{(1)}(\mathbf{w}_h^{n,[k],l})$ in Eq. (20) rather than setting $\boldsymbol{\sigma}_h^{n,[k],l} = \boldsymbol{X}_{\mathtt{r},\sigma}^{[k]}$. If the equation (16) is solved exactly, $\boldsymbol{\sigma}_h^{n,[k],l}$ would be identical to $\mathbf{R}_h^{(1)}(\mathbf{w}_h^{n,[k],l})$. However, it is only solved to a certain accuracy, and hence, the identity does not necessarily hold. The definition in Eq. (20) avoids inconsistencies in $\mathbf{w}_h$, $(\mathbf{w}_h)_t$ and $(\mathbf{w}_h)_{tt}$ during the timestepping procedure and potential instabilities. Because of this, $\boldsymbol{\sigma}_h$ of a previous timestep, stage or correction iterate is no longer an independent variable and hence does not occur as an explicit argument in $\mathbf{R}_h^{(2)}(\mathbf{w}_h^{n,[k-1],l}, \mathbf{R}_h^{(1)}(\mathbf{w}_h^{n,[k-1],l}))$ and in $\mathcal{I}_l\left(\mathbf{w}_h^{n,[k-1],1:s}\right)$ in Eq. (18).*

In order to solve the arising linear system in Eq. (19), we use the matrix-free GMRES approach with extended block-Jacobi preconditioning described in [1]. As initial condition for the GMRES method a zero vector is chosen. Choosing the negative right hand side times the timestep as initial guess as suggested in [23] can sometimes be advantageous. Similar as the authors in [23], we observed that this advantage is problem dependent and can in some cases have an unfavorable influence on the required iterations, which is especially the case for large timesteps. We therefore use the zero vector as initial conditions in all simulations performed in this work. Similar as it has been done in [1], we neglect the Hessian contribution in Eq. (19), when solving the linear system.

3.3. Error estimator of the HBPC method

Controlling the numerical error introduced through the integration scheme per timestep is obviously crucial for multiple purposes. When using classical implicit Runge-Kutta methods, an error estimate is typically obtained via an embedded quadrature rule, see e.g. [24, 25]. This embedded quadrature rule uses the same nodes as the original scheme but utilizes different weights. This offers the opportunity to obtain either higher or lower order embedded schemes, see e.g. [25]. Inspired by these error estimates for Runge-Kutta methods, we define additional quadrature rules of order $\hat{q} = q - 1$ for the HBPC$(6, k_{\max})$ and the HBPC$(8, k_{\max})$ method,

$$\hat{\mathcal{I}}_l\left(\mathbf{w}^1, \dots, \mathbf{w}^s\right) := \Delta t \sum_{j=1}^{s} \hat{B}^{(1)}_{lj} \mathbf{R}^{(1)}(\mathbf{w}^j) + \Delta t^2 \sum_{j=1}^{s} \hat{B}^{(2)}_{lj} \mathbf{R}^{(2)}(\mathbf{w}^j).$$

The coefficients of the tables $\hat{B}^{(1)}$ and $\hat{B}^{(2)}$ are obtained through collocation such that they utilize the same nodes, i.e. $c = \hat{c}$. In the collocation procedure, the (arbitrary) choice is made that $\mathbf{w}_{tt}$ at time instant $c_1 = 0$ is *not* taken into account. This leads to schemes that are one order lower than the original quadrature rules HBPC$(6, k_{\max})$ and HBPC$(8, k_{\max})$, respectively. The Butcher tableaux corresponding to these fifth and seventh order, respectively, methods are given in Eq. (B.2) and Eq. (B.4).

Error Estimate. The error estimate $\|\mathcal{E}_t^{n,[k],l}\|_2$ is then obtained by

$$\|\mathcal{E}_t^{n,[k],l}\|_2 := \left\|\mathbf{w}_h^{n,[k],l} - \tilde{\mathbf{w}}_h^{n,[k],l}\right\|_2, \quad \text{with} \quad \tilde{\mathbf{w}}_h^{n,[k],l} := \mathbf{w}_h^{n-1,[k+1],s} + \hat{\mathcal{I}}_l\left(\mathbf{w}_h^{n,[k^*],1}, \dots, \mathbf{w}_h^{n,[k^*],s}\right), \tag{22}$$

where we have defined k^* as a function of k to be the closest odd integer that is larger or equal than k. Due to the pipelining strategy of Alg. 1, this means that the k^*-th iterate is always the correction with the highest index available on one processor. Note that due to the construction of the parallelization strategy, see also Fig. 2, the processor(s) handling the predictor and the first corrector step for stages $\mathbf{w}^{n,[k],l}$ with $l \neq s$ (in the example in the figure, this would be proc. #3) are somewhat special, as they do not have acceess to the final stage $\mathbf{w}_h^{n,[k],s}$ of their corresponding $k \in \{0, 1\}$. Hence, the error estimates $\mathcal{E}_t^{n,[k],l}$ with $l \neq s$ are only needed for these processor(s). All other processors utilize $\mathcal{E}_t^{n,[k],s}$ for their error estimates.

Alternatively, instead of evaluating the quadrature rule directly, one can perform an additional correction step with $\hat{\mathcal{I}}_l$ to obtain an approximate quantity $\tilde{\tilde{\mathbf{w}}}^{n,[k],l}$. That means, solve the following for $\tilde{\tilde{\mathbf{w}}}^{n,[k],l}$:

$$\begin{aligned}\tilde{\tilde{\mathbf{w}}}_h^{n,[k],l} = \mathbf{w}_h^{n-1,[k+1],s} &+ \theta_1 \Delta t \left(\mathbf{R}^{(1)}(\tilde{\tilde{\mathbf{w}}}_h^{n,[k],l}) - \mathbf{R}^{(1)}(\mathbf{w}_h^{n,[k^*],l})\right) - \theta_2 \frac{\Delta t^2}{2} \left(\mathbf{R}^{(2)}(\tilde{\tilde{\mathbf{w}}}_h^{n,[k],l}, \tilde{\tilde{\boldsymbol{\sigma}}}_h^{n,[k],l}) - \mathbf{R}^{(2)}(\mathbf{w}_h^{n,[k^*],l}, \mathbf{R}^{(1)}(\mathbf{w}_h^{n,[k^*],l}))\right) \\ &+ \hat{\mathcal{I}}_l\left(\mathbf{w}_h^{n,[k^*],1}, \dots, \mathbf{w}_h^{n,[k^*],s}\right),\end{aligned} \tag{23}$$

and following, calculate the error estimate via

$$\|\mathcal{E}_t^{n,[k],l}\|_2 := \left\|\mathbf{w}_h^{n,[k],l} - \tilde{\tilde{\mathbf{w}}}_h^{n,[k],l}\right\|_2. \tag{24}$$

Evaluation of Temporal Error Estimate. The accuracy of the embedded error estimate is evaluated by considering the Navier-Stokes equations (Eq. (1)) with initial conditions

$$\rho(\mathbf{x}, t = 0) = 1 + 0.3 \sin\left(\pi(x_1 + x_2)\right), \quad \mathbf{v} = (0.3, 0.3)^T, \quad \text{and} \quad p = 1, \tag{25}$$

on the domain $\Omega = [-1, 1]^2$, equipped with periodic boundary conditions. Viscosity is chosen to be $\mu = 10^{-3}$ and the reference Mach number is $\varepsilon \in \{1, 10^{-1}\}$. The domain is discretized with $N_E = 64^2$ elements with $N_p = 7$. The 'exact' solution is obtained via an explicit simulation with a fourth order low-storage Runge-Kutta method [26] with very small timestep ($\Delta t \approx 2.9 \cdot 10^{-5}$ and $\Delta t \approx 7.53 \cdot 10^{-6}$ for $\varepsilon = 1$ and $\varepsilon = 10^{-1}$, respectively).

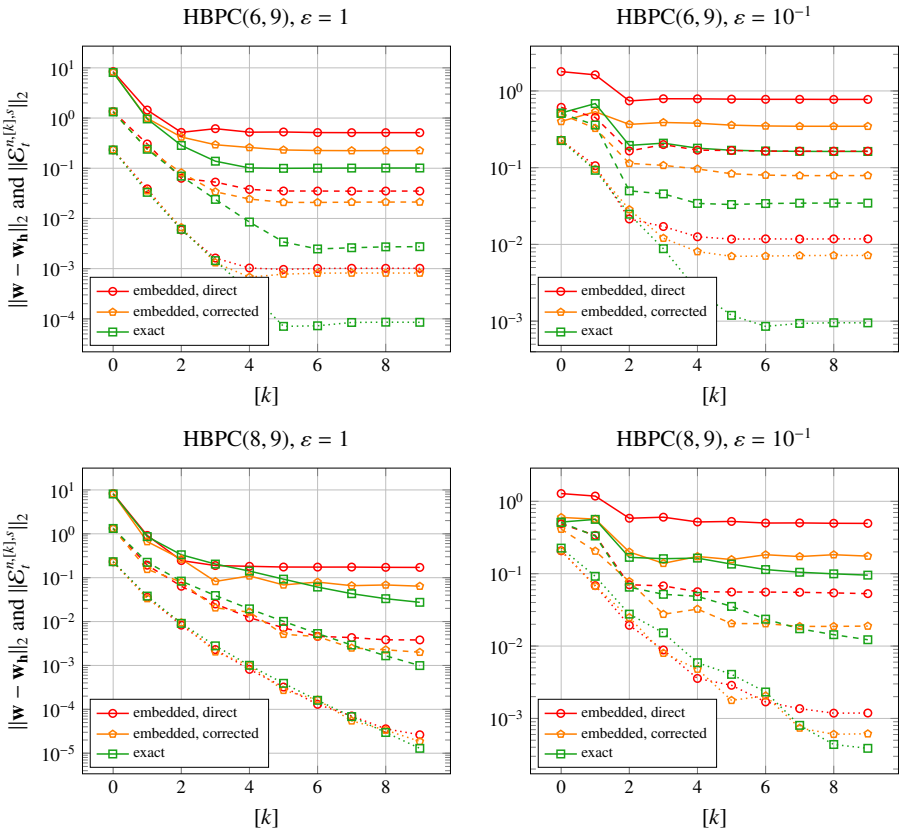

Figure 3. Exact L_2-error and estimated errors via evaluating the embedded formula directly (Eq. (22)) and evaluating the embedded formula within one correction step (Eq. (23)) for HBPC(6, 9) (top) and HBPC(8, 9) scheme (bottom) after the first timestep. (Only one timestep each is performed, as the embedded formulae only measure local (in time) error contributions.) Left column shows results with $\varepsilon = 1$ and $\Delta t = 0.4$ (solid), $\Delta t = 0.2$ (dashed) and $\Delta t = 0.1$ (dotted). Right column shows results with $\varepsilon = 10^{-1}$ and $\Delta t = 0.1$ (solid), $\Delta t = 0.05$ (dashed) and $\Delta t = 0.025$ (dotted).

We now perform a single timestep with different sizes for $\varepsilon = 1$ and $\varepsilon = 10^{-1}$ with the HBPC(6, 9) and the HBPC(8, 9) and report the exact and the estimated errors after the predictor and each correction step in Fig. 3. One can observe a clear trend: the higher the stiffness of the problem, i.e. larger Δt and/or smaller ε, the worse do the error estimators approximate the true error. For larger stiffnesses, the procedure according to Eq. (23) is more accurate than evaluating the embedded formula directly. For lower stiffnesses, the error estimates coincide very well with the true error until some minimum error is reached for some $[k]$. This is due to the fact that the embedded quadrature formula is only of order $\hat{q} = q - 1$ and hence has a lower accuracy than the original quadrature rule. (Technically, the error of the *lower-order* method is approximated.) Similar results are obtained when the accuracy of the embedded error estimator is tested on different meshes (not shown here), which shows the robustness of the error estimator. Summing up, the error estimate in Eq. (23) requiring an additional solving step is slightly more accurate than directly evaluating the embedded quadrature rule; it is recommended for stiff problems.

4. Adaptive Strategy for HBPC Schemes

A key feature for an efficient implicit time discretization method is an adaptation strategy for the iterative solution procedure, see e.g. [5, 27, 28], as it is of utmost importance to keep Newton and GMRES iterations to an absolute

minimum, while obviously guaranteeing a certain quality of the solution. This is very different to explicit schemes, where this part of the solution process simply does not exist. In this section, we are aiming for an adaptive Newton convergence criterion that preserves the accuracy of the time stepping method without 'oversolving' it, i.e., we envision that the error of the Newton procedure, defined by

$$\mathcal{E}_{\mathtt{r}}^{[k],l} := \boldsymbol{X}^{[k]} - \boldsymbol{X}_{\mathtt{r}}^{[k]}, \tag{26}$$

is of the same order as the time discretization error. Hence,

$$\|\mathcal{E}_t^{n,[k],l}\| \approx \|\mathcal{E}_{\mathtt{r}}^{[k],l}\|,$$

where we have omitted the superscript n for the error of Newton's procedure for the ease of presentation. In this way, one does not deteriorate the temporal accuracy, while at the same time one is not overdoing Newton iterations. A rough, but seemingly reliable estimate of Newton's error is devised through an analysis of the equations in Sec. 4.1.

Remark 7. *While it is possible to only use one Newton step per prediction/correction, and take into account more correction steps as similarly done in [29], we have found that this approach does not really work well in our context. In particular the solution quality of the predictor and the corrector do have a significant influence on higher corrections. This can already be seen in the context of ODEs; and has in fact motivated the analysis to follow.*

4.1. Adaptive Newton Strategy

Convergence criteria for Newton's method have been addressed by several authors in the context of flow simulation with implicit timestepping methods relying on Newton-Krylov methods. Basically, two different approaches can be distinguished:

- An absolute tolerance for the Newton increment $\Delta\boldsymbol{X}_{\mathtt{r}}$ has been used in [30]. The inequality $\|\Delta\boldsymbol{X}_{\mathtt{r}}\|_2 \leq \mathrm{TOL}$, specified by a user-defined tolerance $\mathrm{TOL} \in \{10^{-5}, 10^{-7}\}$, is used as a criterion to terminate the Newton iterations.

- More used in practice seem to be convergence criteria based on the Newton residual $\mathtt{N}(\boldsymbol{X})$, which is the quantity to which the discrete solution fails to satisfy the equation. [31], [32] and [33] start with a user defined accuracy TOL. An embedded Runge-Kutta method is then used to determine the corresponding timestep size and a modified tolerance TOL′. While [31] and [32] use $N(\boldsymbol{X}_{\mathtt{r}}) \leq N(\boldsymbol{X}_0) \cdot \mathrm{TOL}/5$ as convergence criterion ([32] also suggests the same treatment for the Newton increment), the authors in [33] use $N(\boldsymbol{X}_{\mathtt{r}}) \leq N(\boldsymbol{X}_0) \cdot \mathrm{TOL}'/10$. A slightly different approach is pursued in [5], where a fixed timestep is prescribed by the user and the convergence criterion $N(\boldsymbol{X}_{\mathtt{r}}) \leq N(\boldsymbol{X}_0) \cdot \min\left(10^{-3}, \|\mathcal{E}_t\|_2/3\right)$ is used, where $\|\mathcal{E}_t\|_2$ is computed through an embedded Runge-Kutta method. An *absolute tolerance for the Newton residual* has been proposed in [28]. They use $N(\boldsymbol{X}_{\mathtt{r}}) \leq \|\mathcal{E}_t\|_2/10$, where the temporal error estimate is again based on an embedded Runge-Kutta method.

None of these approaches can directly be used for the HBPC schemes as different levels of accuracy for the different prediction/correction steps are not taken into account. Inspired by the approach outlined in [28], we derive a Newton convergence criterion that explicitly takes the different levels into account. We find that an absolute convergence criterion based on the Newton increment is a natural choice for this kind of methods.

