Received: 9 December 2016 | Revised: 30 June 2017 | Accepted: 12 August 2017

DOI: 10.1002/nme.5634

RESEARCH ARTICLE

WILEY

Time integration for extended discontinuous Galerkin methods with moving domains

Florian Kummer[1,2] | Björn Müller[1,2] | Thomas Utz[1,2]

[1]Fluid Dynamics, TU Darmstadt, Otto-Berndt-Str 2 Darmstadt, D-64287, Germany

[2]Graduate School CE, Dolivostr 15 Darmstadt, D-64293, Germany

Correspondence
Fluid Dynamics, TU Darmstadt, Otto-Berndt-Str 2, D-64287 Darmstadt, Germany.
Email: kummer@fdy.tu-darmstadt.de

Funding information
Collaborative Research Center 1194/B06; Research Grant WA 2610/2-1; Priority Programme 1506

Summary

We study time integration schemes for discontinuous Galerkin discretizations of problems with moving immersed interfaces. Two approaches have been discussed in literature so far: splitting schemes and space-time methods. Splitting schemes are cheap and easy to implement, but are non-conservative, inherently limited to low orders of accuracy, and require extremely small time steps. Space-time methods, on the other hand, are conservative, allow for large time steps, and generalize to arbitrary orders of accuracy. However, these advantages come at the expense of a severe growth the systems to be solved. Within this work, we present a generic strategy that combines the advantages of both concepts by evaluating numerical fluxes in a moving reference frame and by making use of a conservative cell-agglomeration strategy. We study the performance of this strategy in combination with backward-difference-formulas and explicit Runge-Kutta schemes in the context of the scalar transport equation, the Burgers equation, and the heat equation. Our results indicate that higher order spatial and temporal convergence rates can be achieved without increasing the size of the systems to be solved.

KEYWORDS

cut cell, discontinuous Galerkin, extended/unfitted DG, level set, XDG, XFEM

1 | INTRODUCTION

1.1 | The general problem

We are interested in the numerical treatment of a conservation laws for the scalar quantity u of the general form

$$\partial_t u + \operatorname{div}\left(\vec{f}(u)\right) = 0 \text{ in } \Omega \subset \mathbb{R}^D \tag{1}$$

with solutions that are sufficiently smooth everywhere, but may feature a discontinuity in u or its gradient at a *known*, $(D-1)$-dimensional, oriented manifold $\mathfrak{I}(t)$, which moves with a velocity s in the normal direction $\vec{n}_\mathfrak{I}$ of the interface. At this *interface*, the conservation law only holds in its weak form; ie, it fulfils the well-known Rankine-Hugoniot condition

$$-s\,[\![u]\!] + \left[\!\left[\vec{f}\right]\!\right] \cdot \vec{n}_\mathfrak{I} = 0, \tag{2}$$

where $[\![-]\!]$ denotes the jump of some property, ie, the difference between the limits approaching from both sides, and $\vec{n}_\mathfrak{I}$ denotes a normal field on $\mathfrak{I}(t)$.

The numerical method we are proposing uses the so-called extended discontinuous Galerkin (XDG) method[1] for spatial discretization. This approach may alternatively be called unfitted discontinuous Galerkin (DG)[2] or cut cell DG. For the temporal discretization, we aim to use backward-difference-formulas (BDF) or Runge-Kutta schemes, which are standard when dealing with problems without a moving interface.

To treat the motion of the interface, we consider and compare 2 different options: The first one (see Section 2.1) is a splitting approach; ie, the interface position is constant during one time step and moved afterwards. This requires an extrapolation from the old cut cell mesh to the new one. The second option (see Section 2.2), here termed *moving interface* approach, actually takes into account the motion of the interface *within* a time step.

Both options have to deal with topology changes in the cut cell mesh (see Section 4), ie, when the interface leaves or enters a cell. For the splitting approach, one can solve that by performing a polynomial interpolation. For the moving interface approach, a cell-agglomeration procedure is used where cut cells, which appear/disappear during one time step, are agglomerated with a neighbour cell.

Finally (Section 5), we perform numerical tests on 3 model problems to compare the precision of both methods. These are scalar convection (Section 5.1), the heat equation (Section 5.2), and Burgers equation (Section 5.3).

Alternatively, one might use some *space-time* extended, continuous or discontinuous finite element method (FEM) as done by Lehrenfeld.[3] These methods are probably more accurate, especially for larger time steps. On the other hand, they have more globally coupled degrees of freedom. Since our application is mainly 2-phase flows that include surface tension effects, we have to use small time steps due to the presence of capillary waves. Under such circumstances, cheap time integration schemes are preferred.

1.2 | The development of time-stepping schemes for extended Galerkin methods

To solve partial differential equations (PDEs) with low-regularity solutions, the idea of adapting the FEM approximation space to the location of, eg, jumps in material parameters is quite old. It can be traced back to the 1970s to works by Babuška[4] and Barrett and Elliott,[5] who investigated Poisson problems with a jump in the diffusion parameter along a smooth, static interface. Such a static interface might be approximated by an interface-fitted grid. However, for moving interfaces, one typically wants to avoid repetitive re-meshing. Moës et al[6] introduced the extended finite element methods (XFEMs), where the static background mesh cut in every time step by an interface represented by a level set. Thus, the FEM space changes in every time step. A suitable time integration procedure must take this into account.

One option for the temporal integration is to split the interface evolution from the PDE. Zilian and Legay[7] use this approach to simulate fluid structure interaction using an XFEM spatial discretization. Another prominent work by Heimann et al[8] investigates multiphase flows using an unfitted/extended DG method. In those works, the interface remains fixed during one time step for the PDE and is evolved between 2 time steps.

For the next group of methods, time discretization is performed before spatial discretization. This leads to terms that include products of the basis functions at both the old and new time steps. Using this technique, one gets rid of the extrapolation operation, but this requires integration techniques for the intersection of the old and new cut cells. Merle and Dolbow[9] use a similar approach to simulate the Stefan problem. Chessa et al use this approach to simulate solidification problems[10] as well as multiphase flows in follow-up study[11] using a FEM with gradient enrichment. Fries and Zilian[12] use this method in the context of XFEM with Heaviside enrichment for the simulation of diffusion, advection-diffusion, and Burgers problems.

The last group of methods uses a quite different approach: The interface moving through the spatial domain in time can be interpreted as a fixed interface in the space-time domain. Thus, the whole problem is discretized by an extended Galerkin method in time and space. This approach has gathered quite some attention in the recent past for various applications: For example, Réthoré et al[13] use this approach to simulate crack propagation. Chessa and Belytschko[14] use a piecewise linear space-time XFEM with Heaviside enrichment to simulate linear advection and Burgers problems. Lehrenfeld[3] uses an XFEM function space for the spatial discretization, ie, continuous between non–cut elements in space with a DG discretization in the time domain, ie, discontinuous functions between time slabs to simulate multiphase flows with surface tension. Cheng and Fries[15] present a method using an XFEM space with different gradient enrichments to deal with curved interfaces for steady 2-dimensional (2D) problems and unsteady 1-dimensional (1D) problems. The main benefit of this method is high accuracy, especially for large time step; however, it requires the integration over space-time cut cells. The resulting systems are significantly larger, compared to classical time-stepping schemes, since the degrees of

freedom at multiple time steps are coupled. Very recently, Hansbo et al[16] presented a method that combines a DG method in time with a continuous Galerkin method in space to solve coupled bulk-interface problems.

Kramer and Noble[17] developed an arbitrary Lagrangian-Eulerian (ALE) formulation to discretize problems with moving interfaces in the context of XFEM. The ALE formulations are very similar to space-time formulations: The space-time Galerkin formulation, when written as a time integral over space integrals, leads to similar formulas as the ALE approach.

1.3 | The extended discontinuous Galerkin method

We briefly introduce a notational framework for the DG method and the XDG method. More details about the spatial discretization are given in Kummer.[1] Since this work is not concerned with the numerical analysis of the spatial discretization scheme, we found it useful *not* to use a usual variational notation, as, e.g., used throughout the book of Di Pietro and Ern.[18] Instead, we set up a cell-local basis of the DG, resp. the XDG space, and specify the method with respect to this basis. An advantage of this notation is that it is closer to an actual implementation and is thus more instructional for discussing the differences between different approaches.

