Preprint (2021) 1–22

Two-derivative deferred correction time discretization for the discontinuous Galerkin method

Jonas Zeifang[a], Jochen Schütz[a]

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

Abstract

In this paper, we use an implicit two-derivative deferred correction time discretization approach and combine it with a spatial discretization of the discontinuous Galerkin spectral element method to solve (non-)linear PDEs. The resulting numerical method is high order accurate in space and time. As the novel scheme handles two time derivatives, the spatial operator for both derivatives has to be defined. This results in an extended system matrix of the scheme. We analyze this matrix regarding possible simplifications and an efficient way to solve the arising (non-)linear system of equations. It is shown how a carefully designed preconditioner and a matrix-free approach allow for an efficient implementation and application of the novel scheme. For both, linear advection and the compressible Euler equations, up to eighth order of accuracy in time is shown. Finally, it is illustrated how the method can be used to approximate solutions to the compressible Navier-Stokes equations.

Keywords: Multiderivative Schemes, Discontinuous Galerkin Spectral Element Method, implicit time stepping

1. Introduction

We aim for solving hyperbolic PDEs that can be cast into flux formulation

$$\mathbf{w}_t + \nabla_x \cdot \mathbf{F}(\mathbf{w}) = 0, \tag{1}$$

with state vector $\mathbf{w}$ and flux $\mathbf{F}(\mathbf{w})$. Obviously, upon defining

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

this can be cast as an ODE in some infinite-dimensional function space,

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

While standard time discretization methods (for example Runge-Kutta or BDF methods) only use the information of the first time derivative $\mathbf{w}_t$ for time stepping, two-derivative methods make use of the second time derivative $\mathbf{w}_{tt}$, computed through

$$\mathbf{w}_{tt} = (-\nabla_x \cdot \mathbf{F}(\mathbf{w}))_t = \nabla_x \cdot (-\mathbf{F}(\mathbf{w}))_t = \nabla_x \cdot \left(-\frac{\partial \mathbf{F}}{\partial \mathbf{w}}\mathbf{w}_t\right) = \nabla_x \cdot \left(-\frac{\partial \mathbf{F}}{\partial \mathbf{w}}\mathbf{R}^{(1)}(\mathbf{w})\right) =: \mathbf{R}^{(2)}(\mathbf{w}, \mathbf{R}^{(1)}(\mathbf{w})). \tag{4}$$

Email addresses: `jonas.zeifang@uhasselt.be` (Jonas Zeifang), `jochen.schuetz@uhasselt.be` (Jochen Schütz)

Using this additional derivative introduces more flexibility when designing time stepping methods. This allows to achieve high order of accuracy while using less temporal quadrature nodes than conventional schemes. The class of two-derivative methods belongs to the general class of multistep-multistage-multiderivative methods [1]. Important multiderivative methods are the Taylor methods which are typically referred to the class of Lax-Wendroff methods when considering PDEs [2]; and multiderivative Runge-Kutta schemes [3]. PDE discretizations with Lax-Wendroff methods have been widely addressed in literature, see e.g. the by far not exhaustive list [4, 5, 6, 7, 8]. Also the arbitrary high order derivative Riemann problem (ADER) approach [9] is based on this idea, see [10] for a recent overview on this technique. Explicit two-derivative Runge-Kutta methods have also been used in the context of PDEs, see e.g. [11] where WENO and DG schemes have been combined with a multiderivative Runge-Kutta methods. Other examples for different PDE discretizations with explicit multiderivative Runge-Kutta methods can e.g. be found in [12, 13, 14, 15]. A semi-implicit two-derivative Runge-Kutta method has been used as a generalization of a Lax-Wendroff finite difference approach in [16].

Implicit two-derivative methods have only rarely been used in the context of PDE discretizations. In [17] an implicit Lax-Wendroff approach for a two-derivative two-point method has been introduced. For the spatial discretization a hybridized discontinuous Galerkin (HDG) method has been used. The authors showed that even if the underlying two-derivative implicit method is A-stable, a severe timestep restriction for the PDE discretization can be observed. In a follow-up publication, this timestep restriction is overcome by the introduction of a systematic approach to introduce an additional solution variable per time derivative [18]. In that work, the authors show that the discretization of a linear PDE preserves the stability properties of the ODE solver, corresponding to the method of lines approach. The authors illustrate this with a DG method with a two- and three-derivative two-point time discretization. More details can also be found in [19]. This idea will lay the foundation of the current work.

In this work, differently to [18], higher order time discretization is achieved by using a deferred correction time discretization method. In [20], a fourth order accurate two-derivative deferred correction time discretization method has been presented. Recently, it has been combined with the idea of using multiple stages to obtain higher orders of accuracy [21]. There it has been shown that this class of schemes can be used to solve stiff ODEs while obtaining - at least in principle - arbitrary orders of accuracy. Here, these ODE solvers will be used as time discretization method to solve PDEs, while the spatial discretization is done with the discontinuous Galerkin spectral element method (DGSEM) [22]. The novel scheme is constructed such that it can handle non-linear PDEs, which will be illustrated with the compressible Euler and Navier-Stokes equations. One of the main drawbacks of the approach outlined in [18] is that one obtains an extended and hence larger system matrix. In this work, we show how the linearized system arising from this approach can be solved efficiently. Moreover, our novel approach allows for a straight-forward parallelization of the spatial domain as we introduce a matrix-free discretization and a relatively simple preconditioning strategy. Summing up, the main advances with respect to previous works are

- the use of implicit two-derivative high-order deferred correction methods for a PDE discretization, especially with the DGSEM;
- the handling of non-linear hyperbolic-parabolic PDEs with the ansatz described in [18];
- the introduction of a carefully designed preconditioner which is combined with a matrix-free discretization. This eases the implementation of two-derivative schemes in already existing numerical codes and allows for a straight-forward parallelization of the spatial domain.

The work is structured as follows: In Sec. 2 the two-derivative deferred correction method is introduced in a semi-discrete setting. In the following section (Sec. 3), the fully discrete scheme is derived. After deriving the operators for the first and second temporal derivative $\mathbf{R}^{(1)}$ and $\mathbf{R}^{(2)}$, two approaches how to improve the solution procedure of the arising extended linear system are presented. Both approaches are compared in Sec. 4 for the linear scalar advection and the compressible Euler equations. Furthermore it is shown how the implementation can be pursued matrix-free and the high order temporal accuracy of the novel scheme is illustrated. The extension of the method to the Navier-Stokes equations and some illustrative applications are presented in Sec. 5. Finally, a conclusion is drawn and an outlook is given in Sec. 6.

2. Semi-Discrete Formulation

In this section, the semi-discrete in time formulation of the novel scheme is reviewed. For that purpose, we briefly recall the serial algorithm from [20] and [21] and slightly modify it. The algorithm describes a **p**redictor-**c**orrector approach with $k_{\max}$ correction steps to approximate a two-derivative **H**ermite-**B**irkhoff Runge-Kutta method of order q and is therefore labeled as HBPC$(q, k_{\max})$. As we do not use an IMEX splitting in this paper as it is done in [20, 21], the predictor is modified such that a fourth-order two-point Hermite-Birkhoff Runge-Kutta method is successively used to obtain the predicted solution. Please note that for convenience, we stick with the name HBPC$(q, k_{\max})$ although we use a different predictor than in the original publication [21].

2.1. Hermite-Birkhoff Predictor-Corrector Time Discretization

The spatial operators to calculate $\mathbf{w}_t$ and $\mathbf{w}_{tt}$, viz. $\mathbf{R}^{(1)}$ and $\mathbf{R}^{(2)}$ are used to define the semi-discrete in time formulation. In principle, those operators can be discretized by any spatial discretization such as e.g. a discontinuous Galerkin, a finite volume or a finite difference discretization. In this work, we choose the DGSEM, for which the discrete formulation of $\mathbf{R}^{(1)}(\mathbf{w})$ and $\mathbf{R}^{(2)}(\mathbf{w}, \mathbf{R}^{(1)}(\mathbf{w}))$ will be described in Sec. 3.

Remark 1. $\mathbf{R}^{(2)}$ *depends on two arguments, namely* $\mathbf{w}$ *and* $\mathbf{R}^{(1)}(\mathbf{w})$*. We have explicitly chosen this notation, as it will become important in following sections, where we define an auxiliary variable* $\boldsymbol{\sigma}$ *to equal the discrete form of* $\mathbf{R}^{(1)}(\mathbf{w})$*. To keep the notation short in the following, however, we will abuse notation in this section only and set* $\mathbf{R}^{(2)}(\mathbf{w}) := \mathbf{R}^{(2)}(\mathbf{w}, \mathbf{R}^{(1)}(\mathbf{w}))$ *for the sake of a better readability.*

The method to be presented below relies on two-derivative Butcher tableaux consisting of matrices $B^{(1)}, B^{(2)} \in \mathbb{R}^{s\times s}$ and vector $c \in \mathbb{R}^s$. They define the limiting Hermite-Birkhoff Runge-Kutta scheme and are given in the appendix, Eq. (A.1)-(A.3). More details can be found in [21]. These Butcher tableaux define a quadrature formula $\mathcal{I}_l$ of order q through

$$\mathcal{I}_l := \Delta t \sum_{j=1}^{s} B^{(1)}_{lj} \mathbf{R}^{(1)}(\mathbf{w}^{n,[k],j}) + \Delta t^2 \sum_{j=1}^{s} B^{(2)}_{lj} \mathbf{R}^{(2)}(\mathbf{w}^{n,[k],j})$$

for every stage $1 \le l \le s$. Then, the temporal discretization of a PDE of type (1) with the HBPC$(q, k_{\max})$ method and the spatial operators $\mathbf{R}^{(1)}$ and $\mathbf{R}^{(2)}$ is given by

Algorithm 1 (HBPC$(q, k_{\max})$)**.** *Solve the following expression for* $\mathbf{w}^{n,[0],l}$ *and* $1 \le l \le s$*: First, the initial conditions are filled with* $\mathbf{w}^{-1,[k],s} := w_0$*. To advance the solution to Eq. (1) from time level* t^n *to time level* t^{n+1}*, fill the values* $\mathbf{w}^{n,[0],l}$ *using an implicit fourth order predictor*

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

$$\begin{aligned} \mathbf{w}^{n,[0],1} &:= \mathbf{w}^{n-1,[k_{\max}],s} \\ \mathbf{w}^{n,[0],l} &:= \mathbf{w}^{n,[0],l-1} + \frac{\Delta c_l \Delta t}{2}\left(\mathbf{R}^{(1)}(\mathbf{w}^{n,[0],l-1}) + \mathbf{R}^{(1)}(\mathbf{w}^{n,[0],l})\right) + \frac{(\Delta c_l \Delta t)^2}{12}\left(\mathbf{R}^{(2)}(\mathbf{w}^{n,[0],l-1}) - \mathbf{R}^{(2)}(\mathbf{w}^{n,[0],l})\right), \end{aligned} \tag{5}$$

   with $\Delta c_l := c_l - c_{l-1}$*.*
   Subsequently:

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

$$\begin{aligned} \mathbf{w}^{n,[k+1],1} &:= \mathbf{w}^{n-1,[k_{\max}],s}, \\ \mathbf{w}^{n,[k+1],l} &:= \mathbf{w}^{n-1,[k_{\max}],s} + \Delta t\left(\mathbf{R}^{(1)}(\mathbf{w}^{n,[k+1],l}) - \mathbf{R}^{(1)}(\mathbf{w}^{n,[k],l})\right) - \frac{\Delta t^2}{2}\left(\mathbf{R}^{(2)}(\mathbf{w}^{n,[k+1],l}) - \mathbf{R}^{(2)}(\mathbf{w}^{n,[k],l})\right) + \mathcal{I}_l. \end{aligned} \tag{6}$$

3. ***Update.*** *In order to preserve the first-same-as-last property, we put* $\mathbf{w}^{n+1} := \mathbf{w}^{n,[k_{\max}],s}$*.*

Remark 2. *In [21], it has been shown that the order of accuracy of the class of schemes is given by* $\min(4 + k_{\max}, q)$*. That means that starting with the fourth order of the predictor, the schemes pick up one order of accuracy per correction step. This continues until the order of the used quadrature rule is reached.*

2.2. *Solving for the Stage Values*

To solve for the stage values of the predictor and corrector, see Eq. (5) and Eq. (6), one can apply Newton's method. For both predictor and corrector, the resulting non-linear equations are very similar, they can be written in the generalized form

$$\begin{aligned}\mathbf{G}(\mathbf{W}) &:= \mathbf{g}(\mathbf{W}) - \mathbf{b} = 0,\\ \text{with} \quad \mathbf{g}(\mathbf{W}) &= \mathbf{W} - \alpha_1 \Delta t \mathbf{R}^{(1)}(\mathbf{W}) + \frac{\alpha_2 \Delta t^2}{2}\mathbf{R}^{(2)}(\mathbf{W}).\end{aligned} \tag{7}$$

For the predictor, $\mathbf{W}$, $\mathbf{b}$, α_1 and α_2 are given by

$$\mathbf{W} = \mathbf{w}^{n,[0],l}, \quad \mathbf{b} = \mathbf{w}^{n,[0],l-1} + \alpha_1 \Delta t \mathbf{R}^{(1)}(\mathbf{w}^{n,[0],l-1}) + \frac{\alpha_2 \Delta t^2}{2}\mathbf{R}^{(2)}(\mathbf{w}^{n,[0],l-1}), \quad \alpha_1 = \frac{\Delta c_l}{2}, \quad \text{and} \quad \alpha_2 = \frac{\Delta c_l^2}{6}, \tag{8}$$

and for the corrector steps the quantities are given by

$$\mathbf{W} = \mathbf{w}^{n,[k+1],l}, \quad \mathbf{b} = \mathbf{w}^{n-1,[k_{\max}],s} - \alpha_1 \Delta t \mathbf{R}^{(1)}(\mathbf{w}^{n,[k],l}) + \frac{\alpha_2 \Delta t^2}{2}\mathbf{R}^{(2)}(\mathbf{w}^{n,[k],l}), \quad \text{and} \quad \alpha_1 = \alpha_2 = 1. \tag{9}$$

Consequently, the method consists of subsequent solves of equations of type (7). Hence, two important building blocks remain to be defined to obtain the fully discrete scheme: A discretization for $\mathbf{R}^{(1)}$ and $\mathbf{R}^{(2)}$ has to be set up and an efficient method to solve equations of the type $\mathbf{G}(\mathbf{W}) = 0$ has to be found. Both building blocks will be described in the following section.

