A new efficient explicit Deferred Correction framework: analysis and applications to hyperbolic PDEs and adaptivity

L. Micalizzi*and D.Torlo†

May 26, 2023

Abstract

The Deferred Correction (DeC) is an iterative procedure, characterized by increasing accuracy at each iteration, which can be used to design numerical methods for systems of ODEs. The main advantage of such framework is the automatic way of getting arbitrarily high order methods, which can be put in Runge–Kutta (RK) form. The drawback is the larger computational cost with respect to the most used RK methods. To reduce such cost, in an explicit setting, we propose an efficient modification: we introduce interpolation processes between the DeC iterations, decreasing the computational cost associated to the low order ones. We provide the Butcher tableaux of the new modified methods and we study their stability, showing that in some cases the computational advantage does not affect the stability. The flexibility of the novel modification allows nontrivial applications to PDEs and construction of adaptive methods. The good performances of the introduced methods are broadly tested on several benchmarks both in ODE and PDE contexts.

1 Introduction

A huge amount of phenomena in many different fields can be modeled through ODEs and PDEs, whose analytical solutions are usually not available, hence, many numerical methods have been developed to approximate such solutions. Indeed, the higher is the accuracy needed in the approximation, the more expensive the associated numerical simulations are in terms of computational time and resources employed. If, on the one hand, with modern computers the speed of the simulations has drastically improved, on the other hand, the always stricter tolerances required by modern applications have lead to massive simulations only accessible to supercomputers and, still, characterized by very long computational times. That is why any effort in reducing the computational costs of numerical simulations is of paramount importance. A classical way of reducing them is the adoption of high order methods, which allow to reach lower errors within coarse discretizations.

A wide series of arbitrarily high order methods is based on the DeC approach. Its original formulation has been firstly introduced in 1949 in [17] in a simple prediction-correction time integrator framework. A more elegant version based on spectral integration in time was introduced in 2000 [16], characterized by an iterative procedure allowing to increase the order

*Affiliation: Institute of Mathematics, University of Zurich, Winterthurerstrasse 190, Zurich, 8057, Switzerland. Email: lorenzo.micalizzi@math.uzh.ch.

†Affiliation: SISSA mathLab, SISSA, via Bonomea 265, Trieste, 34136, Italy. Email: davide.torlo@sissa.it.

of accuracy by one at each iteration. In 2003 [29], Minion generalized the DeC framework to obtain an implicit-explicit arbitrarily high order method, with various applications to ODEs and PDEs [30, 23, 19, 28, 36]. Later on, the DeC approach has been generalized by Abgrall [2] to solve hyperbolic PDEs with high order continuous Galerkin (CG) spatial discretizations, overcoming the burden related to the mass matrix leading to numerous applications in the hyperbolic field [3, 6, 27, 14, 7]. The DeC has been also modified in order to preserve physical structures (positivity, entropy, moving equilibria, conservation) [31, 5, 14, 4]. Finally, in [18] it has been pointed out that DeC and ADER methods are very similar iterative time integrators and, when restricted to ODEs, they can be written as RK schemes, see also [21, 37].

The clear advantage of the DeC framework is the possibility to easily increase the order of accuracy, the drawback is the expensive computational cost, due to the iterations and to the high degree of the polynomial reconstruction of the numerical solution considered in each of them. To alleviate such cost, the *ladder* strategy was proposed in implicit DeC algorithms [29, 23, 36], where the reconstruction in time increases the degree at each iteration. Between the iterations, an interpolation procedure links the different reconstructions. Though being the idea used in some works, it has never been deeply studied and analyzed, in particular, for the purely explicit DeC.

Inspired by this idea, in this work, we provide a detailed description of two novel families of efficient explicit DeC methods, based on easy modifications of existing DeC schemes. By explicitly constructing their Butcher tableaux and studying their stability, we show that in some cases the new efficient versions and the classical one have the same stability functions. Moreover, we exploit the modification to build adaptive methods that, given a certain tolerance, automatically choose the order of accuracy to reach such error in the most efficient way. We also apply the efficient modification in the context of mass matrix-free CG–DeC methods [2] for hyperbolic PDEs.

The structure of this work is the following. We start by introducing the DeC procedure in an abstract framework in Section 2 and as a tool for the numerical solution of ODEs systems in Section 3. In Section 4, we introduce the new families of efficient DeC methods. Then, we give their Butcher tableaux in Section 5 and in Section 6 we study in detail their linear stability. In Section 7, we describe the application to the numerical solution of hyperbolic problems with CG spatial discretizations avoiding mass matrices. We propose an adaptive and efficient version of the methods in Section 8. In Section 9, we present numerical results for ODEs and hyperbolic PDEs with various comparisons with the classical DeC methods. Section 10 is dedicated to the conclusions.

2 Abstract DeC formulation

We will first introduce the DeC abstract formulation proposed by Abgrall in [2]. Let us assume that we have two operators between two normed vector spaces $\left(X, \|\cdot\|_X\right)$ and $\left(Y, \|\cdot\|_Y\right)$, namely $\mathcal{L}^1_\Delta, \mathcal{L}^2_\Delta : X \longrightarrow Y$, associated to two discretizations of the same problem and dependent on a same discretization parameter Δ. In particular, assume that $\mathcal{L}^2_\Delta$ corresponds to a high order implicit discretization, while, $\mathcal{L}^1_\Delta$ corresponds to a low order explicit one. We would like to solve $\mathcal{L}^2_\Delta$, i.e., finding $\underline{\boldsymbol{u}}_\Delta \in X$ such that $\mathcal{L}^2_\Delta(\underline{\boldsymbol{u}}_\Delta) = \mathbf{0}_Y$, to get a high order approximation of the solution to the original problem, but this is not easy because of its implicit character. Instead, the low order explicit operator $\mathcal{L}^1_\Delta$ is very easy to solve and, more in general, we assume that it is easy to solve $\mathcal{L}^1_\Delta(\underline{\boldsymbol{u}}) = \underline{\boldsymbol{r}}$ with $\underline{\boldsymbol{r}} \in Y$ given, but the associated accuracy is not sufficient for our intended goals. In the next theorem, we will provide a simple recipe to get an arbitrary high order approximation of the solution of $\mathcal{L}^2_\Delta$ by combining the operators $\mathcal{L}^1_\Delta$ and $\mathcal{L}^2_\Delta$ in an easy iterative procedure.

Theorem 2.1 (DeC accuracy)**.** *Let the following hypotheses hold*

1. ***Existence of a unique solution to*** $\mathcal{L}^2_\Delta$
   $\exists!\, \underline{\boldsymbol{u}}_\Delta \in X$ *solution of* $\mathcal{L}^2_\Delta$ *such that* $\mathcal{L}^2_\Delta(\underline{\boldsymbol{u}}_\Delta) = \mathbf{0}_Y$*;*
2. ***Coercivity-like property of*** $\mathcal{L}^1_\Delta$
   $\exists\, \alpha_1 \geq 0$ *independent of* Δ *such that*

$$\left\|\mathcal{L}^1_\Delta(\underline{\boldsymbol{v}}) - \mathcal{L}^1_\Delta(\underline{\boldsymbol{w}})\right\|_Y \geq \alpha_1 \left\|\underline{\boldsymbol{v}} - \underline{\boldsymbol{w}}\right\|_X, \ \forall \underline{\boldsymbol{v}}, \underline{\boldsymbol{w}} \in X; \tag{1}$$

3. ***Lipschitz-continuity-like property of*** $\mathcal{L}^1_\Delta - \mathcal{L}^2_\Delta$
   $\exists\, \alpha_2 \geq 0$ *independent of* Δ *such that*

$$\left\|\left(\mathcal{L}^1_\Delta(\underline{\boldsymbol{v}}) - \mathcal{L}^2_\Delta(\underline{\boldsymbol{v}})\right) - \left(\mathcal{L}^1_\Delta(\underline{\boldsymbol{w}}) - \mathcal{L}^2_\Delta(\underline{\boldsymbol{w}})\right)\right\|_Y \leq \alpha_2 \Delta \left\|\underline{\boldsymbol{v}} - \underline{\boldsymbol{w}}\right\|_X, \ \forall \underline{\boldsymbol{v}}, \underline{\boldsymbol{w}} \in X. \tag{2}$$

Then, if we iteratively define $\underline{\boldsymbol{u}}^{(p)}$ *as the solution of*

$$\mathcal{L}^1_\Delta(\underline{\boldsymbol{u}}^{(p)}) = \mathcal{L}^1_\Delta(\underline{\boldsymbol{u}}^{(p-1)}) - \mathcal{L}^2_\Delta(\underline{\boldsymbol{u}}^{(p-1)}), \quad p = 1, \ldots, P, \tag{3}$$

we have that

$$\left\|\underline{\boldsymbol{u}}^{(P)} - \underline{\boldsymbol{u}}_\Delta\right\|_X \leq \left(\Delta \frac{\alpha_2}{\alpha_1}\right)^P \left\|\underline{\boldsymbol{u}}^{(0)} - \underline{\boldsymbol{u}}_\Delta\right\|_X. \tag{4}$$

Proof. The proof relies on a direct use of the hypotheses. In particular, we have

$$\begin{aligned}
\left\|\underline{\boldsymbol{u}}^{(P)} - \underline{\boldsymbol{u}}_\Delta\right\|_X &\leq \frac{1}{\alpha_1} \left\|\mathcal{L}^1_\Delta(\underline{\boldsymbol{u}}^{(P)}) - \mathcal{L}^1_\Delta(\underline{\boldsymbol{u}}_\Delta)\right\|_Y && \text{(5a)}\\
&= \frac{1}{\alpha_1} \left\|\mathcal{L}^1_\Delta(\underline{\boldsymbol{u}}^{(P-1)}) - \mathcal{L}^2_\Delta(\underline{\boldsymbol{u}}^{(P-1)}) - \mathcal{L}^1_\Delta(\underline{\boldsymbol{u}}_\Delta)\right\|_Y && \text{(5b)}\\
&= \frac{1}{\alpha_1} \left\|\mathcal{L}^1_\Delta(\underline{\boldsymbol{u}}^{(P-1)}) - \mathcal{L}^2_\Delta(\underline{\boldsymbol{u}}^{(P-1)}) - \mathcal{L}^1_\Delta(\underline{\boldsymbol{u}}_\Delta) + \mathcal{L}^2_\Delta(\underline{\boldsymbol{u}}_\Delta)\right\|_Y && \text{(5c)}\\
&\leq \Delta \frac{\alpha_2}{\alpha_1} \left\|\underline{\boldsymbol{u}}^{(P-1)} - \underline{\boldsymbol{u}}_\Delta\right\|_X, && \text{(5d)}
\end{aligned}$$

where in (5a) we have used (1), in (5b) the definition of the DeC iteration (3), in (5c) the fact that $\mathcal{L}^2_\Delta(\underline{\boldsymbol{u}}_\Delta) = \mathbf{0}_Y$ and, finally, in (5d) we have used (2). By repeating these calculations recursively we get the desired result. □

Let us remark that, due to our assumption on the operator $\mathcal{L}^1_\Delta$, the updating formula (3) represents a simple explicit recipe to approximate arbitrarily well the solution $\underline{\boldsymbol{u}}_\Delta$ of $\mathcal{L}^2_\Delta$. The convergence for $P \to +\infty$ is ensured independently of the starting vector $\underline{\boldsymbol{u}}^{(0)}$ provided that $\Delta \frac{\alpha_2}{\alpha_1} < 1$. The coefficients α_1 and α_2 can be computed once the operators $\mathcal{L}^1_\Delta$ and $\mathcal{L}^2_\Delta$ are defined. In the next sections, we will provide such definitions for different DeC ODE solvers, and the convergence constraint imposed by $\Delta \frac{\alpha_2}{\alpha_1} < 1$ will sum up to a classical timestep restriction for explicit methods.

If the solution $\underline{\boldsymbol{u}}_\Delta$ of $\mathcal{L}^2_\Delta$ is an R-th order accurate approximation of the exact solution $\underline{\boldsymbol{u}}^{ex}$ of the original problem to which the operators are associated, it does not make sense to approximate $\underline{\boldsymbol{u}}_\Delta$ with accuracy higher than R, as we are actually interested in $\underline{\boldsymbol{u}}^{ex}$. In particular, thanks to the accuracy estimate (4), if $\underline{\boldsymbol{u}}^{(0)}$ is an $O(\Delta)$-approximation of $\underline{\boldsymbol{u}}^{ex}$, the optimal choice is $P = R$, i.e., the optimal number of iterations coincides with the accuracy of the operator $\mathcal{L}^2_\Delta$. Any further iteration results in a waste of computational resources.

In the following, we will characterize the operators $\mathcal{L}^1_\Delta$ and $\mathcal{L}^2_\Delta$ for some DeC ODEs solvers, explicitly writing the associated updating formulas. In order to provide a clearer understanding of the methods, we also report their more classical formulation, in Appendix A, in terms of

residual and error functions [16]. However, we will stick to Abgrall's formulation [2] for its compactness, the possibility to directly work on the solution and its flexibility, which allows for applications to more general contexts, such as structure preserving methods [32, 14, 5, 4], mass-matrix free finite element methods [2, 3, 6], ADER-DG methods [18, 25]. All these generalizations and the efficient modifications that we present in this paper are straightforward in Abgrall's formulation, while they are more involved in the classical DeC framework.

3 The DeC for systems of ODEs

We want to solve the Cauchy problem

$$\begin{cases} \frac{d}{dt}\boldsymbol{u}(t) = \boldsymbol{G}(t, \boldsymbol{u}(t)), \quad t \in [0, T], \\ \boldsymbol{u}(0) = \boldsymbol{z}, \end{cases} \tag{6}$$

with $\boldsymbol{u}(t) \in \mathbb{R}^Q$, $\boldsymbol{z} \in \mathbb{R}^Q$ and $\boldsymbol{G} : \mathbb{R}_0^+ \times \mathbb{R}^Q \to \mathbb{R}^Q$ a continuous map Lipschitz continuous with respect to $\boldsymbol{u}$ uniformly with respect to t with a Lipschitz constant L, which ensures the existence of a unique solution. We will present two explicit DeC methods for the numerical solution of such problem, which are based on approximations of its integral form

- bDeC, which was introduced originally in [24] in a more general family of schemes, but fully exploited for its simplicity only starting from [2] in the context of Galerkin solvers for hyperbolic PDEs without mass matrix. In this method, the integral form is approximated on *"big"* intervals, hence the name bDeC.
- sDeC, which has a longer history [16] and more developments [29, 22, 19, 36]. In this method, the integral form is approximated on *"small"* intervals, hence the name sDeC.

Then, we will consider a general family of DeC methods, αDeC, depending on a parameter α, which contains both the previously described formulations as particular cases, as described in [24].

We assume a one-step method setting: at each time interval $[t_n, t_{n+1}]$, we assume to know $\boldsymbol{u}_n \approx \boldsymbol{u}(t_n)$ and we look for $\boldsymbol{u}_{n+1} \approx \boldsymbol{u}(t_{n+1})$. In particular, as in the context of a general consistency analysis, we assume $\boldsymbol{u}_n = \boldsymbol{u}(t_n)$. In this context, the parameter Δ of the DeC is the step size $\Delta t = t_{n+1} - t_n$. A more traditional but equivalent formulation of bDeC and sDeC in terms of error and residual functions [12, 13, 9, 8] is reported in Appendix A.

3.1 bDeC

In the generic time step $[t_n, t_n + \Delta t]$, we introduce $M + 1$ subtimenodes $t_n = t^0 < t^1 < \cdots < t^M = t_n + \Delta t$. Several choices of subtimenodes are possible, but for the following discussion we will consider equispaced ones. In the numerical tests, we will also present results obtained with Gauss–Lobatto (GL) subtimenodes [32, 18, 16], which can obtain higher accuracy for a fixed number of subtimenodes. We will refer to $\boldsymbol{u}(t^m)$ as the exact solution in the subtimenode t^m and to $\boldsymbol{u}^m$ as the approximation of the solution in the same subtimenode. Just for the first subtimenode, we set $\boldsymbol{u}^0 := \boldsymbol{u}_n$.

The bDeC method is based on the integral version of the ODE (6) in each interval $[t^0, t^m]$, which reads

$$\boldsymbol{u}(t^m) - \boldsymbol{u}^0 - \int_{t^0}^{t^m} \boldsymbol{G}(t, \boldsymbol{u}(t))dt = \boldsymbol{0}, \quad m = 1, \dots, M. \tag{7}$$

Starting from this formulation, we define the high order operator $\mathcal{L}^2_\Delta$ and the low order operator $\mathcal{L}^1_\Delta$. We define $\mathcal{L}^2_\Delta : \mathbb{R}^{(M\times Q)} \to \mathbb{R}^{(M\times Q)}$ by approximating the function $\boldsymbol{G}$ in (7) with a high

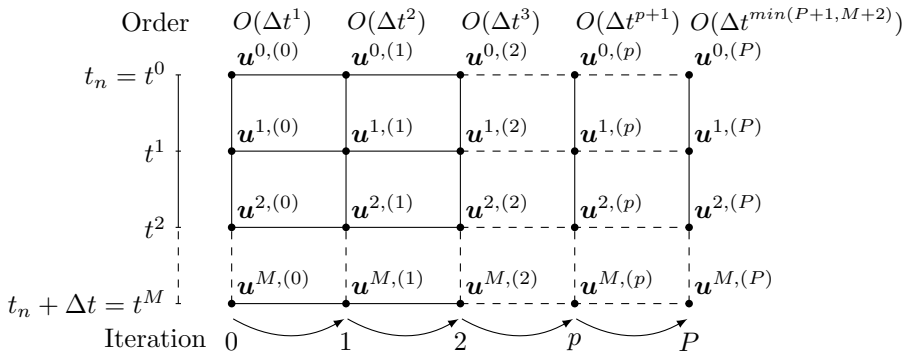

Figure 1: Sketch of the DeC iterative process for equispaced subtimenodes

order interpolation via the Lagrange polynomials ψ^ℓ of degree M associated to the $M+1$ subtimenodes and exact integration of such polynomials

$$\mathcal{L}^2_\Delta(\underline{\boldsymbol{u}}) = \begin{pmatrix} \boldsymbol{u}^1 - \boldsymbol{u}^0 - \Delta t \sum_{\ell=0}^M \theta^1_\ell \boldsymbol{G}(t^\ell, \boldsymbol{u}^\ell) \\ \vdots \\ \boldsymbol{u}^M - \boldsymbol{u}^0 - \Delta t \sum_{\ell=0}^M \theta^M_\ell \boldsymbol{G}(t^\ell, \boldsymbol{u}^\ell) \end{pmatrix}, \text{ with } \underline{\boldsymbol{u}} = \begin{pmatrix} \boldsymbol{u}^1 \\ \vdots \\ \boldsymbol{u}^M \end{pmatrix}, \tag{8}$$

where the normalized coefficients $\theta^m_\ell := \frac{1}{\Delta t}\int_{t^0}^{t^m} \psi^\ell(t)dt$ do not depend on Δt. This leads to the definition of the spaces $X = Y := \mathbb{R}^{M\times Q}$ of Section 2. Let us remark that $\mathcal{L}^2_\Delta$ is defined on the M components $\boldsymbol{u}^m \in \mathbb{R}^Q$ corresponding to the subtimenodes where the solution is unknown, while $\boldsymbol{u}^0$ is an intrinsic datum of the operator. The generic m-th component $\mathcal{L}^{2,m}_\Delta(\underline{\boldsymbol{u}}) = \boldsymbol{0}$ of the global problem $\mathcal{L}^2_\Delta(\underline{\boldsymbol{u}}) = \boldsymbol{0}$ corresponds to a high order discretization of (7). In particular, for equispaced subtimenodes, we have that if $\boldsymbol{u}^m$ is the m-th component of the solution of (8), then, it is an $(M+1)$-th order accurate approximation of $\boldsymbol{u}(t^m)$. The proof is based on a fixed-point argument and can be found in the supplementary material. It is worth noting that $\mathcal{L}^2_\Delta(\underline{\boldsymbol{u}}) = \boldsymbol{0}$ coincides with an implicit RK method with M stages, e.g., when choosing GL subtimenodes one obtains the LobattoIIIA methods.