In a first step, we introduce the nomenclature for a standard DG discretization. We define the following:

- A numerical grid/mesh
$$\mathfrak{K}_h = (K_1, \dots, K_J), \overline{\Omega} = \bigcup_j \overline{K}_j, K_j \cap K_i = \emptyset \text{ for } i \neq j, \tag{3}$$
where Ω is the computational domain.
- The standard DG space of (absolute) degree k on $\mathfrak{K}_h$
$$\mathbb{P}_k(\mathfrak{K}_h) := \left\{ f \in L^2(\Omega); \forall K \in \mathfrak{K}_h : f|_K \text{ is polynomial and deg } (f|_K) \leq k \right\}. \tag{4}$$
- A basis of $\mathbb{P}_k(\mathfrak{K}_h)$, written as a row vector
$$\underline{\phi} = \left(\phi_{j,n}\right)_{j=1,\dots,J,\, n=1,\dots,N_k}$$
with supp $(\phi_{j,n}) = \overline{K}_j$. A scalar field $u \in \mathbb{P}_k(\mathfrak{K}_h)$ can hence be written as
$$u(\vec{x}) = \sum_{j,n} \phi_{j,n}(\vec{x}) \tilde{u}_{j,n} = \underline{\phi}(\vec{x}) \cdot \tilde{u},$$
where the *DG coefficients* $\tilde{u}$ are arranged as a column vector of length $J \cdot N_k$. Consequently, the restriction of u to cell j is denoted as
$$u|_{K_j} = \sum_n \phi_{j,n} \tilde{u}_{j,n} = \underline{\phi}_{j,-} \cdot \tilde{u}_{j,-}.$$
- The local mass matrix in cell K_j,
$$\mathcal{M}_j = \int_\Omega \underline{\phi}_{j,-}^T \cdot \underline{\phi}_{j,-} \, \mathrm{dV}.$$

Now, consider a computational domain that is intersected by the interface $\mathfrak{I}(t)$, which moves with normal velocity
$$s = \begin{cases} \frac{\partial_t \varphi}{|\nabla \varphi|} & \text{at } \mathfrak{I}(t) \\ 0 & \text{elsewhere.} \end{cases} \tag{5}$$

We assume that $\mathfrak{I}(t)$ is prescribed by means of a known level-set function $\varphi(t, \vec{x}) \in C^\infty(\mathbb{R}_{>0} \times \Omega)$. That is, φ defines a disjoint partitioning
$$\Omega = \mathfrak{A}(t) \cup \mathfrak{I}(t) \cup \mathfrak{B}(t), \tag{6}$$
where
$$\begin{aligned} \mathfrak{I}(t) &= \{\vec{x} \in \Omega;\, \varphi(t, \vec{x}) = 0\}, \\ \mathfrak{A}(t) &= \{\vec{x} \in \Omega;\, \varphi(t, \vec{x}) < 0\}, \text{ and} \\ \mathfrak{B}(t) &= \{\vec{x} \in \Omega;\, \varphi(t, \vec{x}) > 0\}. \end{aligned}$$

Within this setting, we introduce the extended nomenclature for the XDG discretization proposed in Kummer[1]:

- A cell is *cut by the level set* $\mathfrak{I}(t)$ at time t, if
$$\oint_{\mathfrak{I}(t) \cap K_j} 1 \, \mathrm{dS} > 0.$$

Such a cell may also be called a *cut background cell*. In contrast, we define *cut cells* at the intersections of K_j with $\mathfrak{A}(t)$ and $\mathfrak{B}(t)$, ie,

$$K_{j,\mathfrak{s}}(t) = K_j \cap \mathfrak{s}(t), \text{ for } \mathfrak{s} \in \{\mathfrak{A}, \mathfrak{B}\}.$$

This implies we define *two* cut cells $K_{j,\mathfrak{A}}$ and $K_{j,\mathfrak{B}}$ for each cut background cell K_j.

- Using these newly defined cut cells, we define a cut cell grid $\mathfrak{K}_h^X(t)$. For ease of notation, we formally allow cut cells of measure zero, so we can notate this grid as

$$\mathfrak{K}_h^X(t) = (K_{1,\mathfrak{A}}(t), K_{1,\mathfrak{B}}(t), \dots, K_{J,\mathfrak{A}}(t), K_{J,\mathfrak{B}}(t)).$$

- The basis $\mathbb{P}_k(\mathfrak{K}_h^X(t))$ of the XDG space analogously to the DG basis. That is,

$$\underline{\phi}^X(t) = (\phi_{j,n}\chi_{\mathfrak{s}(t)})_{j=1,\dots,J,\, n=1,\dots,N_k,\, \mathfrak{s}\in\{\mathfrak{A},\mathfrak{B}\}},$$

where χ_S denotes the characteristic function associated with the set S. Again, we allow zero entries in the row vector $\underline{\phi}^X$ to maintain a vector whose length is independent of the actual position of the interface $\mathfrak{I}(t)$.

- We denote $u \in \mathbb{P}_k(\mathfrak{K}_h^X(t))$

$$u(t,\vec{x}) = \sum_{j,n,\mathfrak{s}} \phi_{j,n,\mathfrak{s}}(t,\vec{x})\tilde{u}_{j,n,\mathfrak{s}}(t) = \underline{\phi}^X(t,\vec{x}) \cdot \tilde{u}(t).$$

To ensure uniqueness of the DG coefficients, we set $\tilde{u}_{j,n,\mathfrak{s}} = 0$ if $\phi_{j,n,\mathfrak{s}}(t,-) \equiv 0$.

- The local mass matrix in cut cell $K_{j,\mathfrak{s}}(t)$

$$\mathcal{M}_{j,\mathfrak{s}}(t) = \int_\Omega \underline{\phi}^{X,T}_{j,-,\mathfrak{s}}(t) \cdot \underline{\phi}^X_{j,-,\mathfrak{s}}(t)\,\mathrm{dV}.$$

- The abbreviations $\mathcal{M}^n_{j,\mathfrak{s}} = \mathcal{M}_{j,\mathfrak{s}}(t^n)$ and $\tilde{u}^n_{j,\mathfrak{s}} = \tilde{u}_{j,-,\mathfrak{s}}(t^n)$ to denote the mass matrix and DG coefficients at a discrete time level t^n.

An important ingredient for the implementation of methods within this extended setting is the numerical integration in cut cells. Optimally, the quadrature error should converge with order $2p$ for linear problems if a DG polynomial order of p is used. Here, we use a technique called hierarchical moment fitting,[1,19] where quadrature rules for cut cells are constructed on the fly by solving linear, under-determined systems to obtain quadrature weights, which fulfil a set of integral identities.

2 | TIME DISCRETIZATION APPROACHES

Within this section, we consider the temporal integration of the generic conservation law (1) from time t^0 to t^1. We assume that there is no topology change between the cut cell grids $\mathfrak{K}_h^X(t^0)$ and $\mathfrak{K}_h^X(t^1)$ in the following sense: If some cell $K_{j,\mathfrak{s}}(t^0)$ has zero/non-zero measure, then $K_{j,\mathfrak{s}}(t)$ has zero/non-zero measure for all $t \in [t^0, t^1]$. Under such assumptions, it is sufficient to investigate a single cell $K_{j,\mathfrak{s}}(t)$. The handling of topology changes will be discussed in Section 4.

2.1 | The splitting approach

The basic idea of the splitting approach is the following: First, move the interface and extrapolate the initial value at t^0 to the new cut cell grid at t^1. Then, the temporal integration is carried out as if the interface would be fixed.

Since we exclude topology changes for now, the extrapolation operation for u^0 onto new cut cells at t^1 can be easily expressed in terms of the DG coefficients $\tilde{u}^0$:

$$\begin{aligned} \mathrm{Ex}_{(t^0,t^1)} : \mathbb{P}_k\left(\mathfrak{K}_h^X(t^0)\right) &\to \mathbb{P}_k\left(\mathfrak{K}_h^X(t^1)\right) \\ \underline{\phi}^X(t^0)\tilde{u}^0 &\mapsto \underline{\phi}^X(t^1)\tilde{u}^0. \end{aligned} \tag{7}$$

In simple words, the coefficients are kept constant while the basis is replaced.

Afterwards, we apply the standard DG Ansatz to Equation 1 in the cut cell $K^1_{j,\mathfrak{s}}$: The conservation law (1) is multiplied by test functions $\underline{\phi}^{X,T}_{j,-,\mathfrak{s}}(t^1)$ (ie, the column vector of XDG-basis functions in cut cell $K^1_{j,\mathfrak{s}}$ at time t^1), integrated in space over the cut cell, and integrated in time over the interval (t^0, t^1). Finally, one performs integration by parts and replaces the

flux normal to cell boundary (ie, $\vec{f}\cdot\vec{n}_{\partial K^1_{j,\mathfrak{s}}}$) by a numerical flux $\hat{F}_{j,\mathfrak{s}}$ to obtain

$$\mathcal{M}^1_{j,\mathfrak{s}}\left(\tilde{u}^1_{j,\mathfrak{s}}-\tilde{u}^0_{j,\mathfrak{s}}\right)+\int_{t^0}^{t^1}\underbrace{\left(\oint_{\partial K^1_{j,\mathfrak{s}}}\underline{\phi}^{X,T}_{j,-,\mathfrak{s}}(t^1)\hat{F}_{j,\mathfrak{s}}\,\mathrm{dS}-\int_{K^1_{j,\mathfrak{s}}}\nabla\underline{\phi}^{X,T}_{j,-,\mathfrak{s}}(t^1)\cdot\vec{f}\,\mathrm{dV}\right)}_{=:\mathcal{F}_{\mathrm{split},j,-,\mathfrak{s}}}\mathrm{d}t=0. \tag{8}$$

Remark on the notation of numerical fluxes: We have $\hat{F}_{j,\mathfrak{s}}\approx\vec{f}\cdot\vec{n}_{\partial K^1_{j,\mathfrak{s}}}$ at the boundary of cell $K_{j,\mathfrak{s}}(t)$. That is, $\hat{F}_{j,\mathfrak{s}}$ is a function that depends on the normal vector $\vec{n}_{\partial K^1_{j,\mathfrak{s}}}$ and the inner and outer limits

$$u^-(\vec{x})=\lim_{\varepsilon\searrow 0}\left(u^+(\vec{x}-\varepsilon\vec{n})\right)\text{ and }u^+(\vec{x})=\lim_{\varepsilon\searrow 0}\left(u^+(\vec{x}+\varepsilon\vec{n})\right).$$

To ensure conservation for the adjacent cells $K_{j,\mathfrak{s}}(t)$ and $K_{i,\mathfrak{r}}(t)$ (for $\mathfrak{s},\mathfrak{r}\in\{\mathfrak{A},\mathfrak{B}\}$, 2 cells are adjacent if $\oint_E 1\,\mathrm{dS}>0$, where $E=\overline{K_{j,\mathfrak{s}}(t)}\cap\overline{K_{j,\mathfrak{r}}(t)}$), the numerical fluxes for those cells must fulfil $\hat{F}_{j,\mathfrak{s}}=-\hat{F}_{i,\mathfrak{r}}$.