3. Fully Discrete Scheme

3.1. *The Discontinuous Galerkin Spectral Element Method*

The discontinuous Galerkin spectral element method has been introduced in [22]. Based on the discretization of the domain Ω with n_E quadrangular (2d) or hexahedral elements (3d) Ω_e, it utilizes high order nodal polynomials to represent the solution inside each element. As it is characteristic for discontinuous Galerkin methods, see e.g. [23, 24], discontinuities are allowed across cell boundaries.

3.1.1. *Calculation of the First Temporal Derivative with the DGSEM*

In this subsection, the discrete formulation of the DGSEM is briefly recalled. We closely follow Hindenlang et al. [25], and refer to the corresponding equations in their work where appropriate. The DGSEM is derived through the weak formulation of Eq. (1)

$$\sum_{e=1}^{n_E} \left(\mathbf{w}_t, \phi\right)_{\Omega_e} - \left(\mathbf{F}(\mathbf{w}), \nabla_x \phi\right)_{\Omega_e} + \left\langle \mathbf{F}^*(\mathbf{w}^L, \mathbf{w}^R) \cdot \mathbf{n}, \phi \right\rangle_{\partial\Omega_e} = 0, \qquad \forall \phi \in \Pi_N, \tag{10}$$

with the test functions ϕ taken from the - for now - not nearer specified set of polynomials Π_N. Element-wise integration over the domain Ω is denoted by the scalar product $(\cdot,\cdot)$, and $\langle \cdot,\cdot \rangle$ denotes the integral over the cell edges $\partial\Omega_e$. On the cell edges with normals $\mathbf{n}$, the flux is substituted by a numerical flux $\mathbf{F}^*$, which depends on "left" values $\mathbf{w}^L$ and "right" values $\mathbf{w}^R$ stemming from two neighboring elements. This equation is transformed into reference space $\boldsymbol{\xi} = (\xi_1, \xi_2)$ and the cells are mapped onto the reference unit element. For the ease of presentation, we restrict ourselves to two spatial dimensions. The steps of this transformation can be found in [25, Eqs. (2)-(16)]. After this transformation, one obtains the transformed fluxes $\mathcal{F}^m$ in direction $m \in \{1,2\}$ and the solution $\mathbf{w}$ in reference space. One important building block of the DGSEM is to represent both with the tensor-product of one dimensional Lagrange polynomials ℓ, each of degree N

$$\mathbf{w}(\boldsymbol{\xi}) \approx \mathbf{w}_h(\boldsymbol{\xi}) := \sum_{i,j=0}^{N} \hat{\mathbf{w}}_{ij} \ell_i(\xi_1)\ell_j(\xi_2), \quad \text{and} \quad \mathcal{F}^m(\boldsymbol{\xi}) \approx \mathcal{F}^m_h(\boldsymbol{\xi}) := \sum_{i,j=0}^{N} \hat{\mathcal{F}}^m(\hat{\mathbf{w}}_{ij})\ell_i(\xi_1)\ell_j(\xi_2), \quad m = 1,2, \tag{11}$$

where $(\hat{\bullet})_{ij}$ denotes the $(i,j)-$th basis coefficient of the polynomial representation. As interpolation points, the $N+1$ nodes of the Gauss-Legendre quadrature are chosen. Following, collocation is performed, i.e. the quadrature rule to approximate the integrals in Eq. (10) uses the same nodes as they are used for the polynomial basis of the solution (Eq. (11)). Additionally, the test functions ϕ are chosen to be the same as the ansatz functions, i.e. Π_N is the set of tensor-products of the one dimensional Lagrange polynomials of degree N.

After some algebraic manipulations, see [25, Eqs. (23)-(37)], the spatial DGSEM operator $\mathbf{R}_h^{(1)}(\mathbf{w})$ (which is an approximation of $\mathbf{w}_t$) in two dimensions for one element is given by

$$\begin{aligned}\left(\hat{\mathbf{w}}_{ij}\right)_t = \left(\mathbf{R}_h^{(1)}(\mathbf{w}_h)\right)_{ij} := -\frac{1}{J_{\text{geo}ij}}\Bigg[&\sum_{\lambda=0}^{N} \hat{\boldsymbol{\mathcal{F}}}^1_{\lambda j}\hat{D}_{i\lambda} + \left([\mathbf{f}^*\hat{s}]_j^{+\xi_1}\hat{\ell}_i(1) + [\mathbf{f}^*\hat{s}]_j^{-\xi_1}\hat{\ell}_i(-1)\right) \\ &+ \sum_{\mu=0}^{N} \hat{\boldsymbol{\mathcal{F}}}^2_{i\mu}\hat{D}_{j\mu} + \left([\mathbf{f}^*\hat{s}]_i^{+\xi_2}\hat{\ell}_j(1) + [\mathbf{f}^*\hat{s}]_i^{-\xi_2}\hat{\ell}_j(-1)\right)\Bigg] \quad \forall\, i,j. \end{aligned} \tag{12}$$

with the abbreviations

$$\hat{\ell}_i := \frac{\ell_i}{\omega_i}, \quad \text{and} \quad \hat{D}_{ij} := -\frac{\omega_i}{\omega_j}\frac{\partial \ell_i(\xi)}{\partial \xi}\bigg|_{\xi=\xi_j},$$

and the Jacobian of the geometrical transformation J_{geo}, see [25, Eqs. (5), (32) and (38)]. Note that ω_i denotes the quadrature weight of the Gauss-Legendre quadrature at position i. At the four edges of the element, which are denoted by $-\xi_1$, $+\xi_1$, $-\xi_2$ and $+\xi_2$, the transformed numerical flux $[\mathbf{f}^*\hat{s}]$ is evaluated, see [25, Eqs. (31)-(33)] for the definition of the transformation and the definition of the surface element $\hat{s}$. The numerical flux depends on the left and right values at the edge and the normal vector of the edge, i.e. $\mathbf{f}^* = \mathbf{f}^*(\mathbf{w}^L, \mathbf{w}^R, \mathbf{n})$. Here, we use a global Lax-Friedrichs flux

$$\mathbf{f}^*(\mathbf{w}^L, \mathbf{w}^R, \mathbf{n}) = \left(\frac{1}{2}\left(\mathbf{F}(\mathbf{w}^L) + \mathbf{F}(\mathbf{w}^R)\right) + \lambda\left(\mathbf{w}^L - \mathbf{w}^R\right)\right)\cdot\mathbf{n}, \tag{13}$$

where λ is a globally constant value. The left and right values $\mathbf{w}^L$, $\mathbf{w}^R$ are obtained by evaluation of the solution polynomial at the cell edges, see [25, Eq. (40)]. For more details on the derivation of the DGSEM, see [22, 25]. The novel scheme described in the current paper has been implemented in the open source code FLEXI[1]; for a recent overview on this code see [26].

3.1.2. Calculation of the Second Temporal Derivative with the DGSEM

For the considered two derivative time discretization methods, an approximation of $\mathbf{w}_{tt}$ is additionally required, see Eq. (4). In [18], the artificial quantity

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

has been introduced in order to facilitate the calculation of the second derivative term. Here, we follow this idea and start from the discretized equation Eq. (10) that we differentiate with respect to time t (see also Eq. (4) for comparison)

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

Following the same steps as outlined for $\mathbf{R}^{(1)}$, one obtains the DGSEM discretization of $\mathbf{R}^{(2)}$ as

$$\begin{aligned}\left(\hat{\mathbf{w}}_{ij}\right)_{tt} = \left(\mathbf{R}_h^{(2)}(\mathbf{w}_h,\boldsymbol{\sigma})\right)_{ij} := -\frac{1}{J_{\text{geo}ij}}\Bigg[&\sum_{\lambda=0}^{N} \frac{\partial\hat{\boldsymbol{\mathcal{F}}}^1_{\lambda j}}{\partial\hat{\mathbf{w}}}\hat{\boldsymbol{\sigma}}_{\lambda j}\hat{D}_{i\lambda} + \left([\widetilde{\mathbf{f}^*}\hat{s}]_j^{+\xi_1}\hat{\ell}_i(1) + [\widetilde{\mathbf{f}^*}\hat{s}]_j^{-\xi_1}\hat{\ell}_i(-1)\right) \\ &+ \sum_{\mu=0}^{N} \frac{\partial\hat{\boldsymbol{\mathcal{F}}}^2_{i\mu}}{\partial\hat{\mathbf{w}}}\boldsymbol{\sigma}_{i\mu}\hat{D}_{j\mu} + \left([\widetilde{\mathbf{f}^*}\hat{s}]_i^{+\xi_2}\hat{\ell}_j(1) + [\widetilde{\mathbf{f}^*}\hat{s}]_i^{-\xi_2}\hat{\ell}_j(-1)\right)\Bigg] \quad \forall\, i,j, \end{aligned} \tag{16}$$

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

where the numerical flux $\widetilde{\mathbf{f}}^* = \widetilde{\mathbf{f}}^*(\mathbf{w}^L, \mathbf{w}^R, \boldsymbol{\sigma}^L, \boldsymbol{\sigma}^R, \mathbf{n})$ is given by

$$\widetilde{\mathbf{f}}^*(\mathbf{w}^L, \mathbf{w}^R, \boldsymbol{\sigma}^L, \boldsymbol{\sigma}^R, \mathbf{n}) = \left(\frac{1}{2}\left(\frac{\partial \mathbf{F}(\mathbf{w}^L)}{\partial \mathbf{w}}\boldsymbol{\sigma}^L + \frac{\partial \mathbf{F}(\mathbf{w}^R)}{\partial \mathbf{w}}\boldsymbol{\sigma}^R\right) + \lambda\left(\boldsymbol{\sigma}^L - \boldsymbol{\sigma}^R\right)\right) \cdot \mathbf{n}.$$

Similar as it is done for $\mathbf{w}^{L/R}$, the values $\boldsymbol{\sigma}^{L/R}$ are obtained by evaluation of the solution polynomial at the cell edges. This completes the spatial discretization and allows us now to formulate the non-linear equation system to be solved for the time stepping procedure.

3.2. Solving the (Non-)Linear System of Equations

3.2.1. The Linear System for the Two-Derivative DGSEM

As we have introduced the artificial quantity $\boldsymbol{\sigma} := \mathbf{R}_h^{(1)}(\mathbf{w})$, we define an extended state vector $\boldsymbol{X} := (\hat{\mathbf{W}}, \hat{\boldsymbol{\sigma}})^T$, where $\hat{\mathbf{W}}$ and $\hat{\boldsymbol{\sigma}}$ contain the coefficients of the polynomial basis of $\mathbf{W}$ and $\boldsymbol{\sigma}$. Note that for ease of presentation, we omit the hat symbol ($\hat{\bullet}$) in the following. For this extended state vector, the residual (7) is modified and reads

$$\mathcal{G}(\boldsymbol{X}) = \begin{pmatrix}\mathcal{G}_1(\boldsymbol{X})\\ \mathcal{G}_2(\boldsymbol{X})\end{pmatrix} := \begin{pmatrix}\mathbf{W}\\ \boldsymbol{\sigma}\end{pmatrix} - \begin{pmatrix}\mathbf{b}\\ 0\end{pmatrix} - \begin{pmatrix}\mathbf{g}(\mathbf{W})\\ \mathbf{R}_h^{(1)}(\mathbf{W})\end{pmatrix} \stackrel{!}{=} 0.$$

For the definitions of $\mathbf{W}$, $\mathbf{b}$ and $\mathbf{g}(\mathbf{W})$ see Eq. (8) and Eq. (9) – of course with $\mathbf{R}^{(1)}$ and $\mathbf{R}^{(2)}$ replaced by their discrete counterparts $\mathbf{R}_h^{(1)}$ and $\mathbf{R}_h^{(2)}$, respectively. To solve for $\boldsymbol{X}$, Newton's method is applied which consists of $\mathtt{r}$ iterative solves

$$\begin{aligned}\frac{\partial \mathcal{G}(\boldsymbol{X}^{\mathtt{r}})}{\partial \boldsymbol{X}} \cdot \Delta \boldsymbol{X} &= -\mathcal{G}(\boldsymbol{X}^{\mathtt{r}})\\ \boldsymbol{X}^{\mathtt{r}+1} &= \boldsymbol{X}^{\mathtt{r}} + \Delta \boldsymbol{X},\end{aligned} \tag{17}$$

with the Newton increment $\Delta \boldsymbol{X} := (\Delta \mathbf{W}, \Delta \boldsymbol{\sigma})^T$. To ease presentation, we will drop the superscript $\mathtt{r}$ in the following. The matrix-vector product of the arising linear system for each Newton's iteration with the system matrix $\mathcal{J}$ is given by

$$\begin{aligned}\mathcal{J}\Delta \boldsymbol{X} := \frac{\partial \mathcal{G}(\boldsymbol{X})}{\partial \boldsymbol{X}} \cdot \Delta \boldsymbol{X} &= \Delta \boldsymbol{X} - \frac{\partial}{\partial \boldsymbol{X}}\begin{pmatrix}\alpha_1 \Delta t \mathbf{R}_h^{(1)}(\mathbf{W}) - \frac{\alpha_2 \Delta t^2}{2}\mathbf{R}_h^{(2)}(\mathbf{W}, \boldsymbol{\sigma})\\ \mathbf{R}_h^{(1)}(\mathbf{W})\end{pmatrix}\Delta \boldsymbol{X}\\ &= \begin{pmatrix}\mathrm{Id} - \alpha_1 \Delta t \frac{\partial \mathbf{R}_h^{(1)}}{\partial \mathbf{W}} + \frac{\alpha_2 \Delta t^2}{2}\frac{\partial \mathbf{R}_h^{(2)}}{\partial \mathbf{W}} & \frac{\alpha_2 \Delta t^2}{2}\frac{\partial \mathbf{R}_h^{(2)}}{\partial \boldsymbol{\sigma}}\\ -\frac{\partial \mathbf{R}_h^{(1)}}{\partial \mathbf{W}} & \mathrm{Id}\end{pmatrix}\begin{pmatrix}\Delta \mathbf{W}\\ \Delta \boldsymbol{\sigma}\end{pmatrix}.\end{aligned} \tag{18}$$

Due to the definition of $\boldsymbol{\sigma}$, see Eq. (14), two important observations can be made for Eq. (18):

- A comparison of Eq. (12) and Eq. (16) directly shows $\frac{\partial \mathbf{R}_h^{(2)}}{\partial \sigma} \equiv \frac{\partial \mathbf{R}_h^{(1)}}{\partial \mathbf{W}}$.
- If the flux is linear, the Hessian contribution is zero, i.e. $\frac{\partial \mathbf{R}_h^{(2)}}{\partial \mathbf{W}} = 0$.