The definition of the low order explicit operator $\mathcal{L}^1_\Delta : \mathbb{R}^{(M\times Q)} \to \mathbb{R}^{(M\times Q)}$ is based on a first order explicit Euler discretization of (7) leading to

$$\mathcal{L}^1_\Delta(\underline{\boldsymbol{u}}) = \begin{pmatrix} \boldsymbol{u}^1 - \boldsymbol{u}^0 - \Delta t \beta^1 \boldsymbol{G}(t^0, \boldsymbol{u}^0) \\ \vdots \\ \boldsymbol{u}^M - \boldsymbol{u}^0 - \Delta t \beta^M \boldsymbol{G}(t^0, \boldsymbol{u}^0) \end{pmatrix}, \tag{9}$$

where the normalized coefficients $\beta^m = \frac{t^m - t^0}{\Delta t}$ are determined only by the distribution of the subtimenodes. The generic m-th component $\mathcal{L}^{1,m}_\Delta(\underline{\boldsymbol{u}}) = \boldsymbol{0}$ of $\mathcal{L}^1_\Delta(\underline{\boldsymbol{u}}) = \boldsymbol{0}$ corresponds to the explicit Euler discretization of (7), hence, it is first order accurate and any system $\mathcal{L}^1_\Delta(\underline{\boldsymbol{u}}) = \underline{\boldsymbol{r}}$ can be readily solved for a given $\underline{\boldsymbol{r}} \in \mathbb{R}^{M\times Q}$. The operators $\mathcal{L}^1_\Delta$ and $\mathcal{L}^2_\Delta$ fulfill the hypotheses required to apply the DeC procedure, the proofs can be found in the supplementary material. In particular, we highlight that $\alpha_1 = 1$, while $\alpha_2 = L \cdot \max_{m=1,\dots,M} \sum_{\ell=1}^M |\theta^m_\ell|$.

Let us now characterize the updating formula (3) to this setting. The vector $\underline{\boldsymbol{u}}^{(p)} \in \mathbb{R}^{(M\times Q)}$ is, in this case, made by M components $\boldsymbol{u}^{m,(p)} \in \mathbb{R}^Q$, associated to the subtimenodes t^m $m = 1, \dots, M$ in which the solution is unknown, while we set $\boldsymbol{u}^{0,(p)} := \boldsymbol{u}_n$ for all p. Then, (3)

gives

$$\boldsymbol{u}^{m,(p)} = \boldsymbol{u}^0 + \Delta t \sum_{\ell=0}^{M} \theta_\ell^m \boldsymbol{G}(t^\ell, \boldsymbol{u}^{\ell,(p-1)}), \quad m = 1, \dots, M. \tag{10}$$

The starting vector $\underline{\boldsymbol{u}}^{(0)}$ for our iterative procedure is chosen as $\boldsymbol{u}^{m,(0)} := \boldsymbol{u}_n$ for all m. At the end of the iteration process, we set $\boldsymbol{u}_{n+1} := \boldsymbol{u}^{M,(P)}$. A graphical sketch of the updating process is shown in Figure 1. As said in Section 2, the optimal number of iterations depends on the accuracy of the operator $\mathcal{L}_\Delta^2$, i.e., $P = M + 1$ for equispaced subtimenodes and $P = 2M$ for GL ones. Further iterations would not increase the order of accuracy of the method. On the other hand, to build a P-th order method, the optimal choice consists of P iterations with $M = P - 1$ for equispaced and $M = \left\lceil \frac{P}{2} \right\rceil$ for GL subtimenodes.

3.2 sDeC

The sDeC operators differ from the bDeC ones by the "smaller" intervals considered to obtain the integral version of the ODE. In fact, adopting the previous definition of the subtimenodes, the sDeC method is based on the integral version of (6) over the intervals $[t^{m-1}, t^m]$ for $m = 1, \dots, M$. This leads to the following definition of the operators $\mathcal{L}_\Delta^1, \mathcal{L}_\Delta^2 : \mathbb{R}^{(M\times Q)} \to \mathbb{R}^{(M\times Q)}$

$$\mathcal{L}_\Delta^{1,m}(\underline{\boldsymbol{u}}) := \boldsymbol{u}^m - \boldsymbol{u}^{m-1} - \Delta t \gamma^m \boldsymbol{G}(t^{m-1}, \boldsymbol{u}^{m-1}), \qquad \text{for } m = 1, \dots, M, \tag{11}$$

$$\mathcal{L}_\Delta^{2,m}(\underline{\boldsymbol{u}}) := \boldsymbol{u}^m - \boldsymbol{u}^{m-1} - \Delta t \sum_{\ell=0}^{M} \delta_\ell^m \boldsymbol{G}(t^\ell, \boldsymbol{u}^\ell), \qquad \text{for } m = 1, \dots, M, \tag{12}$$

with $\gamma^m = \frac{t^m - t^{m-1}}{\Delta t}$ and $\delta_\ell^m := \frac{1}{\Delta t} \int_{t^{m-1}}^{t^m} \psi^\ell(t) dt$ normalized coefficients. As before, $\mathcal{L}_\Delta^{1,m}(\underline{\boldsymbol{u}}) = \boldsymbol{0}$ is a first order explicit discretization, while, $\mathcal{L}_\Delta^{2,m}(\underline{\boldsymbol{u}}) = \boldsymbol{0}$ is a high order implicit one and, further, we have $\boldsymbol{u}^0 := \boldsymbol{u}_n$.

Differently from the previous formulation, in this case we cannot solve the operator $\mathcal{L}_\Delta^1$ in all its components at the same time but we have to do it component by component from $\boldsymbol{u}^1$ to $\boldsymbol{u}^M$. The same holds for the general problem $\mathcal{L}_\Delta^1(\underline{\boldsymbol{u}}) = \underline{\boldsymbol{r}}$ for a fixed $\underline{\boldsymbol{r}} \in \mathbb{R}^{(M\times Q)}$. However, still the computation of its solution can be performed explicitly.

Let us characterize the updating formula (3) to this context. The explicit character of the operator $\mathcal{L}_\Delta^1$ leads to an explicit recipe for the computation of $\underline{\boldsymbol{u}}^{(p)}$ whose components, in this case, must be computed in increasing order

$$\begin{aligned} \boldsymbol{u}^{m,(p)} = \boldsymbol{u}^{m-1,(p)} &+ \Delta t \gamma^m \left(\boldsymbol{G}(t^{m-1}, \boldsymbol{u}^{m-1,(p)}) - \boldsymbol{G}(t^{m-1}, \boldsymbol{u}^{m-1,(p-1)}) \right) \\ &+ \Delta t \sum_{\ell=0}^{M} \delta_\ell^m \boldsymbol{G}(t^\ell, \boldsymbol{u}^{\ell,(p-1)}). \end{aligned} \tag{13}$$

With recursive substitutions, (13) can be equivalently written as

$$\begin{aligned} \boldsymbol{u}^{m,(p)} = \boldsymbol{u}^0 &+ \Delta t \sum_{\ell=0}^{m-1} \gamma^{\ell+1} \left(\boldsymbol{G}(t^\ell, \boldsymbol{u}^{\ell,(p)}) - \boldsymbol{G}(t^\ell, \boldsymbol{u}^{\ell,(p-1)}) \right) \\ &+ \Delta t \sum_{r=1}^{m} \sum_{\ell=0}^{M} \delta_\ell^r \boldsymbol{G}(t^\ell, \boldsymbol{u}^{\ell,(p-1)}). \end{aligned} \tag{14}$$

Now, let us focus on the last term of (14). Exchanging the sums over r and ℓ, thanks to the fact that $\sum_{r=1}^{m} \delta_\ell^r = \theta_\ell^m$, we have

$$\begin{aligned} \boldsymbol{u}^{m,(p)} = \boldsymbol{u}^0 + \Delta t \sum_{\ell=0}^{m-1} \gamma^{\ell+1} \left(\boldsymbol{G}(t^\ell, \boldsymbol{u}^{\ell,(p)}) - \boldsymbol{G}(t^\ell, \boldsymbol{u}^{\ell,(p-1)}) \right) \\ + \Delta t \sum_{\ell=0}^{M} \theta_\ell^m \boldsymbol{G}(t^\ell, \boldsymbol{u}^{\ell,(p-1)}), \end{aligned} \tag{15}$$

which allows to explicitly compute all the components $\boldsymbol{u}^{m,(p)}$ in sequence from $m = 1$ to $m = M$, in opposition to bDeC where a parallel strategy can be adopted. For what concerns the accuracy of the method and the optimal number of iterations, one can refer to what already said in the context of the bDeC formulation.

Let us observe that the sDeC method is equivalent to the DeC method presented in [16] in terms of residuals and error functions. We show the equivalence in Appendix A.

3.3 A general family of DeC methods, αDeC

Following [21], we can construct a family of schemes dependent on a single parameter $\alpha \in [0, 1]$ by a convex combination of the updating formulas of bDeC (10) and sDeC (15):

$$\begin{aligned} \boldsymbol{u}^{m,(p)} = \boldsymbol{u}^0 + \Delta t \sum_{\ell=0}^{M} \theta_\ell^m \boldsymbol{G}(t^\ell, \boldsymbol{u}^{\ell,(p-1)}) \\ + \alpha \left[\Delta t \sum_{\ell=0}^{m-1} \gamma^{\ell+1} \left(\boldsymbol{G}(t^\ell, \boldsymbol{u}^{\ell,(p)}) - \boldsymbol{G}(t^\ell, \boldsymbol{u}^{\ell,(p-1)}) \right) \right]. \end{aligned} \tag{16}$$

Through (16), it is possible to explicitly compute iteration by iteration the different components $\boldsymbol{u}^{m,(p)}$ starting from $m = 1$ until M. Of course, when $\alpha = 0$ we retrieve the bDeC formulation, while for $\alpha = 1$ we get the sDeC one.

3.3.1 Matrix formulation

We will now introduce a compact matrix-formulation of the presented methods. For convenience, we will now introduce the vectors containing as components the quantities related to all the subtimenodes including the initial one, even if $\boldsymbol{u}^0 = \boldsymbol{u}_n$ is never changed along the iterations and it is not an input of the operators previously described. In order to avoid confusion, we refer to the vectors not containing such component with the small letter and to the vectors containing it with the capital letter, i.e.,

$$\underline{\boldsymbol{u}}^{(p)} = \begin{pmatrix} \boldsymbol{u}^{1,(p)} \\ \vdots \\ \boldsymbol{u}^{M,(p)} \end{pmatrix}, \quad \underline{\boldsymbol{U}}^{(p)} = \begin{pmatrix} \boldsymbol{u}^0 \\ \underline{\boldsymbol{u}}^{(p)} \end{pmatrix}. \tag{17}$$

We will also denote the component-wise application of $\boldsymbol{G}$ to the vectors $\underline{\boldsymbol{u}}^{(p)}$ and $\underline{\boldsymbol{U}}^{(p)}$ by

$$\underline{\boldsymbol{G}}(\underline{\boldsymbol{u}}^{(p)}) = \begin{pmatrix} \boldsymbol{G}(t^1, \boldsymbol{u}^{1,(p)}) \\ \vdots \\ \boldsymbol{G}(t^M, \boldsymbol{u}^{M,(p)}) \end{pmatrix}, \quad \underline{\boldsymbol{G}}(\underline{\boldsymbol{U}}^{(p)}) = \begin{pmatrix} \boldsymbol{G}(t^0, \boldsymbol{u}^0) \\ \underline{\boldsymbol{G}}(\underline{\boldsymbol{u}}^{(p)}) \end{pmatrix}. \tag{18}$$

With the previous definitions, it is possible to recast the general updating formula (16) in the following compact form

$$\begin{aligned}\underline{\boldsymbol{U}}^{(p)} &= \underline{\boldsymbol{U}}^{(0)} + \Delta t \Theta \underline{\boldsymbol{G}}(\underline{\boldsymbol{U}}^{(p-1)}) + \Delta t \alpha \Gamma (\underline{\boldsymbol{G}}(\underline{\boldsymbol{U}}^{(p)}) - \underline{\boldsymbol{G}}(\underline{\boldsymbol{U}}^{(p-1)})) \\ &= \underline{\boldsymbol{U}}^{(0)} + \Delta t (\Theta - \alpha\Gamma) \underline{\boldsymbol{G}}(\underline{\boldsymbol{U}}^{(p-1)}) + \Delta t \alpha \Gamma \underline{\boldsymbol{G}}(\underline{\boldsymbol{U}}^{(p)}),\end{aligned} \tag{19}$$

where the vector $\underline{\boldsymbol{U}}^{(0)} \in \mathbb{R}^{((M+1)\times Q)}$ and the matrices $\Theta, \Gamma \in \mathbb{R}^{(M+1)\times(M+1)}$ are defined as

$$\underline{\boldsymbol{U}}^{(0)} = \begin{pmatrix} \boldsymbol{u}_n \\ \vdots \\ \boldsymbol{u}_n \end{pmatrix}, \Theta = \begin{pmatrix} 0 & 0 & \dots & 0 \\ \theta_0^1 & \theta_1^1 & \dots & \theta_M^1 \\ \theta_0^2 & \theta_1^2 & \dots & \theta_M^2 \\ \vdots & \vdots & \ddots & \vdots \\ \theta_0^M & \theta_1^M & \dots & \theta_M^M \end{pmatrix}, \Gamma = \begin{pmatrix} 0 & 0 & \dots & 0 & 0 \\ \gamma^1 & 0 & \dots & 0 & 0 \\ \gamma^1 & \gamma^2 & \ddots & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ \gamma^1 & \gamma^2 & \dots & \gamma^M & 0 \end{pmatrix}, \tag{20}$$

with the matrix Γ being strictly lower-triangular, as the scheme is fully explicit. Let us observe that the first component $\boldsymbol{u}^0$ of $\underline{\boldsymbol{U}}^{(p)}$ is never updated. This is coherent with what we have said so far. The matrices Θ and Γ that we have defined are referred to a scalar ODE ($Q = 1$). In case one wants to adapt them to a vectorial problem, they must be block-expanded.

4 Two novel families of DeC methods

In this section, we will show how to construct two novel families of efficient DeC methods by introducing a modification in the αDeC methods, first focusing on equispaced subtimenodes and then extending the idea to GL ones. The modification is based on the following observation: at any iteration $p < M + 1$, we get a solution $\underline{\boldsymbol{u}}^{(p)}$ that is p-th order accurate using $M + 1$ subtimenodes even though only p would be formally sufficient to provide such accuracy. In other words, the number of subtimenodes is fixed a priori for all iterations in order to get the desired order of accuracy. These subtimenodes are used throughout the whole iterative process, although the formal order of accuracy, for which such nodes are required, is reached only in the final iteration. This represents indeed a waste of computational resources.

The proposed modification consists in starting with only two subtimenodes and increasing their number, iteration by iteration, matching the order of accuracy achieved in the specific iteration. In particular, we introduce intermediate interpolation processes between the iterations in order to retrieve the needed quantities in the new subtimenodes. The idea has been introduced in [29] for implicit methods, but without a systematic theory and related analytical study. We will present here two possible interpolation strategies which will lead to the definition of two general families of efficient DeC methods.

We will use the star symbol $*$ to refer to quantities obtained through the interpolation process. The number of subtimenodes will change iteration by iteration, therefore, it is useful to define the vector $\underline{t}^{(p)} := \left(t^{0,(p)}, \dots, t^{p,(p)}\right)^T$ of the subtimenodes in which we obtain the approximations of the solution at the p-th iteration, with $t^{0,(p)} = t_n$ and $t^{p,(p)} = t_{n+1}$.

4.1 αDeCu

The αDeCu methods are obtained from the αDeC methods by introducing an intermediate interpolation process on the solution $\boldsymbol{u}(t)$ between the iterations. For convenience, we will formulate the methods in terms of the vectors $\underline{\boldsymbol{U}}^{(p)}$ containing the component $\boldsymbol{u}^0 = \boldsymbol{u}_n$ associated to the initial subtimenode.