2.2 | The moving interface approach

To consider the time-dependent domain, it is useful to introduce a space-time notation in which the framework developed so far can be embedded.

Space-time notation: We introduce

- space-time elements (cf Figure 1):

$$K^*_{j,\mathfrak{s}}=\{(t,\vec{x})|\,t^0<t<t^1,\vec{x}\in K_{j,\mathfrak{s}}(t)\} \tag{9}$$

- the boundary $\partial K^*_{j,\mathfrak{s}}$ of the space-time element $K^*_{j,\mathfrak{s}}$ with disjoint decomposition

$$\partial K^*_{j,\mathfrak{s}}=\left(\{t^0\}\times K^0_{j,\mathfrak{s}}\right)\cup\left(\{t^1\}\times K^1_{j,\mathfrak{s}}\right)\cup\left(\{(t,\vec{x})|\,t^0\le t\le t^1,\vec{x}\in\partial K_{j,\mathfrak{s}}(t)\}\right), \tag{10}$$

  where $K^0_{j,\mathfrak{s}}=K_{j,\mathfrak{s}}(t^0)$ and $K^1_{j,\mathfrak{s}}=K_{j,\mathfrak{s}}(t^1)$
- the space-time divergence:

$$\mathrm{div}^*(u,\vec{f})=\partial_t u+\mathrm{div}\left(\vec{f}\right). \tag{11}$$

  Thus, one can rewrite Equation 1 in time-space form:

$$\mathrm{div}^*\left((u,\vec{f})\right)=0. \tag{12}$$

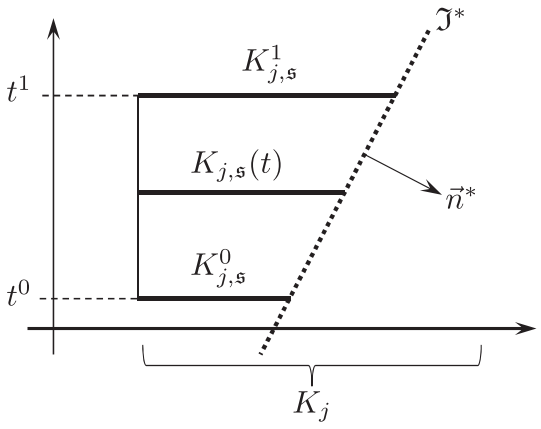

FIGURE 1 Sketch of a space-time cut cell $K^*_{j,\mathfrak{s}}$ for a constant interface velocity s

- space-time outer normals on $\partial K^*_{j,\mathfrak{s}}$ (cf Figure 1):

$$\vec{n}^* = \begin{cases} \frac{1}{\sqrt{1+s^2}}(-s,\vec{n})^T & \text{on } \mathfrak{s}^* \cap \partial K^*_{j,\mathfrak{s}} \\ (-1,0)^T & \text{on } \{t^0\} \times K^0_{j,\mathfrak{s}} \\ (1,0)^T & \text{on} \{t^1\} \times K^1_{j,\mathfrak{s}} \\ (0,\vec{n})^T & \text{elsewhere.} \end{cases}$$

Integral identities: To reduce the space-time notation to a form that can be used with standard time integration schemes such as BDF or Runge-Kutta, the following 2 integral identities for some integrand $q(t,\vec{x})$ are required:

- space-time *volume integral*:

$$\int_{K^*_{j,\mathfrak{s}}} q \,\mathrm{dV}^* = \int_{t^0}^{t^1} \int_{K_{j,\mathfrak{s}}(t)} q \,\mathrm{dV}\,\mathrm{d}t \tag{13}$$

- space-time *boundary integral*:

$$\oint_{\partial K^*_{j,\mathfrak{s}}} q \,\mathrm{dS}^* = \int_{t^0}^{t^1} \oint_{\partial K_{j,\mathfrak{s}}(t)} q\sqrt{1+s^2}\,\mathrm{dS}\,\mathrm{d}t + \int_{K^1_{j,\mathfrak{s}}} q|_{t=t^1}\,\mathrm{dV} - \int_{K^0_{j,\mathfrak{s}}} q|_{t=t^0}\,\mathrm{dV}. \tag{14}$$

DG Ansatz in space-time cell: We formulate a DG Ansatz in the space-time cut cell $K^*_{j,\mathfrak{s}}$ by multiplying with test functions $\underline{\phi}^{X,T}_{j,-,\mathfrak{s}}$ (column vector), integrating over $K^*_{j,\mathfrak{s}}$, performing partial integration, and introducing a numerical flux $\hat{F}^*_{j,\mathfrak{s}}$. Using identities (13) and (14), this yields

$$\begin{aligned} 0 &= \int_{K^*_{j,\mathfrak{s}}} \underline{\phi}^{X,T}_{j,-,\mathfrak{s}} \,\mathrm{div}^*\left((u,\vec{f})^T\right) \mathrm{dV}^* \\ &= \oint_{\partial K^*_{j,\mathfrak{s}}} \underline{\phi}^{X,T}_{j,-,\mathfrak{s}} (u,\vec{f})^T \cdot \vec{n}^* \,\mathrm{dS}^* - \int_{K^*_{j,\mathfrak{s}}} \nabla^* \underline{\phi}^{X,T}_{j,-,\mathfrak{s}} \cdot (u,\vec{f})^T \,\mathrm{dV}^* \\ &= \int_{t^0}^{t^1} \int_{\partial K_{j,\mathfrak{s}}(t)} \underline{\phi}^{X,T}_{j,-,\mathfrak{s}} \underbrace{(-su + \vec{f}\cdot\vec{n}_{K_{j,\mathfrak{s}}})}_{\approx \hat{F}^*_{j,\mathfrak{s}}} \mathrm{dS}\,\mathrm{d}t + \underbrace{\int_{K^1_{j,\mathfrak{s}}} \left(\underline{\phi}^{X,T}_{j,-,\mathfrak{s}} u\right)\Big|_{t=t^1} \mathrm{dV}}_{=\mathcal{M}^1_{j,\mathfrak{s}}\tilde{u}^1_{j,\mathfrak{s}}} \\ &\quad \underbrace{- \int_{K^0_{j,\mathfrak{s}}} \left(\underline{\phi}^{X,T}_{j,-,\mathfrak{s}} u\right)\Big|_{t=t^0} \mathrm{dV}}_{=\mathcal{M}^0_{j,\mathfrak{s}}\tilde{u}^0_{j,\mathfrak{s}}} - \int_{t^0}^{t^1} \int_{K_{j,\mathfrak{s}}(t)} \nabla \underline{\phi}^{X,T}_{j,-,\mathfrak{s}} \cdot \vec{f}\,\mathrm{dV}\,\mathrm{d}t. \end{aligned} \tag{15}$$

Note that $\partial_t \underline{\phi}^{X,T} = 0$ *within* $K^*_{j,\mathfrak{s}}$ (but not on the boundary of $K^*_{j,\mathfrak{s}}$), so the space-time gradient equals $\nabla^* \underline{\phi}^{X,T} = (\partial_t \underline{\phi}^{X,T}, \nabla \underline{\phi}^{X,T}) = (0, \nabla \underline{\phi}^{X,T})$. Note furthermore that, since we defined $s = 0$ away from the interface $\mathfrak{I}$ (cf Equation 5), we have $\hat{F}^*_{j,\mathfrak{s}} = \hat{F}_{j,\mathfrak{s}} \approx \vec{f}\cdot\vec{n}_{K_{j,\mathfrak{s}}}$ on the static part $\partial K_{j,\mathfrak{s}}(t) \backslash \mathfrak{I}(t)$ of the cut cell-boundary. In summary, we thus obtain

$$\mathcal{M}^1_{j,\mathfrak{s}}\tilde{u}^1_{j,\mathfrak{s}} - \mathcal{M}^0_{j,\mathfrak{s}}\tilde{u}^0_{j,\mathfrak{s}} + \int_{t^0}^{t^1} \underbrace{\left(\oint_{\partial K_j(t)} \underline{\phi}^{X,T}_{j,-,\mathfrak{s}} \hat{F}^*_{j,\mathfrak{s}} \,\mathrm{dS} - \int_{K_j(t)} \nabla \underline{\phi}^{X,T}_{j,-,\mathfrak{s}} \cdot \vec{f}\,\mathrm{dV} \right)}_{=: \mathcal{F}_{\mathrm{ts},j,-,\mathfrak{s}}} \mathrm{d}t = 0. \tag{16}$$

To complete the definition of the moving interface approach, we still have to choose a numerical flux function $\hat{F}^*_{j,\mathfrak{s}}$. A natural requirement on this flux is free-stream preservation, namely, that the solution should remain unaltered when moving the interface through an initially constant solution. It directly follows from Equation 16 that $\hat{F}^*_{j,\mathfrak{s}}$ thus has to depend on the interface speed s. We will explore 2 relevant choices in Section 3.