Note that if the discretized system of equations (1) has a linear flux, Newton's method is not required and one can directly solve the linear equation for $\boldsymbol{X}$ instead of $\Delta \boldsymbol{X}$. Nevertheless, utilizing Newton's method can have a beneficial influence on the achieved accuracy, see e.g. [27]. Therefore, we use Newton's method regardless if the system is linear or non-linear.

3.2.2. Notes on the Practical Implementation

For the implementation of the system matrix $\mathcal{J}$, one has to calculate the matrices $\frac{\partial \mathbf{R}_h^{(1)}}{\partial \mathbf{W}}$ and $\frac{\partial \mathbf{R}_h^{(2)}}{\partial \mathbf{W}}$ and assemble them into $\mathcal{J}$. The former is the standard system matrix of the DGSEM, where a detailed derivation of the inner-element dependencies can be found in [28, 29]. For the calculation of the latter, all routines of the standard system matrix can be reused. The only difference is that the physical flux Jacobi matrices have to be substituted by the physical flux Hessian. We solve the linear system with a GMRES method using the PETSc library [30].

3.2.3. A Novel Preconditioner for the Extended Linear System: BJ$_{ext}$

For a better convergence of the GMRES method, a preconditioner can be applied. Typical preconditioners are left ILU(0) preconditioning or an element-wise block-Jacobi preconditioner (BJ). Alternatively, the special structure of the linear system, see Eq. (18), can be exploited to construct a problem-tailored preconditioner. In the following, two slightly different preconditioners are presented, which either take the Hessian contribution in $\mathcal{J}$ into account (BJ$^H_{\text{ext}}$) or neglect this contribution (BJ$_{\text{ext}}$).

BJ$^H_{ext}$ Preconditioner. We start with the idea of using an element-wise block-Jacobi preconditioner and derive an extended block-Jacobi preconditioner. That means that we build up the contributions to the system matrix without considering dependencies of neighboring elements, i.e. only the element-internal influence is taken into account. This preconditioner matrix P is visualized in Fig. 1 on the left. For the construction of P, the four different blocks are given

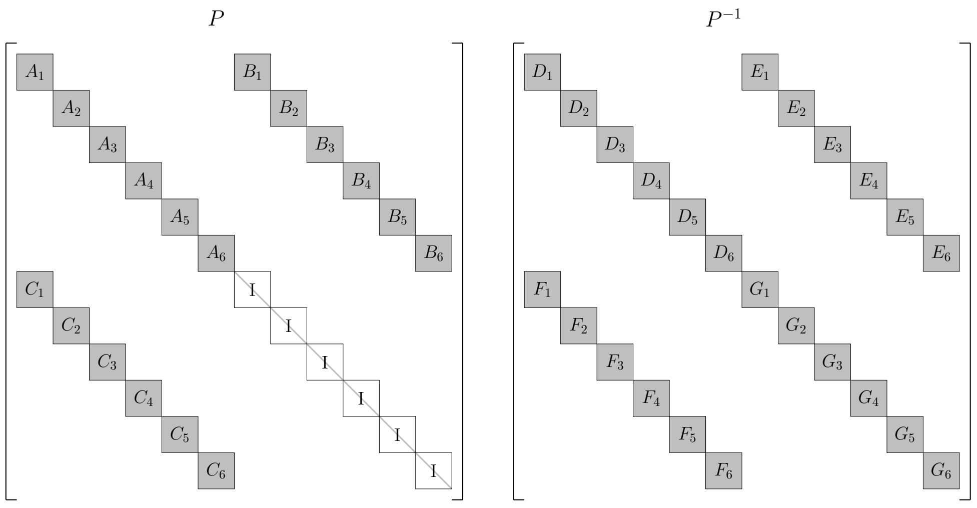

Figure 1. Sketch of extended block-Jacobi preconditioner matrix P (left) and inverse of preconditioner matrix P^{-1} (right) for a setup with six spatial discretization elements ($n_E = 6$).

by

$$A_i = A_i^H := \mathrm{Id} - \alpha_1 \Delta t \frac{\partial \mathbf{R}_h^{(1)}}{\partial \mathbf{W}} + \frac{\alpha_2 \Delta t^2}{2} \frac{\partial \mathbf{R}_h^{(2)}}{\partial \mathbf{W}}\bigg|_i, \quad B_i := \frac{\alpha_2 \Delta t^2}{2} \frac{\partial \mathbf{R}_h^{(2)}}{\partial \boldsymbol{\sigma}}\bigg|_i = \frac{\alpha_2 \Delta t^2}{2} \frac{\partial \mathbf{R}_h^{(1)}}{\partial \mathbf{W}}\bigg|_i, \quad C_i := -\frac{\partial \mathbf{R}_h^{(1)}}{\partial \mathbf{W}}\bigg|_i, \quad i = 1, \ldots, n_E,$$

with n_E denoting the number elements of the spatial discretization. As we have neglected all element-neighbor dependencies and due to the special structure of P, we can directly find the inverse P^{-1} which also consists of four different types of blocks, visualized in Fig. 1, via

$$D_i^H = (A_i - B_i C_i)^{-1}, \quad E_i^H = -(A_i - B_i C_i)^{-1} \cdot B_i, \quad F_i^H = -C_i \cdot (A_i - B_i C_i)^{-1} \text{ and } \quad G_i^H = \mathrm{Id} + C_i \cdot (A_i - B_i C_i)^{-1} \cdot B_i,$$

where the superscript $(\bullet)^H$ indicates that the Hessian contribution has been considered in P. These block matrix contributions can be calculated independently for all elements. Moreover, one can see that they require only one matrix inversion of one element-sized matrix per element, viz.

$$(A_i - B_i C_i)^{-1} = \left(\mathrm{Id} - \alpha_1 \Delta t \frac{\partial \mathbf{R}_h^{(1)}}{\partial \mathbf{W}}\bigg|_i + \frac{\alpha_2 \Delta t^2}{2} \frac{\partial \mathbf{R}_h^{(2)}}{\partial \mathbf{W}}\bigg|_i + \frac{\alpha_2 \Delta t^2}{2} \left(\frac{\partial \mathbf{R}_h^{(1)}}{\partial \mathbf{W}}\bigg|_i \right)^2 \right)^{-1}.$$

As these matrices are relatively small, we calculate those inverses via LU-decomposition.

BJ_{ext} Preconditioner. Alternatively, we define the preconditioner $\mathrm{BJ}_{\mathrm{ext}}$ which neglects the Hessian in P, i.e.

$$A_i := \mathrm{Id} - \alpha_1 \Delta t \left.\frac{\partial \mathbf{R}_h^{(1)}}{\partial \mathbf{W}}\right|_i, \quad i = 1, \dots, n_E,$$

This reveals that only $\left.\frac{\partial \mathbf{R}_h^{(1)}}{\partial \mathbf{W}}\right|_i$ has to be calculated, which is the same as for standard one-derivative methods such as the implicit Euler method. Again, the inverse of the preconditioner P^{-1} consists of four different types of blocks which are now given by

$$D_i = (A_i - B_i C_i)^{-1}, \quad E_i = -B_i \cdot (A_i - B_i C_i)^{-1}, \quad F_i = -C_i \cdot (A_i - B_i C_i)^{-1} \text{ and } G_i = A_i \cdot (A_i - B_i C_i)^{-1}.$$

For the calculation of those block matrices, we have exploited the fact that there holds $A_i B_i = B_i A_i$ and $B_i C_i = C_i B_i$, if the Hessian contribution in A_i is neglected. Again, only one inverse per element has to be calculated, which is

$$(A_i - B_i C_i)^{-1} = \left(\mathrm{Id} - \alpha_1 \Delta t \left.\frac{\partial \mathbf{R}_h^{(1)}}{\partial \mathbf{W}}\right|_i + \frac{\alpha_2 \Delta t^2}{2} \left(\left.\frac{\partial \mathbf{R}_h^{(1)}}{\partial \mathbf{W}}\right|_i \right)^2 \right)^{-1}.$$

This preconditioner has two advantages compared to the $\mathrm{BJ}^H_{\mathrm{ext}}$ preconditioner: During the calculation of E_i and G_i, the number of matrix-matrix multiplication can be reduced by one. This decreases the building costs of the preconditioner. Additionally, the implementation of the preconditioner is simpler as the Hessian does not have to be implemented. The drawback compared to the $\mathrm{BJ}^H_{\mathrm{ext}}$ preconditioner is that P is a slightly worse approximation to $\mathcal{J}$. Both alternatives will be compared in Sec. 4.1.2.

3.2.4. Reducing the Problem Size with the Schur Complement

Another option to reduce the computational and implementational effort is to exploit the special structure of $\mathcal{J}$ by forming a Schur complement to reduce the problem size. Instead of solving Eq. (17), the problem can be reduced to

$$\begin{aligned}
\mathcal{J}^{\mathtt{r}}_{\mathrm{Schur}} \Delta \mathbf{w} &= -\mathcal{G}_1(X^{\mathtt{r}}) + \frac{\alpha_2 \Delta t^2}{2} \frac{\partial \mathbf{R}_h^{(2)}}{\partial \boldsymbol{\sigma}} \cdot \mathcal{G}_2(X^{\mathtt{r}}) \\
\Delta \boldsymbol{\sigma} &= -\mathcal{G}_2(X^{\mathtt{r}}) + \frac{\partial \mathbf{R}_h^{(1)}}{\partial \mathbf{W}} \Delta \mathbf{w} \\
X^{\mathtt{r}+1} &= X^{\mathtt{r}} + \Delta X,
\end{aligned}$$

where $\mathtt{r}$ again denotes the index of the Newton iterate, which will be omitted in the following. The Schur complement system matrix $\mathcal{J}_{\mathrm{Schur}}$ is given by

$$\mathcal{J}_{\mathrm{Schur}} := \mathrm{Id} - \alpha_1 \Delta t \frac{\partial \mathbf{R}_h^{(1)}}{\partial \mathbf{W}} + \frac{\alpha_2 \Delta t^2}{2} \frac{\partial \mathbf{R}_h^{(2)}}{\partial \mathbf{W}} + \frac{\alpha_2 \Delta t^2}{2} \frac{\partial \mathbf{R}_h^{(2)}}{\partial \boldsymbol{\sigma}} \frac{\partial \mathbf{R}_h^{(1)}}{\partial \mathbf{W}} = \mathrm{Id} - \alpha_1 \Delta t \frac{\partial \mathbf{R}_h^{(1)}}{\partial \mathbf{W}} + \frac{\alpha_2 \Delta t^2}{2} \frac{\partial \mathbf{R}_h^{(2)}}{\partial \mathbf{W}} + \frac{\alpha_2 \Delta t^2}{2} \left(\frac{\partial \mathbf{R}_h^{(1)}}{\partial \mathbf{W}} \right)^2. \tag{19}$$

The advantage of this approach is that while $\mathcal{J} \in \mathbb{R}^{2\cdot\mathrm{size}(\mathbf{w}) \times 2\cdot\mathrm{size}(\mathbf{w})}$, the Schur complement system matrix is only $\mathcal{J}_{\mathrm{Schur}} \in \mathbb{R}^{\mathrm{size}(\mathbf{w}) \times \mathrm{size}(\mathbf{w})}$. One drawback of this approach is that the square of DGSEM's system matrix has to be calculated.

In the following, we first consider the linear solver's convergence for the case with and without reducing the problem size with the Schur complement. This allows us to choose the better suited approach. Moreover, the effectiveness of the different preconditioning strategies is investigated. Finally, we illustrate the capabilities of the novel schemes by showing the experimental order of convergence.

4. Numerical Investigations

In this section, some properties of the novel scheme are evaluated. We investigate if reducing the size of the system matrix with a Schur complement is beneficial and evaluate the best suited preconditioner. Moreover, it is shown how a matrix-free approach can be used to facilitate the implementation of the novel method. The linear scalar advection and the non-linear Euler equations of gas dynamics are taken as prototypical hyperbolic PDEs.

4.1. Convergence of the Linear Solver

4.1.1. Linear Scalar Advection

We start by considering the linear scalar advection equation

$$w_t + \nabla_x \cdot (\mathbf{a}w) = 0,$$

with the advection velocity $\mathbf{a}$ being a constant vector. The initial condition $w(\mathbf{x}, t = 0) = \sin(\pi\mathbf{x})$ is chosen, leading to the solution

$$w(\mathbf{x}, t) = \sin\left(\pi\left(\mathbf{x} - \mathbf{a}t\right)\right). \tag{20}$$

The advection velocity is set to $\mathbf{a} = (0.3, 0.3)^T$ on the two dimensional domain $\Omega = [-1, 1]^2$. For the numerical flux function, see Eq. (13), we choose $\lambda = |\mathbf{a} \cdot \mathbf{n}|$, with $\mathbf{n}$ being the normal vector of the element's face.