4.1.1. Newton Error Estimate

In this section, we derive a heuristic that links the Newton error $\mathcal{E}_{\mathtt{r}}^{[k],l}$, see Eq. (26), to the Newton increment $\Delta\boldsymbol{X}_{\mathtt{r}+1}$ of the *following* Newton step and the Newton errors $\mathcal{E}_{\mathtt{r}'}^{[k-1],i}$ of *previous* correction steps. This allows, in a subsequent step, to derive a practically usable criterion on when to terminate Newton's algorithm. Terminating Newton's method (see Eq. (19) and Eq. (20)) at finite $\mathtt{r}$, the introduced error $\mathcal{E}_{\mathtt{r}}^{[k],l}$ is given by

$$\mathcal{E}_{\mathtt{r}}^{[k],l} = \boldsymbol{X}^{[k]} - \boldsymbol{X}_{\mathtt{r}}^{[k]} = \boldsymbol{W}_{\mathrm{old}} + \bar{\Phi}\left(\boldsymbol{X}^{[k]}, \boldsymbol{X}^{[k-1],1:s}\right) - \boldsymbol{X}_{\mathtt{r}}^{[k]} + \bar{\Phi}\left(\boldsymbol{X}_{r}^{[k]}, \boldsymbol{X}_{\mathtt{r}'}^{[k-1],1:s}\right) - \bar{\Phi}\left(\boldsymbol{X}_{r}^{[k]}, \boldsymbol{X}_{\mathtt{r}'}^{[k-1],1:s}\right).$$

Performing a Taylor expansion one obtains

$$\begin{aligned}\boldsymbol{X}^{[k]} - \boldsymbol{X}_{\mathrm{r}}^{[k]} =& \frac{\partial \bar{\Phi}\left(\boldsymbol{X}_{\mathrm{r}}^{[k]}, \boldsymbol{X}_{\mathrm{r}'}^{[k-1],1:s}\right)}{\partial \boldsymbol{X}^{[k]}}\left(\boldsymbol{X}^{[k]} - \boldsymbol{X}_{\mathrm{r}}^{[k]}\right) + \sum_{i=1}^{s} \frac{\partial \bar{\Phi}\left(\boldsymbol{X}_{\mathrm{r}}^{[k]}, \boldsymbol{X}_{\mathrm{r}'}^{[k-1],1:s}\right)}{\partial \boldsymbol{X}^{[k-1],i}}\left(\boldsymbol{X}^{[k-1],i} - \boldsymbol{X}_{\mathrm{r}'}^{[k-1],i}\right) \\ & -\boldsymbol{X}_{\mathrm{r}}^{[k]} + \mathbf{W}_{\mathrm{old}} + \bar{\Phi}\left(\boldsymbol{X}_{r}^{[k]}, \boldsymbol{X}_{\mathrm{r}'}^{[k-1],1:s}\right) + \mathcal{O}\left(\left(\boldsymbol{X}^{[k]} - \boldsymbol{X}_{\mathrm{r}}^{[k]}\right)^2\right) + \mathcal{O}\left(\left(\boldsymbol{X}^{[k-1],1:s} - \boldsymbol{X}_{\mathrm{r}'}^{[k-1],1:s}\right)^2\right).\end{aligned}$$

Next, we truncate the higher order terms[2] and find

$$\mathcal{E}_{\mathrm{r}}^{[k],l} \doteq \left(\mathrm{Id} - \frac{\partial \bar{\Phi}\left(\boldsymbol{X}_{\mathrm{r}}^{[k]}, \boldsymbol{X}_{\mathrm{r}'}^{[k-1],1:s}\right)}{\partial \boldsymbol{X}^{[k]}}\right)^{-1} \cdot \left(\mathbf{W}_{\mathrm{old}} + \bar{\Phi}\left(\boldsymbol{X}_{r}^{[k]}, \boldsymbol{X}_{\mathrm{r}'}^{[k-1],1:s}\right) - \boldsymbol{X}_{\mathrm{r}}^{[k]} + \sum_{i=1}^{s} \frac{\partial \bar{\Phi}\left(\boldsymbol{X}_{\mathrm{r}}^{[k]}, \boldsymbol{X}_{\mathrm{r}'}^{[k-1],1:s}\right)}{\partial \boldsymbol{X}^{[k-1],i}}\left(\boldsymbol{X}^{[k-1],i} - \boldsymbol{X}_{\mathrm{r}'}^{[k-1],i}\right)\right).$$

We then can make use of the definition of Newton's method, see Eq. (19) to simplify the first part of the expression

$$\mathcal{E}_{\mathrm{r}}^{[k],l} \doteq \underbrace{\Delta \boldsymbol{X}_{\mathrm{r}+1}}_{\substack{\text{current Newton} \\ \text{procedure}}} + \underbrace{\left(\mathrm{Id} - \frac{\partial \bar{\Phi}\left(\boldsymbol{X}_{\mathrm{r}}^{[k]}, \boldsymbol{X}_{\mathrm{r}'}^{[k-1],1:s}\right)}{\partial \boldsymbol{X}^{[k]}}\right)^{-1} \cdot \left(\sum_{i=1}^{s} \frac{\partial \bar{\Phi}\left(\boldsymbol{X}_{\mathrm{r}}^{[k]}, \boldsymbol{X}_{\mathrm{r}'}^{[k-1],1:s}\right)}{\partial \boldsymbol{X}^{[k-1],i}} \mathcal{E}_{\mathrm{r}'}^{[k-1],i}\right)}_{\text{accumulation of previous Newton errors}}.$$

The error hence consists of one part, where the Newton errors of previous prediction/correction steps are accumulated and another part influenced by the current Newton procedure, which equals the Newton increment of the next Newton iterate $\Delta \boldsymbol{X}_{\mathrm{r}+1}$. For the predictor, we then directly find

$$\mathcal{E}_{\mathrm{r}}^{[0],l} \doteq \Delta \boldsymbol{X}_{\mathrm{r}+1},$$

as there are no previous prediction/correction steps that can influence the error. For the corrector one finds

$$\begin{aligned}\mathcal{E}_{\mathrm{r}}^{[k],l} \doteq \Delta \boldsymbol{X}_{\mathrm{r}+1} - & \begin{pmatrix} \mathrm{Id} - \theta_1 \Delta t \frac{\partial \mathbf{R}_h^{(1)}}{\partial \mathbf{w}_h^{n,[k],l}} + \frac{\theta_2 \Delta t^2}{2} \frac{\partial \mathbf{R}_h^{(2)}}{\partial \mathbf{w}_h^{n,[k],l}} & \frac{\theta_2 \Delta t^2}{2} \frac{\partial \mathbf{R}_h^{(2)}}{\partial \sigma_h^{n,[k],l}} \\ -\frac{\partial \mathbf{R}_h^{(1)}}{\partial \mathbf{w}_h^{n,[k],l}} & \mathrm{Id} \end{pmatrix}^{-1} \cdot \left(\begin{pmatrix} \theta_1 \Delta t \frac{\partial \mathbf{R}_h^{(1)}}{\partial \mathbf{w}_h^{n,[k-1],l}} - \frac{\theta_2 \Delta t^2}{2}\left(\frac{\partial \mathbf{R}_h^{(2)}}{\partial \mathbf{w}_h^{n,[k-1],l}} + \frac{\partial \mathbf{R}_h^{(2)}}{\partial \sigma_h^{n,[k-1],l}} \frac{\partial \mathbf{R}_h^{(1)}}{\partial \mathbf{w}_h^{n,[k-1],l}}\right) & 0 \\ 0 & 0 \end{pmatrix} \mathcal{E}_{\mathrm{r}'}^{[k-1],l} \right. \\ & \left. - \sum_{i=1}^{s} \begin{pmatrix} \Delta t B_{li}^{(1)} \frac{\partial \mathbf{R}_h^{(1)}}{\partial \mathbf{w}_h^{n,[k-1],i}} + \Delta t^2 B_{li}^{(2)}\left(\frac{\partial \mathbf{R}_h^{(2)}}{\partial \mathbf{w}_h^{n,[k-1],i}} + \frac{\partial \mathbf{R}_h^{(2)}}{\partial \sigma_h^{n,[k-1],i}} \frac{\partial \mathbf{R}_h^{(1)}}{\partial \mathbf{w}_h^{n,[k-1],i}}\right) & 0 \\ 0 & 0 \end{pmatrix} \mathcal{E}_{\mathrm{r}'}^{[k-1],i} \right),\end{aligned}$$

which can be simplified to (please note that due to construction, there holds $\frac{\partial \mathbf{R}_h^{(2)}}{\partial \sigma_h} = \frac{\partial \mathbf{R}_h^{(1)}}{\partial \mathbf{w}_h}$ [3])

$$\begin{aligned}\mathcal{E}_{\mathrm{r}}^{[k],l} \doteq \Delta \boldsymbol{X}_{\mathrm{r}+1} - & \begin{pmatrix} S^{-1}\left(\theta_1 \Delta t \frac{\partial \mathbf{R}_h^{(1)}}{\partial \mathbf{w}_h^{n,[k-1],l}} - \frac{\theta_2 \Delta t^2}{2}\left(\frac{\partial \mathbf{R}_h^{(2)}}{\partial \mathbf{w}_h^{n,[k-1],l}} + \left(\frac{\partial \mathbf{R}_h^{(1)}}{\partial \mathbf{w}_h^{n,[k-1],l}}\right)^2\right)\right) & 0 \\ \frac{\partial \mathbf{R}_h^{(1)}}{\partial \mathbf{w}_h^{n,[k],l}} S^{-1}\left(\theta_1 \Delta t \frac{\partial \mathbf{R}_h^{(1)}}{\partial \mathbf{w}_h^{n,[k-1],l}} - \frac{\theta_2 \Delta t^2}{2}\left(\frac{\partial \mathbf{R}_h^{(2)}}{\partial \mathbf{w}_h^{n,[k-1],l}} + \left(\frac{\partial \mathbf{R}_h^{(1)}}{\partial \mathbf{w}_h^{n,[k-1],l}}\right)^2\right)\right) & 0 \end{pmatrix} \mathcal{E}_{\mathrm{r}'}^{[k-1],l} \\ & + \sum_{i=1}^{s} \begin{pmatrix} S^{-1}\left(\Delta t B_{li}^{(1)} \frac{\partial \mathbf{R}_h^{(1)}}{\partial \mathbf{w}_h^{n,[k-1],i}} + \Delta t^2 B_{li}^{(2)}\left(\frac{\partial \mathbf{R}_h^{(2)}}{\partial \mathbf{w}_h^{n,[k-1],i}} + \left(\frac{\partial \mathbf{R}_h^{(1)}}{\partial \mathbf{w}_h^{n,[k-1],i}}\right)^2\right)\right) & 0 \\ \frac{\partial \mathbf{R}_h^{(1)}}{\partial \mathbf{w}_h^{n,[k],l}} S^{-1}\left(\Delta t B_{li}^{(1)} \frac{\partial \mathbf{R}_h^{(1)}}{\partial \mathbf{w}_h^{n,[k-1],i}} + \Delta t^2 B_{li}^{(2)}\left(\frac{\partial \mathbf{R}_h^{(2)}}{\partial \mathbf{w}_h^{n,[k-1],i}} + \left(\frac{\partial \mathbf{R}_h^{(1)}}{\partial \mathbf{w}_h^{n,[k-1],i}}\right)^2\right)\right) & 0 \end{pmatrix} \mathcal{E}_{\mathrm{r}'}^{[k-1],i},\end{aligned} \tag{27}$$

[2] Please note that this is an assumption that we make. It is not clear – in particular for large Δt or stiff equations – that these terms are small. However, to obtain guidelines for the termination of Newton's algorithm, we will from now on neglect the higher order terms.

[3] That this is true can be seen the easiest from the continuous level: There holds $\mathbf{w}_t = \mathbf{R}^{(1)}(\mathbf{w})$ due to Eq. (3). Differentiating with respect to time yields $\mathbf{w}_{tt} = \frac{\partial \mathbf{R}^{(1)}(\mathbf{w})}{\partial \mathbf{w}} \mathbf{w}_t = \frac{\partial \mathbf{R}^{(1)}(\mathbf{w})}{\partial \mathbf{w}} \boldsymbol{\sigma} =: \mathbf{R}^{(2)}(\mathbf{w}, \boldsymbol{\sigma})$. From this definition, the identity follows in a straightforward way. The same is true for the DG discretization, yet, it is more cumbersome (but not more difficult) to show this, departing from the weak formulations in (13) and (15).

with the Schur complement corresponding to the lower right block given by

$$S := \left(\mathrm{Id} - \theta_1 \Delta t \frac{\partial \mathbf{R}_h^{(1)}}{\partial \mathbf{w}_h^{n,[k],l}} + \frac{\theta_2 \Delta t^2}{2} \left(\frac{\partial \mathbf{R}_h^{(2)}}{\partial \mathbf{w}_h^{n,[k],l}} + \left(\frac{\partial \mathbf{R}_h^{(1)}}{\partial \mathbf{w}_h^{n,[k],l}} \right)^2 \right) \right).$$

We now consider the limits of Eq. (27) and start with $\Delta t \to 0$, i.e.

$$\begin{aligned} \mathcal{E}_{\mathrm{r}}^{[k],l} \doteq \Delta \boldsymbol{X}_{\mathrm{r}+1} - & \begin{pmatrix} \left(\mathrm{Id} + O(\Delta t) + O(\Delta t^2)\right)^{-1} \left(O(\Delta t) + O(\Delta t^2)\right) & 0 \\ \left(\mathrm{Id} + O(\Delta t) + O(\Delta t^2)\right)^{-1} \left(O(\Delta t) + O(\Delta t^2)\right) & 0 \end{pmatrix} \mathcal{E}_{\mathrm{r}'}^{[k-1],l} \\ & + \sum_{i=1}^{s} \begin{pmatrix} \left(\mathrm{Id} + O(\Delta t) + O(\Delta t^2)\right)^{-1} \left(O(\Delta t) + O(\Delta t^2)\right) & 0 \\ \left(\mathrm{Id} + O(\Delta t) + O(\Delta t^2)\right)^{-1} \left(O(\Delta t) + O(\Delta t^2)\right) & 0 \end{pmatrix} \mathcal{E}_{\mathrm{r}'}^{[k-1],i} \to \Delta \boldsymbol{X}_{\mathrm{r}+1}, \quad \Delta t \to 0. \end{aligned}$$

We find that for vanishing Δt, the Newton errors introduced by previous stages and correction steps do not play a role and the error is directly given by the next Newton increment. The limit $\Delta t \to \infty$ is more difficult to obtain. We start by considering the $O(\Delta t^2)$ terms

$$\begin{aligned} \lim_{\Delta t \to \infty} \mathcal{E}_{\mathrm{r}}^{[k],l} \doteq \Delta \boldsymbol{X}_{\mathrm{r}+1} - & \begin{pmatrix} \left(\frac{\partial \mathbf{R}_h^{(2)}}{\partial \mathbf{w}_h^{n,[k],l}} + \left(\frac{\partial \mathbf{R}_h^{(1)}}{\partial \mathbf{w}_h^{n,[k],l}} \right)^2 \right)^{-1} \left(\frac{\partial \mathbf{R}_h^{(2)}}{\partial \mathbf{w}_h^{n,[k-1],l}} + \left(\frac{\partial \mathbf{R}_h^{(1)}}{\partial \mathbf{w}_h^{n,[k-1],l}} \right)^2 \right) & 0 \\ \frac{\partial \mathbf{R}_h^{(1)}}{\partial \mathbf{w}_h^{n,[k],l}} \left(\frac{\partial \mathbf{R}_h^{(2)}}{\partial \mathbf{w}_h^{n,[k],l}} + \left(\frac{\partial \mathbf{R}_h^{(1)}}{\partial \mathbf{w}_h^{n,[k],l}} \right)^2 \right)^{-1} \left(\frac{\partial \mathbf{R}_h^{(2)}}{\partial \mathbf{w}_h^{n,[k-1],l}} + \left(\frac{\partial \mathbf{R}_h^{(1)}}{\partial \mathbf{w}_h^{n,[k-1],l}} \right)^2 \right) & 0 \end{pmatrix} \mathcal{E}_{\mathrm{r}'}^{[k-1],l} \\ & + \sum_{i=1}^{s} \begin{pmatrix} \frac{2B_{li}^{(2)}}{\theta_2} \left(\frac{\partial \mathbf{R}_h^{(2)}}{\partial \mathbf{w}_h^{n,[k],l}} + \left(\frac{\partial \mathbf{R}_h^{(1)}}{\partial \mathbf{w}_h^{n,[k],l}} \right)^2 \right)^{-1} \left(\frac{\partial \mathbf{R}_h^{(2)}}{\partial \mathbf{w}_h^{n,[k-1],i}} + \left(\frac{\partial \mathbf{R}_h^{(1)}}{\partial \mathbf{w}_h^{n,[k-1],i}} \right)^2 \right) & 0 \\ \frac{2B_{li}^{(2)}}{\theta_2} \frac{\partial \mathbf{R}_h^{(1)}}{\partial \mathbf{w}_h^{n,[k],l}} \left(\frac{\partial \mathbf{R}_h^{(2)}}{\partial \mathbf{w}_h^{n,[k],l}} + \left(\frac{\partial \mathbf{R}_h^{(1)}}{\partial \mathbf{w}_h^{n,[k],l}} \right)^2 \right)^{-1} \left(\frac{\partial \mathbf{R}_h^{(2)}}{\partial \mathbf{w}_h^{n,[k-1],i}} + \left(\frac{\partial \mathbf{R}_h^{(1)}}{\partial \mathbf{w}_h^{n,[k-1],i}} \right)^2 \right) & 0 \end{pmatrix} \mathcal{E}_{\mathrm{r}'}^{[k-1],i}. \end{aligned} \tag{28}$$