2.2.1 | Note on temporal accuracy

Typically, one expects from a Galerkin method to exactly reproduce a solution if it is a member of the respective approximation space. A prerequisite for this behaviour is, however, that the integrals of the Galerkin Ansatz are computed exactly.

To illustrate this, we take a closer look at the time-volume integral in the scheme (13), ie, the term

$$\int_{t^0}^{t^1} \int_{K_j(t)} \nabla \underline{\phi}_{j,-,\mathfrak{s}}^{X,T} \cdot \vec{f} \, \mathrm{dV} \, \mathrm{d}t. \tag{17}$$

Note that similar considerations hold for the surface integral. For the sake of simplicity, we assume a 1D situation, a cut cell $K_j(t) = (x_{\mathrm{lo}}, x_{\mathrm{hi}} + st)$ and a flux $\vec{f}$, which linearly depends on u.

Then, for a DG polynomial degree p, the integrand $\nabla \underline{\phi}_{j,-,\mathfrak{s}}^{X,T} \cdot \vec{f}(u)$ is of maximum order $2p-1$ because $\nabla \underline{\phi}_{j,-,\mathfrak{s}}^{X,T}$ is of order $\mathcal{O}(p-1)$ and $\vec{f}(u)$ is of order $\mathcal{O}(p)$. The highest order terms in (17) thus behave as

$$\mathrm{const} \cdot \left(\int_{t^0}^{t^1} (x_{\mathrm{hi}} + st)^{2p} \, \mathrm{d}t - (t^1 - t^0) x_{\mathrm{lo}}^{2p} \right).$$

Observing that the integrand of the temporal integral behaves as $\mathcal{O}(t^{2p})$, it becomes obvious that the time integration scheme must also be able to integrate polynomials like t^{2p} exactly to be able to fulfil the above-mentioned property. Obviously, if the cell volume is more complex than a first-order polynomial of t, the integrand is of even higher order.

3 | FLUXES ACROSS THE MOVING INTERFACE

For the static part of the cut cell surface, a standard Riemann flux (cf Section 2.1) can obviously be used. Accordingly, we set

$$\hat{F}_{j,\mathfrak{s}}^{*} = \hat{F}_{j,\mathfrak{s}} \text{ on } \partial K_{j,\mathfrak{s}}(t) \backslash \mathfrak{F}(t).$$

As discussed in Section 2.2, we require

$$\hat{F}_{j,\mathfrak{s}}^{*} \approx -su + \vec{f} \cdot \vec{n}_{K_{j,\mathfrak{s}}}$$

on the *moving part* of $\partial K_{j,\mathfrak{s}}(t)$ to regain a free-stream preserving scheme. In the following, we present 2 different options to achieve this if the order of accuracy of the time integration scheme is sufficiently high.

The first option is to use a static flux and add the correction term $-su$. However, we still have to specify how this term should be evaluated at boundaries between because $\hat{F}^{*}$ must be single-valued to ensure conservativity. Here, we have selected the downwind value u^{dwnd} of u for causality reasons (future should depend on the past) such that we obtain

$$\hat{F}_{j,\mathfrak{s}}^{*} = -su^{\mathrm{dwnd}} + \hat{F}_{j,\mathfrak{s}}. \tag{18}$$

The second option is to actually consider a flux $\hat{F}_{j,\mathfrak{s}}^{*}$ in a moving frame. This has to be done specifically for the equation of interest. Consider, for example, scalar convection with the flux $\vec{f} = \vec{c}u$. Since the interface moves with velocity s in normal direction, the relative convection velocity at the interface is $(-s\vec{n} + \vec{c})$. Now, one can use the upwind formulation

$$\hat{F}_{j,\mathfrak{s}}^{*} = \begin{cases} (-s\vec{n} + \vec{c}) \cdot \vec{n} u^{-} & \text{if } -s + \vec{c} \cdot \vec{n} \geq 0 \\ (-s\vec{n} + \vec{c}) \cdot \vec{n} u^{+} & \text{if } -s + \vec{c} \cdot \vec{n} < 0. \end{cases} \tag{19}$$

The geometric situation is illustrated in Figure 2.

Example: Scalar convection, corrected vs moving-frame upwind flux: To illustrate the difference between the 2 choices for the flux at the interface, we consider one-dimensional ($D = 1$), scalar, linear convection, with a constant advection velocity and a piecewise constant initial value. That is,

$$\partial_t u + \partial_x (cu) = 0$$

with $c = \mathrm{const.} > 0$ and the initial value, resp. the (weak) exact solution,

$$u_{\mathrm{ex}}(t,x) = \begin{cases} u_{\mathrm{left}} & \text{for } x < ct \\ u_{\mathrm{right}} & \text{for } x > ct \end{cases}.$$

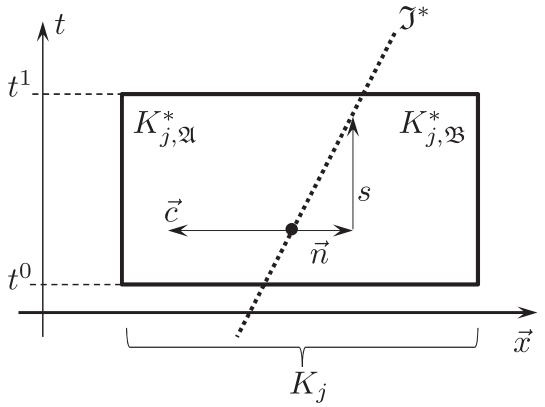

FIGURE 2 The selection of the upwind value at the moving interface: The relative flow velocity at the point • is $-s\vec{n} + \vec{c}$; thus, the discriminator for the upwind selection is $-s + \vec{c} \cdot \vec{n}$. Alternatively, one might also interpret this as the inner product of the space-time velocity with the normal vector, ie, the product $(1, \vec{c}) \cdot (-s, \vec{n})$

Furthermore, let the interface move with the same velocity as the advection speed, ie, $s = c$. In the following, we will illustrate that the moving-frame upwind flux (Equation 19) reproduces the *exact* solution for this case, whereas the upwind flux wit correction term (Equation 18) induces numerical diffusion.

We chose a DG polynomial degree of $p = 0$, consider a singe cell $K_j = (x_{\text{lo}}, x_{\text{hi}})$ ($x_{\text{lo}} < 0, x_{\text{hi}} > 0$), and set the basis as $\phi_{j,1} = 1$ for the sake of simplicity. In the 1D setting, the interface $\mathfrak{F} = \{st\}$ is a single point, and we assume that it is located in K_j for $t \in [t_0, t_1]$. Then, the 1×1 mass matrices for the cut cells $K_{j,\mathfrak{A}}(t) = (x_{\text{lo}}, st)$ and $K_{j,\mathfrak{A}}(t) = (st, x_{\text{hi}})$ are given by

$$\mathcal{M}_{j,\mathfrak{A}} = st - x_{\text{lo}} \text{ and}$$
$$\mathcal{M}_{j,\mathfrak{B}} = x_{\text{hi}} - st,$$

and the initial value translates into the DG coefficients $\tilde{u}^0_{j,1,\mathfrak{A}} = u_{\text{left}}$ and $\tilde{u}^0_{j,1,\mathfrak{B}} = u_{\text{right}}$. For the sake of simplicity, consider an explicit Euler scheme and standard upwind fluxes at the static cell boundary $\{x_{\text{lo}}, x_{\text{hi}}\}$. This leads to the following scheme:

$$\tilde{u}^1_{j,1,\mathfrak{A}} = \frac{st^0 - x_{\text{lo}}}{st^1 - x_{\text{lo}}} u_{\text{left}} - \frac{(t^1 - t^0)}{st^1 - x_{\text{lo}}} \left(-u_{\text{left}} c + \hat{F}^*_{j,\mathfrak{A}}\Big|_{x=st} \right) \tag{20}$$

$$\tilde{u}^1_{j,1,\mathfrak{B}} = \frac{x_{\text{hi}} - st^0}{x_{\text{hi}} - st^1} u_{\text{right}} - \frac{(t^1 - t^0)}{x_{\text{hi}} - st^1} \left(-\hat{F}^*_{j,\mathfrak{B}}\Big|_{x=st} + u_{\text{right}} c \right). \tag{21}$$

We recall that conservativity implies

$$\hat{F}^*_{j,\mathfrak{A}}\Big|_{x=st} = -\hat{F}^*_{j,\mathfrak{B}}\Big|_{x=st}$$

at the moving interface $\{st\}$, which is why the *corrected* upwind flux reduces to

$$\hat{F}^*_{j,\mathfrak{A}}\Big|_{x=st} = -s \begin{cases} u_{\text{right}} & \text{if } c \geq 0 \\ u_{\text{left}} & \text{if } c < 0 \end{cases} + \begin{cases} c u_{\text{left}} & \text{if } c \geq 0 \\ c u_{\text{right}} & \text{if } c < 0 \end{cases}$$

and further to

$$\hat{F}^*_{j,\mathfrak{A}}\Big|_{x=st} = -\hat{F}^*_{j,\mathfrak{B}}\Big|_{x=st} = -s u_{\text{right}} + c u_{\text{left}}.$$

Obviously, this scheme is unable to reproduce the exact solution for $u_{\text{right}} \neq u_{\text{left}}$. Even though it is certainly possible to tune the upwind/downwind discriminators (eg, by using $c \geq 0$ in the flux and $c > 0$ in the correction term) to fix the problem for this particular case, it is easy to verify that this is not a general solution. The same finding holds true for more involved discriminators such as $(-st + c) \geq 0$.