The domain is discretized with $n_E = 16^2$ elements with a polynomial degree of the ansatz functions of $N = 5$. To get a better understanding of the convergence behavior of the novel scheme, we consider the required amount of GMRES iterations with left ILU(0) preconditioning for three different cases and a varying timestep size:

- $\mathcal{J}^{\text{predictor}} = \begin{pmatrix} \text{Id} - \frac{\Delta t}{2}\frac{\partial \mathbf{R}_h^{(1)}}{\partial \mathbf{W}} & \frac{\Delta t^2}{12}\frac{\partial \mathbf{R}_h^{(1)}}{\partial \mathbf{W}} \\ -\frac{\partial \mathbf{R}_h^{(1)}}{\partial \mathbf{W}} & \text{Id} \end{pmatrix}$, which arises from the HBPC$(q, k_{\max})$ predictor, see Eq. (18),
- $\mathcal{J}^{\text{predictor}}_{\text{Schur}} = \text{Id} - \frac{\Delta t}{2}\frac{\partial \mathbf{R}_h^{(1)}}{\partial \mathbf{W}} + \frac{\Delta t^2}{12}\left(\frac{\partial \mathbf{R}_h^{(1)}}{\partial \mathbf{W}}\right)^2$, which arises from the HBPC$(q, k_{\max})$ predictor, using the Schur complement, see Eq. (19), and
- $\mathcal{J}^{\text{Euler}} = \text{Id} - \Delta t\frac{\partial \mathbf{R}_h^{(1)}}{\partial \mathbf{W}}$, which arises from the first order implicit Euler method as a comparison.

Note that even though the problem is linear, we use Newton's method to solve the problem in order to have the same setup as for the non-linear case. The relative convergence criteria are chosen to be $\varepsilon_{\text{GMRES}} = 10^{-3}$ and $\varepsilon_{\text{Newton}} = 10^{-8}$. The maximum number of Krylov subspaces of the GMRES method is chosen to be large enough such that no restart of the GMRES method is performed.

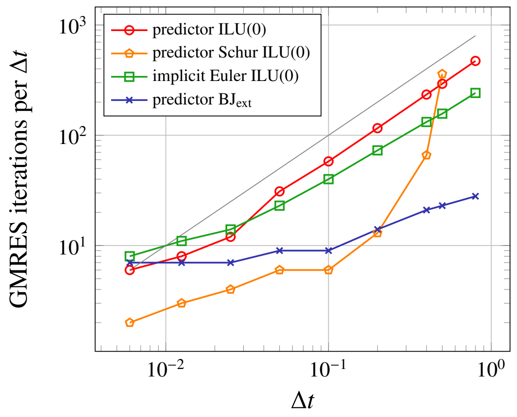

Figure 2. Linear scalar advection equation: required iterations of linear solver to complete one timestep for varying timestep sizes; gray line indicates slope = 1.

Fig. 2 shows the required GMRES iterations per timestep for the three different schemes. The gray line is a reference which indicates slope one. That means, if the slope of a scheme is lower than the reference, one can expect an acceleration of the calculation when choosing a larger timestep size. If the slope is larger than the reference, a speed-up can only be achieved if the reduction in computational costs for setting up the Jacobian and the preconditioner counterbalances the increased costs due to the increased amount of iterations. First, we compare the implicit Euler method and the predictor of the HBPC$(q, k_{\max})$ method with ILU(0) preconditioning each. One can see that the slope for the predictor is slightly steeper than for the implicit Euler scheme and is very close to one. The slope and the number of iterations of the predictor can be reduced drastically by using the $\mathrm{BJ}_{\mathrm{ext}}$ preconditioner. Reducing the problem size with the Schur complement has only a favorable influence for small timesteps. For larger timesteps the iteration count suddenly increases very rapidly and leads to non-convergence for large timesteps. Note that we have observed the same qualitative behavior when using a standard element-wise BJ preconditioner instead of ILU(0) preconditioning for the Schur complement matrix. Summing up, choosing the $\mathrm{BJ}_{\mathrm{ext}}$ preconditioner for the system arising from the HBPC($q, k_{\max}$) predictor without reducing the system size with a Schur complement seems to be the a good choice for a wide range of timestep sizes. The Schur complement discretization is only beneficial for relatively small timesteps.

4.1.2. Euler Equations

Next, we consider the Euler equations of gas dynamics in two dimensions as an example for a non-linear equation system, which read

$$\mathbf{W}_t + \nabla_x \cdot \mathbf{F}(\mathbf{W}) = 0, \quad \text{with} \quad \mathbf{W} = \begin{pmatrix} \rho \\ \rho\mathbf{v} \\ E \end{pmatrix} \quad \text{and} \quad \mathbf{F}(\mathbf{W}) = \begin{pmatrix} \rho\mathbf{v} \\ \rho\mathbf{v}\otimes\mathbf{v} + \frac{1}{\varepsilon^2} p \cdot \mathrm{Id} \\ \mathbf{v}(E+p), \end{pmatrix}, \tag{21}$$

with the density ρ, velocity $\mathbf{v} = (v_1, v_2)^T$, energy E and the reference Mach number ε. The pressure p is calculated via the equation of state for a perfect gas

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

with the isentropic coefficient $\gamma = 1.4$. For the numerical flux function (Eq. (13)), following [31, 32] we choose $\lambda = \mathrm{diag}(\frac{1}{\varepsilon}, 1, 1, \frac{1}{\varepsilon})$. The considered test setup is an extension of the one for the linear scalar advection, see Eq. (20). The initial conditions are hence given by

$$\rho(\mathbf{x}, t) = 1 + 0.3\sin\left(\pi(\mathbf{x} - \mathbf{a}t)\right), \qquad \mathbf{v} = \mathbf{a}, \qquad p = 1, \tag{22}$$

with $\mathbf{a} := (0.3.0.3)^T$ on domain $\Omega = [-1, 1]^2$. The domain is discretized with $n_E = 16^2$ elements with a polynomial degree of the ansatz functions of $N = 5$. The final time is set to $T_{\mathrm{end}} = 0.8$ and convergence tolerances are chosen to be $\varepsilon_{\mathrm{GMRES}} = 10^{-3}$ and $\varepsilon_{\mathrm{Newton}} = 10^{-8}$. The GMRES method performs a restart if no convergence has been reached after 700 iterations.

Similar as for the linear case, see Sec. 4.1.1, we consider the three different system matrices $\mathcal{J}^{\mathrm{predictor}}$, $\mathcal{J}^{\mathrm{predictor}}_{\mathrm{Schur}}$ and $\mathcal{J}^{\mathrm{Euler}}$ to evaluate the convergence properties of the schemes for the non-linear case. Differently as for the linear case, the Hessian contribution is not zero ($\partial\mathbf{R}_h^{(2)}/\partial\mathbf{W} \neq 0$). We therefore additionally investigate the influence on the required iterations when neglecting this contribution in the system matrix.

Influence of Chosen System Matrix. The required linear iterations for those setups for a varying reference Mach number ε are visualized in Fig. 3. One can see an almost linear behavior for the HBPC($q, k_{\max}$) predictor and the implicit Euler scheme. Comparing the predictor and the implicit Euler scheme (both with ILU(0) preconditioner), shows that while the slope of the curve for the implicit Euler scheme remains below one that of the predictor is always slightly larger than one. Going to higher stiffnesses ($\varepsilon = 10^{-2}$) shows that the slope even increases for the predictor.

Reducing the system size with the Schur complement, reveals a strongly non-linear behavior of the required amount of iterations. Starting from a certain threshold, the required iterations increase rapidly such that no solution is obtained already for moderately large timesteps. In both cases, neglecting the Hessian contribution has only a minor influence on the required iterations for moderately large timesteps. For very large timesteps, the linear scaling of the iteration number can only be obtained when taking the Hessian contribution into account. This can consistently be observed for all considered stiffnesses. Therefore, the Hessian contribution in the system matrix is always taken into account in the remainder of this paper.

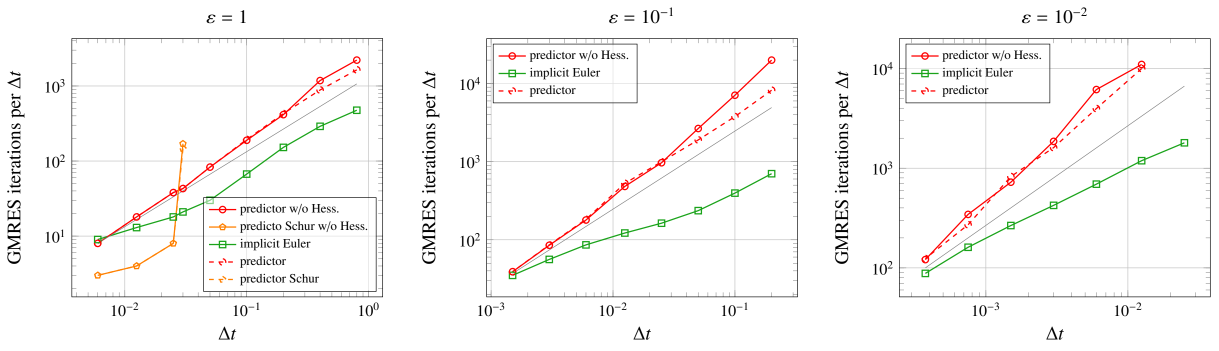

Figure 3. Euler equations: required iterations of linear solver to complete one timestep for varying timestep sizes for different reference Mach numbers ε with and without Hessian contribution; gray line indicates slope = 1. For all schemes an ILU(0) preconditioner is used. Note that missing data points indicate that no convergence of the linear solver could be obtained within a limit of 7000 GMRES iterations per Newton step.

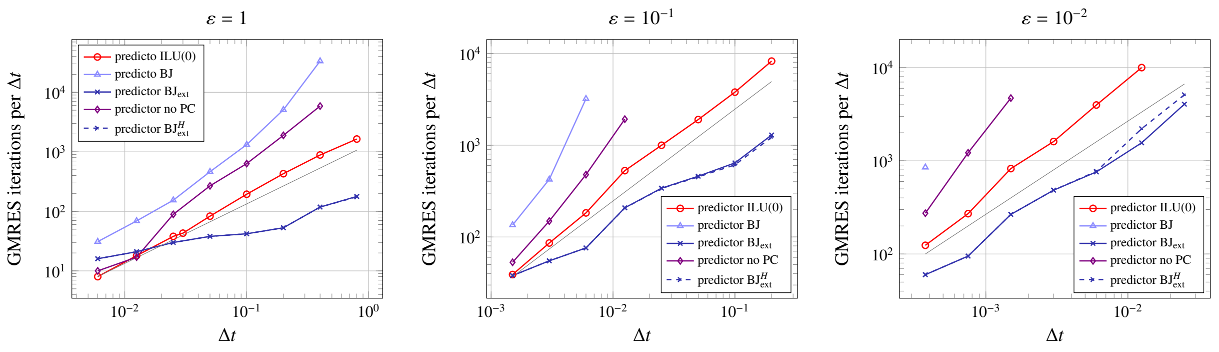

Figure 4. Influence of reference Mach number on required iterations to complete one timestep for different preconditioners for predictor of the HBPC$(q, k_{\max})$ scheme; gray line indicates slope = 1. Note that missing data points indicate that no convergence of the linear solver could be obtained within a limit of 7000 GMRES iterations per Newton step. The values for the predictor with $\mathrm{BJ}_{\mathrm{ext}}$ and the predictor with $\mathrm{BJ}^{H}_{\mathrm{ext}}$ are almost not distinguishable.

Influence of Preconditioner. Additionally, we investigate the influence of the used preconditioner on the required iterations without using the Schur complement. For that purpose, we repeat the previous simulations with the Hessian contribution choosing different preconditioners. The preconditioners are rebuilt before each Newton iteration. We report the results of this series of simulations in Fig. 4. The figure shows that the novel $\mathrm{BJ}_{\mathrm{ext}}$ and $\mathrm{BJ}^{H}_{\mathrm{ext}}$ preconditioners are best suited to reduce the required amount of iterations. They even perform better than the ILU(0) preconditioner, which performs better than using no preconditioner. Interestingly, choosing the standard element-wise BJ preconditioner has an unfavorable influence on the required iterations. One can observe that the slope of the curves increases for an increasing stiffness. A similar behavior has already been observed in [33]. To cure this, one could use adaptive tolerances for the GMRES and Newton's method. Another option is to use asymptotic preserving schemes as it is e.g. done in [33].

As the efficiency of a preconditioner cannot only be evaluated by its ability to reduce the number of iterations but also by its computational costs, we additionally consider the required wallclocktimes for this set of simulations, which are visualized in Fig. 5. One can see a very similar behavior as already observed for the linear iterations when considering the slope of the curves: Choosing the $\mathrm{BJ}_{\mathrm{ext}}/\mathrm{BJ}^{H}_{\mathrm{ext}}$ preconditioner gives the most efficient scheme. When the ILU(0) preconditioner is chosen, in only a few cases a speedup can be achieved by selecting a larger timestep size. For the $\mathrm{BJ}_{\mathrm{ext}}/\mathrm{BJ}^{H}_{\mathrm{ext}}$ preconditioner, a (super-) linear scaling is only observed for large timestep sizes. Regarding the absolute values of the wallclocktime reveals the different costs of the preconditioners. Due to the missing costs of building and applying the preconditioner when choosing no PC, for relatively small timesteps this variant is the most efficient. Still, a good preconditioner such as the novel $\mathrm{BJ}_{\mathrm{ext}}$ preconditioner is required to obtain an efficient scheme for large timesteps. The $\mathrm{BJ}_{\mathrm{ext}}/\mathrm{BJ}^{H}_{\mathrm{ext}}$ preconditioners are relatively expensive and hence for small timesteps

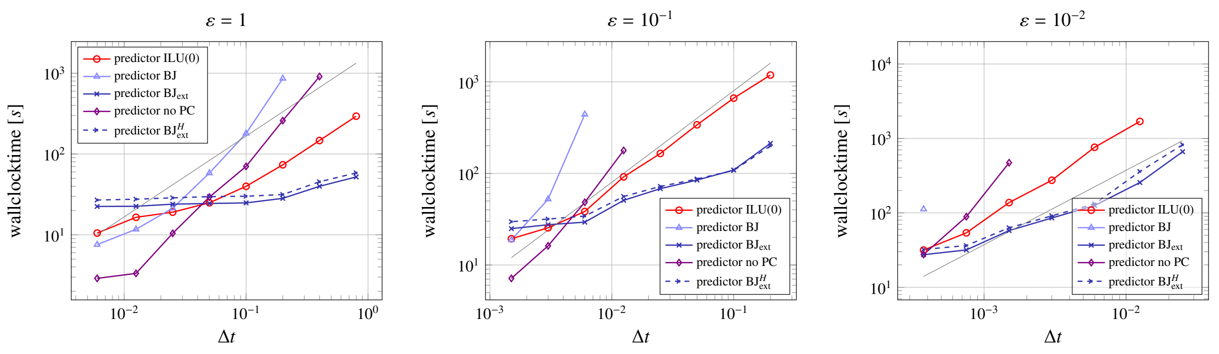

Figure 5. Influence of reference Mach number on required wallclocktime to complete one timestep for different preconditioners for the predictor of the HBPC(q, k_{max}) scheme; gray line indicates slope = 1. Note that missing data points indicate that no convergence of the linear solver could be obtained within a limit of 7000 GMRES iterations per Newton step.

and low stiffnesses are outperformed by the other preconditioners. In a practical application, the large building costs can be reduced by reusing the preconditioner for several Newton steps or even several timesteps. Comparing the novel BJ$_{\text{ext}}$ and BJ$^{H}_{\text{ext}}$ preconditioner shows that neglecting the Hessian contribution increases the efficiency of the preconditioner. While this has only a small influence on the required iterations, see Fig. 4, the reduced cost for building the preconditioner has a significant influence on the required wallclocktimes. Therefore, this approach is pursued in the remainder of this paper.