Remark 8. *The above equation (28) is highly nonlinear, yet, it has an interesting structure. For* linear *equations, where the quantities* $\frac{\partial \mathbf{R}_h^{(2)}}{\partial \mathbf{w}_h^{n,[k],l}}$ *and* $\frac{\partial \mathbf{R}_h^{(1)}}{\partial \mathbf{w}_h^{n,[k],l}}$ *are constant, there holds* $\left(\frac{\partial \mathbf{R}_h^{(2)}}{\partial \mathbf{w}_h^{n,[k],l}} + \left(\frac{\partial \mathbf{R}_h^{(1)}}{\partial \mathbf{w}_h^{n,[k],l}} \right)^2 \right)^{-1} \left(\frac{\partial \mathbf{R}_h^{(2)}}{\partial \mathbf{w}_h^{n,[k-1],l}} + \left(\frac{\partial \mathbf{R}_h^{(1)}}{\partial \mathbf{w}_h^{n,[k-1],l}} \right)^2 \right) = \mathrm{Id}$*, and the equations reduce to*

$$\lim_{\Delta t \to \infty} \mathcal{E}_{\mathrm{r}}^{[k],l} \doteq \Delta \boldsymbol{X}_{\mathrm{r}+1} - \begin{pmatrix} \mathrm{Id} & 0 \\ \frac{\partial \mathbf{R}_h^{(1)}}{\partial \mathbf{w}_h^{n,[k],l}} & 0 \end{pmatrix} \mathcal{E}_{\mathrm{r}'}^{[k-1],l} + \sum_{i=1}^{s} \begin{pmatrix} \frac{2B_{li}^{(2)}}{\theta_2} \mathrm{Id} & 0 \\ \frac{2B_{li}^{(2)}}{\theta_2} \frac{\partial \mathbf{R}_h^{(1)}}{\partial \mathbf{w}_h^{n,[k],i}} & 0 \end{pmatrix} \mathcal{E}_{\mathrm{r}'}^{[k-1],i}.$$

This automatically leads to the estimate

$$\|\mathcal{E}_{\mathrm{r},w}^{[k],l}\|_2 \lesssim \|\Delta \boldsymbol{X}_{\mathrm{r}+1,w}\|_2 + \|\mathcal{E}_{\mathrm{r}',w}^{[k-1],l}\|_2 + \sum_{i=2}^{s} \frac{2|B_{li}^{(2)}|}{\theta_2} \|\mathcal{E}_{\mathrm{r}',w}^{[k-1],i}\|_2, \quad \textit{for } \Delta t \to \infty, \tag{29}$$

where by $\mathcal{E}_{\mathrm{r},w}^{[k],l}$*, we denote the component of* $\mathcal{E}_{\mathrm{r}}^{[k],l}$ *corresponding to the degrees of freedom of* $\mathbf{w}_h^{n,[k],l}$*.* $\mathcal{E}_{\mathrm{r}'}^{[k-1],1} = 0$ *due to the fact that the first stage is trivial to compute and does not need a Newton iteration. Please note that a similar computation is, to our knowledge, not possible for arbitrary nonlinear equations, in particular not for the compressible Navier-Stokes equations. We will, however, use (29) as a heuristic basis for our error estimation procedure.*

Inspired by the behavior of the linear algorithm, we consider the following error estimate for small and large Δt:

$$\begin{aligned} \|\mathcal{E}_{\mathrm{r},w}^{[k],l}\|_2 &\approx \|\Delta \boldsymbol{X}_{\mathrm{r}+1,w}\|_2, && \text{for } \Delta t \to 0, \\ \|\mathcal{E}_{\mathrm{r},w}^{[k],l}\|_2 &\approx \|\Delta \boldsymbol{X}_{\mathrm{r}+1,w}\|_2 + C_l \|\mathcal{E}_{\mathrm{r}',w}^{[k-1],l}\|_2 + \sum_{i=2}^{s} C_i \frac{2|B_{li}^{(2)}|}{\theta_2} \|\mathcal{E}_{\mathrm{r}',w}^{[k-1],i}\|_2, && \text{for } \Delta t \to \infty. \end{aligned} \tag{30}$$

Here, C_i are user-defined constants, which, later, will reduce into one global constant. Please note that we also use this form in case of the modified arguments for the correction steps $k > 1$ (see Eq. (9)).

Remark 9. *Choosing the constants C_i basically means that we* assume *that terms of form*

$$\left\| \left(\frac{\partial \mathbf{R}_h^{(2)}}{\partial \mathbf{w}_h^{n,[k],l}} + \left(\frac{\partial \mathbf{R}_h^{(1)}}{\partial \mathbf{w}_h^{n,[k],l}} \right)^2 \right)^{-1} \left(\frac{\partial \mathbf{R}_h^{(2)}}{\partial \mathbf{w}_h^{n,[k-1],l}} + \left(\frac{\partial \mathbf{R}_h^{(1)}}{\partial \mathbf{w}_h^{n,[k-1],l}} \right)^2 \right) \right\|$$

are bounded.

4.1.2. Newton Convergence Criterion

The findings in Eq. (30) still depend on the various stages, which make the error estimate even more tedious to evaluate than it already is. In the sequel, we do therefore assume that the stage-error is constant within a timestep, and define

$$\|\mathcal{E}_{\mathtt{r},w}^{[k]}\|_2 := \|\mathcal{E}_{\mathtt{r},w}^{[k],s}\|_2, \qquad \text{and} \qquad \|\mathcal{E}_t^{n,[k]}\|_2 := \|\mathcal{E}_t^{n,[k],s}\|_2, \tag{31}$$

so only the last stage is taken into account. The desired goal of the adaptive Newton strategy is that the errors introduced by not solving the non-linear equation (16) exactly are smaller than the errors introduced by the timestepping procedure itself. In formulae, this means

$$\|\mathcal{E}_{\mathtt{r},w}^{[k]}\|_2 \leq \eta \|\mathcal{E}_t^{n,[k]}\|_2, \quad \text{for} \quad k = 0, \dots, k_{\max}, \tag{32}$$

where we have introduced some safety factor $0 < \eta < 1$. η is testcase-dependent, and will be explicitly stated for each numerical result.

Again, this is in good agreement with our numerical experience. Subsequently, based on the findings in Eq. (30) and definitions in Eq. (31), we assume that the Newton error $\|\mathcal{E}_{\mathtt{r},w}^{[k]}\|_2$ can be written as

$$\|\mathcal{E}_{\mathtt{r},w}^{[k]}\|_2 = \|\Delta \boldsymbol{X}_{\mathtt{r}+1,w}^{[k]}\|_2 + C \cdot \|\mathcal{E}_{\mathtt{r}',w}^{[k-1]}\|_2, \tag{33}$$

for a constant C. If we define $\|\mathcal{E}_{\mathtt{r}',w}^{[-1]}\|_2 = 0$, this is valid for all $k \geq 0$. Please note that $\mathtt{r}'$ is a generic constant, as the amount of Newton steps in correction k can be different from the ones in correction $k-1$.

Remark 10. *For $\Delta t \to 0$, there holds $C = 0$, see (30). For all the numerical experiments we made, C was always in the order of one, never exceeding five. In the algorithm itself, it is treated as a user-supplied constant.*

We can recursively unfold formula (33) from $k = k_{\max}$ to obtain

$$\|\mathcal{E}_{\mathtt{r},w}^{[k_{\max}]}\|_2 = \sum_{k=0}^{k_{\max}} C^{k_{\max}-k} \|\Delta \boldsymbol{X}_{\mathtt{r}'+1,w}^{[k]}\|_2.$$

Please note again that $\mathtt{r}'$ is a generic amount of steps and can change from one correction to the other. Under these preliminaries, the inequality to be fulfilled for $k_{\max}$ is given by

$$\|\mathcal{E}_{\mathtt{r},w}^{[k_{\max}]}\|_2 = \sum_{k=0}^{k_{\max}} C^{k_{\max}-k} \|\Delta \boldsymbol{X}_{\mathtt{r}'+1,w}^{[k]}\|_2 \overset{!}{\leq} \eta \|\mathcal{E}_t^{n,[k_{\max}]}\|_2. \tag{34}$$

The scaling with $C^{k_{\max}-k}$ in (34) motivates the following heuristic choice for the Newton increment:

$$\begin{aligned} \|\Delta \boldsymbol{X}_{\mathtt{r}+1,w}^{[0]}\|_2 &\leq \eta \min\left(C_1^{k_{\max}} \|\mathcal{E}_t^{n,[k_{\max}]}\|_2, \|\mathcal{E}_t^{n,[0]}\|_2 \right), \\ \|\Delta \boldsymbol{X}_{\mathtt{r}+1,w}^{[k]}\|_2 &\leq \eta \min\left(C_2 C_1^{k_{\max}-k} \|\mathcal{E}_t^{n,[k_{\max}]}\|_2, \|\mathcal{E}_t^{n,[k]}\|_2 \right), \quad \text{for} \quad 1 \leq k \leq k_{\max}. \end{aligned} \tag{35}$$

Here, $C_1 \sim C^{-1}$ and C_2 are user-defined input parameters to the code. We have included C_2 into this heuristic as we have found that the quality of the predictor typically has a larger influence on the quality than the correction steps; typically, C_2 is taken to be smaller than one. We have found numerically that the $k_{\max}$-dependent choices $C_1 := \frac{k_{\max}-1}{C k_{\max}}$ and $C_2 := 1 - \frac{k_{\max}-1}{k_{\max}}$ seem to work very well; we will stick to this definition in the sequel.

Remark 11. *Eq. (35) shows that for the evaluation of the convergence criterion of Newton's method for the current iterate $[k]$, one requires $||\mathcal{E}_t^{n,[k]}||_2$ and $||\mathcal{E}_t^{n,[k_{\max}]}||_2$. While the former can be evaluated independently on each processor, the latter requires some special treatment. The information about the solution $\mathbf{w}_h^{n,[k_{\max}],s}$ is only available on the processor with rank zero (#0), see Fig. 2. Hence, the temporal error estimate $||\mathcal{E}_t^{n,[k_{\max}]}||_2$ can only be evaluated on this processor. The error information is then communicated to the other processors along the standard information propagation path, i.e. along the diagonal. This leads to a delay of the adaptive procedure's start on the different processors, meaning that, instead of $||\mathcal{E}_t^{n,[k_{\max}]}||_2$, the quantity $||\mathcal{E}_t^{n^*,[k_{\max}]}||_2$ with $n^* < n$ is evaluated. This delay is indicated with the gray shaded area in Fig. 2. As it is crucial to keep the parallel-in-time structure of the scheme, we need to hence make the important assumption that the errors $||\mathcal{E}_t^{n,[k_{\max}]}||_2$ are only changing mildly with n, so that $||\mathcal{E}_t^{n^*,[k_{\max}]}||_2$ is indeed a good approximation for $||\mathcal{E}_t^{n,[k_{\max}]}||_2$. Note that $n - n^*$ cannot exceed $k_{\max}$ due to construction.*

Remark 12. *A similar approach as the one outlined above in Sec. 4.1.1 and Sec. 4.1.2 can be done for standard diagonally implicit Runge Kutta methods. In Appendix C, we briefly introduce a similar adaptive Newton strategy for ESDIRK methods, which will then later be used for efficiency comparisons in Sec. 5.4.*

4.1.3. Newton Error Extrapolation

Considering Eq. (34) and Eq. (35), one can see that we have found a condition for $||\mathcal{E}_{\mathtt{r},w}^{[k]}||_2$ via the Newton increment $||\Delta \boldsymbol{X}_{\mathtt{r}+1,w}^{[k]}||_2$. That means that in order to obtain a condition for the error at iterate $\mathtt{r}$ one has to calculate or estimate the norm of the $(\mathtt{r}+1)$-th Newton increment. As the computation of $\mathcal{E}_{\mathtt{r},w}^{[k]}$ is time-consuming, we propose to use an extrapolation procedure to obtain an estimate for $||\Delta \boldsymbol{X}_{\mathtt{r}+1}||_2$, which is then used within the estimate for $||\mathcal{E}_{\mathtt{r},w}||_2$, see Eq. (34). We either use a linear extrapolation procedure,

$$\begin{aligned} ||\Delta \boldsymbol{X}_{3,w}^{[k]}||_2 &\approx \frac{||\Delta \boldsymbol{X}_{2,w}^{[k]}||_2^2}{||\Delta \boldsymbol{X}_{1,w}^{[k]}||_2}, \\ ||\Delta \boldsymbol{X}_{\mathtt{r}+1,w}^{[k]}||_2 &\approx \frac{||\Delta \boldsymbol{X}_{\mathtt{r}-2,w}^{[k]}||_2 ||\Delta \boldsymbol{X}_{\mathtt{r}-1,w}^{[k]}||_2 + ||\Delta \boldsymbol{X}_{\mathtt{r}-1,w}^{[k]}||_2 ||\Delta \boldsymbol{X}_{\mathtt{r},w}^{[k]}||_2}{||\Delta \boldsymbol{X}_{\mathtt{r}-2,w}^{[k]}||_2^2 + ||\Delta \boldsymbol{X}_{\mathtt{r}-1,w}^{[k]}||_2^2} ||\Delta \boldsymbol{X}_{\mathtt{r},w}^{[k]}||_2, \quad \text{for } \mathtt{r} \geq 3, \end{aligned} \tag{36}$$

or a quadratic extrapolation procedure

$$\begin{aligned} ||\Delta \boldsymbol{X}_{3,w}^{[k]}||_2 &\approx \frac{||\Delta \boldsymbol{X}_{2,w}^{[k]}||_2^3}{||\Delta \boldsymbol{X}_{1,w}^{[k]}||_2^2}, \\ ||\Delta \boldsymbol{X}_{\mathtt{r}+1,w}^{[k]}||_2 &\approx \frac{||\Delta \boldsymbol{X}_{\mathtt{r}-2,w}^{[k]}||_2^2 ||\Delta \boldsymbol{X}_{\mathtt{r}-1,w}^{[k]}||_2 + ||\Delta \boldsymbol{X}_{\mathtt{r}-1,w}^{[k]}||_2^2 ||\Delta \boldsymbol{X}_{\mathtt{r},w}^{[k]}||_2}{||\Delta \boldsymbol{X}_{\mathtt{r}-2,w}^{[k]}||_2^4 + ||\Delta \boldsymbol{X}_{\mathtt{r}-1,w}^{[k]}||_2^4} ||\Delta \boldsymbol{X}_{\mathtt{r},w}^{[k]}||_2^2, \quad \text{for } \mathtt{r} \geq 3, \end{aligned} \tag{37}$$

that considers at maximum the previous three calculated Newton increments. Note that for $\mathtt{r} \geq 3$, a least-squares approximation of the constants is used in both cases. The linear extrapolation procedure has to be applied if a fixed coarse relative tolerance for the GMRES solver is applied [32]. If the relative tolerances converge to zero fast enough, quadratic convergence of Newton's method can be expected. One opportunity to achieve this is to use the *Eisenstat-Walker* procedure introduced in [34].