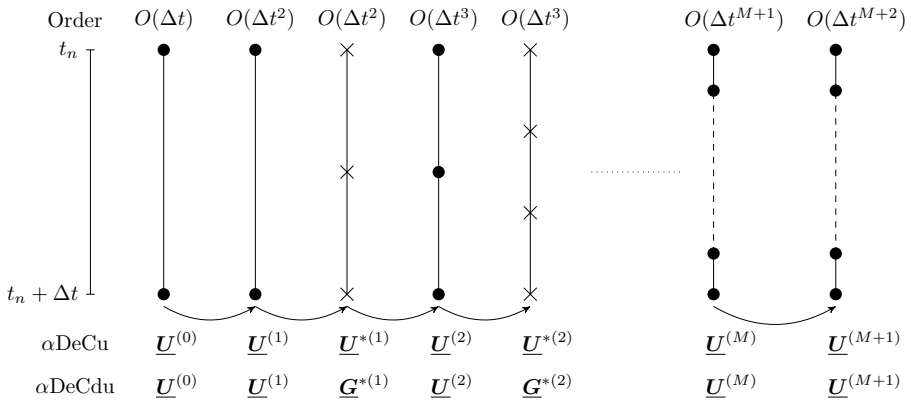

Figure 2: αDeCu and αDeCdu, sketches: dots for computed values, crosses for interpolated ones

We start with $\underline{\boldsymbol{U}}^{(0)} = (\boldsymbol{u}_n, \boldsymbol{u}_n)^T \in \mathbb{R}^{(2\times Q)}$ associated to two subtimenodes, t_n and $t_n + \Delta t$, and we perform the first iteration

$$\underline{\boldsymbol{U}}^{(1)} = \underline{\boldsymbol{U}}^{(0)} + \Delta t(\Theta^{(1)} - \alpha\Gamma^{(1)})\underline{\boldsymbol{G}}(\underline{\boldsymbol{U}}^{(0)}) + \Delta t\alpha\Gamma^{(1)}\underline{\boldsymbol{G}}(\underline{\boldsymbol{U}}^{(1)}) \in \mathbb{R}^{(2\times Q)}. \tag{21}$$

$\underline{\boldsymbol{U}}^{(1)}$ is first order accurate and it yields an $O(\Delta t^2)$-accurate reconstruction on $[t_n, t_{n+1}]$. Here, $\Gamma^{(1)}$ and $\Theta^{(1)}$ are the operators associated to two subtimenodes. Now, we perform the first interpolation, via a suitable interpolation matrix $H^{(1)}$, passing from two to three equispaced subtimenodes

$$\begin{aligned}
\underline{\boldsymbol{U}}^{*(1)} &= H^{(1)}\underline{\boldsymbol{U}}^{(1)} \\
&= H^{(1)}\left[\underline{\boldsymbol{U}}^{(0)} + \Delta t(\Theta^{(1)} - \alpha\Gamma^{(1)})\underline{\boldsymbol{G}}(\underline{\boldsymbol{U}}^{(0)}) + \Delta t\alpha\Gamma^{(1)}\underline{\boldsymbol{G}}(\underline{\boldsymbol{U}}^{(1)})\right] \\
&= \underline{\boldsymbol{U}}_3^{(0)} + \Delta t H^{(1)}(\Theta^{(1)} - \alpha\Gamma^{(1)})\underline{\boldsymbol{G}}(\underline{\boldsymbol{U}}^{(0)}) + \Delta t\alpha H^{(1)}\Gamma^{(1)}\underline{\boldsymbol{G}}(\underline{\boldsymbol{U}}^{(1)}),
\end{aligned} \tag{22}$$

where the last equality is due to the fact that, by consistency, the sum of the elements on the rows of the interpolation matrices $H^{(p)}$ is equal to 1. The subscript 3 has been added to $\underline{\boldsymbol{U}}_3^{(0)} \in \mathbb{R}^{3\times Q}$ to distinguish it from the initial $\underline{\boldsymbol{U}}^{(0)} \in \mathbb{R}^{2\times Q}$. Now, we have $\underline{\boldsymbol{U}}^{*(1)} \in \mathbb{R}^{(3\times Q)}$, still first order accurate. Then, we perform the second iteration

$$\underline{\boldsymbol{U}}^{(2)} = \underline{\boldsymbol{U}}_3^{(0)} + \Delta t(\Theta^{(2)} - \alpha\Gamma^{(2)})\underline{\boldsymbol{G}}(\underline{\boldsymbol{U}}^{*(1)}) + \Delta t\alpha\Gamma^{(2)}\underline{\boldsymbol{G}}(\underline{\boldsymbol{U}}^{(2)}), \tag{23}$$

which gives a second order accurate approximation, i.e., an $O(\Delta t^3)$-accurate approximation. Thus, we continue with another interpolation

$$\begin{aligned}
\underline{\boldsymbol{U}}^{*(2)} &= H^{(2)}\underline{\boldsymbol{U}}^{(2)} \\
&= H^{(2)}\left[\underline{\boldsymbol{U}}_3^{(0)} + \Delta t(\Theta^{(2)} - \alpha\Gamma^{(2)})\underline{\boldsymbol{G}}(\underline{\boldsymbol{U}}^{*(1)}) + \Delta t\alpha\Gamma^{(2)}\underline{\boldsymbol{G}}(\underline{\boldsymbol{U}}^{(2)})\right] \\
&= \underline{\boldsymbol{U}}_4^{(0)} + \Delta t H^{(2)}(\Theta^{(2)} - \alpha\Gamma^{(2)})\underline{\boldsymbol{G}}(\underline{\boldsymbol{U}}^{*(1)}) + \Delta t\alpha H^{(2)}\Gamma^{(2)}\underline{\boldsymbol{G}}(\underline{\boldsymbol{U}}^{(2)}),
\end{aligned} \tag{24}$$

from which we can get $\underline{\boldsymbol{U}}^{(3)}$ $O(\Delta t^4)$-accurate and so on. Proceeding iteratively, at the p-th iteration we have

$$\begin{aligned}
\underline{\boldsymbol{U}}^{*(p-1)} = \underline{\boldsymbol{U}}_{p+1}^{(0)} &+ \Delta t H^{(p-1)}(\Theta^{(p-1)} - \alpha\Gamma^{(p-1)})\underline{\boldsymbol{G}}(\underline{\boldsymbol{U}}^{*(p-2)}) \\
&+ \Delta t\alpha H^{(p-1)}\Gamma^{(p-1)}\underline{\boldsymbol{G}}(\underline{\boldsymbol{U}}^{(p-1)}),
\end{aligned} \tag{25}$$

$$\underline{\boldsymbol{U}}^{(p)} = \underline{\boldsymbol{U}}_{p+1}^{(0)} + \Delta t(\Theta^{(p)} - \alpha\Gamma^{(p)})\underline{\boldsymbol{G}}(\underline{\boldsymbol{U}}^{*(p-1)}) + \Delta t\alpha\Gamma^{(p)}\underline{\boldsymbol{G}}(\underline{\boldsymbol{U}}^{(p)}), \tag{26}$$

where $\underline{\boldsymbol{U}}^{*(p-1)} \in \mathbb{R}^{(p+1)\times Q}$ is $O(\Delta t^p)$-accurate and $\underline{\boldsymbol{U}}^{(p)} \in \mathbb{R}^{(p+1)\times Q}$ is $O(\Delta t^{p+1})$-accurate. Clearly, the DeC operators $\Theta^{(p)}$ and $\Gamma^{(p)}$, used in the p-th iteration, are chosen according to the dimension of involved variables.

Let us notice that $\underline{\boldsymbol{U}}^{(p)} \in \mathbb{R}^{((p+1)\times Q)}$, got at the p-th iteration, is $O(\Delta t^{p+1})$-accurate and associated to $p+1$ subtimenodes but, actually, they would be enough to guarantee $O(\Delta t^{p+2})$-accuracy. For this reason, if the final number of subtimenodes is fixed to be $M+1$, the optimal choice is to perform M iterations to reach such setting and a final $(M+1)$-th iteration without interpolation to saturate the $O(\Delta t^{M+2})$-accuracy associated to the subtimenodes. In this way, we have that the interpolation is performed at each iteration except the first and the last one. Thus, the last iteration reads

$$\underline{\boldsymbol{U}}^{(M+1)} = \underline{\boldsymbol{U}}^{(0)}_{M+1} + \Delta t(\Theta^{(M)} - \alpha\Gamma^{(M)})\underline{\boldsymbol{G}}(\underline{\boldsymbol{U}}^{(M)}) + \Delta t\alpha\Gamma^{(M)}\underline{\boldsymbol{G}}(\underline{\boldsymbol{U}}^{(M+1)}), \tag{27}$$

where the matrices $\Theta^{(M)}$ and $\Gamma^{(M)}$ are the ones used also for the M-th iteration. A useful sketch of the algorithm is represented in Figure 2.

On the other hand, one could also not fix a priori the final number of subtimenodes and stop when certain conditions are met, see an example for adaptive methods in Section 8.

4.2 αDeCdu

Like the αDeCu methods, the αDeCdu methods are based on the introduction of an interpolation process between the iterations. In this case, the interpolated quantity is the function $\boldsymbol{G}(t, \boldsymbol{u}(t))$. The name is due to the fact that formally we interpolate $\frac{d}{dt}\boldsymbol{u}(t) = \boldsymbol{G}(t, \boldsymbol{u}(t))$.

We start with two subtimenodes, associated to t_n and $t_n + \Delta t$, and $\underline{\boldsymbol{U}}^{(0)} \in \mathbb{R}^{(2\times Q)}$ and we perform the first iteration of the αDeC method, as in (21), getting $\underline{\boldsymbol{U}}^{(1)} \in \mathbb{R}^{(2\times Q)}$, which is $O(\Delta t^2)$-accurate. Then, we can compute $\underline{\boldsymbol{G}}(\underline{\boldsymbol{U}}^{(1)})$, whose components allow to get an $O(\Delta t^2)$-accurate global reconstruction of $\boldsymbol{G}(t, \boldsymbol{u}(t))$ in the interval $[t_n, t_n + \Delta t]$ through Lagrange interpolation. We thus perform an interpolation to retrieve the approximated values of $\boldsymbol{G}(t, \boldsymbol{u}(t))$ in three equispaced subtimenodes in the interval $[t_n, t_n + \Delta t]$, getting $\underline{\boldsymbol{G}}^{*(1)} = H^{(1)}\underline{\boldsymbol{G}}(\underline{\boldsymbol{U}}^{(1)}) \in \mathbb{R}^{(3\times Q)}$. Then, we compute

$$\begin{aligned} \underline{\boldsymbol{U}}^{(2)} &= \underline{\boldsymbol{U}}^{(0)}_3 + \Delta t(\Theta^{(2)} - \alpha\Gamma^{(2)})\underline{\boldsymbol{G}}^{*(1)} + \Delta t\alpha\Gamma^{(2)}\underline{\boldsymbol{G}}(\underline{\boldsymbol{U}}^{(2)}) \\ &= \underline{\boldsymbol{U}}^{(0)}_3 + \Delta t(\Theta^{(2)} - \alpha\Gamma^{(2)})H^{(1)}\underline{\boldsymbol{G}}(\underline{\boldsymbol{U}}^{(1)}) + \Delta t\alpha\Gamma^{(2)}\underline{\boldsymbol{G}}(\underline{\boldsymbol{U}}^{(2)}), \end{aligned} \tag{28}$$

which is in $\mathbb{R}^{(3\times Q)}$ and $O(\Delta t^3)$-accurate. We can iteratively continue with interpolations, $\underline{\boldsymbol{G}}^{*(p-1)} = H^{(p-1)}\underline{\boldsymbol{G}}(\underline{\boldsymbol{U}}^{(p-1)})$, and iterations, obtaining the general updating formula

$$\underline{\boldsymbol{U}}^{(p)} = \underline{\boldsymbol{U}}^{(0)}_{p+1} + \Delta t(\Theta^{(p)} - \alpha\Gamma^{(p)})H^{(p-1)}\underline{\boldsymbol{G}}(\underline{\boldsymbol{U}}^{(p-1)}) + \Delta t\alpha\Gamma^{(p)}\underline{\boldsymbol{G}}(\underline{\boldsymbol{U}}^{(p)}), \tag{29}$$

with $\underline{\boldsymbol{U}}^{(p)} \in \mathbb{R}^{((p+1)\times Q)}$ and $O(\Delta t^{p+1})$-accurate. Analogous considerations, as for the αDeCu method, hold on the advantage of performing a final iteration with no interpolation when the final number of subtimenodes is fixed. Also in this case, the reader is referred to Figure 2 for a better understanding of the method.

4.3 αDeCu and αDeCdu with Gauss–Lobatto subtimenodes

As already explained, $M+1$ GL subtimenodes can guarantee an accuracy equal to $2M$. In such a case, if the final number of subtimenodes is fixed, we start with two subtimenodes and we alternate iterations of the αDeC method and interpolations as in the equispaced case, adding one subtimenode at each iteration until reaching the desired $M+1$ subtimenodes, then, we continue with normal iterations of the αDeC until $P = 2M$ to get the maximal order of

accuracy associated to such choice. The updating formulas are identical to the ones already presented. The interpolation is not performed at the first iteration and from the $(M+1)$-th iteration on. On the other hand, if the order P is fixed, the most efficient choice is given by a final number of subtimenodes equal to $M+1$ with $M = \lceil \frac{P}{2} \rceil$ and P iterations.

Contrarily to what one might think, it is not possible to postpone an interpolation process after the saturation of the maximal accuracy associated to some intermediate number of GL subtimenodes adopted in the early iterations. The interpolation processes must mandatorily take place in the first iterations. This is due to the mismatch between the $O(\Delta t^{2p+1})$-accuracy of the operator $\mathcal{L}^2_\Delta$ associated to $p+1$ GL subtimenodes and the $O(\Delta t^{p+1})$-accuracy of the interpolation process with the same number of subtimenodes.

5 The DeC as RK

An explicit RK method with S stages applied in the interval $[t_n, t_{n+1}]$ reads

$$\begin{cases} \boldsymbol{y}^0 = \boldsymbol{u}_n, \\ \boldsymbol{y}^s = \boldsymbol{u}_n + \Delta t \sum_{r=0}^{s-1} a_{s,r} \boldsymbol{G}(t_n + c_r \Delta t, \boldsymbol{y}^r), \quad \text{for } s = 1, \dots, S-1, \\ \boldsymbol{u}_{n+1} = \boldsymbol{u}_n + \Delta t \sum_{r=0}^{S-1} b_r \boldsymbol{G}(t_n + c_r \Delta t, \boldsymbol{y}^r). \end{cases} \tag{30}$$

The coefficients a_{sr}, c_r and b_r uniquely characterize the RK method and can be stored, respectively, into the strictly lower triangular matrix A and the vectors $\boldsymbol{c}$ and $\boldsymbol{b}$, often summarized in a Butcher tableau

$$\begin{array}{c|c} \boldsymbol{c} & A \\ \hline & \boldsymbol{b} \end{array}.$$

It is well known, as presented in [24, 21, 18], that DeC methods can be written into RK form. This also holds for the new methods, αDeCu and αDeCdu. In this section, we will explicitly construct their Butcher tableaux. We will adopt a zero-based numeration and the following convention for slicing. If $\mathcal{M} \in \mathbb{R}^{D_0 \times D_1}$, we denote by $\mathcal{M}_{i:j,k:\ell}$ its slice from the i-th row to the j-th row (included) and from the k-th column to the ℓ-th column (included). We omit the last (first) index in case we want to include all the entries until the end (from the beginning), e.g., $\mathcal{M}_{1:,:6} = M_{1:D_0-1,0:6}$. The same notation is assumed for vectors. We define also the vectors $\underline{\beta}^{(p)} := \left(0, \frac{t^{1,(p)}-t_n}{\Delta t}, \dots, \frac{t^{p,(p)}-t_n}{\Delta t}\right)^T$ of the β^m coefficients in the different iterations of the new methods and, for the original αDeC method, the fixed vector $\underline{\beta} := \left(0, \frac{t^1-t_n}{\Delta t}, \dots, \frac{t^M-t_n}{\Delta t}\right)^T$. In order make the Butcher tableaux as compact as possible, the computation of the solution in the different subtimenodes at the first iteration will be always made through the explicit Euler method. This little modification has no impact on the formal accuracy, since the first iteration is meant to provide a first order approximation of the solution.

We will focus on equispaced subtimenodes. The extension to the GL case is trivial: it suffices to repeat the block without interpolation, related to the final iteration of the standard method, for the needed number of times, $M-1$ in the optimal case.

5.1 αDeC

We recall the general updating formula of the αDeC methods in matricial form

$$\underline{\boldsymbol{U}}^{(p)} = \underline{\boldsymbol{U}}^{(0)} + \Delta t(\Theta - \alpha\Gamma)\underline{\boldsymbol{G}}(\underline{\boldsymbol{U}}^{(p-1)}) + \Delta t \alpha \Gamma \underline{\boldsymbol{G}}(\underline{\boldsymbol{U}}^{(p)}). \tag{31}$$

If we align each iteration one after the other and we consider the approximation in each subtimenode of each iteration as a RK stage, we can pass to the RK formulation. Indeed, we do not

$\boldsymbol{c}$	$\boldsymbol{u}^0$	$\underline{\boldsymbol{u}}^{(1)}$	$\underline{\boldsymbol{u}}^{(2)}$	$\underline{\boldsymbol{u}}^{(3)}$	$\cdots$	$\underline{\boldsymbol{u}}^{(M)}$	$\underline{\boldsymbol{u}}_{:M-1}^{(M+1)}$	A
0	0							$\boldsymbol{u}^0$
$\underline{\beta}_{1:}$	$\underline{\beta}_{1:}$	$\underline{\underline{0}}$						$\underline{\boldsymbol{u}}^{(1)}$
$\underline{\beta}_{1:}$	$\Theta_{1:,0}$	$(\Theta-\alpha\Gamma)_{1:,1:}$	$\alpha\Gamma_{1:,1:}$	$\underline{\underline{0}}$				$\underline{\boldsymbol{u}}^{(2)}$
$\underline{\beta}_{1:}$	$\Theta_{1:,0}$	$\underline{\underline{0}}$	$(\Theta-\alpha\Gamma)_{1:,1:}$	$\alpha\Gamma_{1:,1:}$	$\underline{\underline{0}}$			$\underline{\boldsymbol{u}}^{(3)}$
	$\vdots$	$\vdots$		$\ddots$	$\ddots$			$\vdots$
	$\vdots$	$\vdots$			$\ddots$	$\ddots$		$\vdots$
$\underline{\beta}_{1:M-1}$	$\Theta_{1:M-1,0}$	$\underline{\underline{0}}$	$\cdots$	$\cdots$	$\underline{\underline{0}}$	$(\Theta-\alpha\Gamma)_{1:M-1,1:}$	$\alpha\Gamma_{1:M-1,1:M-1}$	$\underline{\boldsymbol{u}}_{:M-1}^{(M+1)}$
$\boldsymbol{b}$	$\Theta_{M,0}$	$\underline{0}$	$\cdots$	$\cdots$	$\underline{0}$	$(\Theta-\alpha\Gamma)_{M,1:}$	$\alpha\Gamma_{M,1:M-1}$	$\underline{\boldsymbol{u}}^{M,(M+1)}$

Table 1: RK structures for the original αDeC with equispaced subtimenodes, $\boldsymbol{c}$ at the left $\boldsymbol{b}$ at the bottom, A in the middle

$\boldsymbol{c}$	$\boldsymbol{u}^0$	$\underline{\boldsymbol{u}}^{(1)}$	$\underline{\boldsymbol{u}}^{(2)}$	$\underline{\boldsymbol{u}}^{(3)}$	$\cdots$	$\underline{\boldsymbol{u}}^{(M-1)}$	$\underline{\boldsymbol{u}}^{(M)}$	A
0	0							$\boldsymbol{u}^0$
$\underline{\beta}_{1:}$	$\underline{\beta}_{1:}$	$\underline{\underline{0}}$						$\underline{\boldsymbol{u}}^{(1)}$
$\underline{\beta}_{1:}$	$\Theta_{1:,0}$	$\Theta_{1:,1:}$	$\underline{\underline{0}}$					$\underline{\boldsymbol{u}}^{(2)}$
$\underline{\beta}_{1:}$	$\Theta_{1:,0}$	$\underline{\underline{0}}$	$\Theta_{1:,1:}$	$\underline{\underline{0}}$				$\underline{\boldsymbol{u}}^{(3)}$
	$\vdots$	$\vdots$		$\ddots$	$\ddots$			$\vdots$
	$\vdots$	$\vdots$			$\ddots$	$\ddots$		$\vdots$
$\underline{\beta}_{1:}$	$\Theta_{1:,0}$	$\underline{\underline{0}}$	$\cdots$	$\cdots$	$\underline{\underline{0}}$	$\Theta_{1:,1:}$	$\underline{\underline{0}}$	$\underline{\boldsymbol{u}}^{(M)}$
$\boldsymbol{b}$	$\Theta_{M,0}$	$\underline{0}$	$\cdots$	$\cdots$	$\cdots$	$\underline{0}$	$\Theta_{M,1:}$	$\underline{\boldsymbol{u}}^{M,(M+1)}$

Table 2: RK structures for the original bDeC with equispaced subtimenodes, $\boldsymbol{c}$ at the left $\boldsymbol{b}$ at the bottom, A in the middle

repeat the redundant states, i.e., all the $\boldsymbol{u}^{0,(p)} = \boldsymbol{u}_n$, and we keep only $\boldsymbol{u}^0$ as representative of all of them. This leads to the RK formulation (30) with Butcher tableau as in Table 1, where we added on top and on the right side the references to the different iteration steps. The number of stages of this formulation amounts to $S = MP$ for any type of subtimenodes. If $\alpha = 0$, the αDeC method reduces to the bDeC method and the Butcher tableau simplifies to Table 2. In such case, we observe that we do not need the whole vector $\underline{\boldsymbol{u}}^{(P)}$, but we can just compute the component associated to the final subtimenode with the only $\underline{\boldsymbol{u}}^{(P-1)}$, leading to a total number of RK stages equal to $S = M(P-1)+1$.