4.2. Matrix-Free Discretization

A matrix-free implementation allows to increase the flexibility of a solver and reduces the required memory consumption. Moreover, it has turned out that is is also beneficial in terms of computational time for high order discretizations of complex settings [34]. For the two-derivative DGSEM discretization it additionally facilitates a parallel distribution of the spatial domain on different processors.

Matrix-Free Approach. Differently to the previously described matrix-based approach one does not explicitly form the system matrix $\mathcal{J}$ and multiply the vector ΔX. Instead, one approximates the matrix vector product $\mathcal{J}\Delta X$ via a finite difference, see e.g. [35] for an overview on matrix-free approaches. For the second derivative method, the matrix-vector product given in Eq. (18) for a non-linear system is approximated by

$$\mathcal{J}\Delta X \approx \begin{pmatrix} \Delta \mathbf{W} - \alpha_1 \Delta t \frac{\mathbf{R}_h^{(1)}(\mathbf{W}+\varepsilon^w_{\text{FD}}\Delta\mathbf{W}) - \mathbf{R}_h^{(1)}(\mathbf{W})}{\varepsilon^w_{\text{FD}}} + \frac{\alpha_2\Delta t^2}{2}\frac{\mathbf{R}_h^{(2)}(\mathbf{W}+\varepsilon^w_{\text{FD}}\Delta\mathbf{W},\boldsymbol{\sigma}) - \mathbf{R}_h^{(2)}(\mathbf{W},\boldsymbol{\sigma})}{\varepsilon^w_{\text{FD}}} + \frac{\alpha_2\Delta t^2}{2}\frac{\mathbf{R}_h^{(2)}(\mathbf{W},\boldsymbol{\sigma}+\varepsilon^\sigma_{\text{FD}}\Delta\boldsymbol{\sigma}) - \mathbf{R}_h^{(2)}(\mathbf{W},\boldsymbol{\sigma})}{\varepsilon^\sigma_{\text{FD}}} \\ -\frac{\mathbf{R}_h^{(1)}(\mathbf{W}+\varepsilon^w_{\text{FD}}\Delta\mathbf{W}) - \mathbf{R}_h^{(1)}(\mathbf{W})}{\varepsilon^w_{\text{FD}}} + \Delta\boldsymbol{\sigma} \end{pmatrix}, \tag{23}$$

with $\varepsilon^{(\bullet)}_{\text{FD}}$ being a small user-defined value. Inspired by [35], we set

$$\varepsilon^{(\bullet)}_{\text{FD}} := \frac{\sqrt{\varepsilon_{\text{machine}}}}{\varepsilon \|\Delta(\bullet)\|_2},$$

with $\varepsilon_{\text{machine}}$ being approximately machine accuracy. Here, we use the fortran intrinsic *epsilon* function, which gives approximately $\varepsilon_{\text{machine}} \approx 2\cdot 10^{-16}$. Additionally, both the reference Mach number and the perturbed quantity are taken into account, so $(\bullet)$ is either $(\bullet) = \mathbf{W}$ or $(\bullet) = \boldsymbol{\sigma}$. For a linear system, the matrix-vector product simplifies to

$$\mathcal{J}\Delta X = \begin{pmatrix} \Delta \mathbf{W} - \alpha_1 \Delta t \frac{\mathbf{R}_h^{(1)}(\mathbf{W}+\varepsilon^w_{\text{FD}}\Delta\mathbf{W}) - \mathbf{R}_h^{(1)}(\mathbf{W})}{\varepsilon^w_{\text{FD}}} + \frac{\alpha_2\Delta t^2}{2}\frac{\mathbf{R}_h^{(1)}(\boldsymbol{\sigma}+\varepsilon^\sigma_{\text{FD}}\Delta\boldsymbol{\sigma}) - \mathbf{R}_h^{(1)}(\boldsymbol{\sigma})}{\varepsilon^\sigma_{\text{FD}}} \\ -\frac{\mathbf{R}_h^{(1)}(\mathbf{W}+\varepsilon^w_{\text{FD}}\Delta\mathbf{W}) - \mathbf{R}_h^{(1)}(\mathbf{W})}{\varepsilon^w_{\text{FD}}} + \Delta\boldsymbol{\sigma} \end{pmatrix} = \begin{pmatrix} \Delta\mathbf{W} - \alpha_1\Delta t \mathbf{R}_h^{(1)}(\Delta\mathbf{W}) + \frac{\alpha_2\Delta t^2}{2}\mathbf{R}_h^{(1)}(\Delta\boldsymbol{\sigma}) \\ -\mathbf{R}_h^{(1)}(\Delta\mathbf{W}) + \Delta\boldsymbol{\sigma} \end{pmatrix}. \tag{24}$$

Eq. (23) and Eq. (24) show that two or three different operator evaluations have to be performed per GMRES iteration for the linear or non-linear case, respectively.

Parallel Implementation. One advantage of using a matrix-free implementation is the straight forward parallelization. Differently to the matrix-based approach, the Jacobian $\mathcal{J}$ does not have to be distributed among the processors. While the parallelization relies on a domain decomposition, and $\mathcal{J}$ consists of four blocks, each corresponding to the whole spatial discretization, a parallel matrix-based implementation would require some restructuring of already existing codes. One would reorder the solution vector $\boldsymbol{X}$ such that $\mathbf{W}$ and $\boldsymbol{\sigma}$ of each element are alternating in the solution vector. Most importantly, the assembling routines and the routines to evaluate the right hand side would have to be rewritten, which is a tedious task.

Instead, we can distribute $\mathbf{W}$ and $\boldsymbol{\sigma}$ in a similar manner on the processors, such that for both variables the same domain decomposition is used. I.e. one processor contains the same segments of $\mathbf{W}$ *and* $\boldsymbol{\sigma}$, which are then assembled consecutively in the solution vector. With this, the parallelization of the spatial operator does not have to be changed, see [26] for an overview on the parallelization strategy for the FLEXI DGSEM code.

As the ILU(0) preconditioner requires the information of the whole matrix, it is not in line with the matrix-free approach. Differently, the $\mathrm{BJ}_{\mathrm{ext}}$ and $\mathrm{BJ}_{\mathrm{ext}}^{H}$ preconditioners require only the formation of element-wise small matrices which can be set up independently of each other. Additionally, the application of the preconditioner does not require any communication between different processors. It consists of independent evaluations of matrix-vector products

$$P^{-1} \cdot \boldsymbol{X} = \begin{pmatrix} D_1 \cdot \mathbf{W}_1 + E_1 \cdot \boldsymbol{\sigma}_1 \\ \vdots \\ D_{n_E} \cdot \mathbf{W}_{n_E} + E_{n_E} \cdot \boldsymbol{\sigma}_{n_E} \\ F_1 \cdot \mathbf{W}_1 + G_1 \cdot \boldsymbol{\sigma}_1 \\ \vdots \\ F_{n_E} \cdot \mathbf{W}_{n_E} + G_{n_E} \cdot \boldsymbol{\sigma}_{n_E} \end{pmatrix},$$

and is hence well suited for parallel computations. A dedicated evaluation of the parallel performance of the novel method will be addressed in a future work.

Comparison of Matrix-Free and Matrix-Based Implementation. In order to validate the matrix-free implementation and to illustrate the influence of using an approximation of the Jacobian-vector product on the required iterations, we compare matrix-free and matrix-based simulations. We use the same initialization for the Euler equations (Eq. (22)) with the same discretization parameters as in Sec. 4.1.2. In Fig. 6 the matrix-free and the matrix-based approach

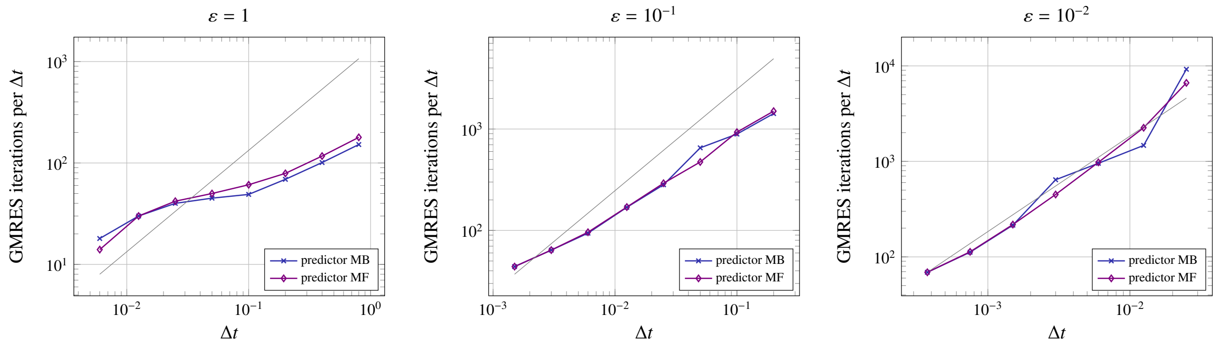

Figure 6. Influence on required GMRES iterations of choosing a matrix-based (MB) or a matrix-free (MF) discretization. To improve convergence, right $\mathrm{BJ}_{\mathrm{ext}}$ preconditioning is applied.

are compared regarding the required linear iterations. One can see that the required iterations are very similar. This illustrates that for this setup, a matrix-free approach can be chosen which allows for an efficient spatial parallelization. Consequently, the results in the remainder of this paper are obtained with the matrix-free approach choosing the $\mathrm{BJ}_{\mathrm{ext}}$ preconditioner.

4.3. Experimental Order of Convergence

4.3.1. Linear Scalar Advection

In order to validate the implementation and to illustrate the high order accuracy of the HBPC($q, k_{\max}$) schemes, we perform a set of simulations with fixed spatial discretization where we vary the timestep size and the time discretization method. The spatial domain is discretized with $n_E = 32^2$ elements with $N = 7$. The final time of the simulation is set to $T_{\text{end}} = 0.8$ and the convergence criteria for the GMRES method and Newton's method are set to $\varepsilon_{\text{GMRES}} = 10^{-5}$ and $\varepsilon_{\text{Newton}} = 10^{-12}$.

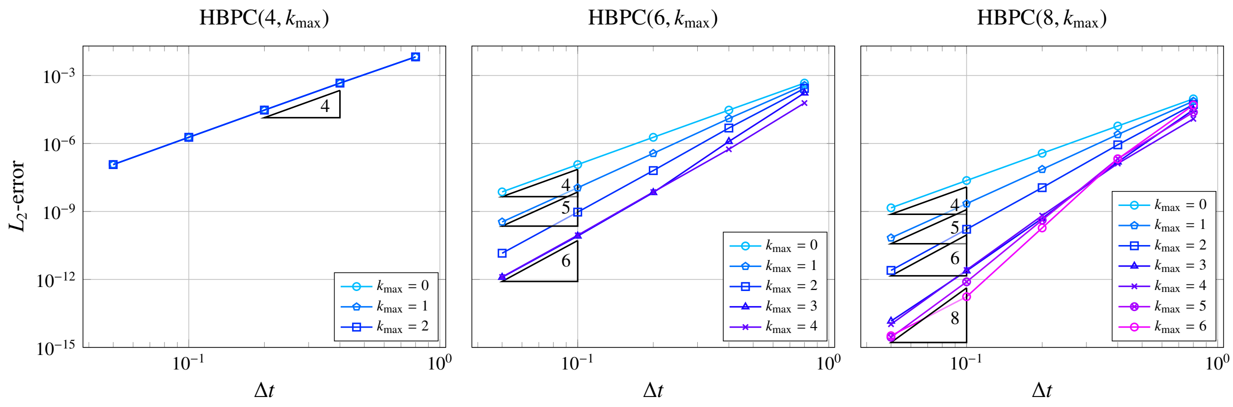

Figure 7. L_2-error for linear scalar advection of sine wave with $T_{\text{end}} = 0.8$ for different HBPC($q, k_{\max}$) methods. The spatial domain is discretized with $n_E = 32^2$ elements with $N = 7$. Note that the L^2-error of the initial spatial projection is approximately $1.1 \cdot 10^{-15}$.

The resulting L_2-errors of this series of simulations are visualized in Fig. 7. The figure shows that the temporal error decreases with the desired order of convergence for all considered HBPC($q, k_{\max}$) schemes: If the HBPC($4, k_{\max}$) scheme is chosen, the corrector steps do not have any influence on the solution. This is an expected behavior as the limiting fourth order Hermite-Birkhoff Runge-Kutta scheme is already used as a predictor. For the HBPC($6, k_{\max}$) and HBPC($8, k_{\max}$) schemes one can see that starting with the order of the predictor, the schemes pick up one order of accuracy per correction step until the maximum order of the underlying quadrature rule is obtained. The only exception from this behavior is the HBPC($8, k_{\max}$) scheme with $k_{\max} \geq 4$. Here, for some refinement steps a slightly larger order of convergence as expected can be observed. One can see that using more correction steps than required to reach the maximum order of convergence can have a favorable influence on the accuracy of the method.