4.1.4. Application of Adaptive Newton Procedure

In this section, all ingredients of the adaptive Newton strategy are combined and validated. The temporal error is estimated according to Eq. (22) and is evaluated only once after the first timestep. This error estimate is then used to determine Newton's convergence condition via Eq. (35), where the actual Newton increment is obtained via the extrapolation procedure, introduced in Sec. 4.1.3. The constants C and η are chosen to be 0.5 and 0.1, respectively. We either use a fixed relative GMRES tolerance of $\varepsilon_{\text{GMRES}} = 5 \cdot 10^{-2}$, which corresponds to the same criterion as used in [28], with the linear extrapolation procedure of Newton's increment given in Eq. (36). Alternatively, we apply the adaptive Eisenstat-Walker procedure [34] to determine the relative GMRES tolerances and utilize the quadratic extrapolation of Newton's increment given by Eq. (37).

We choose the same setup used in Sec. 3.3 for the Navier-Stokes equations with the initial conditions given by Eq. (25) with $\varepsilon = 1$, $\mathcal{N}_p = 7$ and $\mathcal{N}_E = 32^2$. The final time is set to $T_{end} = 1.0$. In order to evaluate the adaptive Newton procedure, simulations with fixed relative tolerances are used. Additionally, an absolute convergence criterion

$$\|\Delta \boldsymbol{X}^{[k]}_{\mathrm{r}+1,w}\|_2 \leq 10^{-14} \frac{\sqrt{\mathrm{nDOF}}}{\min(1,\varepsilon)}, \tag{38}$$

is used for all cases including the simulations with adaptive Newton strategy, where nDOF describes the total number of spatial degrees of freedom and ε is the stiffness parameter of the considered physical equations.[4] As initial guess for Newton's method, i.e. $\boldsymbol{X}^{[k]}_0$, we choose the solution of the second order explicit Taylor step for the predictor and the corresponding stage value of the previous iterate $[k-1]$ for the correction steps. We start counting Newton's iterations after $k_{\max}$ timesteps (compare Fig. 2) to ensure that the full capabilities of the adaptive strategy are evaluated. Solution steps that cannot use the adaptive Newton strategy (gray shaded area in Fig. 2) utilize a relative convergence criterion of $\varepsilon_{\mathrm{Newton,rel}} = 10^{-10}$.

The resulting errors and average Newton iterations per stage of this series of simulations are visualized in Fig. 4. Missing points for the fixed tolerance $\varepsilon_{\mathrm{Newton}} = 10^{-2}$ indicate a diverging solution. Fig. 4, leftmost column, displays the numerical errors made for the adaptive choice of the Newton tolerance and several fixed Newton tolerances. It is only for HBPC(6, 5) with the finest timestep size that the error curves associated to the adaptive strategy and the finest tolerance deviate slightly; in all the other cases, the obtained 'adaptive' error is equal to the one with the finest fixed Newton tolerance. This means that the adaptive strategy is succesful in the sense that it does not underresolve the algebraic systems of equations. The adaptive strategy is also successful w.r.t. the reduction of the required Newton and GMRES iterations, see Fig. 4 (middle, right). One can see that with the adaptive strategy, always at least two Newton iterations are performed. This is most likely caused by the fact that by choosing the Newton increment as convergence condition, we "lag" one Newton iteration, and the extrapolation procedure can only be applied after two Newton increments have been calculated.

Repetition of Introductory Example. We now repeat the illustrative example shown in the introduction (see Fig. 1) with all the ingredients introduced in the previous sections. These are the parallelizable timestepping procedure described in Alg. 1, an improved initial Newton guess given by Eq. (21) and the adaptive Newton strategy with linear error extrapolation given by Eq. (35) and Eq. (36). Again, the constants C and η are chosen to be 0.5 and 0.1, respectively. Similar as for the introductory example, we use $\varepsilon_{\mathrm{GMRES}} = 10^{-3}$ for the linear solver.

The total number of GMRES iterations and the normalized GMRES iterations are shown in Fig. 5. Compared to the simulation with the serial algorithm, using a fixed relative tolerance for the residual of Newton's method of $\varepsilon_{\mathrm{Newton}} = 10^{-10}$, one can clearly see a much stronger dependency of the required absolute number of iterations on the chosen timestep size. Moreover, the absolute values are much smaller than the ones reported in Fig. 1. Considering the normalized GMRES iterations, one can see that the relative cost of the predictor has been reduced and the costs of the correction steps have been homogenized. Moreover, the curves for the different timestep sizes coincide very well. The only exception from this behavior are the highest iterates, i.e. $k = 6, 7$, for the smallest timestep for which a reduced number of iterations is reported. This drop in iterations is most likely caused by hitting the absolute tolerance (see Eq. (38)).

5. Parallel Performance

In this section, we investigate the parallel performance of the novel scheme. For that purpose, we first investigate the performance of the spatial parallelization. Next, we combine the spatial parallelization with the novel parallelization in time.

All simulations were performed on the *VSC Genius* cluster using up to 36 nodes. Each node has 192 GB RAM and consists of 18 cores, each equipped with 2 Xeon Gold 6240 CPUs@2.6 GHz (Cascadelake) processors. The

[4] $\sqrt{\mathrm{nDOF}}$ is the Euclidean norm of the vector of size nDOF with each element being one. This quantity serves hence as a reference value – the larger nDOF, the lesser one can expect that fine target accuracies can be reached. The ε in the denominator is a safety factor to account for the stiffness of the problem.

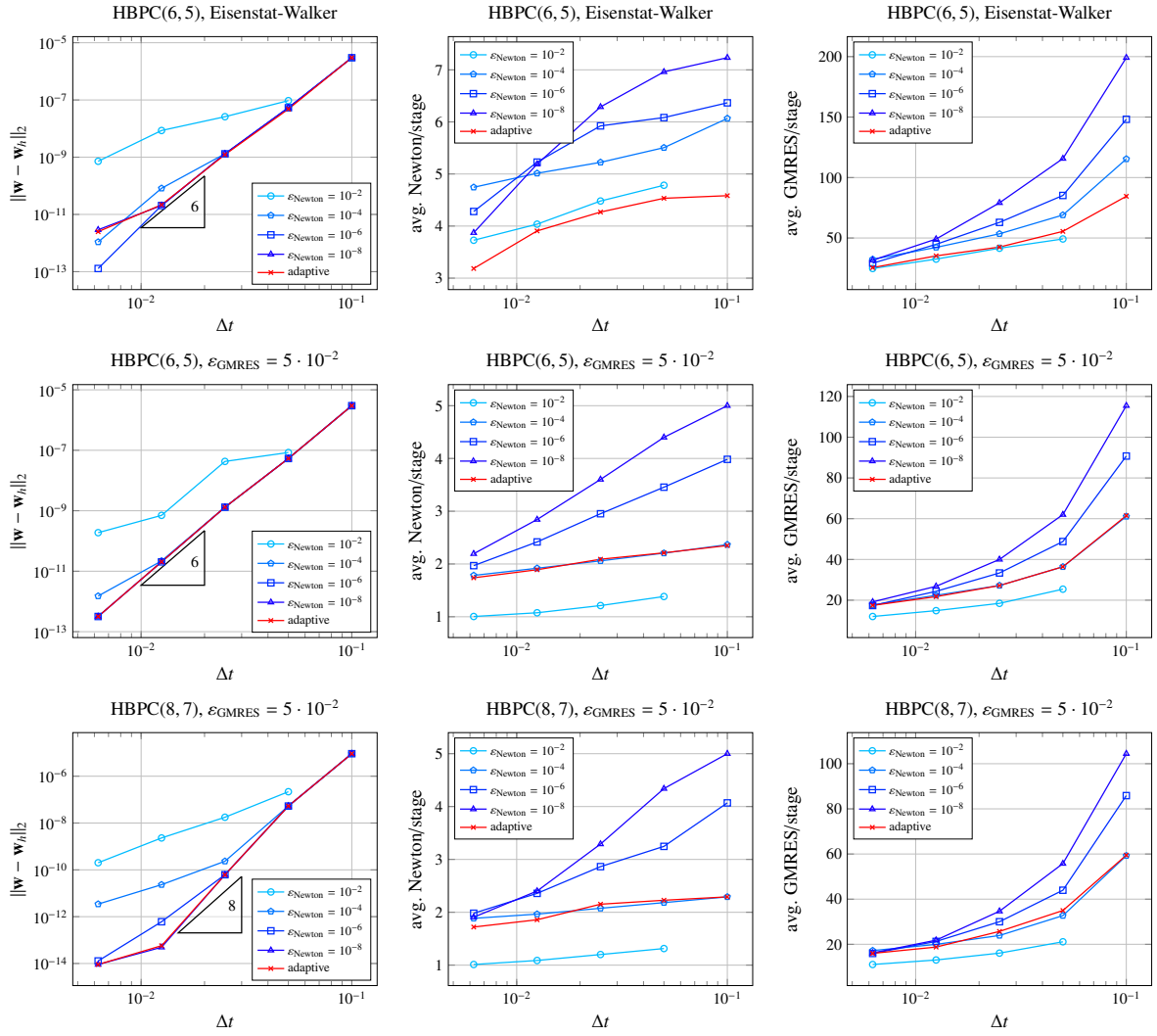

Figure 4. Resulting L_2-error (left), average Newton iterations per stage (middle) and average GMRES iterations per stage (right) for HBPC(6, 5) with Eisenstat-Walker GMRES tolerance (top), HBPC(6, 5) with fixed GMRES tolerance (middle) and HBPC(8, 7) method with fixed GMRES tolerance (bottom) when choosing different convergence criteria for Newton's method. Adaptive Newton strategy is performed according to Eq. (35) with Newton increment extrapolation (Eq. (36) and Eq. (37)) and temporal error estimate according to Eq. (22).

connection between nodes is established with an Infiniband EDR network (25 GB/s bandwidth). The Fortran-written simulation code is compiled with the GCC compiler (6.4.0). It uses OpenMPI (v3.1.1) for the implementation of processor communication and OpenBLAS (v0.3.17) for the efficient implementation of the preconditioner's matrix inversion via LU-decomposition and matrix-matrix/matrix-vector multiplications. The single-derivative base-line code in which the novel method is implemented into is the open source code FLEXI[5], see also [35].

5.1. Parallel Performance of Spatial Parallelization

The spatial parallelization is based on a domain decomposition via a space-filling curve. Each processor is responsible for its own set of elements and information between different processors is done via the surfaces of adjacent elements. A detailed overview on the parallelization strategy, including the communication pattern and an evaluation of the parallel efficiency with an explicit timestepping procedure can be found in [35].

[5] www.flexi-project.org, GNU GPL v3.0

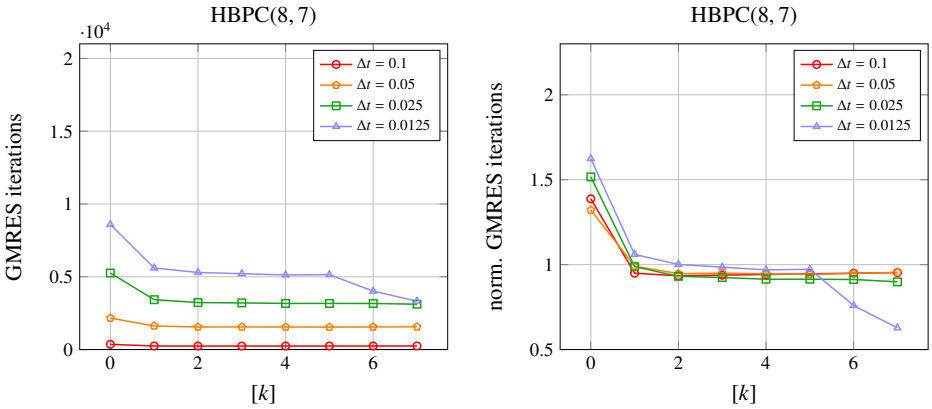

Figure 5. Repetition of the numerical experiment from Sec. 1, Fig. 1, but with parallel-in-time algorithm (Alg. 1), adaptive Newton strategy (Eq. (35) and Eq. (36)) and improved initial Newton guess (Eq. (21)). Cumulated number (left) and normalized (right) GMRES iterations per prediction/correction step are shown. Normalization of the GMRES iterations per $[k]$ has been done with the mean GMRES iterations per timestep.

We benchmark the spatial parallel performance of the implicit two-derivative method with two different settings: a two dimensional setup with $\mathcal{N}_p = 7$ and a three dimensional setup with $\mathcal{N}_p = 5$. For both cases we use Eq. (25) as initial condition, where $\mathbf{v} = (0.25, 0.25, 0.25)^T$ for the 3d and $\mathbf{v} = (0.3, 0.3)^T$ for the 2d-setup. The temporal discretization is done with the implicit two-derivative Taylor method and $\Delta t = 0.1$. We perform $\mathcal{N}_T = 10$ timesteps and use the relative tolerances $\varepsilon_{\text{Newton,rel}} = 10^{-3}$ and $\varepsilon_{\text{GMRES}} = 5 \cdot 10^{-2}$. The preconditioner is built once, and kept fixed for the whole simulation. Similar as it has been done in [1], we neglect the Hessian contribution when solving the linear system. A measure for the computational cost is the performance index (PID)

$$\text{PID} = \frac{\mathcal{T} \cdot \#\text{processors}}{\text{nDOF} \cdot \mathcal{N}_T \cdot (s-1)},$$

where $\mathcal{T}$ denotes the wallclocktime. PID measures the average time *per degree of freedom* that is necessary to perform one implicit stage of a single timestep. Note that, differently than for an explicit scheme, the absolute value of the PID does not transfer to different settings as it highly depends on the chosen test setup, i.e. on the implicit parameters and initial conditions[6]. It can hence serve only as a relative measure. For the investigation of the parallel efficiency, a series of simulations on different meshes with varying number of processors is performed. For the 2d simulations, the smallest mesh has 12×12 elements to discretize the domain $\Omega = [-1, 1]^2$. Larger grids are obtained by doubling the number of elements and extending the domain Ω accordingly. The domain is extended to account for the fact that the CFL number should stay constant (note that $\Delta t = 0.1$ in this testcase), as otherwise, the behavior of the implicit algorithm changes drastically. The largest mesh has 192×192 elements for the domain $\Omega = [-128, 128]^2$. The meshes for the 3d simulations range from $6 \times 6 \times 3$ elements ($\Omega = [-2, 2] \times [-2, 2] \times [-1, 1]$) up to $24 \times 12 \times 12$ elements ($\Omega = [-8, 8] \times [-4, 4] \times [-4, 4]$). The lowest load, i.e. the lowest number of DOF per processor, is either obtained by using 1152 processors, or having 2 or 3 elements per processor for the 2d and 3d case, respectively. Each simulation is performed three times. The average, the minimum and the maximum PID of the simulations are reported in Fig. 6 (left). From those values, the weak and strong parallel efficiency as well as the speedup can be derived, see Fig. 6. Note that the minimum number of processors is 36, corresponding to one node.

One can see that the PID is a relatively constant quantity for small processor numbers. However, there is a strong increase if the load decreases below approximately 1000 DOF per processor and the number of processors increases. This increase additionally comes with an increasing variance of the measured PID. This is due to the increasing relative amount of communication and its high dependency on fluctuations of the machine's performance. Regarding the parallel efficiency, one can still observe a weak and strong efficiency of approx. 80% when using 1152 processors for the 3d simulations. For the 2d case, a significant dependency of the strong parallel efficiency on the amount of

[6] In particular, the conditioning of the nonlinear system of equations (16) plays an important role, which can be different for different test cases.