5.2 bDeCu

Let us recall the general updating formulas of the αDeCu methods

$$\begin{aligned}\underline{\boldsymbol{U}}^{*(p-1)} &= \underline{\boldsymbol{U}}_{p+1}^{(0)} + \Delta t H^{(p-1)}(\Theta^{(p-1)} - \alpha\Gamma^{(p-1)})\underline{\boldsymbol{G}}(\underline{\boldsymbol{U}}^{*(p-2)}) \\ &\quad + \Delta t \alpha H^{(p-1)}\Gamma^{(p-1)}\underline{\boldsymbol{G}}(\underline{\boldsymbol{U}}^{(p-1)}), \qquad (32)\\ \underline{\boldsymbol{U}}^{(p)} &= \underline{\boldsymbol{U}}_{p+1}^{(0)} + \Delta t(\Theta^{(p)} - \alpha\Gamma^{(p)})\underline{\boldsymbol{G}}(\underline{\boldsymbol{U}}^{*(p-1)}) + \Delta t\alpha\Gamma^{(p)}\underline{\boldsymbol{G}}(\underline{\boldsymbol{U}}^{(p)}), \qquad (33)\end{aligned}$$

to which we need to add an initial iteration made with Euler and either a final iteration or, in the context of GL subtimenodes, some final iterations (M in the optimal case) of the standard

$\boldsymbol{c}$	$\boldsymbol{u}^0$	$\underline{\boldsymbol{u}}^{*(1)}$	$\underline{\boldsymbol{u}}^{*(2)}$	$\underline{\boldsymbol{u}}^{*(3)}$	$\cdots$	$\underline{\boldsymbol{u}}^{*(M-2)}$	$\underline{\boldsymbol{u}}^{*(M-1)}$	$\underline{\boldsymbol{u}}^{(M)}$	A	dim
0	0								$\boldsymbol{u}^0$	1
$\underline{\beta}^{(2)}_{1:}$	$\underline{\beta}^{(2)}_{1:}$	$\underline{\underline{0}}$							$\underline{\boldsymbol{u}}^{*(1)}$	2
$\underline{\beta}^{(3)}_{1:}$	$W^{(2)}_{1:,0}$	$W^{(2)}_{1:,1:}$	$\underline{\underline{0}}$						$\underline{\boldsymbol{u}}^{*(2)}$	3
$\underline{\beta}^{(4)}_{1:}$	$W^{(3)}_{1:,0}$	$\underline{\underline{0}}$	$W^{(3)}_{1:,1:}$	$\underline{\underline{0}}$					$\underline{\boldsymbol{u}}^{*(3)}$	4
	$\vdots$	$\vdots$		$\ddots$	$\ddots$				$\vdots$	$\vdots$
	$\vdots$	$\vdots$			$\ddots$	$\ddots$			$\vdots$	$\vdots$
$\underline{\beta}^{(M)}_{1:}$	$W^{(M-1)}_{1:,0}$	$\underline{\underline{0}}$	$\cdots$	$\cdots$	$\underline{\underline{0}}$	$W^{(M-1)}_{1:,1:}$	$\underline{\underline{0}}$	$\underline{\underline{0}}$	$\underline{\boldsymbol{u}}^{*(M-1)}$	M
$\underline{\beta}^{(M)}_{1:}$	$W^{(M)}_{1:,0}$	$\underline{\underline{0}}$	$\cdots$	$\cdots$	$\cdots$	$\underline{\underline{0}}$	$W^{(M)}_{1:,1:}$	$\underline{\underline{0}}$	$\underline{\boldsymbol{u}}^{(M)}$	M
$\boldsymbol{b}$	$W^{(M+1)}_{M,0}$	$\underline{\underline{0}}$	$\cdots$	$\cdots$	$\cdots$	$\cdots$	$\underline{\underline{0}}$	$W^{(M+1)}_{M,1:}$	$\underline{\boldsymbol{u}}^{M,(M+1)}$	

Table 3: RK structures for the bDeCu method, $\boldsymbol{c}$ at the left $\boldsymbol{b}$ at the bottom, A in the middle

αDeC method performed without interpolation. In this case, the stages of the RK method are given by all the components of the vectors $\underline{\boldsymbol{U}}^{(p)}$ and $\underline{\boldsymbol{U}}^{*(p)}$ (excluding the redundant states). From easy computations, one can see that for $\alpha \neq 0$ the number of stages of the αDeCu method coincides with the number of stages of the αDeC method without computational advantage under this point of view. For this reason, we focus on the bDeCu method ($\alpha = 0$), for which we have a substantial computational advantage. In such case, the updating formulas (32) and (33) reduce to

$$\underline{\boldsymbol{U}}^{*(p-1)} = \underline{\boldsymbol{U}}^{(0)}_{p+1} + \Delta t H^{(p-1)} \Theta^{(p-1)} \underline{\boldsymbol{G}}(\underline{\boldsymbol{U}}^{*(p-2)}), \tag{34}$$

$$\underline{\boldsymbol{U}}^{(p)} = \underline{\boldsymbol{U}}^{(0)}_{p+1} + \Delta t \Theta^{(p)} \underline{\boldsymbol{G}}(\underline{\boldsymbol{U}}^{*(p-1)}). \tag{35}$$

The right-hand sides of the previous equations involve the computation of $\boldsymbol{G}$ in interpolated states $\underline{\boldsymbol{U}}^*$ only and, in particular, the update of $\underline{\boldsymbol{U}}^{*(p-1)}$ only depends on $\underline{\boldsymbol{U}}^{*(p-2)}$. This means that the scheme can be rewritten in terms of the vectors $\underline{\boldsymbol{U}}^{*(p)}$ only (plus $\underline{\boldsymbol{U}}^{M,(P)}$), drastically reducing the number of stages. The RK coefficients are reported in Table 3, in which we have

$$W^{(p)} := \begin{cases} H^{(p)}\Theta^{(p)} \in \mathbb{R}^{(p+2)\times(p+1)}, & \text{if } p = 2, \dots, M-1, \\ \Theta^{(M)} \in \mathbb{R}^{(M+1)\times(M+1)}, & \text{if } p \geq M. \end{cases} \tag{36}$$

The total number of RK stages is given by $S = M(P-1) + 1 - \frac{(M-1)(M-2)}{2}$, so $\frac{(M-1)(M-2)}{2}$ less with respect to the original method. The formula holds for both equispaced and GL subtimenodes.

Remark 5.1 (On the relation between stages and computational cost)**.** *The number of stages is not completely explanatory of the computational costs of the new algorithms. In the context of the novel methods, the cost associated to the computation of the different stages is not homogeneous, especially in applications to PDEs, as some of them are "properly" computed through the updating formula* (16) *of the original scheme, while the others are got through an interpolation process which is much cheaper. As an example,* (32) *can be computed as* $\underline{\boldsymbol{U}}^{*(p-1)} = H^{(p-1)}\underline{\boldsymbol{U}}^{(p-2)}$*. In particular, as already specified, the novel* α*DeCu methods for* $\alpha \neq 0$ *are characterized by the same number of stages as the original* α*DeC, nevertheless, roughly half of them is computed through interpolation. For this reason, they have been numerically investigated for* $\alpha = 1$*.*

$\boldsymbol{c}$	$\boldsymbol{u}^0$	$\underline{\boldsymbol{u}}^{(1)}$	$\underline{\boldsymbol{u}}^{(2)}$	$\underline{\boldsymbol{u}}^{(3)}$	$\cdots$	$\underline{\boldsymbol{u}}^{(M-2)}$	$\underline{\boldsymbol{u}}^{(M-1)}$	$\underline{\boldsymbol{u}}^{(M)}$	$\underline{\boldsymbol{u}}_{:M-1}^{(M+1)}$	A	dim
0	0									$\boldsymbol{u}^0$	1
$\underline{\beta}_{1:}^{(1)}$	$\underline{\beta}_{1:}^{(1)}$	$\underline{0}$								$\underline{\boldsymbol{u}}^{(1)}$	1
$\underline{\beta}_{1:}^{(2)}$	$X_{1:,0}^{(2)}$	$X_{1:,1:}^{(2)}$	$Y_{1:,1:}^{(2)}$							$\underline{\boldsymbol{u}}^{(2)}$	2
$\underline{\beta}_{1:}^{(3)}$	$X_{1:,0}^{(3)}$	$\underline{0}$	$X_{1:,1:}^{(3)}$	$Y_{1:,1:}^{(3)}$						$\underline{\boldsymbol{u}}^{(3)}$	3
	$\vdots$	$\vdots$		$\ddots$	$\ddots$					$\vdots$	$\vdots$
	$\vdots$	$\vdots$			$\ddots$	$\ddots$				$\vdots$	$\vdots$
$\underline{\beta}_{1:}^{(M-1)}$	$X_{1:,0}^{(M-1)}$	$\underline{0}$	$\cdots$	$\cdots$	$\underline{0}$	$X_{1:,1:}^{(M-1)}$	$Y_{1:,1:}^{(M-1)}$	$\underline{0}$		$\underline{\boldsymbol{u}}^{(M-1)}$	$M-1$
$\underline{\beta}_{1:}^{(M)}$	$X_{1:,0}^{(M)}$	$\underline{0}$	$\cdots$	$\cdots$	$\cdots$	$\underline{0}$	$X_{1:,1:}^{(M)}$	$Y_{1:,1:}^{(M)}$		$\underline{\boldsymbol{u}}^{(M)}$	M
$\underline{\beta}_{1:M-1}^{(M)}$	$X_{1:M-1,0}^{(M+1)}$	$\underline{0}$	$\cdots$	$\cdots$	$\cdots$	$\cdots$	$\underline{0}$	$X_{1:M-1,1:}^{(M+1)}$	$Y_{1:M-1,1:M-1}^{(M+1)}$	$\underline{\boldsymbol{u}}_{1:M-1}^{(M+1)}$	$M-1$
$\boldsymbol{b}$	$X_{M,0}^{(M+1)}$	$\underline{0}$	$\cdots$	$\cdots$	$\cdots$	$\cdots$	$\underline{0}$	$X_{M,1:}^{(M+1)}$	$Y_{M,1:M-1}^{(M+1)}$	$\underline{\boldsymbol{u}}^{M,(M+1)}$	

Table 4: RK structures for the αDeCdu method with equispaced subtimenodes, $\boldsymbol{c}$ at the left $\boldsymbol{b}$ at the bottom, A in the middle

5.3 αDeCdu

Again, we start by recalling the updating formulas of the method

$$\underline{\boldsymbol{U}}^{(p)} = \underline{\boldsymbol{U}}_{p+1}^{(0)} + \Delta t(\Theta^{(p)} - \alpha\Gamma^{(p)})H^{(p-1)}\underline{\boldsymbol{G}}(\underline{\boldsymbol{U}}^{(p-1)}) + \Delta t\alpha\Gamma^{(p)}\underline{\boldsymbol{G}}(\underline{\boldsymbol{U}}^{(p)}), \tag{37}$$

supplemented with an initial Euler step and a final iteration or, for GL subtimenodes, at most M final iterations of αDeC without interpolation. The usual identification of subtimenodes and RK stages leads to the Butcher tableau in Table 4, in which we have

$$X^{(p)} := \begin{cases} (\Theta^{(p)} - \alpha\Gamma^{(p)})H^{(p-1)} \in \mathbb{R}^{(p+1)\times p}, & \text{if } p = 2,\dots,M, \\ \Theta^{(M)} - \alpha\Gamma^{(M)} \in \mathbb{R}^{(M+1)\times(M+1)}, & \text{if } p > M, \end{cases} \tag{38}$$

$$Y^{(p)} := \begin{cases} \alpha\Gamma^{(p)} \in \mathbb{R}^{(p+1)\times(p+1)}, & \text{if } p = 2,\dots,M, \\ \alpha\Gamma^{(M)} \in \mathbb{R}^{(M+1)\times(M+1)}, & \text{if } p > M. \end{cases} \tag{39}$$

The number of stages in this case amounts to $S = MP - \frac{M(M-1)}{2}$, with a computational advantage of $\frac{M(M-1)}{2}$ with respect to the original method. Also in this case, it is worth giving a particular attention to the method given by $\alpha = 0$. Again, the possibility to compute $\underline{\boldsymbol{u}}^{M,(P)}$ without any need for the other components of $\underline{\boldsymbol{u}}^{(P)}$ further reduces the number of stages to $S = M(P-1) + 1 - \frac{M(M-1)}{2}$.

We conclude this section with two tables, Table 5 and Table 6, containing the number of stages of the original methods and of the novel ones, respectively for equispaced and GL subtimenodes, up to order 13 with associated theoretical speed up factors computed as the ratios between the stages of the original methods and the stages of the modified methods.

6 Stability analysis

In this section, we study the stability of the novel DeC schemes. We will prove two original results. First, the stability functions of bDeCu and bDeCdu coincide with the bDeC ones and do not depend on the distribution of the subtimenodes but only on the order. Second, if we fix the subtimenodes distribution and the order, the αDeCdu methods coincide with the αDeCu

		αDeC			bDeC				
		RK stages		speed up	RK stages			speed up	
P	M	αDeC/αDeCu	αDeCdu	αDeCdu	bDeC	bDeCu	bDeCdu	bDeCu	bDeCdu
2	1	2	2	1.000	2	2	2	1.000	1.000
3	2	6	5	1.200	5	5	4	1.000	1.250
4	3	12	9	1.333	10	9	7	1.111	1.429
5	4	20	14	1.429	17	14	11	1.214	1.545
6	5	30	20	1.500	26	20	16	1.300	1.625
7	6	42	27	1.556	37	27	22	1.370	1.682
8	7	56	35	1.600	50	35	29	1.429	1.724
9	8	72	44	1.636	65	44	37	1.477	1.757
10	9	90	54	1.667	82	54	46	1.519	1.783
11	10	110	65	1.692	101	65	56	1.554	1.804
12	11	132	77	1.714	122	77	67	1.584	1.821
13	12	156	90	1.733	145	90	79	1.611	1.835

Table 5: Number of stages for the original (αDeC, bDeC) and novel (αDeCu, αDeCdu, bDeCu, bDeCdu) methods with equispaced subtimenodes and speed up factor

		αDeC			bDeC				
		RK stages		speed up	RK stages			speed up	
P	M	αDeC/αDeCu	αDeCdu	αDeCdu	bDeC	bDeCu	bDeCdu	bDeCu	bDeCdu
2	1	2	2	1.000	2	2	2	1.000	1.000
3	2	6	5	1.200	5	5	4	1.000	1.250
4	2	8	7	1.143	7	7	6	1.000	1.167
5	3	15	12	1.250	13	12	10	1.083	1.300
6	3	18	15	1.200	16	15	13	1.067	1.231
7	4	28	22	1.273	25	22	19	1.136	1.316
8	4	32	26	1.231	29	26	23	1.115	1.261
9	5	45	35	1.286	41	35	31	1.171	1.323
10	5	50	40	1.250	46	40	36	1.150	1.278
11	6	66	51	1.294	61	51	46	1.196	1.326
12	6	72	57	1.263	67	57	52	1.175	1.288
13	7	91	70	1.300	85	70	64	1.214	1.328

Table 6: Number of stages for the original (αDeC, bDeC) and novel (αDeCu, αDeCdu, bDeCu, bDeCdu) methods with GL subtimenodes and speed up factor

methods on linear problems. For all the schemes, we will show the stability region using some symbolical and numerical tools.

Let us start by reviewing some known results for RK methods [10, 38]. The linear stability of a RK scheme is tested on Dahlquist's problem $u' = \lambda u$, where $\lambda \in \mathbb{C}$ with $Re(\lambda) < 0$. Being the RK schemes linear, we can write a general RK iteration as $u_{n+1} = R(\lambda \Delta t) u_n$, with $R(\cdot)$ the stability function of the method. The stability function is defined as

$$R(z) = 1 + z\boldsymbol{b}^T (I - zA)^{-1} \mathbf{1}, \tag{40}$$

where $\mathbf{1}$ is a vector with all the entries equal to 1. The set of complex numbers z such that $|R(z)| < 1$ is called stability region. We remark that the stability function for explicit RK methods is a polynomial. In fact, the inverse of $(I - zA)$ can be written in Taylor expansion as

$$(I - zA)^{-1} = \sum_{r=0}^{\infty} z^r A^r = I + zA + z^2 A^2 + \dots \tag{41}$$

and, since A is strictly lower triangular, it is nilpotent, i.e., there exists an integer r such that $A^r = \underline{\underline{0}}$ and the minimum of these natural numbers $\mathcal{N}$ is called degree of nilpotence. By definition of $\mathcal{N}$, it is clear that $A^{\mathcal{N}+r} = \underline{\underline{0}}$ for all $r \geq 0$. Moreover, it is also clear that $\mathcal{N} \leq S$, where S is the number of stages of the explicit RK method and the dimension of the matrix A. Hence, $R(z)$ is a polynomial in z with degree at most equal to S. We recall that [38], if a RK method is of order P, then

$$R(z) = 1 + z + \frac{z^2}{2!} + \dots + \frac{z^P}{P!} + O(z^{P+1}). \tag{42}$$

So, we know the first $P + 1$ terms of the stability functions $R(\cdot)$ for all the DeCs of order P presented above. Further, the following result holds.

Theorem 6.1. *The stability function of any bDeC, bDeCu and bDeCdu method of order P is*

$$R(z) = \sum_{r=0}^{P} \frac{z^r}{r!} = 1 + z + \frac{z^2}{2!} + \dots + \frac{z^P}{P!}, \tag{43}$$

and does not depend on the distribution of the subtimenodes.