The *moving-frame* upwind flux (see (19)) is given by

$$\hat{F}^*_{j,\mathfrak{A}}\Big|_{x=st} = \begin{cases} (-s + c) u_{\text{left}} & \text{if } (-s + c) \geq 0 \\ (-s + c) u_{\text{right}} & \text{if } (-s + c) < 0, \end{cases}$$

and since $s = c$, we obtain

$$\hat{F}^*_{j,\mathfrak{A}}\Big|_{x=st} = \hat{F}^*_{j,\mathfrak{B}}\Big|_{x=st} = 0.$$

Therefore, Equations 20 and 21 reduce to

$$\tilde{u}^1_{j,1,\mathfrak{A}} = u_{\text{left}} \quad \text{and} \quad \tilde{u}^1_{j,1,\mathfrak{B}} = u_{\text{right}},$$

which is obviously the *exact* solution.

Example: Burgers equation, corrected vs moving-frame upwind flux: As another example, we consider the Burgers equation

$$\partial_t u + \partial_x \left(\frac{1}{2} u^2 \right) = 0,$$

with the initial value, resp. the exact (weak) solution of the Riemann problem,

$$u_{\text{ex}}(t,x) = \begin{cases} u_{\text{left}} & \text{for } x < \sigma t \\ u_{\text{right}} & \text{for } x > \sigma t \end{cases}.$$

Assuming $u_{\text{left}} > u_{\text{right}} > 0$ leads to a discontinuous solution and the shock speed following from the Rankine-Hugoniot conditions is

$$\sigma = \frac{1}{2} \left(u_{\text{left}} + u_{\text{right}} \right). \tag{22}$$

Since we assume positivity of u,while the upwind value is u_{left}, the downwind value is u_{right}. Again, the interface is moved with the shock speed, ie,

$$s = \sigma, \tag{23}$$

and by taking the same balances as in Equations 20 and 21, we observe that to obtain the exact solution, the numerical flux at the moving interface has to be

$$\hat{F}^*_{j,\mathfrak{A}}\Big|_{x=st} = \hat{F}^*_{j,\mathfrak{B}}\Big|_{x=st} = \frac{1}{2} u_{\text{left}} u_{\text{right}}. \tag{24}$$

Proceeding in the same fashion as in the last paragraph, it becomes apparent that a corrected upwind flux is not able to reproduce the exact solution. To overcome this dilemma, we propose to use the flux

$$\hat{F}^*_{j,\mathfrak{A}}\Big|_{x=st} = \left(\frac{1}{2} u^{\text{upwnd}} - s \right) u^{\text{upwnd}}, \tag{25}$$

where the selection of the upwind value u^{upwnd} is based on the relative speed between the shock σ and the interface speed s, ie,

$$u^{\text{upwnd}} = \begin{cases} u_{\text{left}} & \text{if } (\sigma - s) \geq 0 \\ u_{\text{right}} & \text{if } (\sigma - s) < 0 \end{cases}. \tag{26}$$

Finally, we note the selection of the upwind value *seems* numerically unstable in the limit case of $\sigma = s$, because it then depends on the round-off error whether u_{left} or u_{left} is selected for the upwind value. However, inserting relation (22) into the numerical flux (25) shows that the numerical flux reproduces the exact solution (24) for *both* possible choices for the downwind value.

4 | THE APPEARANCE AND DISAPPEARANCE OF CUT CELLS

So far, we have excluded topology changes in the sense that cut background cells remain cut within the complete time interval $[t_0, t_1]$. We will rectify this restriction in the following by introducing a cell-agglomeration strategy that handles *appearing* and *disappearing* cut cells.

Before we are able to consider these topology changes, the cell-local schemes (8) and (16) have to be put in a global form. By combining the equations for each cell in a large column vector and b taking the limit $t^1 \searrow t^0$, we formally get rid of the specific time integration scheme.

We obtain

$$\mathcal{M}(t)\, \partial_t(\tilde{u}(t)) + \mathcal{F}_{\text{split}}(\tilde{u}(t)) = 0 \tag{27}$$

for the splitting scheme (8), whereas the the space-time scheme Equation 16 yields

$$\partial_t(\mathcal{M}(t)\, \tilde{u}(t)) + \mathcal{F}_{\text{ts}}(\tilde{u}(t)) = 0. \tag{28}$$

It is straightforward to apply standard time integration schemes (ordinary differential equation solvers) like Runge-Kutta or BDF to these forms. For the sake of simplicity, we restrict the discussion in this section to explicit Euler, which implies

$$\frac{1}{\Delta t}\mathcal{M}^1(\tilde{u}^1 - \tilde{u}^0) + \mathcal{F}_{\text{split}}(\tilde{u}^0) = 0 \tag{29}$$

resp.

$$\frac{1}{\Delta t}(\mathcal{M}^1\tilde{u}^1 - \mathcal{M}^0\tilde{u}^0) + \mathcal{F}_{\text{ts}}(\tilde{u}^0) = 0. \tag{30}$$

Besides the obvious problem of a prohibitive Courant-Friedrichs-Lewy restriction for small cut cells, there is also a conceptual problem when the topology changes. That is, if a cell either appears ($|K^0_{j,\mathfrak{s}}| = 0$ and $|K^0_{j,\mathfrak{s}}| > 0$), the mass matrix contain singular blocks. Note that in the moving interface case, the same issue also occurs for disappearing cells ($|K^0_{j,\mathfrak{s}}| > 0$ and $|K^0_{j,\mathfrak{s}}| = 0$).

Furthermore, one has to consider that cut cells can become arbitrarily small, depending on the position of the interface $\mathfrak{I}$. Since the mass matrix for such cells may become very ill-conditioned, especially for higher polynomial orders, one also requires a special treatment for those cells. In previous works,[1,20] we proposed a cell-agglomeration technique to overcome those issues. The basic idea is to join a cut cell with its largest neighbour cell if its volume fraction is below a certain threshold. This factor is usually a number between 0.1 and 0.2. An extensive study has been performed in Kummer.[1]

Cell agglomeration: Without loss of generality (W.l.o.g.), we discuss the cell agglomeration resp. construction of the agglomerated DG basis on the grid

$$\mathfrak{K}_h = (K_1, \dots, K_j, K_i, \dots, K_J).$$

We assume that the cells K_j and K_i are adjacent; ie, they share a common edge $E = \overline{K_j} \cap \overline{K_i}$ with $\oint_E 1\,\mathrm{dS} > 0$. We intend to agglomerate K_j and K_i; ie, we create an agglomerated cell $K_j^{\text{agg}} = K_j \cup K_i$ and replace the 2 original cells with the agglomerated one. As a result, we create a grid

$$\mathfrak{K}_h^{\text{agg}} = (K_1, \dots, K_j \cup K_i, \dots, K_J)$$

with $J - 1$ cells.

Since $\mathbb{P}_k(\mathfrak{K}_h^{\text{agg}})$ is a subspace of $\mathbb{P}_k(\mathfrak{K}_h)$, a DG basis on the agglomerated grid can be written as

$$\underline{\phi}^{\text{agg}} = \underline{\phi}\mathcal{Q}, \tag{31}$$

where $\mathcal{Q}$ is a real-valued matrix of the size $JN_k \times (J-1)N_k$. In other words, the matrix $\mathcal{Q}$ is the injection operator from agglomerated DG space to original DG space, which is given as

$$\begin{aligned} \mathbb{P}_k(\mathfrak{K}_h^{\text{agg}}) &\to \mathbb{P}_k(\mathfrak{K}_h) \\ \underline{\phi}^{\text{agg}}\tilde{u}^{\text{agg}} &\mapsto \underbrace{\underline{\phi}\,\mathcal{Q}\tilde{u}^{\text{agg}}}_{=\tilde{u}}. \end{aligned}$$

4.1 | Applying agglomeration to DG discretizations:

- In general, if one wants to solve a linear problem on $\mathbb{P}_k(\mathfrak{K}_h)$, it can be written in the representation

  $$\mathcal{A}\tilde{u} = \tilde{b},$$

  where $\mathcal{A}$ is a quadratic matrix. Then, the L^2-projection of this system onto $\mathbb{P}_k(\mathfrak{K}_h^{\text{agg}})$ is

  $$\mathcal{Q}^T\mathcal{A}\mathcal{Q}\tilde{u}^{\text{agg}} = \mathcal{Q}^T\tilde{b}. \tag{32}$$

  The solution $\tilde{u}^{\text{agg}}$ on the agglomerated grid can be extrapolated to the original grid by the relation

  $$u = \underline{\phi}\,\mathcal{Q}\tilde{u}^{\text{agg}}. \tag{33}$$

- Especially if $\mathcal{M}$ is the mass matrix of the original basis ϕ, the mass matrix $\mathcal{M}^{\text{agg}}$ of the agglomerated DG basis ϕ^{agg} is given as

  $$\mathcal{M}^{\text{agg}} = \mathcal{Q}^T\mathcal{M}\mathcal{Q}.$$

- Agglomeration in the form that is presented here (with basis functions that are C^∞ in the agglomerated cells) requires that the basis functions in 2 adjacent cells K_j and K_i are members of the same polynomial ring, ie, are polynomials in the same variables. This is usually not the case if the background mesh contains curved elements.