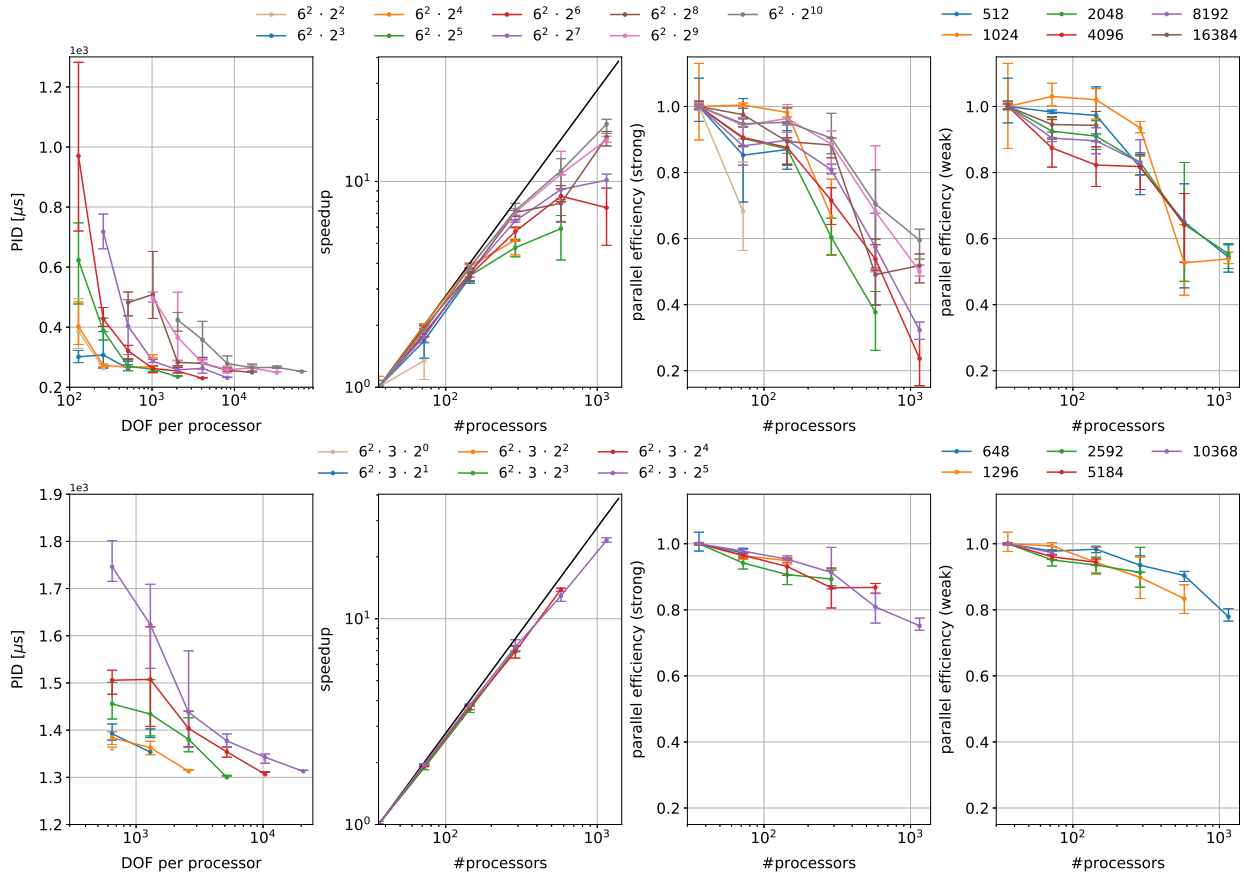

Figure 6. Performance index (left), speedup (second to left), strong parallel efficiency (second to right) and weak parallel efficiency (right) for the 2d setup with $\mathcal{N}_p = 7$ (top) and the 3d setup with $\mathcal{N}_p = 5$ (bottom). For the PID, the strong scaling and the speedup, the legend indicates the different number of elements of different meshes. For the weak scaling, different lines correspond to different loads, i.e. different number of DOF per processor.

DOF can be observed. Depending on the number of DOF, it decreases until approx. 30% to 60% when using 1152 processors. Considering the weak parallel efficiency, one can observe a decrease towards approx. 55% when using 1152 processors. The main findings from this investigation are:

- Good weak scaling on up to more than 1000 processors indicates that the code is well-suited for large-scale applications.
- One can decrease the computational load per processor almost until the finest possible granularity (one element per processor) and still observe some speedup.

The significant differences between the 2d and the 3d simulations can be explained with the different ratios between internal work and communication, especially due to the increased work for matrix-matrix/matrix-vector multiplications.

5.2. *Parallel Performance of Temporal Parallelization*

Next, we consider the parallel performance of the temporal parallelization. Setting up a fair evaluation problem for the parallel-in-time speedup is a non-trivial task [36]. One not only has to choose the problem, but also iterative procedures' parameters such that they are representative for the desired applications. We start with the same setup as used in the previous subsection with the initial conditions given by Eq. (25) with $\varepsilon = 1$, $\mathcal{N}_p = 7$ and use the adaptive Newton strategy introduced in Sec. 4. Again, the well-known values 0.5 and 0.1 are assigned to the constants C and

η, respectively. Differently to the previous subsection, we choose $\mathcal{N}_E = 24^2$ for the domain $\Omega = [-1, 1]^2$ and the final time is set to $T_{end} = 10.0$. We then perform simulations with Alg. 1 running the HBPC$(6, k_{\max})$ and the HBPC$(8, k_{\max})$ serially and in parallel using $\Delta t = \{0.1, 0.05, 0.025\}$ which corresponds to performing $\mathcal{N}_T = \{100, 200, 400\}$ timesteps. The large number of timesteps ensures that the theoretical limit of the speedup for $\mathcal{N}_T \to \infty$ given by Eq. (12) is within reach. The preconditioner is rebuilt every 10^{th} timestep and the errors of the simulations are calculated by using the result of an explicit reference simulation with a fourth order low-storage Runge-Kutta method with a very small timestep ($\Delta t \approx 3.4 \cdot 10^{-4}$).

For the serial simulations we use one node with 36 processors corresponding to a load of 1024 DOF per processor for all $k_{\max} \in \{1, 3, 5, 7, 9\}$. For the parallel simulation we use multiple nodes, e.g. the parallel HBPC$(8, 7)$ method uses 6 nodes with 36 processor each[7], resulting in 216 processors. Note that the load (i.e. DOF per processor) remains the same for all parallel-in-time and the serial simulations.

Parallel-in-Time Speedup. In Fig. 7 we report the results of this series of simulations. On the left one can see the errors of the simulations w.r.t. the required wallclocktime $\mathcal{T}$. While different colors correspond to different timestep sizes, the solid and the dashed lines indicate the parallel and the serial simulations, respectively. Points being connected by a line indicate simulations with increasing $k_{\max} \in \{1, 3, 5, 7, 9\}$. One can see that the parallel method is more efficient (i.e. has a better relation between accuracy and wallclocktime) than the serial method, which is indicated by a much steeper slope of the parallel methods. This behavior can be observed consistently for all considered timestep sizes and both, the HBPC$(6, k_{\max})$ and the HBPC$(8, k_{\max})$ method.

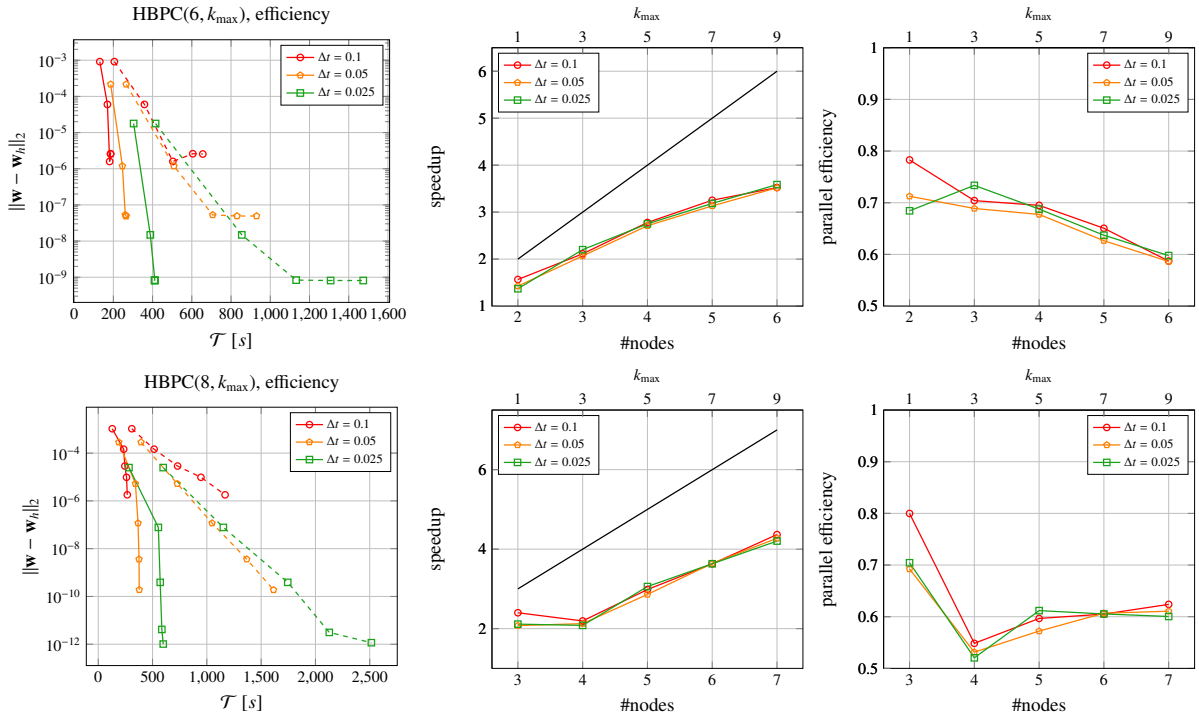

Figure 7. Efficiency of temporal parallelization for HBPC$(6, k_{\max})$ (top) and HBPC$(8, k_{\max})$ method using different timestep sizes. The left plot shows required wallclocktime $\mathcal{T}$ vs. resulting error using different $k_{\max}$ for the parallel-in-time (solid) and serial method (dashed). The speedup (middle) gives the relation between serial and parallel wallclocktime. The black line indicates perfect speedup. The right plot displays the parallel efficiency, i.e. the relation between the actual speedup and the perfect speedup.

[7] HBPC$(8, 7)$ is a method with four stages, the first stage being trivial. Hence, the splitted circles in Fig. 2 would be split into three rather than two. That means that three nodes are necessary for the handling of predictor and first corrector; and then one node each is necessary for $k = 2, 3$, $k = 4, 5$ and $k = 6, 7$. In total, this yields the six nodes used.

Calculating the ratio between the wallclocktimes of the serial and the parallel simulations, one obtains the speedup enabled by the temporal parallelization, visualized in Fig. 7 (middle). The black line indicates ideal speedup, i.e. when using 6 nodes, one expects a speedup of 6. Note that we have neglected the influence of a finite number of timesteps on the ideal speedup, see Eq. (12). Calculating the ratio between the ideal speedup and the actual achieved one, gives the parallel efficiency shown in Fig. 7 (right). Here, one can see that the parallel-in-time scheme achieves a parallel efficiency of approximately 60% when using $k_{\max} = 9$, corresponding to 6 and 7 nodes for the HBPC(6, 9) and the HBPC(8, 9) method, respectively.

A parallel efficiency of 60% on up to 7 partitions is in the same range as it has been reported for other PinT methods in literature: For solving ODEs with the implicit RIDC method, efficiencies of 90% and 69% were reported for 4 and 8 processors, respectively. For the HBPC scheme simulating ODEs, efficiencies up to 65% and 48% are measured for 4 and 18 processors. An inverted dual time stepping procedure is used in [37] and efficiencies of 95% and 45% are reported for 4 and 20 processors. Combined with additional spatial parallelization, an efficiency of 50% is obtained on 12 processors [9]. The reporting of parallel efficiencies of methods based on the parareal algorithm is difficult as the performance heavily depends on the problem to be solved [36]. Especially for hyperbolic dominated problems, the parareal algorithm can have relatively low efficiencies. In [38], fluid structure interaction problems with the incompressible Navier-Stokes equations are solved, parareal is used for the temporal parallelization and an efficiency up to 22% on 20 processors is reported. The compressible Navier-Stokes equations are solved in [8] and the authors show that, in combination with a spatial parallelization, the parareal algorithm shows efficiencies up to approx. 40% on 16 processors.

Work Distribution among the Parallel-in-Time Partitions. To obtain some insight in the parallel efficiency, we consider the work distribution among the different parallel-in-time partitions in Fig. 8. We visualize the normalized GMRES iterations performed on each partition for all the parallel-in-time simulations. One important property of the novel method can be seen from the different graphs in Fig. 8: The curves for different $k_{\max}$ and different timestep sizes are very close to each other. This indicates that the adaptive strategy manages relatively well to balance the load over the processors.

Overall, the figure shows a qualitatively similar behavior for all simulations: While the partitions being responsible for one single stage of the predictor and the first corrector have the least load, the partition being responsible for $k = \{2, 3\}$ has the highest load. For higher iterates, the load decreases again. Comparing Fig. 8 and the parallel efficiency in Fig. 7, one can see that the value of the inverse of the maximum normalized GMRES iterations over all partitions translates almost directly to the achieved parallel efficiency. This highlights the importance of the homogenization of the work distribution among the different prediction/correction steps.

5.3. Space-Time Parallel Performance

Next, we consider the parallel performance of the combined spatial and temporal parallelization. For that purpose, we again consider the initial conditions of the sine density wave given by Eq. (25). Two different representative configurations are investigated: A two dimensional problem on the domain $\Omega = [-2, 2]^2$ which is discretized with 24×24 elements using $\mathcal{N}_p = 7$, and a three dimensional problem on the domain $\Omega = [-2, 2]^3$ which is discretized with $8 \times 8 \times 9$ elements using $\mathcal{N}_p = 5$. For the 2d example the HBPC(8, 7) and for the 3d example the HBPC(6, 5) scheme is used leading to an 8^{th} order and a 6^{th} order scheme in space and time, respectively. The final time is set to $T_{end} = 5$ and the timestep is $\Delta t = 0.05$, leading to $\mathcal{N}_T = 100$ timesteps. The preconditioner is rebuilt every 10^{th} timestep and the linear solver tolerance is set to $\varepsilon_{\text{GMRES}} = 5 \cdot 10^{-2}$. C and η are set as before to 0.5 and 0.1, respectively.

In Fig. 9 the parallel speedups and the parallel efficiencies of a pure spatial and a mixed spatial/temporal parallelization are reported. Increasing the number of processors for the spatial discretization, the parallel efficiency decreases up to approx. 40% (2d) and 75% (3d). It is not possible to use more processors with the pure spatial parallelization for the considered test setups as the very right points of the red curves in Fig. 9 correspond to the finest possible granularity (i.e. one element per processor). If one wants to further reduce the required wallclocktime, spatial and temporal parallelization can be combined. One can clearly see that the temporal parallelization gives a further speedup. For the 2d simulation, one can see that the parallel efficiency can even be improved by combining spatial and temporal parallelization. Due to the very high parallel efficiency of the spatial parallelization for the 3d case, one cannot observe an increase in the parallel efficiency. However, due to the possibility to use more processors, a speedup can still be achieved.