Proof. The proof of this theorem relies only on the block structure of the matrix A for such schemes. In all these cases, the matrix A can be written as

$$A = \begin{pmatrix} 0 & 0 & 0 & \dots & 0 & 0 \\ \star & 0 & 0 & \dots & 0 & 0 \\ \star & \star & 0 & \dots & 0 & 0 \\ \star & 0 & \star & \dots & 0 & 0 \\ \dots & \dots & \dots & \dots & \dots & \dots \\ \star & 0 & 0 & \cdots & \star & 0 \end{pmatrix}, \tag{44}$$

where $\star$ are some non-zero block matrices and the 0 are some zero block matrices of different sizes. The number of blocks in each row and column of A is P, the order of the scheme. By induction, we can prove that A^k has zeros in the main block diagonal, and in all the $k-1$ block diagonals below the main diagonal, i.e., $(A^k)_{i,j} = 0$ if $i < j + k$, where the indices here refer to the blocks. Indeed, it is true that $A_{i,j} = 0$ if $i < j + 1$. Now, let us consider the entry $(A^{k+1})_{i,j}$ with $i < j + k + 1$, i.e., $i - k < j + 1$. Such entry is defined as $(A^{k+1})_{i,j} = \sum_w (A^k)_{i,w} A_{w,j}$, and we will prove that all the terms of the sum are 0. Let $w < j + 1$, then $A_{w,j} = 0$ because

of the structure of A; while, if $w \geq j+1 > i-k$, we have that $i < w+k$, so $(A^k)_{i,w} = 0$ by induction.

In particular, this means that $A^P = \underline{\underline{0}}$, because any block row index i is smaller than $j+P$ for any block column index j, as P is the number of the blocks that we have in each row and column. Hence,

$$(I - zA)^{-1} = \sum_{r=0}^{\infty} z^r A^r = \sum_{r=0}^{P-1} z^r A^r = I + zA + z^2 A^2 + \cdots + z^{P-1} A^{P-1}. \tag{45}$$

Plugging this result into (40), we can state that the stability function $R(z)$ is a polynomial of degree P, the order of the scheme. Since all the terms of degree lower or equal to P must agree with the expansion of the exponential function (42), the stability function must be (43). Finally, let us notice that no assumption has been made on the distribution of the subtimenodes, hence, the result is general for any distribution. □

In the following, we will show that, given a certain order P and a distribution of subtimenodes, the αDeCu and αDeCdu methods are equivalent on linear problems and, as a consequence, they share the same stability functions.

Theorem 6.2 (Equivalence on linear problems)**.** *Given an order P, a distribution of subtimenodes and $\alpha \in [0,1]$, the schemes αDeCu and αDeCdu applied to linear systems are equivalent.*

Proof. Without loss of generality, we can focus on Dahlquist's equation $u' = \lambda u$. Since the schemes are linear, the same arguments would apply component-wise also on linear systems of equations. Let us start by explicitly writing down the general updating formula (29) of the αDeCdu method for Dahlquist's equation

$$\underline{\boldsymbol{U}}^{(p)} = \underline{\boldsymbol{U}}^{(0)}_{p+1} + \Delta t \lambda (\Theta^{(p)} - \alpha \Gamma^{(p)}) H^{(p-1)} \underline{\boldsymbol{U}}^{(p-1)} + \Delta t \lambda \alpha \Gamma^{(p)} \underline{\boldsymbol{U}}^{(p)}. \tag{46}$$

For the αDeCu method, the updating formula (26) becomes

$$\underline{\boldsymbol{U}}^{(p)} = \underline{\boldsymbol{U}}^{(0)}_{p+1} + \Delta t \lambda (\Theta^{(p)} - \alpha \Gamma^{(p)}) \underline{\boldsymbol{U}}^{*(p-1)} + \Delta t \lambda \alpha \Gamma^{(p)} \underline{\boldsymbol{U}}^{(p)}, \tag{47}$$

now, using the definition of $\underline{\boldsymbol{U}}^{*(p-1)} = H^{(p-1)} \underline{\boldsymbol{U}}^{(p-1)}$, we obtain

$$\underline{\boldsymbol{U}}^{(p)} = \underline{\boldsymbol{U}}^{(0)}_{p+1} + \Delta t \lambda (\Theta^{(p)} - \alpha \Gamma^{(p)}) H^{(p-1)} \underline{\boldsymbol{U}}^{(p-1)} + \Delta t \lambda \alpha \Gamma^{(p)} \underline{\boldsymbol{U}}^{(p)}, \tag{48}$$

which coincides with (46). This means that, at each iteration, the two modified schemes coincide. □

In Figure 3, we depict the stability region of all the presented methods from order 3 to 13. We remark that there is no difference in terms of stability between bDeC, bDeCu and bDeCdu, nor dependence on the distribution of the subtimenodes, as well as sDeCu and sDeCdu have the same stability regions for fixed subtimenodes.

7 Application to hyperbolic PDEs

In this section, we apply the novel explicit efficient DeC techniques to hyperbolic PDEs. We will focus on the CG framework, which is particularly challenging with respect to FV and DG formulations, due to the presence of a global sparse mass matrix. In particular, we will consider two strategies that allow to avoid the related issues. We will describe the operators $\mathcal{L}^1_\Delta$ and $\mathcal{L}^2_\Delta$ for the two strategies in the bDeC formulation and see how to apply the bDeCu efficient modification. The proofs of the properties of the operators are provided in the supplementary material.

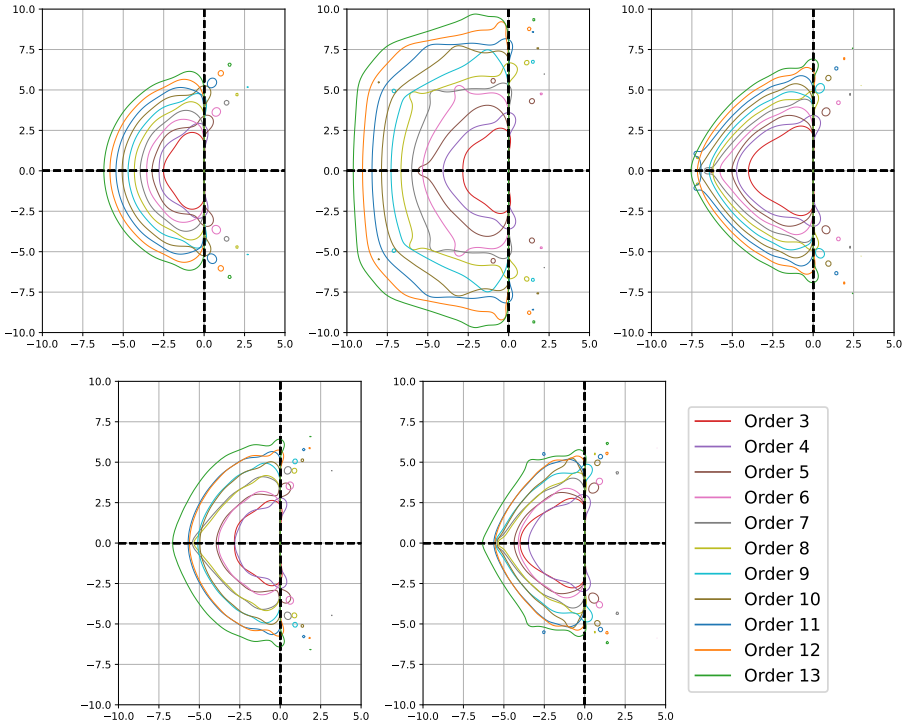

Figure 3: Stability regions for various schemes with order from 3 to 13: bDeC, bDeCu and bDeCdu (equivalent) for any distribution of subtimenodes (top left), sDeC for equispaced subtimenodes (top center), sDeCu and sDeCdu (equivalent) for equispaced subtimenodes (top right), sDeC for GL subtimenodes (bottom left), sDeCu and sDeCdu (equivalent) (bottom center), legend (bottom right)

7.1 Continuous Galerkin FEM

The general form of a hyperbolic system of balance laws reads

$$\frac{\partial}{\partial t}\boldsymbol{u}(\boldsymbol{x},t) + \mathrm{div}_{\boldsymbol{x}}\boldsymbol{F}(\boldsymbol{u}(\boldsymbol{x},t)) = \boldsymbol{S}(\boldsymbol{x},\boldsymbol{u}(\boldsymbol{x},t)), \quad (\boldsymbol{x},t) \in \Omega \times \mathbb{R}_0^+, \tag{49}$$

where $\boldsymbol{u} : \Omega \times \mathbb{R}_0^+ \to \mathbb{R}^Q$, with some initial condition $\boldsymbol{u}(\boldsymbol{x},0) = \boldsymbol{u}_0(\boldsymbol{x})$ on the space domain $\Omega \subseteq \mathbb{R}^D$, and boundary conditions on $\partial\Omega$. We consider a tessellation $\mathcal{T}_h$ of $\overline{\Omega}$ with characteristic length h, made by convex closed polytopals K, and we introduce the space of continuous piecewise polynomial functions $V_h := \{g \in C^0(\overline{\Omega}) \; s.t. \; g|_K \in \mathbb{P}_M(K) \; \forall K \in \mathcal{T}_h\}$. We choose a basis $\{\varphi_i\}_{i=1,\dots,I}$ of V_h, e.g., the Lagrange polynomials or the Bernstein polynomials, which is such that each basis function φ_i can be associated to a degree of freedom (DoF) $\boldsymbol{x}_i \in \overline{\Omega}$ and such that $supp\{\varphi_i\} = \cup_{K \in K_i} K$ with $K_i := \{K \in \mathcal{T}_h \; s.t. \; \boldsymbol{x}_i \in K\}$. Further, we assume a normalization of the basis functions yielding $\sum_{i=1}^I \varphi_i \equiv 1$. Then, we project the weak formulation in space of the PDE (49) over V_h, i.e., we look for $\boldsymbol{u}_h(\boldsymbol{x},t) = \sum_{j=1}^I \boldsymbol{c}_j(t)\varphi_j(\boldsymbol{x}) \in V_h^Q$ such that for any $i = 1, \dots, I$

$$\int_\Omega \left(\frac{\partial}{\partial t}\boldsymbol{u}_h(\boldsymbol{x},t) + \mathrm{div}_{\boldsymbol{x}}\boldsymbol{F}(\boldsymbol{u}_h(\boldsymbol{x},t)) - \boldsymbol{S}(\boldsymbol{x},\boldsymbol{u}_h(\boldsymbol{x},t)) \right) \varphi_i(\boldsymbol{x}) d\boldsymbol{x} + \boldsymbol{ST}_i(\boldsymbol{u}_h) = \boldsymbol{0}, \tag{50}$$

where the stabilization term $\boldsymbol{ST}_i(\boldsymbol{u}_h)$ is added to avoid the instabilities associated to central schemes. Thanks to the assumption on the support of the basis functions, it is possible to recast (50) as

$$\sum_{K \in K_i} \sum_{\boldsymbol{x}_j \in K} \left(\int_K \varphi_i(\boldsymbol{x})\varphi_j(\boldsymbol{x}) d\boldsymbol{x} \right) \frac{d}{dt}\boldsymbol{c}_j(t) + \boldsymbol{\phi}_i(\boldsymbol{c}(t)) = \boldsymbol{0}, \quad i = 1, \dots, I, \tag{51}$$

where $\boldsymbol{c}$ is the vector of all $\boldsymbol{c}_i$ and the space residuals $\boldsymbol{\phi}_i(\boldsymbol{c}(t))$ are defined as

$$\boldsymbol{\phi}_i(\boldsymbol{c}(t)) = \sum_{K \in K_i} \int_K \left(\mathrm{div}_{\boldsymbol{x}}\boldsymbol{F}(\boldsymbol{u}_h(\boldsymbol{x},t)) - \boldsymbol{S}(\boldsymbol{x},\boldsymbol{u}_h(\boldsymbol{x},t)) \right) \varphi_i(\boldsymbol{x}) d\boldsymbol{x} + \boldsymbol{ST}_i(\boldsymbol{u}_h). \tag{52}$$

We would like to solve this system of ODEs in time without solving any linear system at each iteration nor inverting the huge mass matrix.

The first possibility consists in adopting particular basis functions, which, combined with the adoption of the induced quadrature formulas, allow to achieve a high order lumping of the mass matrix. This leads to a system of ODEs like the one described in the previous section and, hence, the novel methods can be applied in a straightforward way. Examples of such basis functions are given by the Lagrange polynomials associated to the GL points in one-dimensional domains and the Cubature elements in two-dimensional domains, introduced in [15] and studied in [20, 33, 26, 27]. The second strategy, introduced by Abgrall in [2] and based on the concept of residual [1, 34, 3, 6], exploits the abstract DeC formulation presented in Section 2, introducing a first order lumping in the mass matrix of the operator $\mathcal{L}_\Delta^1$, resulting in a fully explicit scheme, as we will explain in detail in the following.

7.2 DeC for CG

In this section, we will define the operators $\mathcal{L}_\Delta^1$ and $\mathcal{L}_\Delta^2$ of the DeC formulation for CG FEM discretizations proposed by Abgrall in [2]. In this context, the parameter Δ of the DeC is the mesh parameter h of the space discretization. We assume CFL conditions of the type $\Delta \approx \Delta t \approx h$.

The definition of the high order implicit operator $\mathcal{L}^2_\Delta$ is not very different from the one seen in the context of the bDeC method for ODEs. We denote by $\boldsymbol{c}(t^m)$ the exact solution of the ODE (51) in the subtimenode t^m and by $\boldsymbol{c}^m$ its approximation, containing, respectively, all components $\boldsymbol{c}_i(t^m)$ and $\boldsymbol{c}_i^m$. As usual, for the first subtimenode we set $\boldsymbol{c}^0 = \boldsymbol{c}(t^0) = \boldsymbol{c}(t_n) = \boldsymbol{c}_n$. Starting from the exact integration of (51) over $[t^0, t^m]$ and replacing $\boldsymbol{\phi}_i(\boldsymbol{c}(t))$ by its M-th order interpolation in time associated to the $M+1$ subtimenodes, we get the definition of the operator $\mathcal{L}^2_\Delta : \mathbb{R}^{(I\times Q\times M)} \to \mathbb{R}^{(I\times Q\times M)}$ as

$$\mathcal{L}^2_\Delta(\underline{\boldsymbol{c}}) = \left(\mathcal{L}^2_{\Delta,1}(\underline{\boldsymbol{c}}), \mathcal{L}^2_{\Delta,2}(\underline{\boldsymbol{c}}), \dots, \mathcal{L}^2_{\Delta,I}(\underline{\boldsymbol{c}})\right), \quad \forall \underline{\boldsymbol{c}} \in \mathbb{R}^{(I\times Q\times M)}, \tag{53}$$

where, for any $i = 1, \dots, I$ and $m = 1, \dots, M$, we have

$$\mathcal{L}^{2,m}_{\Delta,i}(\underline{\boldsymbol{c}}) = \sum_{K\in K_i}\sum_{\boldsymbol{x}_j\in K}\left(\int_K \varphi_i(\boldsymbol{x})\varphi_j(\boldsymbol{x})d\boldsymbol{x}\right)\left(\boldsymbol{c}_j^m - \boldsymbol{c}_j^0\right) + \Delta t\sum_{\ell=0}^{M}\theta_\ell^m\boldsymbol{\phi}_i(\boldsymbol{c}^\ell). \tag{54}$$

The solution $\underline{\boldsymbol{c}}_\Delta$ to $\mathcal{L}^2_\Delta(\underline{\boldsymbol{c}}) = \boldsymbol{0}$ is $(M+1)$-th order accurate. Unfortunately, such problem is a huge nonlinear system difficult to directly solve. According to the DeC philosophy, we introduce the operator $\mathcal{L}^1_\Delta$ making use of low order approximations of (51) in order to achieve an explicit formulation. In particular, we use the forward Euler time discretization and a first order mass lumping, obtaining $\mathcal{L}^1_\Delta : \mathbb{R}^{(I\times Q\times M)} \to \mathbb{R}^{(I\times Q\times M)}$

$$\mathcal{L}^1_\Delta(\underline{\boldsymbol{c}}) = \left(\mathcal{L}^1_{\Delta,1}(\underline{\boldsymbol{c}}), \mathcal{L}^1_{\Delta,2}(\underline{\boldsymbol{c}}), \dots, \mathcal{L}^1_{\Delta,I}(\underline{\boldsymbol{c}})\right), \quad \forall \underline{\boldsymbol{c}} \in \mathbb{R}^{(I\times Q\times M)}, \tag{55}$$

whose components, for any $i = 1, \dots, I$ and $m = 1, \dots, M$, are defined as

$$\mathcal{L}^{1,m}_{\Delta,i}(\underline{\boldsymbol{c}}) := C_i\left(\boldsymbol{c}_i^m - \boldsymbol{c}_i^0\right) + \Delta t\beta^m\boldsymbol{\phi}_i(\boldsymbol{c}^0), \tag{56}$$

with $C_i := \int_\Omega \varphi_i(\boldsymbol{x})d\boldsymbol{x}$.

Remark 7.1 (Choice of the basis functions)**.** *For any m and i, we can explicitly compute $\boldsymbol{c}_i^m$ from $\mathcal{L}^{1,m}_{\Delta,i}(\underline{\boldsymbol{c}}) = \boldsymbol{0}$ if and only if $C_i \neq 0$. This means that the construction of the operator $\mathcal{L}^1_\Delta$ is not always well-posed for any arbitrary basis of polynomials. For example, with Lagrange polynomials of degree 2 on triangular meshes, we have $\int_\Omega \varphi_i(\boldsymbol{x})d\boldsymbol{x} = 0$ for some i. However, the construction is always well-posed with Bernstein bases, which verify $C_i > 0\ \forall i$.*

Let us characterize the iterative formula (3) in this context. We have

$$\mathcal{L}^1_\Delta(\underline{\boldsymbol{c}}^{(p)}) = \mathcal{L}^1_\Delta(\underline{\boldsymbol{c}}^{(p-1)}) - \mathcal{L}^2_\Delta(\underline{\boldsymbol{c}}^{(p-1)}), \quad p = 1, \dots, P, \tag{57}$$

where $\underline{\boldsymbol{c}}^{(p)} \in \mathbb{R}^{(I\times Q\times M)}$ consists of M subtimenodes components $\boldsymbol{c}^{m,(p)}$, each of them containing I DoF components $\boldsymbol{c}_i^{m,(p)}$. Just like in the ODE case, procedure (57) results in an explicit iterative algorithm due to the fact that the operator $\mathcal{L}^1_\Delta$ is explicit. After a direct computation, the update of the component associated to the general DoF i in the m-th subtimenode at the p-th iteration reads

$$\begin{aligned}\boldsymbol{c}_i^{m,(p)} = \boldsymbol{c}_i^{m,(p-1)} - \frac{1}{C_i}\Bigg[&\sum_{K\in K_i}\sum_{\boldsymbol{x}_j\in K}\left(\boldsymbol{c}_j^{m,(p-1)} - \boldsymbol{c}_j^0\right)\int_K \varphi_i(\boldsymbol{x})\varphi_j(\boldsymbol{x})d\boldsymbol{x}\\ &+ \Delta t\sum_{\ell=0}^{M}\theta_\ell^m\boldsymbol{\phi}_i(\boldsymbol{c}^{\ell,(p-1)})\Bigg].\end{aligned} \tag{58}$$

We remark that also in this case we assume $\boldsymbol{c}_i^{m,(p)} = \boldsymbol{c}_i(t_n)$ whenever p or m are equal to 0. For what concerns the optimal number of iterations, analogous considerations to the ones made in the ODE case hold. Finally, it is worth observing that the resulting DeC schemes cannot be written in RK form due to the difference between the mass matrices in $\mathcal{L}^1_\Delta$ and $\mathcal{L}^2_\Delta$. In fact, such DeC formulation is not obtained via a trivial application of the method of lines.