Note that the presented method is not limited to eighth order of accuracy in time. If one chooses another quadrature rule in Alg. 1 and a sufficient amount of corrector steps, in principle, arbitrary order can be achieved.

4.3.2. Euler Equations

A similar investigation is performed for the non-linear Euler equations with $\varepsilon = 1$. Again, the domain is discretized with $n_E = 32^2$ elements with a polynomial degree of the ansatz functions of $N = 7$ and $T_{\text{end}} = 0.8$ is chosen as final time. The convergence tolerances are chosen to be $\varepsilon_{\text{GMRES}} = 10^{-5}$ and $\varepsilon_{\text{Newton}} = 10^{-12}$.

The total L_2-errors of these simulations of the test setup described in Eq. (22) are visualized in Fig. 8. Again, the expected order of accuracy can be observed for all considered schemes in most cases. Similar as for the linear case, the HBPC($8, k_{\max}$) schemes shows a slightly larger order of convergence as one would expect for some refinement steps and $k_{\max} \geq 4$.

5. Application to the Navier-Stokes Equations

In this section, we extend the the presented method to handle the Navier-Stokes equations which are given by

$$\mathbf{w}_t + \nabla_x \cdot (\mathbf{F}(\mathbf{w}) - \mathbf{F}^{\nu}(\mathbf{w}, \nabla_x \mathbf{w})) = 0, \quad \text{with} \quad \mathbf{w} = \begin{pmatrix} \rho \\ \rho\mathbf{v} \\ E \end{pmatrix} \quad \text{and} \quad \mathbf{F}^{\nu}(\mathbf{w}, \nabla_x \mathbf{w}) = \begin{pmatrix} 0 \\ \boldsymbol{\tau} \\ \boldsymbol{\tau} \cdot \mathbf{v} + \mathbf{q}, \end{pmatrix}. \tag{25}$$

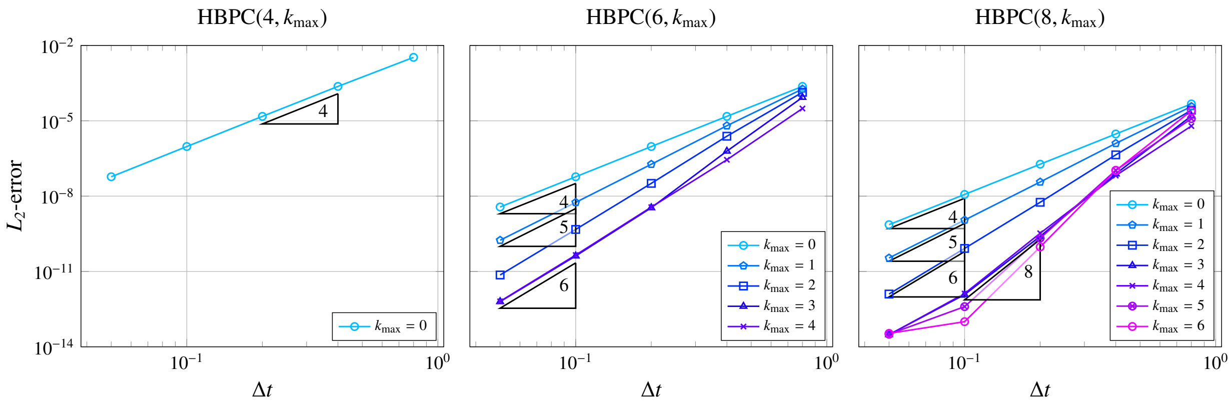

Figure 8. L_2-error for Euler equations with advection of sine wave with $T_{\text{end}} = 0.8$ for different HBPC($q, k_{\max}$) methods. The spatial domain is discretized with $n_E = 32^2$ elements with $N = 7$. Note that the L^2-error of the initial spatial projection is approximately $1.2 \cdot 10^{-15}$.

They consist of the inviscid Euler flux $\mathbf{F}(\mathbf{w})$ (21), and an additional viscous flux $\mathbf{F}^v(\mathbf{w}, \nabla_x \mathbf{w})$. The viscous flux depends on the viscous stress tensor

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

Here, we have used dynamic viscosity μ, heat flux $\mathbf{q} = \lambda \nabla_x T$, thermal conductivity $\lambda = \frac{c_p \mu}{Pr}$, specific heat capacity $c_p = \frac{R\gamma}{\gamma - 1}$ and the specific gas constant $R = \frac{1}{\gamma \varepsilon^2}$. The temperature T is defined via the ideal gas law $p = \rho R T$ and the fluid specific Prandtl number is set to $Pr = 0.72$. The dynamic viscosity μ is selected differently for the considered testcases.

5.1. Fully Discrete Scheme for the Navier-Stokes Equations

From Eq. (25) one can see that the viscous flux depends on the state vector $\mathbf{w}$ and its spatial gradients $\nabla_x \mathbf{w}$. More specifically, it depends on the state vector and the gradients of velocity and temperature. For the discretization of this second order PDE system, we follow the BR2 lifting approach [36]. The idea of the lifting procedure is to rewrite the second order Navier-Stokes system as an extended system of first order PDEs, see e.g. [37] for an overview on this technique. For that purpose, one additional so-called lifting equation per spatial direction is introduced for the gradient vector $\mathbf{d} = (\mathbf{d}^1, \ldots, \mathbf{d}^m)$. It contains the spatial gradients of the velocity and the temperature in the different spatial directions

$$\mathbf{d} = \nabla_x \mathbf{w}_{\text{grad}}, \quad \text{with} \quad \mathbf{w}_{\text{grad}} = (\mathbf{v}, T)^T\,. \tag{26}$$

5.1.1. Calculation of the First Temporal Derivative

The equations to obtain the first temporal derivative $\mathbf{w}_t$ are then given by

$$\begin{pmatrix} \mathbf{w}_t \\ 0 \end{pmatrix} = \begin{pmatrix} \nabla_x \cdot (-\mathbf{F}(\mathbf{w}) + \mathbf{F}^v(\mathbf{w}, \mathbf{d})) \\ \mathbf{d} - \nabla_x \mathbf{w}_{\text{grad}} \end{pmatrix}. \tag{27}$$

In order to obtain a discretization of the gradients $\mathbf{d}$, we follow the BR2 lifting approach [36], where the weak form of Eq. (26) is used to obtain a discretization for the gradients $\mathbf{d}$. Summing up, Eq. (27) in weak formulation which gives the discrete spatial operator $\mathbf{R}_h^{(1)} = \mathbf{R}_h^{(1)}(\mathbf{w}, \mathbf{d}(\mathbf{w}))$ is given by

$$\sum_{e=1}^{n_E} (\mathbf{w}_t, \phi)_{\Omega_e} - (\mathbf{F}(\mathbf{w}) - \mathbf{F}^v(\mathbf{w}, \mathbf{d}), \nabla_x \phi)_{\Omega_e} + \left\langle \mathbf{F}^*(\mathbf{w}^L, \mathbf{w}^R) \cdot \mathbf{n} - \mathbf{F}^{v*}(\mathbf{w}^L, \mathbf{w}^R, \mathbf{d}^L, \mathbf{d}^R) \cdot \mathbf{n}, \phi \right\rangle_{\partial\Omega_e} = 0, \quad \forall \phi \in \Pi_N, \tag{28}$$

$$\text{with} \quad \sum_{e=1}^{n_E} (\mathbf{d}, \boldsymbol{\phi})_{\Omega_e} + \left(\mathbf{w}_{\text{grad}}, \nabla_x \boldsymbol{\phi}\right)_{\Omega_e} - \left\langle \mathbf{w}^*_{\text{grad}}(\mathbf{w}^L_{\text{grad}}, \mathbf{w}^R_{\text{grad}}) \cdot \mathbf{n}, \boldsymbol{\phi} \right\rangle_{\partial\Omega_e} = 0, \quad \forall \boldsymbol{\phi} \in \Pi_N^2. \tag{29}$$

The viscous flux at the surfaces $\mathbf{F}^{v*}$ and the lifting surface flux $\mathbf{w}^*_{\text{grad}}$ are given by the arithmetic means

$$
\begin{aligned}
\mathbf{F}^{v*}(\mathbf{w}^L,\mathbf{w}^R,\mathbf{d}^L,\mathbf{d}^R) &= \frac{1}{2}\left(\mathbf{F}^v(\mathbf{w}^L,\mathbf{d}^L)+\mathbf{F}^v(\mathbf{w}^R,\mathbf{d}^R)\right)\\
\mathbf{w}^*_{\text{grad}}(\mathbf{w}^L_{\text{grad}},\mathbf{w}^R_{\text{grad}}) &= \frac{1}{2}\left(\mathbf{w}^L_{\text{grad}}+\mathbf{w}^R_{\text{grad}}\right).
\end{aligned}
\tag{30}
$$

The gradients at the cell edges $\mathbf{d}^L$, $\mathbf{d}^R$ are *not* obtained by straightforward evaluation of the solution polynomial of $\mathbf{d}$. Instead, a local lifting equation is defined for each edge. The local lifting equations are similar as the global lifting equation given in Eq. (29), but with a modified surface contribution. For each edge, only the surface integral contribution of the edge itself is taken into account and is scaled with a stabilization parameter. The obtained polynomial representation is then evaluated at the corresponding edge to obtain $\mathbf{d}^L$ or $\mathbf{d}^R$. More details on the BR2 lifting procedure, especially the specific formulation for the DGSEM, can be found in [38, Eqs. (3.81)-(3.86)]. Note that the implementation of the lifting procedure has been done in the strong form, which is identically to the weak form for the DGSEM [38]. It is not indispensable to use the BR2 lifting procedure - other procedures would also be possible, see e.g. [37] for an overview on different lifting procedures. Here, we choose the BR2 scheme because of its compact stencil as it has only dependencies on direct neighbors, see e.g. [39].

5.1.2. *Calculation of the Second Temporal Derivative*

In order to obtain an equation for the discretization of the second temporal derivative, we pursue a similar approach as for the pure hyperbolic equation (see Eq. (4)) and extend the system by the lifting equation

$$
\begin{pmatrix}\mathbf{w}_{tt}\\ 0\end{pmatrix} = \begin{pmatrix}\nabla_x\cdot\left(-\frac{\partial\mathbf{F}}{\partial\mathbf{w}}\mathbf{w}_t\right)+\nabla_x\cdot\left(\frac{\partial\mathbf{F}^v}{\partial\mathbf{w}}\mathbf{w}_t\right)+\nabla_x\cdot\left(\frac{\partial\mathbf{F}^v}{\partial\mathbf{d}}\mathbf{d}_t\right)\\ \mathbf{d}_t-\nabla_x\left(\frac{\partial\mathbf{w}_{\text{grad}}}{\partial\mathbf{w}}\mathbf{w}_t\right)\end{pmatrix},
\tag{31}
$$

where the matrix $\frac{\partial\mathbf{w}_{\text{grad}}}{\partial\mathbf{w}}$ is given by the relation of velocity and momentum and by the ideal gas law. We now make use of the auxiliary variable $\boldsymbol{\sigma}$ (see Eq. (14)) and additionally introduce the auxiliary variable $\boldsymbol{\eta} := \mathbf{d}_t$. With this, Eq. (31) reads

$$
\begin{pmatrix}\mathbf{w}_{tt}\\ 0\end{pmatrix} = \begin{pmatrix}\nabla_x\cdot\left(-\frac{\partial\mathbf{F}}{\partial\mathbf{w}}\boldsymbol{\sigma}\right)+\nabla_x\cdot\left(\frac{\partial\mathbf{F}^v}{\partial\mathbf{w}}\boldsymbol{\sigma}\right)+\nabla_x\cdot\left(\frac{\partial\mathbf{F}^v}{\partial\mathbf{d}}\boldsymbol{\eta}\right)\\ \boldsymbol{\eta}-\nabla_x\left(\frac{\partial\mathbf{w}_{\text{grad}}}{\partial\mathbf{w}}\boldsymbol{\sigma}\right)\end{pmatrix}.
\tag{32}
$$

The discretization of Eq. (32) follows the same steps as the discretization for the first temporal derivative (see Eq. (4) and Eq. (15) for a comparison). The discrete operator $\mathbf{R}_h^{(2)} = \mathbf{R}_h^{(2)}(\mathbf{w},\mathbf{d}(\mathbf{w}),\boldsymbol{\sigma},\boldsymbol{\eta}(\boldsymbol{\sigma},\mathbf{w}))$ to obtain the second temporal derivative is then defined by

$$
\begin{aligned}
\sum_{e=1}^{n_E}(\mathbf{w}_{tt},\phi)_{\Omega_e} &- \left(\frac{\partial\mathbf{F}(\mathbf{w})}{\partial\mathbf{w}}\cdot\boldsymbol{\sigma},\nabla_x\phi\right)_{\Omega_e} + \left(\frac{\partial\mathbf{F}^v(\mathbf{w},\mathbf{d})}{\partial\mathbf{w}}\cdot\boldsymbol{\sigma},\nabla_x\phi\right)_{\Omega_e} + \left(\frac{\partial\mathbf{F}^v(\mathbf{w},\mathbf{d})}{\partial\mathbf{d}}\cdot\boldsymbol{\eta},\nabla_x\phi\right)_{\Omega_e}\\
&+\left\langle\frac{\partial\mathbf{F}^*(\mathbf{w}^L,\mathbf{w}^R)}{\partial\mathbf{w}^L}\boldsymbol{\sigma}^L\cdot\mathbf{n}+\frac{\partial\mathbf{F}^*(\mathbf{w}^L,\mathbf{w}^R)}{\partial\mathbf{w}^R}\boldsymbol{\sigma}^R\cdot\mathbf{n},\phi\right\rangle_{\partial\Omega_e}\\
&-\left\langle\frac{1}{2}\left(\frac{\partial\mathbf{F}^v(\mathbf{w}^L,\mathbf{d}^L)}{\partial\mathbf{w}^L}\boldsymbol{\sigma}^L+\frac{\partial\mathbf{F}^v(\mathbf{w}^R,\mathbf{d}^R)}{\partial\mathbf{w}^R}\boldsymbol{\sigma}^R\right)\cdot\mathbf{n},\phi\right\rangle_{\partial\Omega_e}\\
&-\left\langle\frac{1}{2}\left(\frac{\partial\mathbf{F}^v(\mathbf{w}^L,\mathbf{d}^L)}{\partial\mathbf{d}^L}\boldsymbol{\eta}^L+\frac{\partial\mathbf{F}^v(\mathbf{w}^R,\mathbf{d}^R)}{\partial\mathbf{d}^R}\boldsymbol{\eta}^R\right)\cdot\mathbf{n},\phi\right\rangle_{\partial\Omega_e} = 0, \quad \forall\phi\in\Pi_N,
\end{aligned}
$$

with

$$
\sum_{e=1}^{n_E}(\boldsymbol{\eta},\boldsymbol{\phi})_{\Omega_e}+\left(\left(\frac{\partial\mathbf{w}_{\text{grad}}}{\partial\mathbf{w}}\boldsymbol{\sigma}\right),\nabla_x\boldsymbol{\phi}\right)_{\Omega_e}-\left\langle\frac{1}{2}\left(\frac{\partial\mathbf{w}^L_{\text{grad}}}{\partial\mathbf{w}^L}\boldsymbol{\sigma}^L+\frac{\partial\mathbf{w}^R_{\text{grad}}}{\partial\mathbf{w}^R}\boldsymbol{\sigma}^R\right)\cdot\mathbf{n},\boldsymbol{\phi}\right\rangle_{\partial\Omega_e} = 0, \quad \forall\boldsymbol{\phi}\in\Pi_N^2.
\tag{33}
$$

Note that we have directly used the simple structure of the viscous numerical flux and the lifting numerical flux (Eq. (30)) in the definition of $\mathbf{R}_h^{(2)}$. The values of $\boldsymbol{\eta}$ at the cell edges ($\boldsymbol{\eta}^{L/R}$) are obtained for each edge by the local forms of the lifting equation (33).