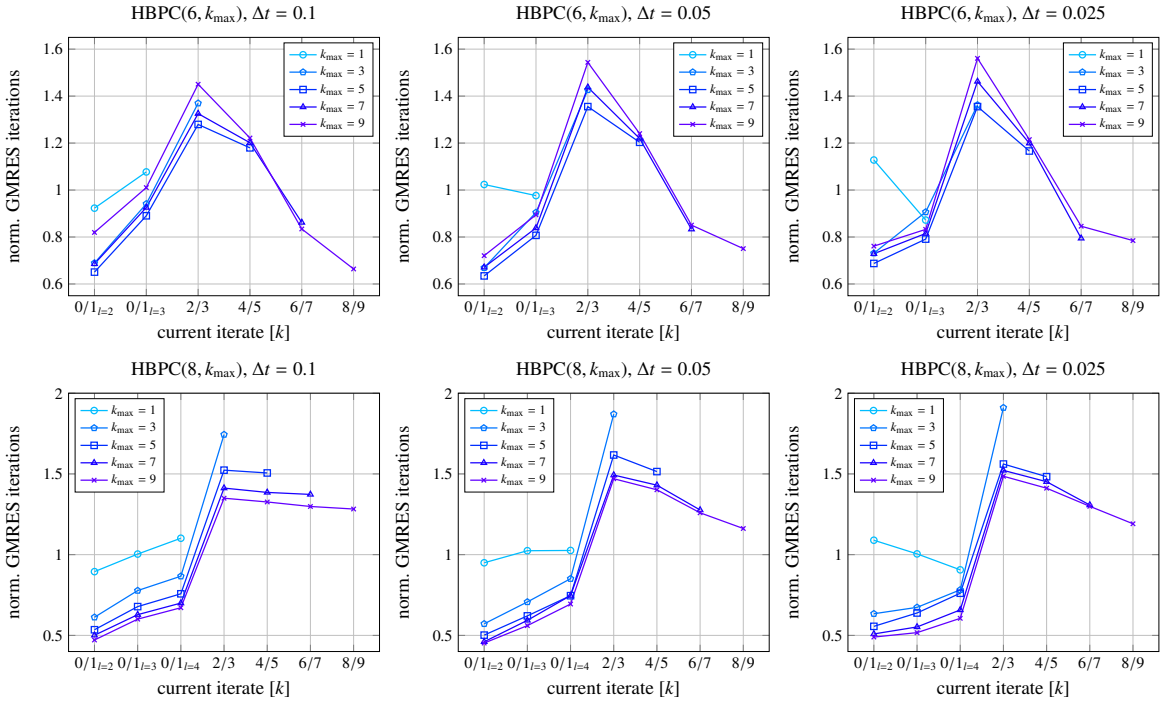

Figure 8. Normalized GMRES iterations per iterate [k] for the HBPC(6, $k_{\max}$) (top) and the HBPC(8, $k_{\max}$) (bottom) method using $\Delta t = 0.1$ (left), $\Delta t = 0.05$ (middle) and $\Delta t = 0.025$ (right). Note that the indexed iterates correspond to the different ranks of the temporal parallelization. Note that the iterates $k = 0/1$ are split up into the different implicit stages, see Fig. 2. For the normalization of the GMRES iterations, the mean GMRES iterations of all iterates k per simulation is used.

Concluding, Fig. 9 shows that the temporal parallelization gives an efficiency gain if the pure spatial parallelization has an efficiency lower than the temporal one (i.e. lower than approx. 60%). I.e. the worse the parallel efficiency of the spatial operator, the more one can benefit from the temporal parallelization. Moreover, one can see that if the spatial parallelization reaches its limit, a further wallclocktime reduction can be achieved with the temporal parallelization.

5.4. *Efficiency Comparisons*

In this final subsection, we compare the efficiency of the novel parallel-in-time two-derivative method with classical sequential-in-time single-derivative Runge-Kutta methods. We will use the 4^{th} order ARK4(3)6L[2]SA method [39, Appendix D] (abbreviated with ESDIRK4-6) and the 6^{th} order ESDIRK6(5)9L[2]SA method [40, Table 13] (abbreviated with ESDIRK6-9) as high order implicit ESDIRK methods[8]. For both, an adaptive Newton procedure that is based on the same principles as the one derived in Sec. 4 is used. Details on this can be found in Appendix C. We set $C = 0.5$ and $\eta = 0.1$.

5.4.1. *Density Sine Wave*

We start by considering the previously used example with the initial data given by Eq. (25) with $\varepsilon = 1$ and $\mathcal{N}_p = 7$ on the domain $\Omega = [-1, 1]^2$, discretized with $\mathcal{N}_E = 24 \times 24$ and with $T_{end} = 10$. We run the simulation by choosing different timesteps with the parallel-in-time HBPC(6, 5), HBPC(6, 7), HBPC(8, 7) and HBPC(8, 9) schemes. Additionally, the ESDIRK4-6 and ESDIRK6-9 are used as a reference. The simulations are performed on one node with 36 processors; for the parallel-in-time methods the respective multiples are used. The preconditioners are rebuilt every 10^{th} timestep and $\varepsilon_{\text{GMRES}} = 5 \cdot 10^{-2}$ is chosen for the linear solver. Initially, when no adaptive Newton criterion is available, we choose $\varepsilon_{\text{Newton,rel}} = 10^{-8}$. C and η are set to 0.5 and 0.1, respectively.

[8] While HBPC(8, 7) and HBPC(8, 9) are schemes of order eight, we did, despite a thorough literature study, not find an eighth-order diagonally implicit Runge-Kutta scheme, see also [25]. Hence, only fourth- and sixth-order Runge-Kutta schemes are used for comparison.

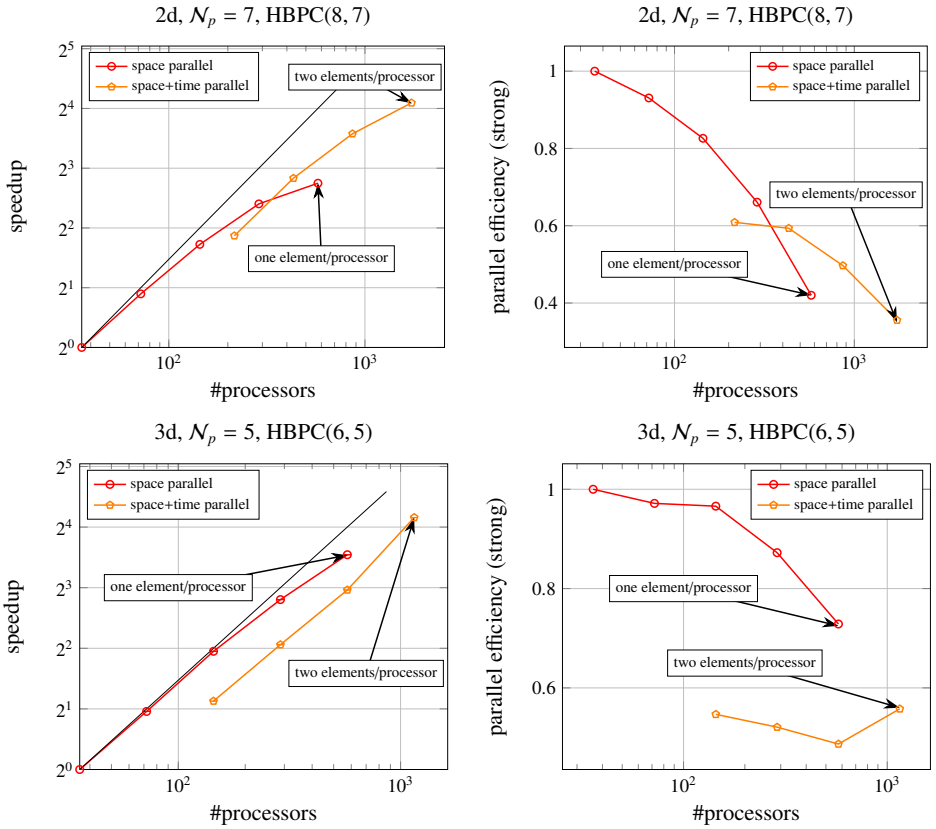

Figure 9. Parallel speedup (left) and strong parallel efficiency (right) for a pure spatial and a mixed spatial/temporal parallelization for a 2d example using $\mathcal{N}_p = 7$ and HBPC(8, 7) (top) and a 3d example using $\mathcal{N}_p = 5$ and HBPC(6, 5) (bottom). Note that the very right point of the pure spatial parallelization (red, circles) corresponds to the finest possible granularity, i.e. one element per processor. The very right point of the mixed spatial/temporal parallelization corresponds to two elements per processor.

Fig. 10 shows the temporal convergence (left) and the efficiency (right) of the different methods. One can see that all methods show the desired order of convergence. The sixth order HBPC schemes show a smaller error than the sixth order ESDIRK method. Considering the efficiency, one can see that if small errors are desirable, the higher order methods pay off. Moreover, one can see that the sixth order HBPC method is superior to the ESDIRK6-9 scheme for smaller errors.

5.4.2. *Cylinder Flow*

Next, we consider the two dimensional flow around a cylinder. Similar as it has been done by other authors, see e.g. [41, 42], we consider the aerodynamic coefficients as a quality measure. The flow parameters for the cylinder flow with diameter $D = 1$ are given by the reference Mach number $\varepsilon = 0.1$ and the Reynolds number $\mathrm{Re}_D = 200$. The initial conditions are given by the constant state

$$\rho_0 = 1, \quad \mathbf{v}_0 = (1, 0)^T, \quad p_0 = \frac{1}{\gamma},$$

and the cylindrical domain with an outer diameter of $D_\infty = 200$ is discretized using $\mathcal{N}_E = 1200$ with $\mathcal{N}_p = 5$. On the cylinder surface, wall boundary conditions are prescribed. At the outer boundary, Dirichlet boundary conditions with the constant initial state are used. A more detailed description of the used mesh is available in [5, Sec. 5.1.1]. We run the simulation with a fourth order low-storage Runge-Kutta time discretization (ERK4) [26] up to $T = 400$. At this time, the flow is fully developed and a vortex shedding has been established. In the interval from $T = 400$ to $T = 500$, we obtain $C_D \approx 1.35$, a fluctuating lift coefficient of $C_{L'} \approx 0.501$ and a Strouhal number of $\mathrm{Sr} \approx 0.197$.

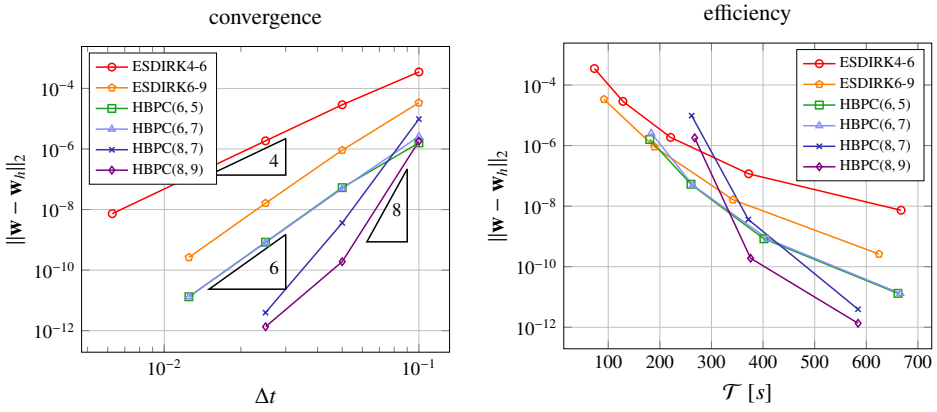

Figure 10. Temporal convergence (left) and required wallclocktime (right) to achieve a certain error for different implicit schemes for the 2d sine wave problem with $\varepsilon = 1$ using different timestep sizes.

Those aerodynamic measures agree well with data from literature, see e.g. [43, 44, 45]. Differences are most likely due to the relatively coarse spatial resolution.

To evaluate the performance of the novel scheme, we use the mean drag coefficient

$$C_D := \frac{2\bar{F}_x}{\rho_0 \|\mathbf{v}_0\|_2^2 D},$$

where $\bar{F}_x$ denotes the mean drag force at the cylinder surface as a measure for the accuracy. We restart the simulation at $T = 400$ and run it approximately for two vortex shedding periods ($T_{end} = 410$) using different timestepping methods and timestep sizes. As this is a quasi-steady flow, we estimate the temporal error only once during the adaptive Newton procedure. The aerodynamic forces are measured at the time intervals $\Delta t = 0.5$ and $\bar{F}_x$ is calculated via mean of these discrete values. A simulation with a very small explicit timestep for the ERK4 serves as a reference solution to calculate an error measure.

As we are using relatively large timesteps and $\varepsilon = 0.1$, we use Eq. (23) to estimate the temporal error and do not perform an explicit step for the initial Newton guess. Moreover, we set $C = 1$ in the adaptive Newton procedure, see Eq. (35). For the linear solver, the adaptive Eisenstat-Walker procedure is used. During the startup procedure of the adaptive Newton strategy, a relative tolerance of $\varepsilon_{\text{Newton,rel}} = 10^{-4}$ is used. The ESDIRK schemes initially use a Newton tolerance of $\varepsilon_{\text{Newton,rel}} = 10^{-6}$. We choose the safety factor $\eta = 0.1$ for the ESDIRK4-6 and the ESDIRK6-9 in Eq. (C.2).

In Fig. 11 the convergence and efficiency considering the drag coefficient are visualized on the left and right, respectively. One can see that the ESDIRK4-6 reaches the desired order of convergence. The sixth order schemes have difficulties to reach the desired order for large timesteps but reach the asymptotic regime for small timesteps. The eighth order HBPC(8, 7) scheme also achieves almost its desired order for the small timesteps. The figure shows that the HBPC(6, 5) scheme has smaller errors than the ESDIRK methods using the same timestep. However, the HBPC(8, 7) scheme could outperform the sixth order schemes in terms of error only for a few timesteps. Considering the efficiency, the ESDIRK4-6 seems to be superior to the other methods. The ESDIRK6-9 and the HBPC(6, 5) are within reach, but are only able to outperform the ESDIRK4-6 for very few settings. One reason for the good behavior of the forth order method could be its L-stability. As both, the ESDIRK6-9 and the HBPC($6, k_{\max}$) are not L-stable, ESDIRK4-6 is naturally better suited for stiff problems. Note that the results of this investigation depend on the equipped error measure and physical setting. Using another error measure or using another mesh and/or Reynolds number could lead to slightly different results.

5.4.3. Taylor-Green-Vortex

Finally, we consider the three dimensional Taylor-Green-Vortex (TGV). It is a prototypical periodic test case to study the transition to turbulence and its decay. The initial data for the non-dimensional equations (see also [46]) are

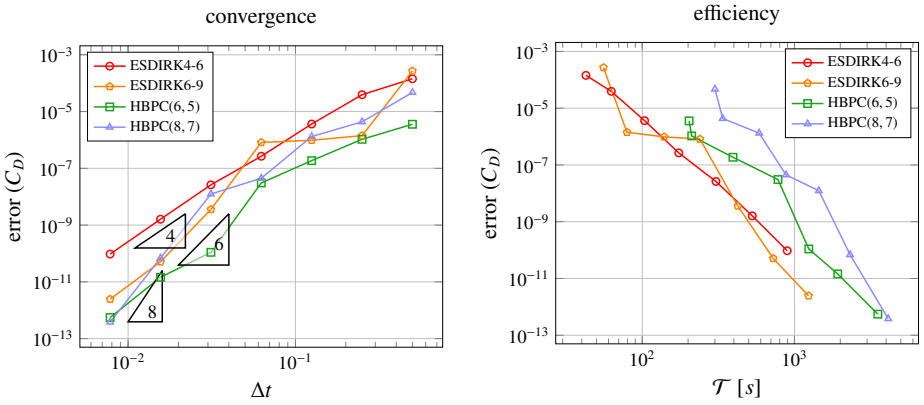

Figure 11. Temporal convergence (left) of mean drag coefficient C_D and required wallclocktime (right) to achieve a certain error for different implicit schemes for the cylinder flow problem at $\mathrm{Re}_D = 200$ using different timestep sizes.

given by

$$\rho = 1, \quad \mathbf{v} = \begin{pmatrix} \cos(x)\cos(y)\cos(z) \\ -\cos(x)\sin(y)\cos(z) \\ 0 \end{pmatrix}, \quad \text{and} \quad p = \frac{\rho}{\gamma} + \frac{\rho\varepsilon^2}{16}\left(\cos(2x) + \cos(2y)\right)\left(\cos(2z) + 2\right),$$

on the periodic domain $\Omega = [0, 2\pi]^3$, which we discretize with $N_E = 16 \times 16 \times 16$ and $N_p = 3$. We use $\varepsilon = 10^{-1}$, a Reynolds number of $\mathrm{Re} = 800$ and $T_{end} = 10$. A measure for the turbulent decay is the dissipation rate of the kinetic energy

$$\frac{\partial E_{\mathrm{kin}}}{\partial t} = \frac{\mu}{\rho\|\Omega\|}\int_\Omega \nabla_x \mathbf{v} : \nabla_x \mathbf{v} d\mathbf{x}.$$

We use the dissipation rate as an error measure by calculating the L_1-norm of the measured dissipation rates in the time intervals $\Delta t = 0.25$. An explicit simulation with a very small timestep (ERK4, $\Delta t \approx 1.4 \cdot 10^{-3}$) serves as a reference. Similar as it has been done in Sec. 5.4.2, the ESDIRK4-6 and the ESDIRK6-9 use $\eta = 0.1$. For the HBPC(6, 5), the temporal error is estimated according to Eq. (23) and $C = 0.5$, $\eta = 0.1$. During the startup procedure of the adaptive Newton strategy, a relative tolerance of $\varepsilon_{\mathrm{Newton,rel}} = 10^{-4}$ is used. The ESDIRK schemes initially use a Newton tolerance of $\varepsilon_{\mathrm{Newton,rel}} = 10^{-6}$. At every 10-th timestep, the temporal error is estimated and the preconditioners are rebuilt. Simulations are performed on one node with 36 processors, or the respective multiples for the temporal parallelization. Please note that we do not report on HBPC(8, 7) results in this section. Due to limited computational resources, and the large amount of degrees of freedom present, we could not test the method sufficiently such that fair and reliable results can be guaranteed. Only very preliminary results are available that show that the method does not suffer from stability issues, and numerical errors are at least in the order of the other methods shown here.