Cell-agglomeration policy in the splitting case: As already noted, in this case, the interface is moved first, before the time integration is performed. We use agglomeration to extend the extrapolation operator (7) to cases with topology changes.

A cut cell $K_{j,\mathfrak{s}}(t)$ is agglomerated to its largest neighbour cell (must be in the same species $\mathfrak{s}$ and must have a volume greater than zero for all times between t^0 and t^1) if its volume is zero at time t^0 and greater than 0 for time t^1. Following this agglomeration policy creates an agglomerated cut cell grid, which is free of topology changes. Note that other strategies for the definition of agglomeration pairs may be beneficial depending on the application scenario. Here, we adopt this strategy because it is general and easy to implement. It was shown in Kummer[1] that this choice leads to well-conditioned system matrices, at least in the context of the incompressible Navier-Stokes equations. Other strategies could be, eg, selecting the downstream neighbour or the one where $K_{j,\mathfrak{s}}(t)$ shares the largest interface with. However, in this paper, we only investigate 2D test cases, and because of the necessary restrictions on the agglomeration partner (same species, positive volume for all times), the number of possible choices is typically one or two. Therefore, the effect of a different strategy would be hard to measure since the results will be very similar. In 3D settings, however, the strategy for the choice of the agglomeration partner should be investigated more deeply.

Then, the extrapolation operator (7) can be defined on the agglomerated grid, ie,

$$\begin{aligned} \mathrm{Ex}_{(t^0,t^1)} \,:\, \mathbb{P}_k(\mathfrak{K}_h^{X,\mathrm{agg}}(t^0)) &\to \mathbb{P}_k(\mathfrak{K}_h^{X,\mathrm{agg}}(t^1)) \\ \underline{\phi}^{X,\mathrm{agg}}(t^0)\ \tilde{u}^0 &\mapsto \underline{\phi}^{X,\mathrm{agg}}(t^1)\ \tilde{u}^0. \end{aligned} \tag{34}$$

After applying this trivial extrapolation, one can use the injection (33) to prolongate $\underline{\phi}^{X,\mathrm{agg}}(t^1)\tilde{u}^0$ into the original grid.

For the actual time integration, we also recommend to use cell agglomeration as in our previous works[1,20] to avoid problems related to condition number and/or Courant-Friedrichs-Lewy restrictions caused by very small cut cells.

Cell-agglomeration policy in the moving interface case: For the moving interface case, we have to deal not only with cut cells, which appear during one time step, but we also have to consider disappearing cells. Therefore, we agglomerate a cut cell $K_{j,\mathfrak{s}}(t)$ to its largest neighbour cell (which must be in the same species $\mathfrak{s}$ and must not be an appearing or disappearing cell and also not one of the zero-measure cells, which we formally introduced, ie, cells for which $|K_{j,\mathfrak{s}}(t)| = 0$ for $t^0 \le t \le t^1$) if one or more of the following conditions hold:

- the cut cell has a very small volume fraction, ie, $\frac{|K^0_{j,\mathfrak{s}}|}{|K_j|} < \alpha^{\mathrm{agg}}$ or $\frac{|K^1_{j,\mathfrak{s}}|}{|K_j|} < \alpha^{\mathrm{agg}}$;
- a cut cell appears: $|K^0_{j,\mathfrak{s}}| = 0$ and $|K^1_{j,\mathfrak{s}}| > 0$;
- a cut cell disappears: $|K^0_{j,\mathfrak{s}}| > 0$ and $|K^1_{j,\mathfrak{s}}| = 0$.

The last 2 conditions illustrated in Figure 3. Again, following this agglomeration policy creates an agglomerated cut cell grid, which is free of topology changes. The initial value u^0 has to be projected onto the agglomerated XDG space by relation (32), where the matrix $\mathcal{A}$ in (32) is set to the identity matrix. Then, the time integration can be performed on the agglomerated XDG space, and the solution at t^1 can be injected back onto the original XDG space by (33).

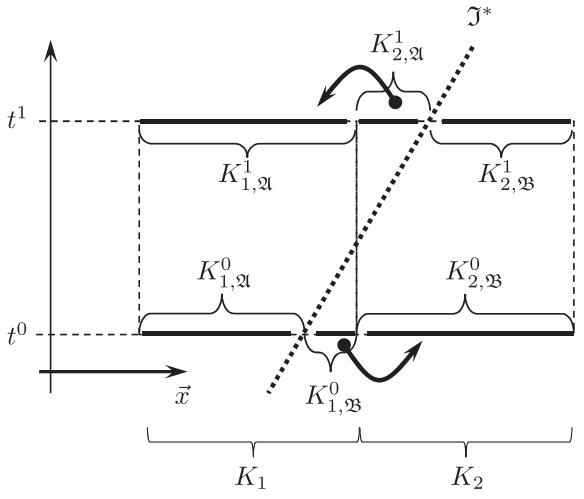

FIGURE 3 Sketch of the agglomeration strategy for appearing ($K^1_{2,\mathfrak{A}}$) and disappearing ($K^0_{1,\mathfrak{B}}$) cut cells in a one-dimensional setting

5 | NUMERICAL TESTS

In the following, we are going to investigate 3 different problems with increasing complexity. They have been chosen to model different aspects of multiphase flow problems. The first problem is given by the scalar advection of discontinuous initial value (cf Section 5.1). This simple setting allows us to evaluate the performance of both time discretization schemes for a simple interface condition and in the absence of non-linear effects. Second, we study the heat equation with a discontinuous diffusion coefficient (cf Section 5.2). Even though this problem is still linear in the bulk, it allows us to investigate the influence of jump conditions that resemble the continuous strain condition for multiphase flows. Finally, we consider the Burgers equation with a shock solution (cf Section 5.3) to study the implications of the presence of non-linear effects in the bulk.

In all numerical experiments, we use equidistant grids with a characteristic mesh size denoted by h. Time is discretized using constant step sizes denoted $\Delta t = t_{\text{end}}/\Theta$, where Θ denotes the total number of time steps. To evaluate the performance of the different approaches, we simulate problem configurations where we know the exact (manufactured) solution $u_{\text{ex}}(t_{\text{end}}, -)$ at a specific end time t_{end}. This allows us to study the behaviour of 2 different error measures for some of the numerical solution u:

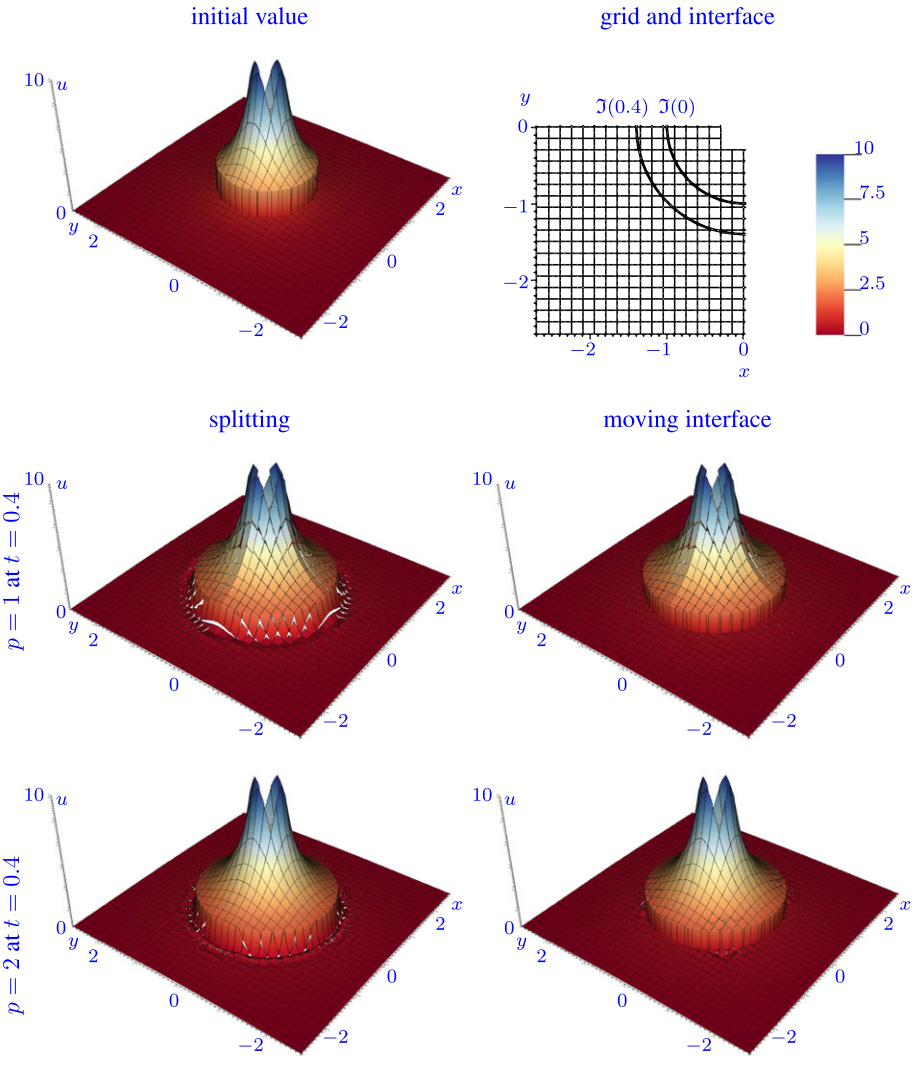

FIGURE 4 Plot of the initial value as well as the solution for $t = 0.4$ for the scalar transport using flux (37) and exact solution (41). One observes 2 issues: First, the jump is completely lost in the splitting calculation; ie, the jump at the interface represented by the extended discontinuous Galerkin discretization is close to zero. What remains is only a steep, but finite gradient. Second, the $p = 2$ calculation with moving interface (bottom right) is near the stability limit, as seen by the oscillations in the outer domain; those, however, would vanish with a finer temporal resolution

$$\text{bulk error:} \qquad \mathrm{ERR}_{\Omega} := \left(\int_{\Omega} (u_{\mathrm{ex}}(t_{\mathrm{end}}, .) - u)^2 \, \mathrm{dV} \right)^{1/2} \tag{35}$$

$$\text{interface error:} \qquad \mathrm{ERR}_{\mathfrak{I}} := \left(\oint_{\mathfrak{I}(t_{\mathrm{end}})} (u_{\mathrm{ex}}(t_{\mathrm{end}}, .) - u)^2 \, \mathrm{dS} \right)^{1/2}. \tag{36}$$

The second error measure is of particular interest for problems where interface conditions have a strong influence on the bulk, for example, multiphase flows with surface tension.