5.1.3. *Solving the Non-Linear System*

Finally, the non-linear equation system which is solved by Newton's method, see Eq. (18), has to be modified accordingly. The matrix-vector product of the system matrix $\mathcal{J}$ and the increment vector ΔX is then given by

$$\mathcal{J}\Delta X = \begin{pmatrix} \mathrm{Id} - \alpha_1 \Delta t \left(\frac{\partial \mathbf{R}_h^{(1)}}{\partial \mathbf{W}} + \frac{\partial \mathbf{R}_h^{(1)}}{\partial \mathbf{d}} \frac{\partial \mathbf{d}}{\partial \mathbf{W}} \right) + \frac{\alpha_2 \Delta t^2}{2} \left(\frac{\partial \mathbf{R}_h^{(2)}}{\partial \mathbf{W}} + \frac{\partial \mathbf{R}_h^{(2)}}{\partial \mathbf{d}} \frac{\partial \mathbf{d}}{\partial \mathbf{W}} + \frac{\partial \mathbf{R}_h^{(2)}}{\partial \boldsymbol{\eta}} \frac{\partial \boldsymbol{\eta}}{\partial \mathbf{W}} \right) & \frac{\alpha_2 \Delta t^2}{2} \left(\frac{\partial \mathbf{R}_h^{(2)}}{\partial \boldsymbol{\sigma}} + \frac{\partial \mathbf{R}_h^{(2)}}{\partial \boldsymbol{\eta}} \frac{\partial \boldsymbol{\eta}}{\partial \boldsymbol{\sigma}} \right) \\ -\left(\frac{\partial \mathbf{R}_h^{(1)}}{\partial \mathbf{W}} + \frac{\partial \mathbf{R}_h^{(1)}}{\partial \mathbf{d}} \frac{\partial \mathbf{d}}{\partial \mathbf{W}} \right) & \mathrm{Id} \end{pmatrix} \begin{pmatrix} \Delta \mathbf{W} \\ \Delta \boldsymbol{\sigma} \end{pmatrix}.$$

Again, due to the definitions of $\boldsymbol{\sigma}$ and $\boldsymbol{\eta}$ a tedious but straightforward analysis reveals that

$$\frac{\partial \mathbf{R}_h^{(1)}}{\partial \mathbf{d}} \frac{\partial \mathbf{d}}{\partial \mathbf{W}} \equiv \frac{\partial \mathbf{R}_h^{(2)}}{\partial \boldsymbol{\eta}} \frac{\partial \boldsymbol{\eta}}{\partial \boldsymbol{\sigma}}.$$

Similar as it has been done for the Euler equations, we neglect the Hessian contribution $\frac{\partial \mathbf{R}_h^{(2)}}{\partial \mathbf{W}} + \frac{\partial \mathbf{R}_h^{(2)}}{\partial \mathbf{d}} \frac{\partial \mathbf{d}}{\partial \mathbf{W}} + \frac{\partial \mathbf{R}_h^{(2)}}{\partial \boldsymbol{\eta}} \frac{\partial \boldsymbol{\eta}}{\partial \mathbf{W}}$ when constructing the preconditioner. In the following, the matrix-free approach (see Eq. (23)) with the BJ_{ext} preconditioner is used for the simulations.

5.2. *Numerical Validation*

For the numerical validation of the presented algorithm, we choose the same test setup as used for the Euler equations given in Eq. (22). Additionally, viscosity with $\mu = 10^{-3}$ is taken into account. Again, $n_E = 32^2$ elements with $N = 7$ are used for the spatial discretization. As the exact solution is no longer known, the solution of a simulation with a very small explicit timestep (4^{th} order [40], CFL = 0.1) is taken as reference solution at $T_{end} = 0.8$. The convergence tolerances are chosen to be $\varepsilon_{\text{GMRES}} = 10^{-3}$ and $\varepsilon_{\text{Newton}} = 10^{-10}$. We additionally use an absolute convergence tolerance for Newton's method $||\Delta\mathbf{W}||_2 \leq 10^{-12}$. This became necessary as the initial relative Newton norm for correction steps can sometimes be already very low, such that roundoff errors prevent convergence of the relative tolerance.

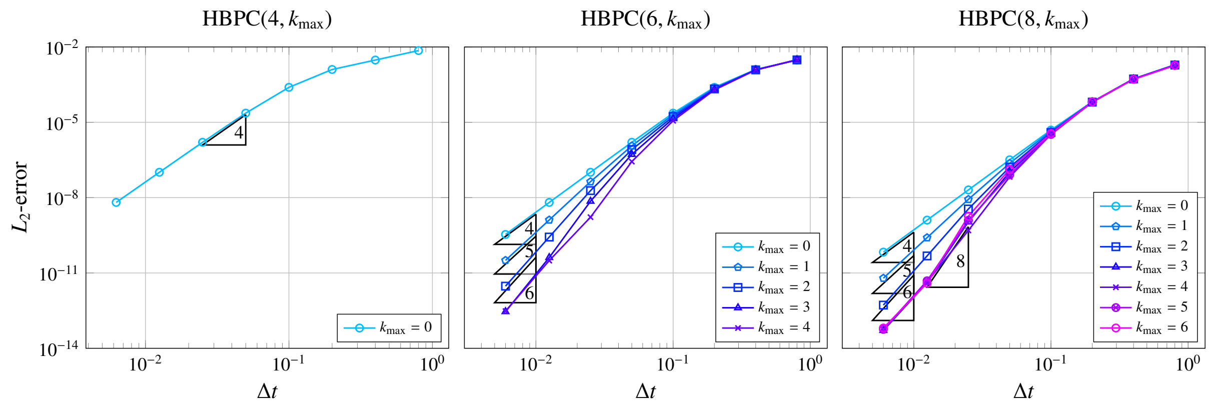

Figure 9. L_2-error for Navier-Stokes equations with advection and diffusion of sine wave with $T_{\text{end}} = 0.8$ for different HBPC($q, k_{\max}$) methods. The spatial domain is discretized with $n_E = 32^2$ elements with $N = 7$.

The resulting L_2-errors of the simulations are visualized in Fig. 9. One can observe that the desired orders are reached for all considered HBPC($q, k_{\max}$) schemes. For the HBPC($6, k_{\max}$) schemes, performing more correction steps than required to reach the maximum order improves the achieved accuracy. For the HBPC($8, k_{\max}$) schemes, the desired order cannot be observed during the last refinement step and $k_{\max} \geq 3$ due to occurring roundoff errors.

A comparison with the Euler case without viscosity (Fig. 8) shows that one needs more temporal refinement steps to reach the asymptotic regime. This is most likely due to the very fast parabolic characteristics. When considering the setup with viscosity, the explicit timestep has to be reduced by approximately a factor of 5 compared to the inviscid setting. This indicates the dominance of the fast parabolic characteristics and hence why the asymptotic regime is shifted to smaller timestep sizes.

5.3. Illustrative Applications

We finally show some illustrative applications of the novel method to typical flow problems.

5.3.1. Lid-Driven Cavity

The first example is the lid-driven cavity flow. The quadratic domain $\Omega = [0,1]^2$, which is discretized with $n_E = 16^2$ elements, has wall boundary conditions at the left, right and lower boundary. At the upper boundary, a constant flow field

$$\rho = 1,\ \mathbf{v} = (1,0)^T,\ p = \frac{1}{\gamma},$$

is prescribed. The domain is initialized with the same density and pressure, but with zero velocity. We select $\mu = 2.5 \cdot 10^{-2}$, resulting in a Reynolds number of $Re = 400$, and a reference Mach number of $\varepsilon = 0.1$. The spatial and temporal domain are discretized with a sixth order method choosing $N = 5$ and HBPC(6, 2) with $\Delta t = 0.02$. Tolerances of the implicit method are set to $\varepsilon_{\text{Newton}} = 10^{-3}$ and $\varepsilon_{\text{GMRES}} = 10^{-2}$. An illustration of the steady state

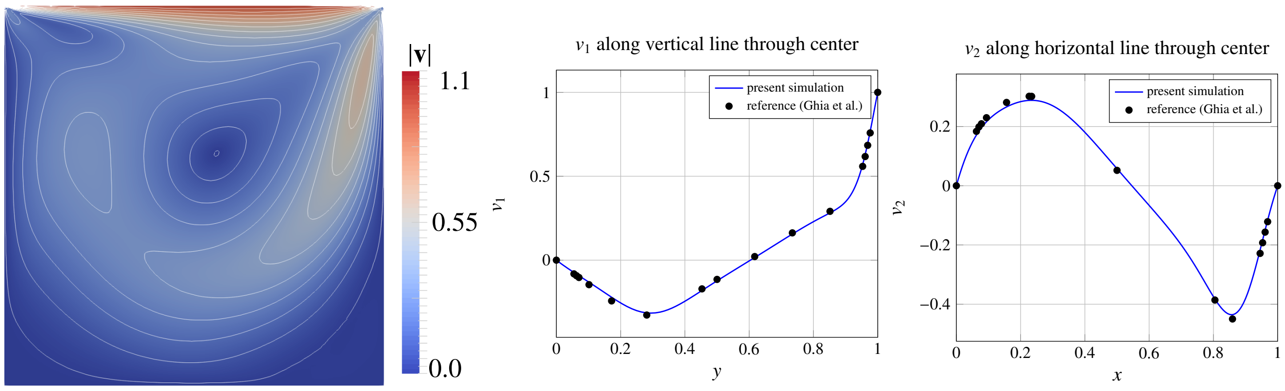

Figure 10. Velocity magnitude (coloring and isocontours) at steady state ($T_{end} = 50$) for lid-driven cavity flow problem (left). Additional validation is provided by comparison of present simulation with data from Ghia et. al [41]: v_1-velocity along vertical line through geometrical center of cavity (middle) and v_2-velocity along horizontal line through geometrical center of cavity (right).

solution at $T_{end} = 50$ is depicted in Fig. 10 (left). The results of a comparison of our solution with the simulations from [41][2] is shown in Fig. 10 (middle and right). The present solution matches the reported results in literature very well. As this is a steady state flow example, the high order accurate time discretization would not be necessary in this case. Nevertheless, this example illustrates that the novel method is capable of simulating such problems.

5.3.2. Flow Around a Cylinder

The next example is the two dimensional flow around a cylinder with a Reynolds number $Re_D = 200$ and a reference Mach number $\varepsilon = 0.1$. The diameter of the cylinder is chosen to be $D = 1$. Initially, the flow field is set to a constant state

$$\rho = 1,\ \mathbf{v} = (1,0)^T,\ p = \frac{1}{\gamma},$$

and the viscosity is set to $\mu = 5 \cdot 10^{-3}$. We use a cylindrical mesh with $n_E = 1000$ elements and a sixth order discretization in space and time with $N = 5$ and the HBPC(6, 2) scheme. At the farfield, the initial condition is prescribed as Dirichlet boundary condition, and at the cylinder surface, wall boundaries are applied. A detailed description of the used mesh can be found in [28, Sec. 5.1.1]. The convergence tolerances for Newton's method and the linear solver are set to $\varepsilon_{\text{Newton}} = 10^{-5}$ and $\varepsilon_{\text{GMRES}} = 10^{-3}$, respectively.

[2]Please note that in [41, Table II], we have left out the point 117, as this number seems to be erroneous.

For this setup, one can expect a vortex shedding behind the cylinder. One measure of the solution quality is the shedding frequency of the wake, which can be analyzed by the frequency of the lift forces at the cylinder. The frequency f is typically related to the lift forces, the cylinder diameter D and the freestream velocity v_1, which results in the so-called Strouhal number $\mathrm{Sr} = \frac{fD}{v_1}$. We compare two different simulations with relatively large timesteps $\Delta t = 0.1$ and $\Delta t = 0.25$. These timestep sizes are approximately 200 ($\Delta t = 0.1$) and 500 times ($\Delta t = 0.25$) larger than it would be possible with a 4th order explicit low-storage Runge-Kutta scheme [40].