The dissipation rate for the ESDIRK4-6, ESDIRK6-9 and the HBPC(6, 5), each with $\Delta t = 0.25$ are shown in Fig. 12 (left). Virtually, all schemes coincide; deviations from the DNS data [47] are due to the too coarse spatial resolution. In Fig. 12 (middle and right) the convergence and the efficiency w.r.t. the dissipation rate for the ESDIRK4-6, the ESDIRK6-9 and the HBPC(6, 5) are shown. One can see that the HBPC(6, 5) has smaller errors than the ESDIRK methods for the same chosen timestep sizes. However, this advantage does not directly transfer to higher efficiencies. This motivates further research on the development of the HBPC methods as outlined in the next section, possibly offering higher accuracies and efficiencies for stiff problems.

6. Conclusion and Outlook

In this work, we have shown the application of a parallel-in-time implicit two-derivative discontinuous Galerkin method to the Navier-Stokes equations. As time discretization the HBPC scheme has been used. In previous works,

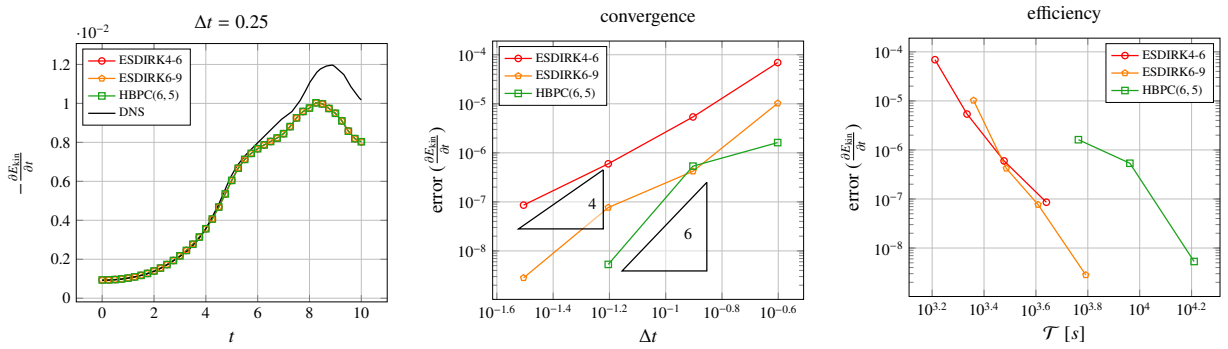

Figure 12. Taylor-Green vortex at Re = 800 and $\varepsilon = 10^{-1}$: temporal evolution of kinetic energy dissipation rate (left) with $\Delta t = 0.25$, DNS data from [47]. Temporal convergence (middle) and required wallclocktime (right) to achieve a certain error for different implicit schemes using different timestep sizes.

this time discretization has been combined with the discontinous Galerkin method [1] to solve PDEs and the concept of time parallelism has been shown for ODEs [3]. The present work tackles practical aspects of combining a space-parallel discontinuous Galerkin PDE discretization with the time-parallel HBPC scheme.

A homogeneous distribution of linear iterations over the different processors has been identified as a key for parallel efficiency. Two main ingredients have been introduced for that purpose: an adaptive procedure for Newton's method, and an additional distribution of the predictor's and first corrector's stages to different processors. It has been shown that the temporal parallelization reaches a parallel efficiency of approx. $70-60\%$ on $4-7$ partitions. The pure spatial parallelization has been shown to be well suited for parallel computing as it provides $50-80\%$ parallel efficiency for very fine granularities on more than 1000 processors. Combining spatial and temporal parallelization offers the possibility to obtain further speedup and in some cases also an improved efficiency over the pure spatial parallelization. This has also been demonstrated for settings with more than 1000 processors, highlighting the capability of the novel method to solve large-scale problems. Furthermore, the novel method has been compared with serial-in-time ESDIRK methods in terms of efficiency. We have shown that in some cases the novel method can outperform these schemes.

We consider the current paper as a milestone towards making two-derivative predictor corrector schemes a viable alternative to established schemes in applications from compressible flows. Obviously, the results in Sec. 5 show that there is still room for improvement. Active research lines include the identification of more suited background schemes in Eq. (6) (other collocation points, extension to general linear methods); the use of IMEX schemes within this context, see [2, 3, 48] for first attempts in the context of ODEs; and Jacobian-free high-derivative schemes, where more than two temporal derivatives are added to the algorithm using a suitable finite difference approach, see [49] for first attempts. Further developments will consider full adaptivity in space and time, hence making use of spatial error estimators to determine both hp-adaptivity and non-constant timesteps.

Acknowledgments

J. Zeifang was funded by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) - project no. 457811052. A. Thenery Manikantan was funded by the "Bijzonder Onderzoeksfonds" (BOF) from UHasselt - project no. BOF21KP12. We acknowledge the VSC (Flemish Supercomputer Center) for providing computing resources. The VSC is funded by the Research Foundation - Flanders (FWO) and the Flemish Government.

Declaration 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.

Appendix A. Navier-Stokes fluxes

For the Navier-Stokes equations (1), inviscid and viscous fluxes $\mathbf{F}(\mathbf{w})$ and $\mathbf{F}^v(\mathbf{w}, \nabla_x \mathbf{w})$ are given by

$$\mathbf{F}(\mathbf{w}) = \begin{pmatrix} \rho \mathbf{v} \\ \rho \mathbf{v} \otimes \mathbf{v} + \frac{1}{\varepsilon^2} p \cdot \mathrm{Id} \\ \mathbf{v}(E + p) \end{pmatrix}, \quad \text{and} \quad \mathbf{F}^v(\mathbf{w}, \nabla_x \mathbf{w}) = \begin{pmatrix} 0 \\ \boldsymbol{\tau} \\ \boldsymbol{\tau} \cdot \mathbf{v} + \mathbf{q} \end{pmatrix}. \tag{A.1}$$

Pressure is coupled to density, momentum and energy via the ideal gas equation of state,

$$p = (\gamma - 1)\left(E - \frac{\varepsilon^2}{2}\rho \|\mathbf{v}\|_2^2\right).$$

The viscous stress tensor $\boldsymbol{\tau}$ and the heat flux $\mathbf{q}$ are defined as

$$\boldsymbol{\tau} := \mu\left(\nabla_x \mathbf{v} + (\nabla_x \mathbf{v})^T - \frac{2}{3}(\nabla_x \cdot \mathbf{v})\,\mathrm{Id}\right), \quad \text{and} \quad \mathbf{q} := \lambda_T \nabla_x T,$$

where T denotes temperature, μ dynamic viscosity, the thermal conductivity $\lambda_T = \frac{c_p \mu}{Pr}$, specific heat capacity $c_p = \frac{R\gamma}{\gamma - 1}$, specific gas constant $R = \frac{1}{\gamma \varepsilon^2}$, the ideal gas law $p = \rho R T$ and the fluid specific Prandtl number $Pr = 0.72$.

Appendix B. Butcher Tables of the background Hermite-Birkhoff Runge-Kutta Methods

We consider the following quadrature rules:

- A sixth-order method ($q = 6$) with three stages ($s = 3$, one being fully explicit), as also used in [19, 3]

$$c = \begin{pmatrix} 0 \\ \frac{1}{2} \\ 1 \end{pmatrix}, \quad B^{(1)} = \begin{pmatrix} 0 & 0 & 0 \\ \frac{101}{480} & \frac{8}{30} & \frac{55}{2400} \\ \frac{7}{30} & \frac{16}{30} & \frac{7}{30} \end{pmatrix}, \quad B^{(2)} = \begin{pmatrix} 0 & 0 & 0 \\ \frac{65}{4800} & -\frac{25}{600} & -\frac{25}{8000} \\ \frac{1}{60} & 0 & -\frac{1}{60} \end{pmatrix}, \tag{B.1}$$

  with the fifth-order ($\hat{q} = 5$) embedded quadrature rule

$$\hat{c} = \begin{pmatrix} 0 \\ \frac{1}{2} \\ 1 \end{pmatrix}, \quad \hat{B}^{(1)} = \begin{pmatrix} 0 & 0 & 0 \\ \frac{31}{240} & \frac{4}{15} & \frac{5}{48} \\ \frac{2}{15} & \frac{8}{15} & \frac{1}{3} \end{pmatrix}, \quad \hat{B}^{(2)} = \begin{pmatrix} 0 & 0 & 0 \\ 0 & -\frac{23}{240} & -\frac{1}{60} \\ 0 & -\frac{1}{15} & -\frac{1}{30} \end{pmatrix}. \tag{B.2}$$

- An eighth-order method ($q = 8$) with four stages ($s = 4$, one being fully explicit), as also used in [3]

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

  with the seventh-order ($\hat{q} = 7$) embedded quadrature rule

$$\hat{c} = \begin{pmatrix} 0 \\ \frac{1}{3} \\ \frac{2}{3} \\ 1 \end{pmatrix}, \quad \hat{B}^{(1)} = \begin{pmatrix} 0 & 0 & 0 & 0 \\ \frac{212}{2835} & \frac{47}{1680} & \frac{6}{35} & \frac{171}{2891} \\ \frac{214}{2835} & \frac{19}{105} & \frac{12}{35} & \frac{191}{2835} \\ \frac{8}{105} & \frac{117}{560} & \frac{18}{35} & \frac{337}{1680} \end{pmatrix}, \quad \hat{B}^{(2)} = \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & -\frac{299}{4237} & -\frac{97}{1890} & -\frac{51}{9599} \\ 0 & -\frac{59}{945} & -\frac{62}{945} & -\frac{17}{2835} \\ 0 & -\frac{33}{560} & -\frac{3}{70} & -\frac{19}{1680} \end{pmatrix}, \tag{B.4}$$

Appendix C. Adaptive Criterion for ESDIRK Method

For the comparisons in Sec. 5.4, we use two different ESDIRK methods. The 4^{th} order ARK4(3)6L[2]SA method [39, Appendix D] (abbreviated with ESDIRK4-6) and the 6^{th} order ESDIRK6(5)9L[2]SA method [40, Table 13] (abbreviated with ESDIRK6-9)[9]. For those single-derivative implicit ESDIRK methods, the non-linear equation to be solved for each stage l is given by

$$\boldsymbol{X}^l = \mathbf{w}_{\text{old}} + \Phi\left(\boldsymbol{X}^l, \boldsymbol{X}^{1:l-1}\right),$$

with $\boldsymbol{X}^l := \mathbf{w}_h^{n,l}$, where $\mathbf{w}_h^{n,l}$ denotes the discrete $\mathbf{w}$ at time t^n and stage l. For the ease of notation, we use $\boldsymbol{X}^{1:l-1} := \left(\boldsymbol{X}^1, \boldsymbol{X}^2, \dots, \boldsymbol{X}^{l-1}\right)$. The function Φ is given by

$$\Phi\left(\boldsymbol{X}^l, \boldsymbol{X}^{1:l-1}\right) = \Delta t B_{ll}^{(1)} \mathbf{R}_h^{(1)}(\mathbf{w}_h^{n,l}) + \Delta t \sum_{i=1}^{l-1} B_{li}^{(1)} \mathbf{R}_h^{(1)}(\mathbf{w}_h^{n,i}).$$

Note that the introduction of $\boldsymbol{X}^l$ would not have been necessary for this method. Nevertheless, we introduce it here to highlight the similarities with the two-derivative method, see Eq. (16). As initial guess for Newton's method we use $\mathbf{w}_{h,0}^{n,l} = \mathbf{w}_h^{n,l-1}$. After defining the error introduced by Newton's method

$$\mathcal{E}_{\mathtt{r}}^l := \boldsymbol{X}^l - \boldsymbol{X}_{\mathtt{r}}^l,$$

where the subscript $\mathtt{r}$ denotes the solution of the $\mathtt{r}$-th Newton step, we follow the steps outlined in Sec. 4.1.1 and find

$$\begin{aligned}\mathcal{E}_{\mathtt{r}}^l \doteq & \Delta \boldsymbol{X}_{\mathtt{r}+1} + \left(\text{Id} - \frac{\partial \Phi\left(\boldsymbol{X}^l, \boldsymbol{X}^{1:l-1}\right)}{\partial \boldsymbol{X}^l}\right)^{-1} \cdot \left(\sum_{i=1}^{l-1} \frac{\partial \Phi\left(\boldsymbol{X}^l, \boldsymbol{X}^{1:l-1}\right)}{\partial \boldsymbol{X}^i} \mathcal{E}_{\mathtt{r}'}^i\right) \\ = & \Delta \boldsymbol{X}_{\mathtt{r}+1} + \left(\text{Id} - \Delta t B_{ll}^{(1)} \frac{\partial \mathbf{R}_h^{(1)}}{\partial \mathbf{w}_h^{n,l}}\right)^{-1} \cdot \left(\Delta t \sum_{i=1}^{l-1} B_{li}^{(1)} \frac{\partial \mathbf{R}_h^{(1)}}{\partial \mathbf{w}_h^{n,i}} \mathcal{E}_{\mathtt{r}'}^i\right).\end{aligned}$$

The limits of this equations can then be found as

$$\begin{aligned}\|\mathcal{E}_{\mathtt{r}}^l\|_2 &\approx \|\Delta \boldsymbol{X}_{\mathtt{r}+1}\|_2 && \text{for } \Delta t \to 0, \\ \|\mathcal{E}_{\mathtt{r}}^l\|_2 &\approx \|\Delta \boldsymbol{X}_{\mathtt{r}+1}\|_2 + \sum_{i=2}^{l-1} \tilde{C}_i \left|\frac{B_{li}^{(1)}}{B_{ll}^{(1)}}\right| \|\mathcal{E}_{\mathtt{r}'}^i\|_2 && \text{for } \Delta t \to \infty,\end{aligned}$$

for some constants $\tilde{C}_i \approx \left\| \left(\frac{\partial \mathbf{R}_h^{(1)}}{\partial \mathbf{w}_h^{n,l}}\right)^{-1} \cdot \left(\frac{\partial \mathbf{R}_h^{(1)}}{\partial \mathbf{w}_h^{n,i}}\right) \right\|_2$, which equal one for a linear system. Note that due to the explicit evaluation of the first stage it holds $\mathcal{E}_{\mathtt{r}}^1 = 0$. We can hence find a condition for the Newton increment of stage l via

$$\|\Delta \boldsymbol{X}_{\mathtt{r}+1}^l\|_2 + C \sum_{i=2}^{l-1} \|\Delta \boldsymbol{X}_{\mathtt{r}+1}^i\|_2 + O(C^2) + \dots + O(C^{l-2}) \leq \eta \|\mathcal{E}_t\|_2. \tag{C.1}$$

As for the used ESDIRK methods the last stage directly gives the solution at the new timestep, we have defined $\|\mathcal{E}_t\|_2 := \|\mathcal{E}_t^s\|_2$. Similar as for the HBPC methods, C is a function of the timestep, though, we choose $C = 0.5$ to be constant in this paper and truncate all higher order terms of C in Eq. (C.1). This corresponds to neglecting secondary effects of error propagation from previous stages. Demanding that the Newton increments of all stages are below a common threshold, we can then find

$$\|\Delta \boldsymbol{X}_{\mathtt{r}+1}^l\|_2 \leq \frac{1}{1 + C(s-2)} \eta \|\mathcal{E}_t\|, \qquad \text{for} \quad l = 2, \dots, s, \tag{C.2}$$

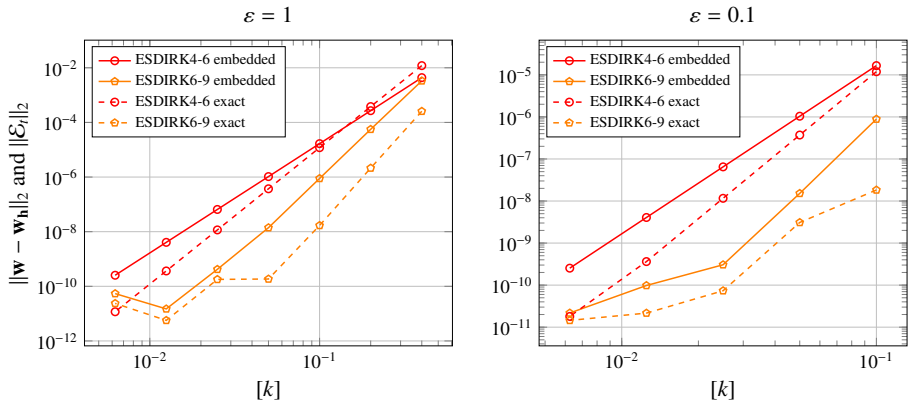

Figure C.13. Accuracy of embedded error estimate for ESDIRK4-6 and ESDIRK6-9 using the initial conditions given by Eq. (25) , $\mathcal{N}_p = 7$ and $\mathcal{N}_E = 32^2$. The reference Mach number is set to $\varepsilon = 1$ (left) and $\varepsilon = 10^{-1}$ (right).

which can be combined with the Newton error extrapolation described in Eq. (36) and Eq. (37). Note that, similar as for the HBPC schemes, we have introduced a safety factor η that accounts for the non-exact error estimate via the embedded temporal error estimate. In Fig. C.13 the accuracy of the embedded error estimates are shown for the sine wave example and the same setup as used in Sec. 4.1.4. The results suggest that choosing $\eta \leq 0.1$ is reasonable.