7.3 bDeCu for CG

As for ODEs, it is possible to modify the original DeC for hyperbolic problems to get a new more efficient method by introducing interpolation processes between the iterations. The underlying idea is the same, we increase the number of subtimenodes as the accuracy of the approximation increases. At the general iteration p, the interpolation process allows to get $\underline{\boldsymbol{c}}^{*(p-1)}$ from $\underline{\boldsymbol{c}}^{(p-1)}$ and then we perform the iteration via (58) getting

$$\begin{aligned}\boldsymbol{c}_i^{m,(p)} = \boldsymbol{c}_i^{*m,(p-1)} - \frac{1}{C_i}\Bigg[\sum_{K\in K_i}\sum_{\boldsymbol{x}_j\in K}\left(\boldsymbol{c}_j^{*m,(p-1)} - \boldsymbol{c}_j^0\right)\int_K \varphi_i(\boldsymbol{x})\varphi_j(\boldsymbol{x})d\boldsymbol{x} \\ + \Delta t\sum_{\ell=0}^{M}\theta_\ell^m \boldsymbol{\phi}_i(\boldsymbol{c}^{*l,(p-1)})\Bigg].\end{aligned} \tag{59}$$

8 Application to adaptivity

In this section, we will see how to exploit the interpolation processes in the new schemes, αDeCu and αDeCdu, to design adaptive methods. In the context of an original αDeC method with a fixed number of subtimenodes, iteration by iteration, we increase the order of accuracy with respect to the solution $\underline{\boldsymbol{u}}_\Delta$ of the operator $\mathcal{L}_\Delta^2$. For this reason, performing a number of iterations higher than the order of accuracy of the discretization adopted in the construction of the operator $\mathcal{L}_\Delta^2$ is formally useless, as we have already pointed out in Section 2. Instead, in the context of an αDeCu or αDeCdu method, we could in principle keep adding subtimenodes, through interpolation, always improving the accuracy of the approximation with respect to the exact solution of (6), until a convergence condition on the final component of $\underline{\boldsymbol{u}}^{(p)}$ (always associated to t_{n+1}) is met, e.g.,

$$\frac{\left\|\underline{\boldsymbol{u}}^{p,(p)} - \underline{\boldsymbol{u}}^{p-1,(p-1)}\right\|}{\|\underline{\boldsymbol{u}}^{p,(p)}\|} \leq \varepsilon \tag{60}$$

with ε a desired tolerance. This leads to a p-adaptive version of the presented algorithms.

9 Numerical results

In this section, we will numerically investigate the new methods, showing the computational advantage with respect to the original ones. Since the αDeC, αDeCu and αDeCdu methods of order 2 coincide, we will focus on methods from order 3 on.

9.1 ODE tests

We will assess here the properties of the new methods on different ODEs tests, checking their computational costs, their errors and their adaptive versions. We will focus on the methods got for $\alpha = 0$ (bDeC) and $\alpha = 1$ (sDeC).

9.1.1 Linear system

The first test is a very simple 2×2 system of equations

$$\begin{cases} u' = -5u + v, \\ v' = 5u - v, \end{cases} \qquad \begin{pmatrix} u_0 \\ v_0 \end{pmatrix} = \begin{pmatrix} 0.9 \\ 0.1 \end{pmatrix}, \tag{61}$$

with exact solution $u(t) = u_0 + (1 - e^{-6t})(-5u_0 + v_0)$ and $v(t) = 1 - u(t)$. We assume a final time $T = 1$. In Figure 4, we plot the error decay for all methods with respect to Δt for

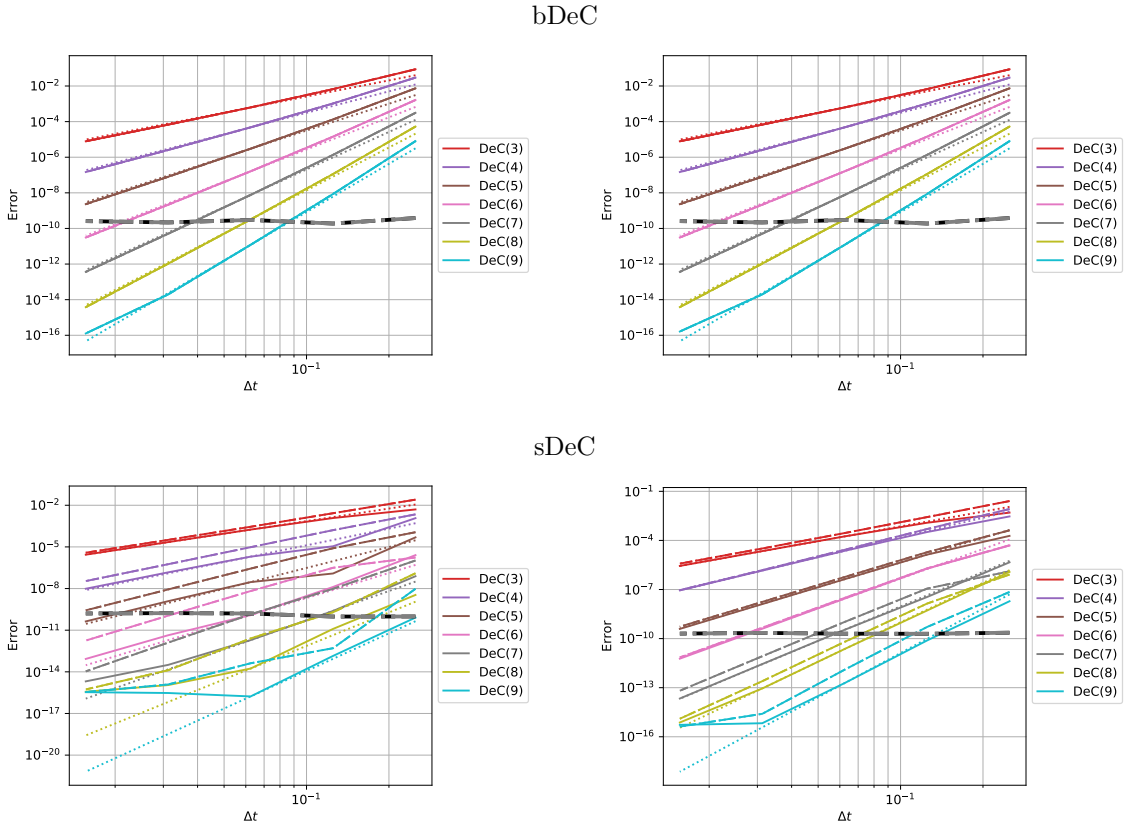

Figure 4: Linear system: Error decay for DeC with continuous line, DeCu with dashed line, DeCdu with dash-dotted line, reference order with dotted line, adaptive DeCu with dashed black line, adaptive DeCdu with dash-dotted gray line. Equispaced subtimenodes on the left and GL on the right

all orders from 3 to 9 and the expected order of convergence is achieved in all cases. We can see that the bDeC, bDeCu and bDeCdu methods have the same error, since they coincide on linear problems, as shown in Theorem 6.1. The sDeC methods show a more irregular behavior and, on average, the errors with the sDeCu and sDeCdu, which coincide due to Theorem 6.2, are slightly larger than the one of sDeC for a fixed Δt. In Figure 5, we plot the error against the computational time of the methods. For bDeC methods there is a huge advantage in using the novel methods: the Pareto front is composed only by the novel methods. In particular, for equispaced subtimenodes there is a larger reduction in computational cost than for GL ones, as predicted by theory. For sDeC methods the situation is not as clear as in the bDeC case. We can systematically see a difference between sDeCu and sDeCdu, being the latter more efficient than the former. In the context of GL subtimenodes, the sDeCdu is slightly better than the original sDeC method from order 5 on in the mesh refinement. We also tested the adaptive versions

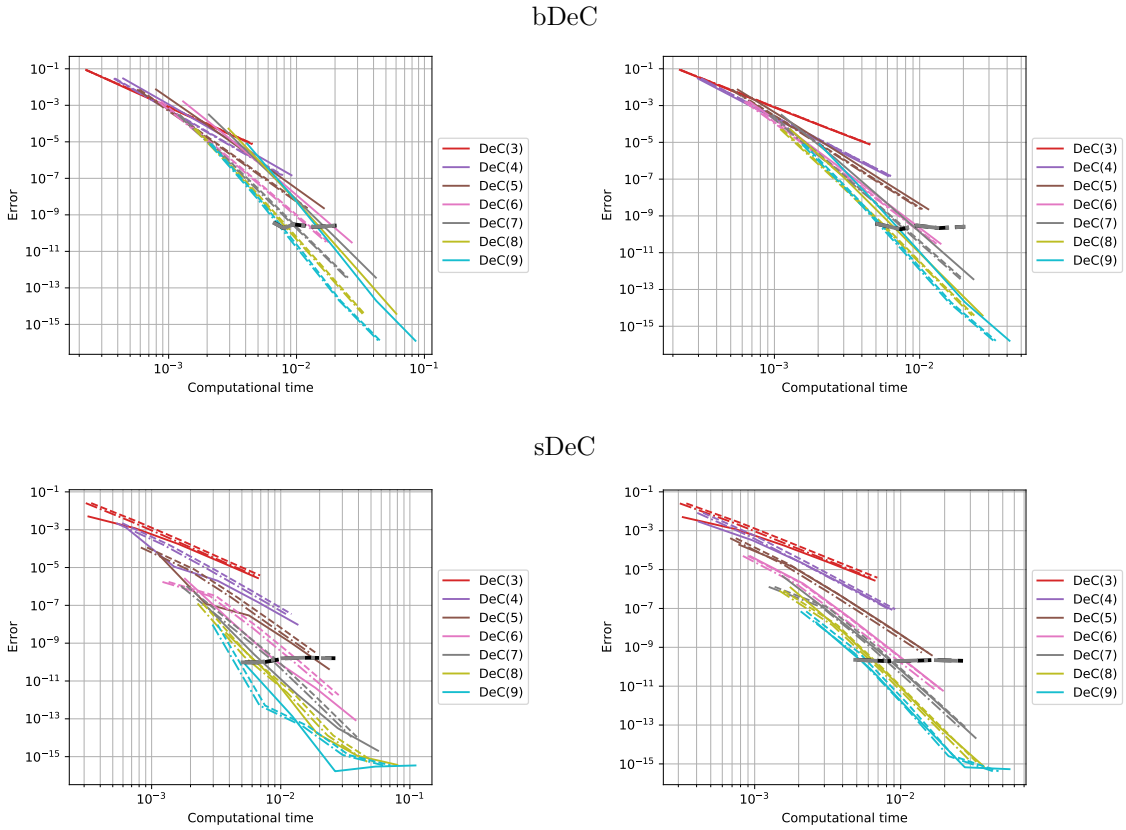

Figure 5: Linear system: Error with respect to computational time for DeC with continuous line, DeCu with dashed line, DeCdu with dash-dotted line, adaptive DeCu with dashed black line, adaptive DeCdu with dash-dotted gray line. Equispaced subtimenodes on the left and GL on the right

of the methods, characterized by the convergence criterion (60) with a tolerance $\varepsilon = 10^{-8}$. As we observe in Figure 4, the error of these methods (in black and gray) is constant and independent of Δt. The required computational time, see Figure 5, is comparable to the one of very high order schemes. In Figure 6, we report the average number of iterations $\pm$ half standard deviation for different adaptive methods with respect to the time discretization. As expected, the smaller the timestep, the smaller is the number of iterations necessary to reach the expected accuracy. In Figure 7, we display, for different Δt, the speed up factor of the bDeCdu method with respect to the bDeC method computed as the ratio between the computational times required by the bDeCdu and the bDeC method. For equispaced subtimenodes we see that, as the order increases, the interpolation process reduces the computational time by an increasing factor, which is almost 2 for order 9. For GL subtimenodes the reduction is smaller but still remarkable, close to $\frac{4}{3}$ in the asymptotic limit.

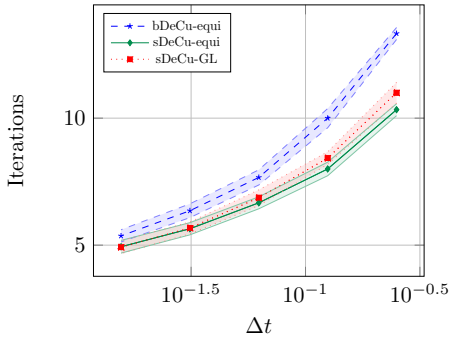

Figure 6: Linear test: Average number of iterations (± half standard deviation) of some adaptive DeC for different time steps

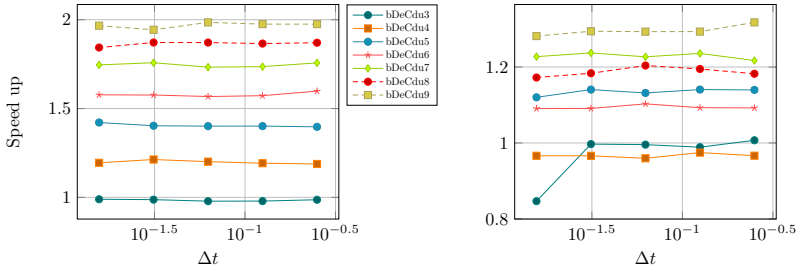

Figure 7: Linear system test: Speed up factor for the bDeCdu method. Equispaced subtimenodes on the left and GL on the right

9.1.2 Vibrating system

Let us consider a vibrating system defined by the following ODE

$$\begin{cases} my'' + ry' + ky = F\cos(\Omega t + \varphi), \quad t \in \mathbb{R}_0^+, \\ y(0) = A, \\ y'(0) = B, \end{cases} \tag{62}$$

with $m, k, \Omega > 0$, $r, F, \psi \geq 0$. Its exact solution [11] reads $y^{ex}(t) = y_h(t) + y_p(t)$ with $y_p(t) = Y_p \cos(\Omega t + \psi)$ particular solution of the whole equation characterized by

$$Y_p = \frac{F}{\sqrt{(-m\Omega^2 + k)^2 + \Omega^2 r^2}}, \qquad \psi = \varphi - \arg\left(-m\Omega^2 + k + i\Omega r\right) \tag{63}$$

and $y_h(t)$ general solution of the homogeneous equation

$$y_h(t) = \begin{cases} C_1 e^{\lambda_1 t} + C_2 e^{\lambda_2 t}, & \text{if } r > 2\sqrt{km}, \\ C_1 e^{\lambda t} + C_2 t e^{\lambda t}, & \text{if } r = 2\sqrt{km}, \\ e^{-\frac{r}{2m}t}\left(C_1\cos(\omega t) + C_2\sin(\omega t)\right), & \text{if } r < 2\sqrt{km}, \end{cases} \tag{64}$$

where $\omega = \frac{1}{2m}\sqrt{4km - r^2}$, λ_1 and λ_2 are the real roots of the characteristics polynomial associated to (62), which are equal to λ when $r = 2\sqrt{km}$. C_1 and C_2 are two constants

computed by imposing the initial conditions $y(0) = A$ and $y'(0) = B$. The mathematical steps needed to get the solution are reported in the supplementary material. The second order scalar ODE (62) can be rewritten in a standard way as a vectorial first order ODE. In the test, we have set $m = 5$, $r = 2$, $k = 5$, $F = 1$, $\Omega = 2$, $\varphi = 0.1$, $A = 0.5$ and $B = 0.25$ with a final time $T = 4$. In Figure 8, we show the error decay for all methods. Differently from the linear

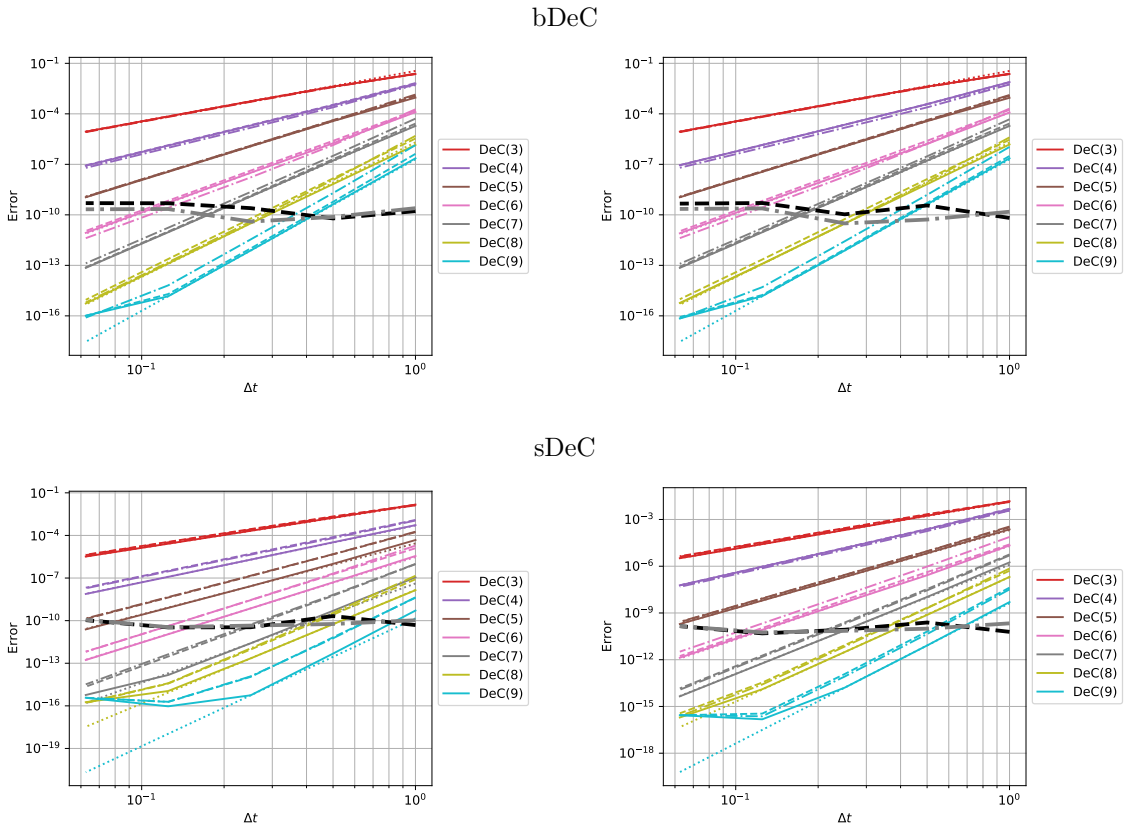

Figure 8: Vibrating system: Error decay for DeC with continuous line, DeCu with dashed line, DeCdu with dash-dotted line, reference order with dotted line, adaptive DeCu with dashed black line, adaptive DeCdu with dash-dotted gray line. Equispaced subtimenodes on the left and GL on the right

case, here bDeC, bDeCu and bDeCdu are not equivalent. Nevertheless, in terms of errors, they behave in a similar way and, also comparing equispaced and GL subtimenodes, we do not observe large deviations. On average the novel schemes are slightly less accurate for a fixed Δt, even if this is not true for all orders of accuracy. For the sDeC, there is a larger difference in the errors between sDeC and sDeCu or sDeCdu, though being the order of accuracy always correct. These effects are visible also in Figure 9. For bDeC with equispaced subtimenodes, the advantages of using the novel methods are evident: the error is almost the same and the computational time reduces by almost half for high order schemes. For bDeC methods with GL subtimenodes the computational advantage of the novel methods is not as big as the one