5.1 | Scalar transport

Problem statement: We consider Equation 1 with the flux

$$\vec{f} := u\vec{c} \tag{37}$$

and

$$\vec{c} := \frac{1}{\sqrt{x^2 + y^2}} \begin{pmatrix} x \\ y \end{pmatrix}. \tag{38}$$

The domain $\Omega := (-2.7, 2.7)^2 \backslash (-0.3, 0.3)^2$ is partitioned as

$$\mathfrak{A}(t) = \{\vec{x} \in \Omega \,|\, |\vec{x}| < 1 + t\},\ \mathfrak{I}(t) = \{\vec{x} \in \mathbb{R}^2 \,|\, |\vec{x}| = 1 + t\},\ \mathfrak{B}(t) = \Omega \backslash \mathfrak{A}(t) \backslash \mathfrak{I}(t), \tag{39}$$

which implies that the constant interface speed is given by

$$s = 1. \tag{40}$$

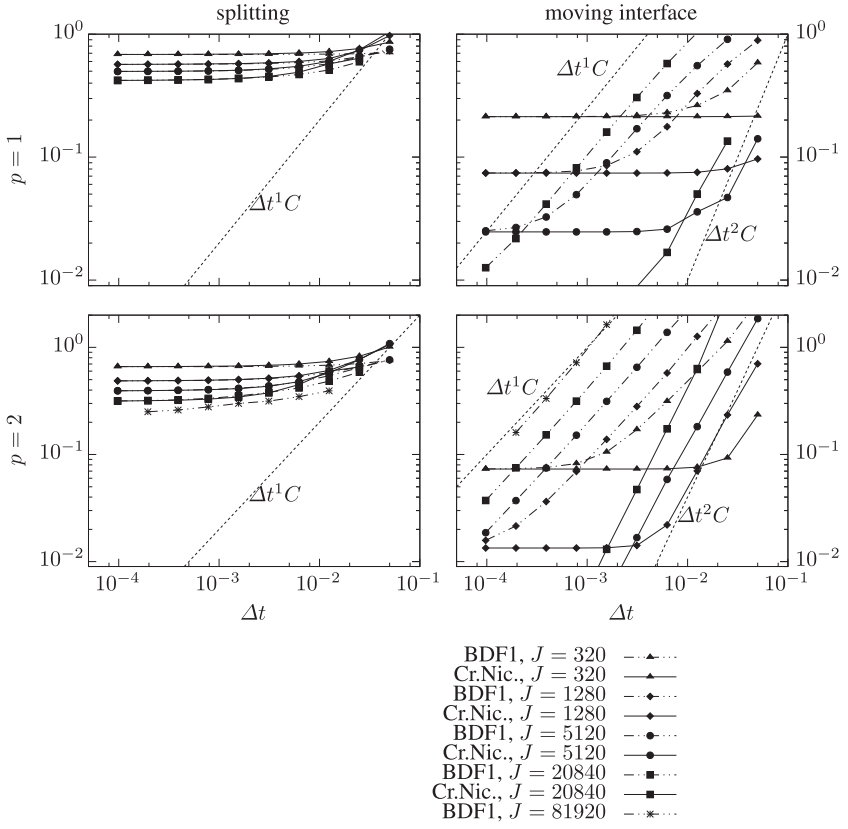

FIGURE 5 Comparison of the temporal convergence of the bulk error for the splitting and the moving interface approach applied to the scalar transport equation. J denotes the number of background cells; p denotes the polynomial degree of the discontinuous Galerkin Ansatz

We choose the initial value, resp. the exact solution, as

$$u_{\text{ex}}(t,x,y) = \begin{cases} \frac{3}{\sqrt{x^2+y^2}} & \sqrt{x^2+y^2} < 1+t \\ \frac{1}{\sqrt{x^2+y^2}} & \sqrt{x^2+y^2} > 1+t \end{cases} \tag{41}$$

with a moving discontinuity at $\mathfrak{F}(t)$. Additionally, $u_{\text{ex}}(t,x,y)$ is used as an inflow boundary condition where applicable.

The coarsest grid consists of 18×18 equidistant, quadrilateral cells. We simulate up to an end time of $t_{\text{end}} = 0.05$ using Crank-Nicolson and BDF schemes for DG polynomial degrees of 1, 2, and 3. As a numerical flux, we use a standard upwind flux at static boundaries and the moving-frame upwind flux (cf Equation 19) at the moving interface.

Results and discussion: A plot of the mesh, initial value, and exemplary numerical solution is shown in Figure 4. In a first step, we compare the performance of splitting with the moving interface approach. A study of the temporal convergence of the bulk error is given in Figure 5. The results reveal that the moving interface approach outperforms splitting by orders of magnitude. Using sufficiently fine spatial resolutions, we recover the expected convergence rates of the BDF1 and the Crank-Nicolson scheme. Splitting, on the other hand, is unable to deliver the expected convergence rates using the given spatial resolutions. Further tests indicate that convergence can indeed only be restored at extreme, impractical spatial resolutions.

Figure 6 shows a spatial convergence study for the same test case. Again, we observe that the moving interface approach delivers superior results in almost all configurations. The only exception occurs for very imprecise temporal integration (long time steps and/or low order time integration). However, the error is so high under these circumstances that the XDG scheme offers no advantage over a standard first-order method on a static mesh.

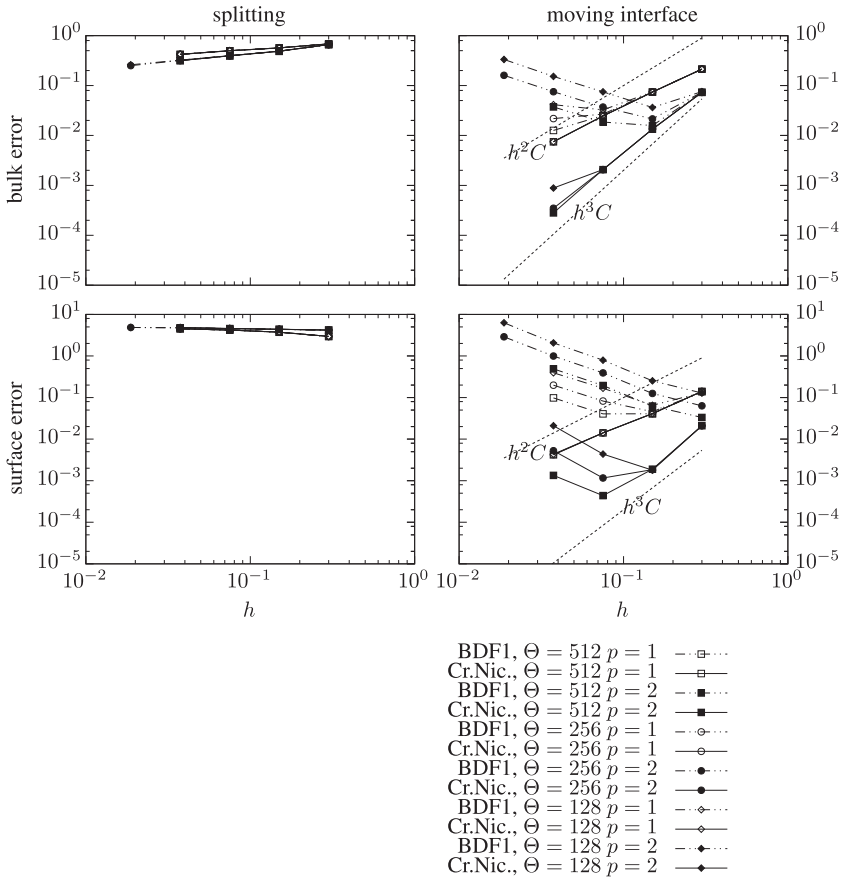

FIGURE 6 Spatial convergence study of the bulk and interface errors for both discretization approaches applied to the scalar transport equation. Θ denotes the number of time steps; p denotes the polynomial degree of the discontinuous Galerkin Ansatz. One observes especially for the low order BDF1 scheme that, if the temporal integration is insufficient, higher spatial resolutions may *increase* the error. Using a higher order temporal discretization, one is able to overcome these issues