We now combine the embedded error estimate and the adaptive Newton criterion given by Eq. (C.2). The effectiveness of this adaptive strategy is highlighted in Fig. C.14. One can see that the novel adaptation strategy is at least as good as the standard strategies based on the Newton residual and outperforms them for large timesteps.

References

[1] J. Zeifang, J. Schütz, Implicit two-derivative deferred correction time discretization for the discontinuous Galerkin method, Journal of Computational Physics 464 (2022) 111353.
[2] J. Schütz, D. Seal, An asymptotic preserving semi-implicit multiderivative solver, Applied Numerical Mathematics 160 (2021) 84–101.
[3] J. Schütz, D. C. Seal, J. Zeifang, Parallel-in-time high-order multiderivative IMEX solvers, Journal of Scientific Computing 90 (54) (2022) 1–33.
[4] J. Zeifang, J. Schütz, D. Seal, Stability of implicit multiderivative deferred correction methods, BIT Numerical Mathematics (2022).
[5] S. Vangelatos, On the efficiency of implicit discontinuous Galerkin spectral element methods for the unsteady compressible Navier-Stokes equations, Ph.D. thesis, University of Stuttgart (2019).
[6] E. Ferrer, G. Rubio, G. Ntoukas, W. Laskowski, O. Mariño, S. Colombo, A. Mateo-Gabín, H. Marbona, F. Manrique de Lara, D. Huergo, J. Manzanero, A. Rueda-Ramírez, D. Kopriva, E. Valero, HORSES3D: A high-order discontinuous Galerkin solver for flow simulations and multi-physics applications, Computer Physics Communications 287 (2023) 108700.
[7] F. D. Witherden, A. M. Farrington, P. E. Vincent, PyFR: An open source framework for solving advection–diffusion type problems on streaming architectures using the flux reconstruction approach, Computer Physics Communications 185 (11) (2014) 3028–3040.
[8] T. Lunet, J. Bodart, S. Gratton, X. Vasseur, Time-parallel simulation of the decay of homogeneous turbulence using parareal with spatial coarsening, Computing and Visualization in Science 19 (1) (2018) 31–44.
[9] W. Chen, Y. Ju, C. Zhang, Parallel-in-time-space Chebyshev pseudospectral method for unsteady fluid flows (2021).
URL https://www.researchgate.net/publication/350049339
[10] R. Croce, D. Ruprecht, R. Krause, Parallel-in-space-and-time simulation of the three-dimensional, unsteady Navier–Stokes equations for incompressible flow, in: Modeling, Simulation and Optimization of Complex Processes-HPSC 2012, Springer, 2014, pp. 13–23.
[11] M. J. Gander, 50 years of time parallel time integration, in: Multiple shooting and time domain decomposition methods, Vol. 9 of Contributions in Mathematical and Computational Sciences, Springer, Cham, 2015, pp. 69–113.
[12] B. W. Ong, J. B. Schroder, Applications of time parallelization, Computing and Visualization in Science 23 (1-4) (2020) 11.
[13] Parallel-in-Time.
URL https://parallel-in-time.org
[14] C. W. Gear, Parallel methods for ordinary differential equations, Calcolo 25 (1) (1988) 1–20.
[15] W. L. Miranker, W. Liniger, Parallel methods for the numerical integration of ordinary differential equations, Mathematics of Computation 21 (1967) 303–320.
[16] A. J. Christlieb, C. B. Macdonald, B. W. Ong, Parallel high-order integrators, SIAM Journal on Scientific Computing 32 (2) (2010) 818–835.

[9] Note that in the original publication [40, Table 13] there is a typo for $\hat{b}_4$. The value should be $\hat{b}_4 = \frac{1376520686137389}{1064235527052079}$.

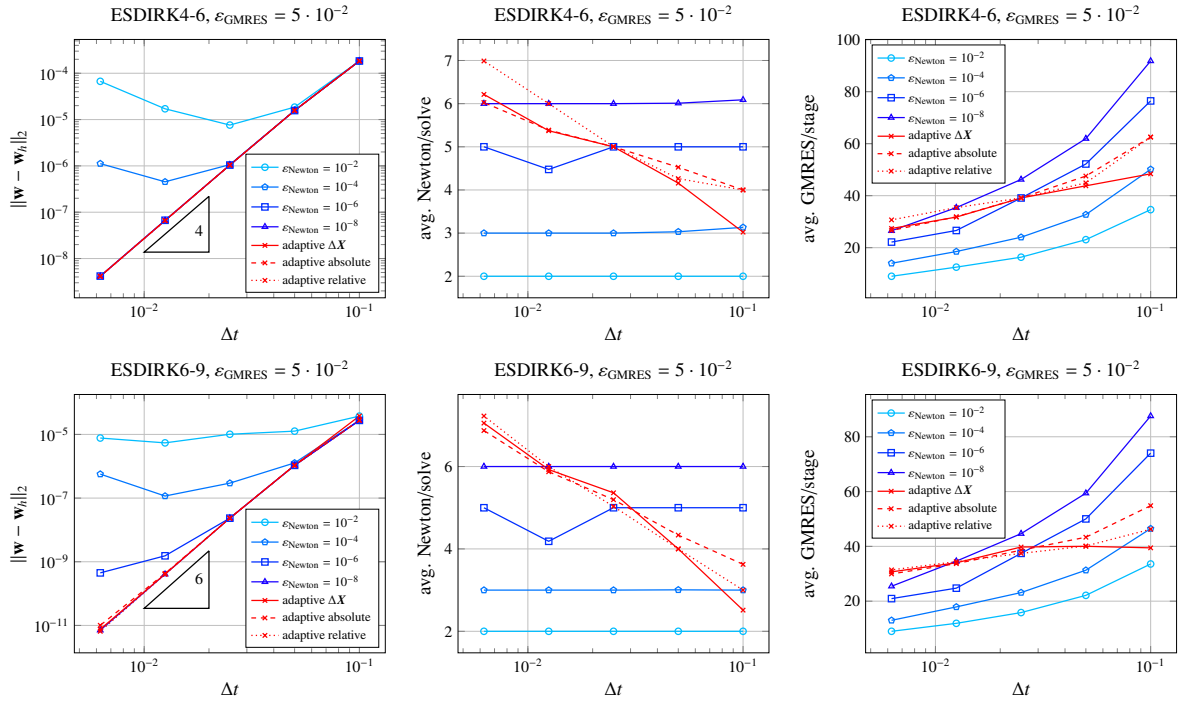

Figure C.14. Resulting L_2-error (left), average Newton iterations per stage (middle) and average GMRES iterations per stage (right) for ESDIRK4-6 (top) and ESDIRK6-9 (bottom) with fixed GMRES tolerance when choosing different convergence criteria for Newton's method. Adaptive Newton strategy is performed according to Eq. (C.2) (adaptive $\Delta \boldsymbol{X}$), $N(\boldsymbol{X}_\mathrm{r}) \leq \eta \|\mathcal{E}_t\|_2$ (adaptive absolute) or $N(\boldsymbol{X}_\mathrm{r}) \leq \eta \|\mathcal{E}_t\|_2 N(\boldsymbol{X}_0)$ (adaptive relative). Note that the legend in the middle figures has been omitted for clarity but is the same as for the left and the right plot.

[17] A. Christlieb, B. Ong, Implicit parallel time integrators, Journal of Scientific Computing 49 (2) (2011) 167–179.
[18] A. J. Christlieb, C. B. Macdonald, B. W. Ong, R. J. Spiteri, Revisionist integral deferred correction with adaptive step-size control, Communications in Applied Mathematics and Computational Science 10 (1) (2015) 1–25.
[19] J. Schütz, D. C. Seal, A. Jaust, Implicit multiderivative collocation solvers for linear partial differential equations with discontinuous Galerkin spatial discretizations, Journal of Scientific Computing 73 (2017) 1145–1163.
[20] D. A. Kopriva, Implementing spectral methods for partial differential equations: Algorithms for scientists and engineers, Springer Science & Business Media, 2009.
[21] F. Hindenlang, G. Gassner, C. Altmann, A. Beck, M. Staudenmaier, C.-D. Munz, Explicit discontinuous Galerkin methods for unsteady problems, Computers & Fluids 61 (2012) 86–93.
[22] F. Bassi, S. Rebay, G. Mariotti, S. Pedinotti, M. Savini, A high-order accurate discontinuous Finite Element method for inviscid and viscous turbomachinery flows, Proceedings of 2nd European Conference on Turbomachinery, Fluid Dynamics and Thermodynamics (1997) 99–108.
[23] V. Linders, P. Birken, Locally conservative and flux consistent iterative methods, arXiv preprint arXiv:2206.10943 (2022).
[24] A. Kværno, Singly diagonally implicit Runge–Kutta methods with an explicit first stage, BIT Numerical Mathematics 44 (3) (2004) 489–502.
[25] C. A. Kennedy, M. H. Carpenter, Diagonally implicit Runge–Kutta methods for ordinary differential equations. A review, Tech. Rep. TM-2016-219173, NASA Langley Research Center (2016).
[26] M. Carpenter, C. Kennedy, Fourth-order 2N-storage Runge-Kutta schemes, Tech. rep., NASA Langley Research Center (1994).
[27] M. Franciolini, A. Crivellini, A. Nigro, On the efficiency of a matrix-free linearly implicit time integration strategy for high-order discontinuous Galerkin solutions of incompressible turbulent flows, Computers & Fluids 159 (2017) 276–294.
[28] Y. Pan, Z.-G. Yan, J. Peiró, S. Sherwin, Development of a balanced adaptive time-stepping strategy based on an implicit JFNK-DG compressible flow solver, Communications on Applied Mathematics and Computation 4 (2022) 728–757.
[29] M. Han Veiga, P. Öffner, D. Torlo, DeC and ADER: similarities, differences and a unified framework, Journal of Scientific Computing 87 (1) (2021) Paper No. 2, 35.
[30] V. Dolejší, M. Holík, J. Hozman, Efficient solution strategy for the semi-implicit discontinuous Galerkin discretization of the Navier–Stokes equations, Journal of Computational Physics 230 (11) (2011) 4176–4200.
[31] D. Blom, P. Birken, H. Bijl, F. Kessels, A. Meister, A. van Zuilen, A comparison of Rosenbrock and ESDIRK methods combined with iterative solvers for unsteady compressible flows, Advances in Computational Mathematics 42 (2016) 1401–1426.
[32] P. Birken, Numerical Methods for Unsteady Compressible Flow Problems, Numerical Analysis and Scientific Computing, Chapman & Hall, 2021.
[33] G. Noventa, F. Massa, F. Bassi, A. Colombo, N. Franchina, A. Ghidoni, A high-order discontinuous Galerkin solver for unsteady incompressible turbulent flows, Computers & Fluids 139 (2016) 248–260.

[34] S. C. Eisenstat, H. F. Walker, Choosing the forcing terms in an inexact Newton method, SIAM Journal on Scientific Computing 17 (1) (1996) 16–32.
[35] N. Krais, A. Beck, T. Bolemann, H. Frank, D. Flad, G. Gassner, F. Hindenlang, M. Hoffmann, T. Kuhn, M. Sonntag, et al., FLEXI: A high order discontinuous Galerkin framework for hyperbolic–parabolic conservation laws, Computers & Mathematics with Applications 81 (2021) 186–219.
[36] S. Götschel, M. Minion, D. Ruprecht, R. Speck, Twelve ways to fool the masses when giving parallel-in-time results, in: Workshops on Parallel-in-Time Integration, Springer, 2020, pp. 81–94.
[37] W. Chen, Y. Ju, C. Zhang, A parallel inverted dual time stepping method for unsteady incompressible fluid flow and heat transfer problems, Computer Physics Communications 260 (2021) 107325.
[38] N. Margenberg, T. Richter, Parallel time-stepping for fluid–structure interactions, Mathematical Modelling of Natural Phenomena 16 (2021) 20.
[39] C. A. Kennedy, M. H. Carpenter, Additive Runge-Kutta schemes for convection-diffusion-reaction equations, Applied Numerical Mathematics 44 (2003) 139–181.
[40] C. A. Kennedy, M. H. Carpenter, Diagonally implicit Runge–Kutta methods for stiff ODEs, Applied Numerical Mathematics 146 (2019) 221–244.
[41] H. Bijl, M. H. Carpenter, V. N. Vatsa, C. A. Kennedy, Implicit time integration schemes for the unsteady compressible Navier–Stokes equations: laminar flow, Journal of Computational Physics 179 (1) (2002) 313–329.
[42] A. Nigro, C. De Bartolo, F. Bassi, A. Ghidoni, Up to sixth-order accurate A-stable implicit schemes applied to the discontinuous Galerkin discretized Navier–Stokes equations, Journal of Computational Physics 276 (2014) 136–162.
[43] L. Qu, C. Norberg, L. Davidson, S.-H. Peng, F. Wang, Quantitative numerical analysis of flow past a circular cylinder at Reynolds number between 50 and 200, Journal of Fluids and Structures 39 (2013) 347–370.
[44] C. Liang, S. Premasuthan, A. Jameson, High-order accurate simulation of low-Mach laminar flow past two side-by-side cylinders using spectral difference method, Computers & Structures 87 (11-12) (2009) 812–827.
[45] J. Meneghini, F. Saltara, C. Siqueira, J. Ferrari Jr., Numerical simulation of flow interference between two circular cylinders in tandem and side-by-side arrangements, Journal of Fluids and Structures 15 (2) (2001) 327–350.
[46] J. Zeifang, J. Schütz, K. Kaiser, A. Beck, M. Lukáčová-Medvid'ová, S. Noelle, A novel full-Euler low Mach number IMEX splitting, Communications in Computational Physics 27 (2020) 292–320.
[47] M. E. Brachet, D. I. Meiron, S. A. Orszag, B. Nickel, R. H. Morf, U. Frisch, Small-scale structure of the Taylor–Green vortex, Journal of Fluid Mechanics 130 (1983) 411–452.
[48] E. Theodosiou, J. Schütz, D. Seal, An explicitness-preserving IMEX-split multiderivative method, UHasselt CMAT Preprint UP2301 (2023). URL https://www.uhasselt.be/media/zvvbnhh0/up2301.pdf
[49] J. Chouchoulis, J. Schütz, J. Zeifang, Jacobian-free explicit multiderivative Runge-Kutta methods for hyperbolic conservation laws, Journal of Scientific Computing 90 (96) (2022).