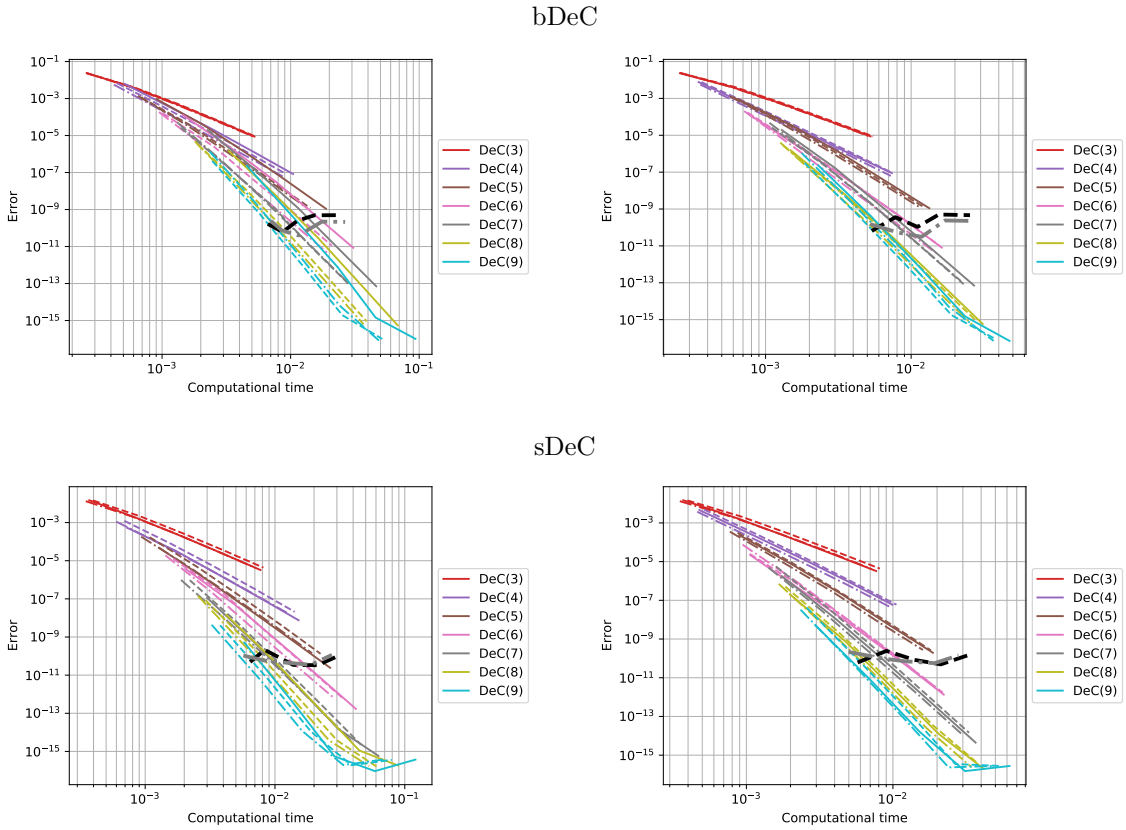

Figure 9: Vibrating system: Error with respect to computational time for DeC with continuous line, DeCu with dashed line, DeCdu with dash-dotted line, adaptive DeCu with dashed black line, adaptive DeCdu with dash-dotted gray line. Equispaced subtimenodes on the left and GL on the right

registered in the previous case as expected from theory, see Table 5 and Table 6, but still pretty visible. For what concerns the sDeC methods with equispaced subtimenodes, the performance of sDeCdu is similar to the one of sDeC until order 5, while, from order 6 on, the novel method is definitely more convenient. The sDeCu method is always less efficient than the sDeCdu one; in particular, only for very high orders it appears to be preferable to the standard method. The general trend of the sDeC methods with GL subtimenodes is that the sDeCdu and the sDeCu always perform, respectively, slightly better and slightly worse than the original sDeC. The results of the adaptive methods for this test are qualitatively similar to the ones seen in the context of the previous test: the methods produce a constant error for any Δt. Also in this case, the threshold for the relative error has been chosen equal to 10^{-8}. Finally, in Figure 10, we display the speed up factor of the new bDeCdu methods with respect to the original bDeC: as expected from theory, it increases with the order of accuracy.

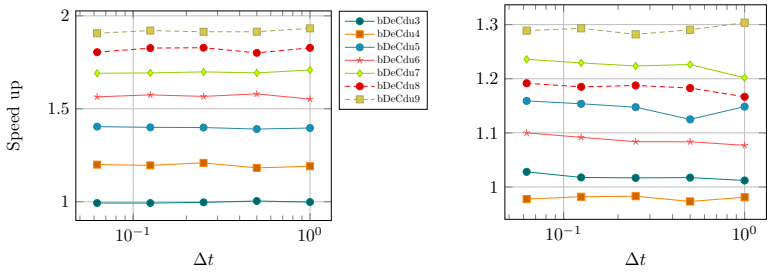

Figure 10: Vibrating system test: Speed up factor for the bDeCdu method. Equispaced subtimenodes on the left and GL on the right

9.2 Hyperbolic PDE tests

For hyperbolic PDEs, we will focus on the bDeC and the bDeCu methods with equispaced subtimenodes. The order of the DeC will be chosen to match the spatial discretization one. We will use two stabilizations discussed in [26, 27]: continuous interior penalty (CIP) and orthogonal subscale stabilization (OSS). The CIP stabilization is defined as

$$\boldsymbol{ST}_i(\boldsymbol{u}_h) = \sum_{f \in \mathcal{F}_h} \alpha_f^{\text{CIP}} \int_f [\![\nabla_{\nu_f} \varphi_i]\!] \cdot [\![\nabla_{\nu_f} \boldsymbol{u}_h]\!] \mathrm{d}\sigma(\boldsymbol{x}), \tag{65}$$

where $\alpha_f^{\text{CIP}} = \delta^{\text{CIP}} \bar{\rho}_f h_f^2$, $\mathcal{F}_h$ is the set of the $(D-1)$-dimensional faces shared by two elements of $\mathcal{T}_h$, $[\![\cdot]\!]$ is the jump across the face f, ∇_{ν_f} is the partial derivative in the direction ν_f normal to the face f, $\bar{\rho}_f$ is a local reference value for the spectral radius of the normal Jacobian of the flux, h_f is the diameter of f and δ^{CIP} is a parameter that must be tuned.

The OSS stabilization is given by

$$\boldsymbol{ST}_i(\boldsymbol{u}_h) = \sum_{K \in \mathcal{T}_h} \alpha_K^{\text{OSS}} \int_K \nabla_{\boldsymbol{x}} \varphi_i \left(\nabla_{\boldsymbol{x}} \boldsymbol{u}_h - \boldsymbol{w}_h \right) \mathrm{d}\boldsymbol{x}, \tag{66}$$

where $\alpha_K^{\text{OSS}} = \delta^{\text{OSS}} \bar{\rho}_K h_K$, $\boldsymbol{w}_h$ is the L^2 projection of $\nabla_{\boldsymbol{x}} \boldsymbol{u}_h$ onto $V_h^{Q \times D}$, $\bar{\rho}_K$ is a local reference value for the spectral radius of the normal Jacobian of the flux, h_K is the diameter of K and δ^{OSS} is a parameter that must be tuned.

9.2.1 1D Linear Advection Equation

We consider the linear advection equation (LAE), $u_t + u_x = 0$, with periodic boundary conditions on the domain $\Omega = [0, 1]$, initial condition $u_0(x) = \cos(2\pi x)$ and final time $T = 1$. The exact solution is given by $u(x, t) = u_0(x - t)$. For the spatial discretization, we considered three families of polynomial basis functions with degree n: Bn, the Bernstein polynomials [3, 2]; Pn, the Lagrange polynomials associated to equispaced nodes; PGLn, the Lagrange polynomials associated to the GL nodes [26]. For Bn and Pn, we used the bDeC version for hyperbolic PDEs (58) introduced by Abgrall; for PGLn, we adopted the bDeC formulation for ODEs (10), as, in this case, the adopted quadrature formula associated to the Lagrangian nodes leads to a high order mass lumping. For all of them, we used the CIP stabilization (65) with the coefficients δ^{CIP} reported in Table 7 found in [26] to minimize the dispersion error, even if, differently from there, we assumed here a constant CFL $= 0.1$. In particular, since the coefficients for P3 and PGL4 were not provided, we used for the former the same coefficient as for B3, while, for the

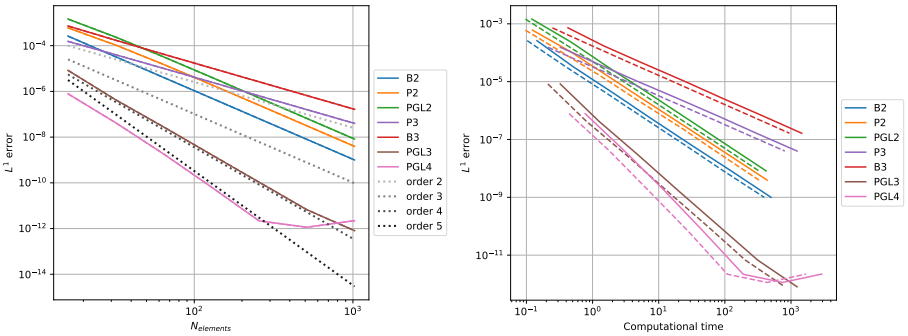

Figure 11: 1D LAE: bDeC with continuous line, bDeCu with dashed line, reference order with dotted line. Convergence analysis on the left and error with respect to computational time on the right

	B2	P2	PGL2	B3	P3*	PGL3	PGL4*
δ^{CIP}	0.016	0.00242	0.00346	0.00702	0.00702	0.000113	0.000113

Table 7: Coefficients δ^{CIP} used for LAE in one dimension. *The coefficients adopted for P3 and PGL4 are not provided in [26].

latter the same coefficient as for PGL3. The results of the convergence analysis and of the computational cost analysis are displayed in Figure 11. For a fixed number of elements, the errors of the bDeC and of the bDeCu methods are essentially identical, leading to a remarkable computational advantage of the novel method with respect to the original bDeC, visible in the plot on the right, where the error against the computational time is depicted. The formal order of accuracy is recovered in all the cases but for B3 and P3 for which we get only second order for both bDeC and bDeCu.

Remark 9.1 (Issues with the DeC for PDEs)**.** *The loss of accuracy for bDeC4 and B3 elements has been registered in other works, e.g. [26, 27, 3]. Even in the original paper [2], the author underlines the necessity to perform more iterations than theoretically expected for orders greater than* 3 *to recover the formal order of accuracy. According to authors' opinion the problem deserves a particular attention, for this reason, the results related to B3 and P3 have not been omitted. The pathology seems to have effect only in the context of unsteady tests and it is maybe due to a high order weak instability. The phenomenon is currently under investigation; more details can be found in the supplementary material. However, we remark that this issue does not occur for elements that allow a proper mass lumping like PGL (or Cubature in 2D).*

The speed up factor of the novel bDeCu with respect to the original method is reported in Figure 12. The obtained speed up factors are higher than ODE ones, because in the implementation of the DeC for PDEs the major cost is not given by the flux evaluation of previously computed stages, but by the evolution of the new stages. This slightly changes the expected and the observed speed up, providing even larger computational advantages.

9.2.2 2D Shallow Water Equations

We consider the Shallow Water (SW) equations onto $\Omega = (0,3) \times (0,3) \in \mathbb{R}^2$, defined, in the form (49), by

$$\boldsymbol{u} = \begin{pmatrix} H \\ H\boldsymbol{v} \end{pmatrix}, \quad \boldsymbol{F}(\boldsymbol{u}) = \begin{pmatrix} H\boldsymbol{v} \\ H\boldsymbol{v} \otimes \boldsymbol{v} + g\frac{H^2}{2}\mathbb{I} \end{pmatrix}, \quad \boldsymbol{S}(\boldsymbol{x}, \boldsymbol{u}) = 0, \tag{67}$$

where H is the water height, $\boldsymbol{v} = (v_1, v_2)^T \in \mathbb{R}^2$ is the vertically averaged speed of the flow, g is the gravitational constant, $\mathbb{I} \in \mathbb{R}^{D\times D}$ is the identity matrix and $D = 2$ is the number of physical dimensions. The test is a $C^6(\Omega)$ compactly supported unsteady vortex from the collection presented in [35] given by

$$\boldsymbol{u} = \boldsymbol{u}^\infty + \begin{cases} \boldsymbol{u}_{r_0}(r), & \text{if } r = ||\boldsymbol{x} - \boldsymbol{x}_m(t)||_2 < r_0, \\ 0, & \text{else,} \end{cases} \tag{68}$$

where $\boldsymbol{u}^\infty = (1,1,1)^T$, $\boldsymbol{x}_m(t) = \boldsymbol{x}_c + t \cdot (1,1)^T$ and

$$\boldsymbol{u}_{r_0}(r) = \begin{pmatrix} \frac{1}{g}\left(\frac{\Gamma}{\omega}\right)^2 (\lambda(\omega r) - \lambda(\pi)) \\ \Gamma\left(1 + \cos(\omega r)\right)^2 (x_2 - x_{m,2}) \\ -\Gamma\left(1 + \cos(\omega r)\right)^2 (x_1 - x_{m,1}) \end{pmatrix}, \ \Gamma = \frac{12\pi\sqrt{g\Delta H}}{r_0\sqrt{315\pi^2 - 2048}} \tag{69}$$

with $\omega = \frac{\pi}{r_0}$ and the function λ defined by

$$\begin{aligned} \lambda(s) = {} & \frac{20}{3}\cos(s) + \frac{27}{16}\cos(s)^2 + \frac{4}{9}\cos(s)^3 + \frac{1}{16}\cos(s)^4 + \frac{20}{3}s\sin(s) \\ & + \frac{35}{16}s^2 + \frac{27}{8}s\cos(s)\sin(s) + \frac{4}{3}s\cos(s)^2\sin(s) + \frac{1}{4}s\cos(s)^3\sin(s). \end{aligned} \tag{70}$$

We set $g = 9.81$, $r_0 = 1$, $\Delta H = 0.1$, $\boldsymbol{x}_c = (1,1)^T$ with a final time $T = 1$ and Dirichlet boundary conditions. For the spatial discretization, we considered two basis functions: Bn, the Bernstein polynomials; Cn, the Cubature elements introduced in [15]. As they allow a high order mass lumping, for Cn elements we used the bDeC (10) for ODEs and OSS stabilization (66), instead, for Bn we considered the PDE formulation (58) and CIP stabilization (65). The tests with B2 have been run with CFL $= 0.1$ and $\delta^{\text{CIP}} = 0.04$; for C2 elements we have set CFL $= 0.12$ and $\delta^{\text{OSS}} = 0.07$, the optimal coefficients minimizing the dispersion error of the original bDeC according to the linear analysis performed in [27]; for C3 we adopted CFL $= 0.015$ and $\delta^{\text{OSS}} = 0.2$. The results of the convergence analysis and of the computational cost analysis are displayed in Figure 13. The errors produced by the novel and the original bDeC method are so close that the lines coincide. The resulting computational advantage can be seen in the plot on the right. The formal order of accuracy is recovered in all the cases and the speed up factor, in Figure 12, proves the convenience in using the novel bDeCu formulation instead of the original bDeC. Let us observe that, according to Table 5, the number of stages of bDeC3 and bDeCu3 is identical, nevertheless, as observed in Remark 5.1, the number of stages does not strictly correspond to the computational time. If we do not consider the "cheap" stages computed via interpolation, we get the theoretical speed up factor $\frac{5}{4} = 1.25$, which is what we obtained in the numerical test for B2. We conclude this section with one last observation: the computational advantage registered with B2 is much higher with respect to C2 and C3 ones, because we have run the simulations with different codes: the results obtained with B2 are obtained with a Fortran implementation, while, for C2 and C3 we have used Parasol, a Python implementation developed by Sixtine Michel [27] and kindly provided to us. A more careful implementation would increase further the speed up factors associated to such elements.