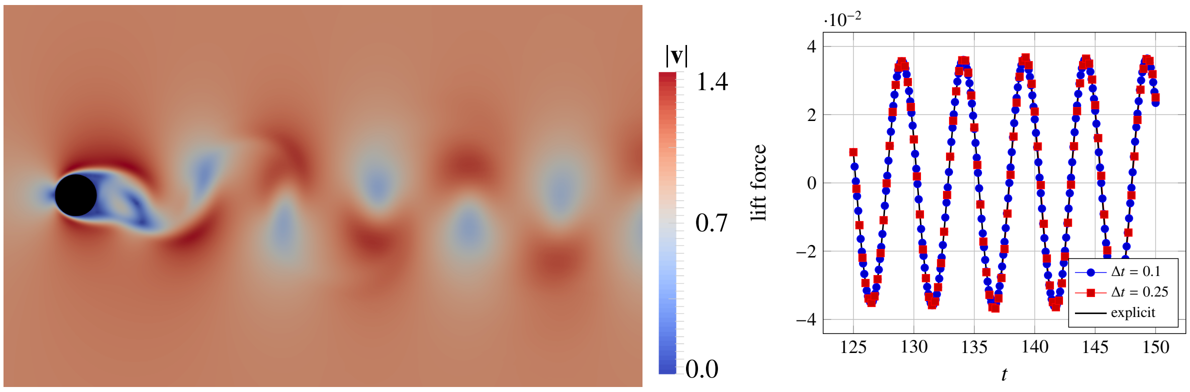

Figure 11. Velocity magnitude at $t = 125$ of cylinder flow with $\varepsilon = 0.1$ and $Re_D = 200$ (left) and temporal evolution of lift forces at cylinder surface (right) choosing different timestep sizes of the HBPC(6, 2) scheme and explicit reference calculation.

Besides the instantaneous flow field at $t = 125$, the temporal evolution of the lift forces from $t = 125$ to $t = 150$ are visualized in Fig. 11. One can see that the calculations with both timestep sizes are very similar and match the explicit reference calculation very well: The Strouhal number is $\mathrm{Sr} = 0.1971$ for $\Delta t = 0.1$ and $\mathrm{Sr} = 0.1968$ for $\Delta t = 0.25$. This is in very good agreement to what has been reported in literature, see e.g. [42] ($\mathrm{Sr} = 0.1957$) and [43] ($\mathrm{Sr} = 0.196$).

5.3.3. Taylor-Green-Vortex

Finally, the three dimensional Taylor-Green-Vortex (TGV) is simulated. This illustrates that the method is also applicable to solve three dimensional problems. The TGV is a typical problem to study the transition to turbulence and its decay. It is initialized with large vortices, which then decompose into smaller vortices. When they are small enough they are dissipated by the viscosity of the fluid and their kinetic energy is transferred into internal energy. A measure of this mechanism is the dissipation rate of the kinetic energy

$$\frac{\partial E_{\text{kin}}}{\partial t} = \frac{2\mu}{\rho \|\Omega\|} \int_\Omega \nabla_x \mathbf{v} : \nabla_x \mathbf{v} d\mathbf{x},$$

with the volume of the computational domain $\|\Omega\|$. Similar as it has been done in [32], we adopt the initialization of the TGV to the non-dimensional equations. The initial data are given by

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

We use $n_E = 32^3$ elements with $N = 3$ to discretize the periodic domain $\Omega = [0, 2\pi]^3$. For the temporal discretization the HBPC(4, 0) scheme with $\Delta t = 0.25$ is chosen. The viscosity is set to $\mu = 1.25 \cdot 10^{-3}$, resulting in a unit Reynolds number of $Re = 800$. In Fig. 12, the dissipation rate of the kinetic energy is shown for the novel scheme, an explicit reference simulation and a DNS solution, taken from [44]. One can see, that the DNS is matched very well and only the peak in the decay rate at $t \approx 9$ is slightly underestimated. This is most probably caused by a too coarse spatial resolution. The solution with the HBPC(4, 0) and the explicit time discretization agree very good although the timestep size of the implicit method is approximately 200 times larger than for the explicit method.

Summing up, the presented testcases illustrate the applicability of the novel scheme for the simulation of complex flow phenomena.

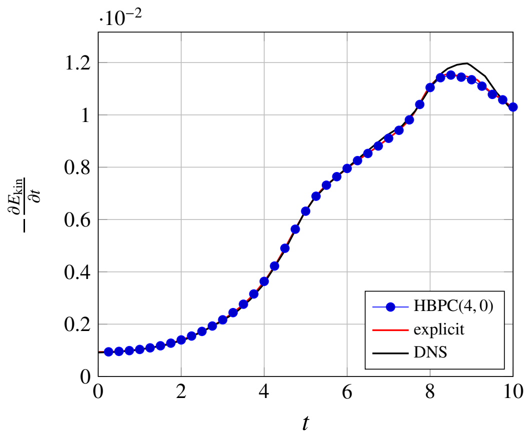

Figure 12. Temporal evolution of kinetic energy dissipation rate for $Re = 800$ TGV. The HBPC(4, 0) scheme uses a timestep size of $\Delta t = 0.25$ and the explicit 4th order scheme [40] $\Delta t \approx 1.25 \cdot 10^{-3}$. Both use a spatial resolution of $n_E = 32^3$ elements with $N = 3$. DNS data are taken from [44].

6. Conclusion and Outlook

In this work, we have shown how two-derivative deferred correction methods can be combined with the DGSEM to obtain a numerical method that is high order in space and time. It has been illustrated that, due to the implicit nature of the time discretization, very large timesteps can be used. For the discretization of the second temporal derivative, the approach of [18] is adopted to handle non-linear equations. Different options how to set up the non-linear system to be solved implicitly have been investigated. The preferred method comes with a matrix-free approach, a novel and relatively simple preconditioner and allows for a straight-forward parallelization of the spatial domain. The flexibility of the novel approach in handling complex applications has been illustrated by the simulation of typical benchmark problems for the Navier-Stokes equations.

The most obvious future development for the novel scheme can be deduced from [21]: the handling of a mixed implicit-explicit (IMEX) flux splitting, such as e.g. [45, 32] and a temporal parallelization. Both will be addressed in a future work. In addition, improvements are possible in terms of efficiency and flexibility of the new method. The use of adaptive tolerances and timestep sizes can ease the setup of simulations. We have observed that the choice of the numerical dissipation in Eq. (13) is crucial for both stability and efficiency. A detailed study, also regarding asymptotic consistency and other numerical flux functions is worth pursuing. Additionally, ideas from the ADER community, see e.g. [46, 10], might help to improve the flexibility in using different Riemann solvers.

Acknowledgments

The authors would like to thank David Seal and Alexander Jaust for the discussions on multiderivative timestepping schemes. J. Zeifang was funded by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) - project no. 457811052. We acknowledge the Institute of Aerodynamics and Gas Dynamics at the University of Stuttgart and 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 Competing 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.

References

[1] E. Hairer, G. Wanner, Multistep-multistage-multiderivative methods for ordinary differential equations, Computing (Arch. Elektron. Rechnen) 11 (3) (1973) 287–303.

[2] P. Lax, B. Wendroff, Systems of conservation laws, Communications on Pure and Applied Mathematics 13 (2) (1960) 217–237.

[3] K. Kastlunger, G. Wanner, On Turan type implicit Runge-Kutta methods, Computing 9 (1972) 317–325.

[4] A. J. Christlieb, Y. Güçlü, D. C. Seal, The Picard integral formulation of weighted essentially nonoscillatory schemes, SIAM Journal on Numerical Analysis 53 (4) (2015) 1833–1856.

[5] Y. Jiang, C.-W. Shu, M. Zhang, An alternative formulation of finite difference weighted ENO schemes with Lax-Wendroff time discretization for conservation laws, SIAM Journal on Scientific Computing 35 (2) (2013) A1137–A1160.

[6] S. A. Moe, J. A. Rossmanith, D. C. Seal, Positivity-preserving discontinuous Galerkin methods with Lax–Wendroff time discretizations, Journal of Scientific Computing 71 (1) (2017) 44–70.

[7] J. Qiu, M. Dumbser, C.-W. Shu, The discontinuous Galerkin method with Lax–Wendroff type time discretizations, Computer Methods in Applied Mechanics and Engineering 194 (42-44) (2005) 4528–4543.

[8] D. Zorío, A. Baeza, P. Mulet, An approximate Lax–Wendroff-type procedure for high order accurate schemes for hyperbolic conservation laws, Journal of Scientific Computing 71 (2017) 246–273.

[9] V. A. Titarev, E. F. Toro, ADER: Arbitrary high order Godunov approach, Journal of Scientific Computing 17 (1) (2002) 609–618.

[10] S. Busto, S. Chiocchetti, M. Dumbser, E. Gaburro, I. Peshkov, High order ADER schemes for continuum mechanics, Frontiers in Physics 8 (2020) 32.

[11] D. Seal, Y. Güçlü, A. Christlieb, High-order multiderivative time integrators for hyperbolic conservation laws, Journal of Scientific Computing 60 (2014) 101–140.

[12] X. Ji, F. Zhao, W. Shyy, K. Xu, A family of high-order gas-kinetic schemes and its comparison with Riemann solver based high-order methods, Journal of Computational Physics 356 (2018) 150–173.

[13] Z. He, F. Gao, B. Tian, J. Li, Implementation of finite difference weighted compact nonlinear schemes with the two-stage fourth-order accurate temporal discretization, Communications in Computational Physics 27 (2020) 1470–1484.

[14] L. Pan, K. Xu, Q. Li, J. Li, An efficient and accurate two-stage fourth-order gas-kinetic scheme for the Euler and Navier-Stokes equations, Journal of Computational Physics 326 (2016) 197 – 221.

[15] J. Chouchoulis, J. Schütz, J. Zeifang, Jacobian-free explicit multiderivative Runge-Kutta methods for hyperbolic conservation laws, arXiv preprint arXiv:2107.06633 (2021).

[16] A. Tsai, R. Chan, S. Wang, Two-derivative Runge-Kutta methods for PDEs using a novel discretization approach, Numerical Algorithms 65 (2014) 687–703.

[17] A. Jaust, J. Schütz, D. C. Seal, Implicit multistage two-derivative discontinuous Galerkin schemes for viscous conservation laws, Journal of Scientific Computing 69 (2016) 866–891.

[18] J. Schütz, D. 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.

[19] A. Jaust, Novel implicit unconditionally stable time-stepping for DG-type methods and related topics, Ph.D. thesis, Hasselt University (2018).

[20] J. Schütz, D. Seal, An asymptotic preserving semi-implicit multiderivative solver, Applied Numerical Mathematics 160 (2021) 84–101.

[21] J. Schütz, D. C. Seal, J. Zeifang, Parallel-in-time high-order multiderivative IMEX solvers, arXiv preprint arXiv:2101.07846 (2021).

[22] D. A. Kopriva, Implementing spectral methods for partial differential equations: Algorithms for scientists and engineers, Springer Science & Business Media, 2009.

[23] W. Reed, T. Hill, Triangular mesh methods for the neutron transport equation, Tech. rep., Los Alamos Scientific Laboratory (1973).

[24] C.-W. Shu, A brief survey on discontinuous Galerkin methods in computational fluid dynamics, Advances in Mechanics 43 (2013) 541–554.

[25] 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.

[26] 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.

[27] N. J. Higham, Accuracy and stability of numerical algorithms, SIAM, 2002.

[28] 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).

[29] J. Zeifang, A discontinuous Galerkin method for droplet dynamics in weakly compressible flows, Verlag Dr. Hut, 2020.

[30] S. Balay, J. Brown, K. Buschelman, V. Eijkhout, W. D. Gropp, D. Kaushik, M. G. Knepley, L. C. McInnes, B. F. Smith, H. Zhang, PETSc users manual, Tech. Rep. ANL-95/11 - Revision 3.1, Argonne National Laboratory (2010).

[31] K. Kaiser, J. Schütz, A high-order method for weakly compressible flows, Communications in Computational Physics 22 (4) (2017) 1150–1174.

[32] 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.

[33] J. Zeifang, K. Kaiser, A. Beck, J. Schütz, C.-D. Munz, Efficient high-order discontinuous Galerkin computations of low Mach number flows, Communications in Applied Mathematics and Computational Science 13 (2018) 243–270.

[34] 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.

[35] D. A. Knoll, D. E. Keyes, Jacobian-free Newton–Krylov methods: a survey of approaches and applications, Journal of Computational Physics 193 (2004) 357–397.

[36] 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.

[37] D. N. Arnold, F. Brezzi, B. Cockburn, L. D. Marini, Unified analysis of discontinuous Galerkin methods for elliptic problems, SIAM Journal on Numerical Analysis 39 (2002) 1749–1779.

[38] F. Hindenlang, Mesh curving techniques for high order parallel simulations on unstructured meshes, Ph.D. thesis, University of Stuttgart (2014).

[39] S. Ortleb, A comparative Fourier analysis of discontinuous Galerkin schemes for advection–diffusion with respect to BR1, BR2, and local discontinuous Galerkin diffusion discretization, Mathematical Methods in the Applied Sciences 43 (13) (2020) 7841–7863.

[40] M. Carpenter, C. Kennedy, Fourth-order 2N-storage Runge-Kutta schemes, Tech. rep., NASA Langley Research Center (1994).

[41] U. Ghia, K. N. Ghia, C. Shin, High-Re solutions for incompressible flow using the Navier-Stokes equations and a multigrid method, Journal of Computational Physics 48 (3) (1982) 387–411.

[42] B. Rajani, A. Kandasamy, S. Majumdar, Numerical simulation of laminar flow past a circular cylinder, Applied Mathematical Modelling 33 (3) (2009) 1228–1247.

[43] 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.

[44] 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.

[45] E. F. Toro, M. E. Vázquez-Cendón, Flux splitting schemes for the Euler equations, Computers & Fluids 70 (2012) 1–12.

[46] E. Toro, V. Titarev, Solution of the generalized Riemann problem for advection–reaction equations, Proceedings of the Royal Society of London. Series A: Mathematical, Physical and Engineering Sciences 458 (2018) (2002) 271–281.

Appendix A. Butcher Tables of the Limiting Hermite-Birkhoff Runge-Kutta Methods

We consider the following quadrature rules:

- A fourth-order method ($q = 4$) with two stages ($s = 2$, one being fully explicit), which exactly corresponds to the method used in [20]:

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

- A sixth-order method ($q = 6$) with three stages ($s = 3$, one being fully explicit), as also used in [21, 18]:

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

- An eigth-order method ($q = 8$) with four stages ($s = 4$, one being fully explicit), as also used in [21]:

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