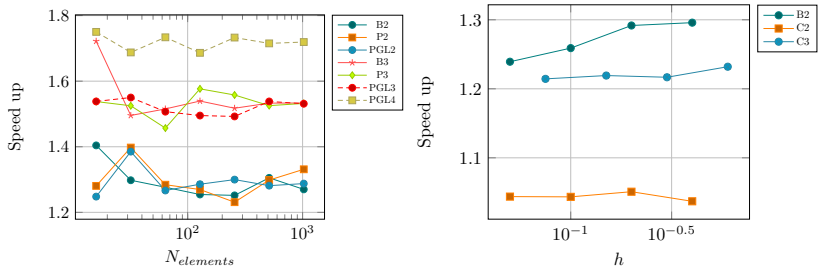

Figure 12: Speed up in the hyperbolic tests of bDeCu with respect to bDeC. 1D LAE on the left and 2D SW on the right

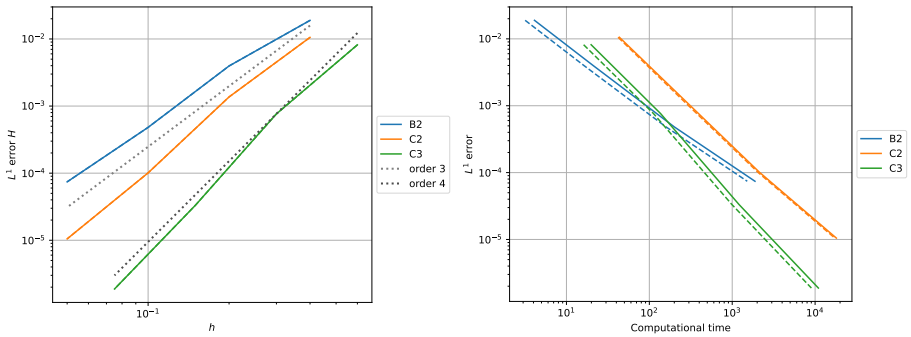

Figure 13: 2D SW: bDeC with continuous line, bDeCu with dashed line, reference order with dotted line. Convergence analysis on the left and error with respect to computational time on the right

10 Conclusions and further developments

In this work, we have investigated analytical and numerical aspects of two novel families of efficient explicit DeC methods. The novel methods are constructed by introducing interpolation processes between the iterations, which increase the degree of the discretization in order to match the accuracy of the approximation associated to the iterations. In particular, we proved that for some of the novel methods the stability region coincides with the one of the original methods. The novel methods have been tested on classical benchmarks in the ODE context revealing, in most of the cases, a remarkable computational advantage with respect to the original ones. Furthermore, the interpolation strategies have been used to design adaptive schemes. Finally, we successfully proved the good performances of the novel methods in the context of the numerical solution of hyperbolic PDEs with continuous space discretizations. Overall, we believe that the approach proposed in this work can alleviate the computational costs not only of DeC methods but also of other schemes with a similar structure. For this reason, investigations of other numerical frameworks are planned and, in particular, we are working on applications to hyperbolic PDEs (with FV and ADER schemes), in which also the order of the space reconstruction is gradually increased iteration by iteration. We hope to spread broadly this technique in the community in order to save computational time and resources in the numerical solution of differential problems, as only little effort is required to

embed the novel modification in an existing DeC code.

Supplementary information

The interested reader is referred to the supplementary material for all the proofs omitted in this document for the sake of compactness.

Acknowledgments

L. Micalizzi has been funded by the SNF grant 200020_204917 "Structure preserving and fast methods for hyperbolic systems of conservation laws" and by the Forschungskredit grant FK-21-098. D. Torlo has been funded by a SISSA Mathematical Fellowship. The authors warmly acknowledge Sixtine Michel for providing the code Parasol.

Conflict of interest

On behalf of all authors, the corresponding author states that there is no conflict of interest.

Compliance with Ethical Standards

On behalf of all authors, the corresponding author is available to collect documentation of compliance with ethical standards and send upon request.

Funding

This work received financial support by the Swiss National Foundation (Switzerland) and Scuola Internazionale Superiore di Studi Avanzati (Italy).

A Residual formulations

Here, we report the residual formulations of the original bDeC and sDeC methods presented in Section 3. In particular, we will present the spectral DeC formulation in terms of residuals introduced in [16] and prove that it is equivalent to the sDeC method. Then, we will see how to get, with a little modification of the presented spectral DeC formulation, the residual formulation of the bDeC method.

A.1 Link between spectral DeC and sDeC

We want to solve system (6) in the interval $[t_n, t_{n+1}]$ getting $\boldsymbol{u}_{n+1} \approx \boldsymbol{u}(t_{n+1})$ from $\boldsymbol{u}_n = \boldsymbol{u}(t_n)$. Also in this case, we consider an iterative procedure on the approximated values $\boldsymbol{u}^{m,(p)}$ of the solution in the subtimenodes $m = 1, \dots, M$, collected in the vector $\underline{\boldsymbol{u}}^{(p)}$, with $\boldsymbol{u}^{0,(p)} := \boldsymbol{u}_n$ fixed. Given $\underline{\boldsymbol{u}}^{(p-1)}$, we consider the interpolation polynomial $\boldsymbol{\mathcal{I}}(\underline{\boldsymbol{u}}^{(p-1)}, t) := \sum_{m=0}^{M} \boldsymbol{u}^{m,(p-1)} \psi^m(t)$. The spectral DeC relies on the definition at each iteration p of two support variables, namely the error function $\boldsymbol{e}^{(p)}(t)$ with respect to the exact solution and the residual function $\boldsymbol{r}^{(p-1)}(t)$ respectively given by

$$\boldsymbol{e}^{(p)}(t) := \boldsymbol{u}(t) - \boldsymbol{\mathcal{I}}(\underline{\boldsymbol{u}}^{(p-1)}, t), \tag{71a}$$

$$\boldsymbol{r}^{(p-1)}(t) := \boldsymbol{u}_n + \int_{t_n}^{t} \boldsymbol{G}(s, \boldsymbol{\mathcal{I}}(\underline{\boldsymbol{u}}^{(p-1)}, s))ds - \boldsymbol{\mathcal{I}}(\underline{\boldsymbol{u}}^{(p-1)}, t). \tag{71b}$$

By integrating the original ODE (6), making use of the definitions of the error function $\boldsymbol{e}^{(p)}(t)$ and of the residual function $\boldsymbol{r}^{(p-1)}(t)$ and differentiating again, we get that the error function satisfies the ODE

$$\begin{cases} \frac{d}{dt}\boldsymbol{e}^{(p)}(t) = \boldsymbol{G}(t, \boldsymbol{\mathcal{I}}(\underline{\boldsymbol{u}}^{(p-1)}, t) + \boldsymbol{e}^{(p)}(t)) - \boldsymbol{G}(t, \boldsymbol{\mathcal{I}}(\underline{\boldsymbol{u}}^{(p-1)}, t)) + \frac{d}{dt}\boldsymbol{r}^{(p-1)}(t), \\ \boldsymbol{e}^{(p)}(t_n) = 0. \end{cases} \tag{71c}$$

We can numerically solve such ODE in each subinterval $[t^{m-1}, t^m]$ through the explicit Euler method starting from $m = 1$ on, thus getting

$$\begin{aligned} \boldsymbol{e}^{m,(p)} = \boldsymbol{e}^{m-1,(p)} + \Delta t \gamma^m \big[\boldsymbol{G}(t^{m-1}, \boldsymbol{u}^{m-1,(p-1)} + \boldsymbol{e}^{m-1,(p)}) \\ - \boldsymbol{G}(t^{m-1}, \boldsymbol{u}^{m-1,(p-1)}) \big] + \boldsymbol{r}^{m,(p-1)} - \boldsymbol{r}^{m-1,(p-1)}, \end{aligned} \tag{71d}$$

with the integrals in the residual function approximated through a spectral integration, i.e., $\boldsymbol{r}^{m,(p-1)} := \boldsymbol{u}_n + \int_{t_n}^{t^m} \sum_{\ell=0}^{M} \boldsymbol{G}(t^\ell, \boldsymbol{u}^{\ell,(p-1)}) \psi^\ell(t) dt - \boldsymbol{u}^{m,(p-1)}$. We have used for $\boldsymbol{e}^{m,(p)}$ and $\boldsymbol{r}^{m,(p-1)}$ the usual convention adopted throughout the manuscript, with m standing for the subtimenode t^m to which such quantities are associated. Indeed, we have $\boldsymbol{e}^{0,(p)} = \boldsymbol{0}$ and $\boldsymbol{r}^{0,(p-1)} = \boldsymbol{0}$. The computed errors are then used to get new approximated values of the solution $\boldsymbol{u}^{m,(p)} := \boldsymbol{u}^{m,(p-1)} + \boldsymbol{e}^{m,(p)}$, allowing to repeat the described process with new error and residual functions, $\boldsymbol{e}^{(p+1)}(t)$ and $\boldsymbol{r}^{(p)}(t)$, analogously defined. The procedure gains one order of accuracy at each iteration until the accuracy of the discretization is saturated and, at the end of the iteration process with P iterations, one can set $\boldsymbol{u}_{n+1} := \boldsymbol{u}^{M,(P)}$. By explicit computation, we have that (71d) is equivalent to the sDeC updating formula (13). In fact, recalling the definition of $\boldsymbol{u}^{m,(p)}$ and $\boldsymbol{r}^{m,(p-1)}$, we get

$$\begin{aligned} \boldsymbol{e}^{m,(p)} = \boldsymbol{e}^{m-1,(p)} + \Delta t \gamma^m \left[\boldsymbol{G}(t^{m-1}, \boldsymbol{u}^{m-1,(p)}) - \boldsymbol{G}(t^{m-1}, \boldsymbol{u}^{m-1,(p-1)}) \right] \\ + \int_{t^{m-1}}^{t^m} \sum_{\ell=0}^{M} \boldsymbol{G}(t^\ell, \boldsymbol{u}^{\ell,(p-1)}) \psi^\ell(t) dt - \boldsymbol{u}^{m,(p-1)} + \boldsymbol{u}^{m-1,(p-1)}, \end{aligned} \tag{71e}$$

from which, recalling the definition of δ_ℓ^m, follows

$$\begin{aligned} \boldsymbol{u}^{m,(p)} = \boldsymbol{u}^{m-1,(p)} + \Delta t \gamma^m \left[\boldsymbol{G}(t^{m-1}, \boldsymbol{u}^{m-1,(p)}) - \boldsymbol{G}(t^{m-1}, \boldsymbol{u}^{m-1,(p-1)}) \right] \\ + \Delta t \sum_{\ell=0}^{M} \delta_\ell^m \boldsymbol{G}(t^\ell, \boldsymbol{u}^{\ell,(p-1)}). \end{aligned} \tag{71f}$$

A.2 bDeC

The residual formulation of the bDeC method is obtained in a similar way. Keeping the same definitions of $\boldsymbol{\mathcal{I}}(\underline{\boldsymbol{u}}^{(p-1)}, t)$, $\boldsymbol{e}^{(p)}(t)$ and $\boldsymbol{r}^{(p-1)}(t)$, we have that (71c) still holds. We solve it through the explicit Euler method in each subinterval $[t^0, t^m]$ obtaining

$$\begin{aligned} \boldsymbol{e}^{m,(p)} = \boldsymbol{e}^{0,(p)} + \Delta t \beta^m \left[\boldsymbol{G}(t^0, \boldsymbol{u}^{0,(p-1)} + \boldsymbol{e}^{0,(p)}) - \boldsymbol{G}(t^0, \boldsymbol{u}^{0,(p-1)}) \right] \\ + \boldsymbol{r}^{m,(p-1)} - \boldsymbol{r}^{0,(p-1)}, \end{aligned} \tag{72a}$$

with the same definition for $\boldsymbol{r}^{m,(p-1)}$ through spectral integration. This is the residual formulation of the bDeC method. Recalling that $\boldsymbol{e}^{0,(p)} = \boldsymbol{r}^{0,(p-1)} = \boldsymbol{0}$, we get

$$\boldsymbol{e}^{m,(p)} = \boldsymbol{r}^{m,(p-1)} = \boldsymbol{u}_n + \int_{t_n}^{t^m} \sum_{\ell=0}^{M} \boldsymbol{G}(t^\ell, \boldsymbol{u}^{\ell,(p-1)}) \psi^\ell(t) dt - \boldsymbol{u}^{m,(p-1)}, \tag{72b}$$

from which, recalling the definition of $\boldsymbol{u}^{m,(p)}$ and of θ_ℓ^m, finally follows

$$\boldsymbol{u}^{m,(p)} = \boldsymbol{u}_n + \Delta t \sum_{\ell=0}^{M} \theta_\ell^m \boldsymbol{G}(t^\ell, \boldsymbol{u}^{\ell,(p-1)}), \tag{72c}$$

which is nothing but (10).

References

[1] Rémi Abgrall. Residual distribution schemes: current status and future trends. *Computers & Fluids*, 35(7):641–669, 2006.

[2] Rémi Abgrall. High order schemes for hyperbolic problems using globally continuous approximation and avoiding mass matrices. *J. Sci. Comput.*, 73(2-3):461–494, 2017.

[3] Rémi Abgrall, Paola Bacigaluppi, and Svetlana Tokareva. High-order residual distribution scheme for the time-dependent Euler equations of fluid dynamics. *Computers & Mathematics with Applications*, 78(2):274–297, 2019.

[4] Rémi Abgrall and Ksenya Ivanova. Staggered residual distribution scheme for compressible flow. *arXiv*, 2111.10647, 2022.

[5] Rémi Abgrall, Élise Le Mélédo, Philipp Öffner, and Davide Torlo. Relaxation Deferred Correction Methods and their Applications to Residual Distribution Schemes. *The SMAI Journal of computational mathematics*, 8:125–160, 2022.

[6] Rémi Abgrall and Davide Torlo. High order asymptotic preserving deferred correction implicit-explicit schemes for kinetic models. *SIAM Journal on Scientific Computing*, 42(3):B816–B845, 2020.

[7] Paola Bacigaluppi, Rémi Abgrall, and Svetlana Tokareva. "A posteriori" limited high order and robust schemes for transient simulations of fluid flows in gas dynamics. *Journal of Computational Physics*, 476:111898, 2023.

[8] Sebastiano Boscarino and Jing-Mei Qiu. Error estimates of the integral deferred correction method for stiff problems. *ESAIM: Mathematical Modelling and Numerical Analysis*, 50(4):1137–1166, 2016.

[9] Sebastiano Boscarino, Jing-Mei Qiu, and Giovanni Russo. Implicit-explicit integral deferred correction methods for stiff problems. *SIAM Journal on Scientific Computing*, 40(2):A787–A816, 2018.

[10] John Charles Butcher. *Numerical methods for ordinary differential equations*. John Wiley & Sons, Auckland, 2016.

[11] Federico Cheli and Giorgio Diana. *Advanced dynamics of mechanical systems*. Springer, Cham, 2015.

[12] Andrew Christlieb, Benjamin Ong, and Jing-Mei Qiu. Comments on high-order integrators embedded within integral deferred correction methods. *Communications in Applied Mathematics and Computational Science*, 4(1):27–56, 2009.

[13] Andrew Christlieb, Benjamin Ong, and Jing-Mei Qiu. Integral deferred correction methods constructed with high order Runge–Kutta integrators. *Mathematics of Computation*, 79(270):761–783, 2010.

[14] M. Ciallella, L. Micalizzi, P. Öffner, and D. Torlo. An arbitrary high order and positivity preserving method for the shallow water equations. *Computers & Fluids*, page 105630, 2022.

[15] Gary Cohen, Patrick Joly, Jean E. Roberts, and Nathalie Tordjman. Higher order triangular finite elements with mass lumping for the wave equation. *SIAM Journal on Numerical Analysis*, 38(6):2047–2078, 2001.

[16] Alok Dutt, Leslie Greengard, and Vladimir Rokhlin. Spectral deferred correction methods for ordinary differential equations. *BIT*, 40(2):241–266, 2000.

[17] Leslie Fox and ET Goodwin. Some new methods for the numerical integration of ordinary differential equations. In *Mathematical Proceedings of the Cambridge Philosophical Society*, volume 45, pages 373–388. Cambridge University Press, 1949.

[18] Maria Han Veiga, Philipp Öffner, and Davide Torlo. Dec and Ader: similarities, differences and a unified framework. *Journal of Scientific Computing*, 87(1):1–35, 2021.

[19] Jingfang Huang, Jun Jia, and Michael Minion. Accelerating the convergence of spectral deferred correction methods. *Journal of Computational Physics*, 214(2):633–656, 2006.

[20] Sébastien Jund and Stéphanie Salmon. Arbitrary High-Order Finite Element Schemes and High-Order Mass Lumping. *International Journal of Applied Mathematics & Computer Science*, 17(3):375–393, 2007.

[21] David Ketcheson and Umair bin Waheed. A comparison of high-order explicit Runge–Kutta, extrapolation, and deferred correction methods in serial and parallel. *Communications in Applied Mathematics and Computational Science*, 9(2):175–200, 2014.

[22] Anita T Layton and Michael L Minion. Conservative multi-implicit spectral deferred correction methods for reacting gas dynamics. *Journal of Computational Physics*, 194(2):697–715, 2004.

[23] Anita T Layton and Michael L Minion. Implications of the choice of quadrature nodes for Picard integral deferred corrections methods for ordinary differential equations. *BIT Numerical Mathematics*, 45(2):341–373, 2005.

[24] Yuan Liu, Chi-Wang Shu, and Mengping Zhang. Strong stability preserving property of the deferred correction time discretization. *Journal of Computational Mathematics*, 26(5):633–656, 2008.

[25] Lorenzo Micalizzi, Davide Torlo, and Walter Boscheri. Efficient iterative arbitrary high order methods: an adaptive bridge between low and high order. *arXiv*, 2212.07783, 2022.

[26] Sixtine Michel, Davide Torlo, Mario Ricchiuto, and Rémi Abgrall. Spectral analysis of continuous FEM for hyperbolic PDEs: influence of approximation, stabilization, and time-stepping. *Journal of Scientific Computing*, 89(2):1–41, 2021.

[27] Sixtine Michel, Davide Torlo, Mario Ricchiuto, and Rémi Abgrall. Spectral analysis of high order continuous fem for hyperbolic pdes on triangular meshes: influence of approximation, stabilization, and time-stepping. *Journal of Scientific Computing*, 94(3):49, 2023.

[28] Michael Minion. A hybrid parareal spectral deferred corrections method. *Communications in Applied Mathematics and Computational Science*, 5(2):265–301, 2011.

[29] Michael L Minion. Semi-implicit spectral deferred correction methods for ordinary differential equations. *Communications in Mathematical Sciences*, 1(3):471–500, 2003.

[30] Michael L Minion. Semi-implicit projection methods for incompressible flow based on spectral deferred corrections. *Applied numerical mathematics*, 48(3-4):369–387, 2004.

[31] Philipp Öffner and Davide Torlo. Arbitrary high-order, conservative and positivity preserving Patankar-type deferred correction schemes. *Applied Numerical Mathematics*, 153:15–34, 2020.

[32] Philipp Öffner and Davide Torlo. Arbitrary high-order, conservative and positivity preserving Patankar-type deferred correction schemes. *Appl. Numer. Math.*, 153:15–34, 2020.

[33] Richard Pasquetti and Francesca Rapetti. Cubature points based triangular spectral elements: An accuracy study. *Journal of Mathematical Study*, 51(1):15–25, 2018.

[34] Mario Ricchiuto and Remi Abgrall. Explicit Runge–Kutta residual distribution schemes for time dependent problems: second order case. *Journal of Computational Physics*, 229(16):5653–5691, 2010.

[35] Mario Ricchiuto and Davide Torlo. Analytical travelling vortex solutions of hyperbolic equations for validating very high order schemes. *arXiv*, 2109.10183, 2021.

[36] Robert Speck, Daniel Ruprecht, Matthew Emmett, Michael Minion, Matthias Bolten, and Rolf Krause. A multi-level spectral deferred correction method. *BIT Numerical Mathematics*, 55(3):843–867, 2015.

[37] Davide Torlo. *Hyperbolic problems: high order methods and model order reduction.* PhD thesis, University Zurich, 2020.

[38] Gerhard Wanner and Ernst Hairer. *Solving ordinary differential equations II: Stiff and Differential-Algebraic Problems*, volume 375. Springer Berlin Heidelberg, Berlin, 1996.