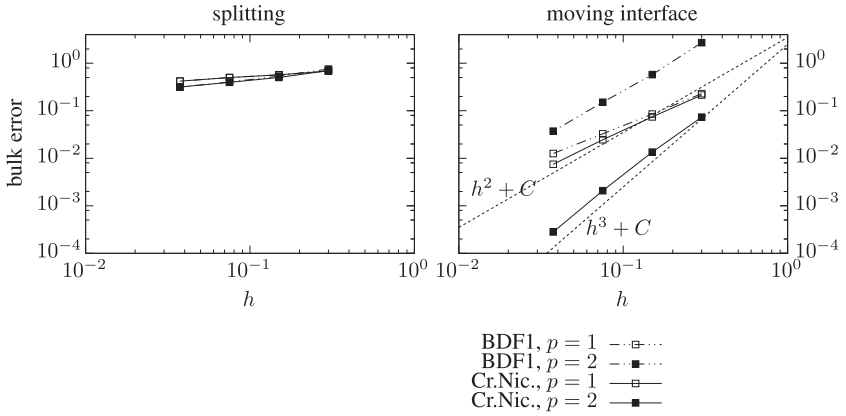

FIGURE 7 Modified spatial convergence study of the bulk errors for both discretization approaches applied to the scalar transport equation. In contrast to Figure 6, we now assume $\Delta t^r \sim h^{p+1}$. Here, p denotes the polynomial degree of the discontinuous Galerkin Ansatz, and r denotes the expected convergence rate of the time integrator (ie, $r = 1$ for BDF1 and $r = 2$ for Crank-Nicolson). In particular, we use 512 time steps on the finest mesh (ie, $\Delta t_0 \approx 10^{-4}$ for mesh size h_0) and adapt the step size on the coarser meshes according to $\Delta t = \Delta t_0 (h/h_0)^{(p+1)/r}$. Once again, we observe that the splitting approach leads to extremely low convergence rates. But also for the moving interface approach, *increasing* the spatial approximation order may *increase* the discretization error when a low order time integrator such as BDF1 is used. For Crank-Nicolson, on the other hand, the expected behaviour can be recovered

A different view on the spatial convergence behaviour for this case is given in Figure 7. Here, we examine the bulk error under h-refinement, while adapting the time step size such that spatial and temporal discretization errors should converge at the same rate. The convergence rates for the splitting approach are below 0.5 in all cases, while the performance of the moving interface approach strongly depends on the selected time integrator. Most prominently, we once again observe *increasing* the spatial resolution *increases* the discretization error using BDF1. The Crank-Nicolson scheme, on the other hand, shows the expected behaviour.

In a second step, we study the results of the moving interface approach using higher order time integrators in detail. Results for a temporal convergence study are given in Figure 8. Here, it becomes evident that higher order temporal convergence can be obtained if the spatial resolution is sufficient. On the other hand, the results of the corresponding spatial convergence study depicted in Figure 9 provide evidence that the moving interface approach is also able to deliver higher order spatial convergence rates, but only if the temporal integration is sufficiently accurate. In particular, we observe that an increase in spatial resolution can *increase* the total error if the temporal resolution is not adjusted properly. This can be seen most prominently in case of the surface error, which is much more sensitive than the bulk error.

In sum, these findings suggest that spatial and temporal accuracy are not independent in the moving interface case, which can be understood as a direct consequence of the approximation of the space-time integrals discussed in Section 2.2.1. Consequently, time integration has to be more accurate if the polynomial degree p is increased and vice versa. For example, this is directly reflected by the data for $p = 3$ using BDF3 in Figure 9: An increase in spatial resolution by a factor of 2 requires an increase of the temporal resolution by a factor of approximately 4 to recover the full convergence rate.

5.2 | Heat equation with discontinuous, time-dependent diffusion coefficient

Problem statement: Within the domain $\Omega := (-1, 1)^2$, we consider Equation 1 with the flux

$$\vec{f} := \mu \nabla u, \text{with } \mu := \begin{cases} \mu_{\mathfrak{A}} & x > \frac{4}{10}(1+t) \\ \mu_{\mathfrak{B}} & x < \frac{4}{10}(1+t) \end{cases}. \tag{42}$$

Note that we do *not* use the Rankine-Hugoniot condition for the heat equation, but the jump conditions

$$[\![u]\!] = 0, \tag{43}$$

$$\left[\!\left[\mu \nabla u \cdot \vec{n}_{\mathfrak{I}}\right]\!\right] = 0. \tag{44}$$

The motivation for these jump conditions is that they are very similar to the strain conditions for incompressible 2-phase flows (see, eg, Kummer[1] and Wang and Oberlack[21]), which are our main application for the XDG method.

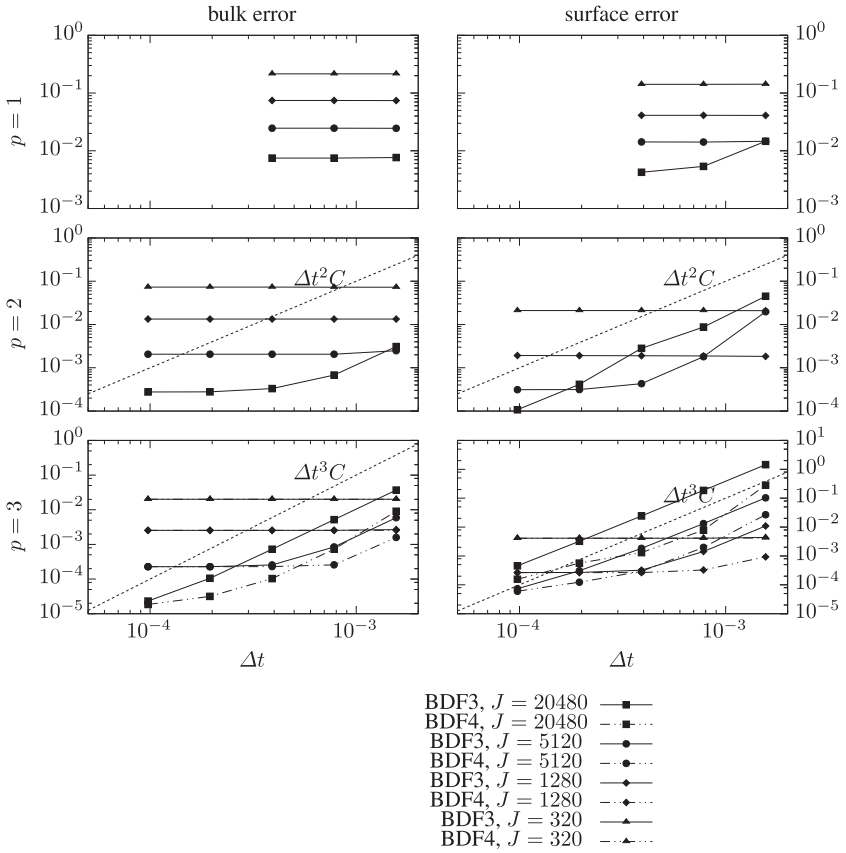

FIGURE 8 Temporal convergence study for the moving interface discretization applied to the scalar transport equation. J denotes the number of background cells; p denotes the polynomial degree of the discontinuous Galerkin Ansatz. A horizontal trend (ie, the error does not decrease any further under temporal refinement) indicates that the spatial error component is dominating

In this pseudo-1D setting, it is straightforward to find a nonpolynomial function, which fulfils the jump conditions for all times, for example,

$$u_{\text{ex}}(t,x,y) = \begin{cases} o_{\mathfrak{A}} + h_{\mathfrak{A}} \cos(x\pi 5/8) & x > \frac{4}{10}(1+t) \\ o_{\mathfrak{B}} + h_{\mathfrak{B}} \cos(x\pi/5) & x < \frac{4}{10}(1+t) \end{cases}. \tag{45}$$

We have chosen the functions in (45) as

$$\begin{aligned}
h_{\mathfrak{A}} &= 1 - t - t^2 \\
h_{\mathfrak{B}} &= -\frac{5}{16}\frac{\sin(c_0 + c_0 t)}{\sin(c_1 + c_1 t)}(t^2 + t - 1) \\
o_{\mathfrak{A}} &= \frac{1}{16\sin(c_1 + c_1 t)}\big[16\cos(c_0 + c_0 t)t^2 \sin(c_1 + c_1 t) - 5\sin(c_0 + c_0 t)\cos(c_1 + c_1 t)t^2 \\
&\quad + c_2 \sin(c_0 + c_0 t)t^2 + 16\cos(c_0 + c_0 t)t\sin(c_1 + c_1 t) \\
&\quad - 5\sin(c_0 + c_0 t)\cos(c_1 + c_1 t)t + c_2 \sin(c_0 + c_0 t)t \\
&\quad - 16\cos(c_0 + c_0 t)\sin(c_1 + c_1 t) + 5\sin(c_0 + c_0 t)\cos(c_1 + c_1 t) \\
&\quad - c_2 \sin(c_0 + c_0 t)\big] \\
o_{\mathfrak{B}} &= c_3 \frac{\sin(c_0 + c_0 t)}{\sin(c_1 + c_1 t)}(t^2 + t - 1)
\end{aligned}$$

with $c_0 = 0.7853981635, c_1 = 0.2513274123, c_2 = 4.045084972$, and $c_3 = 0.2528178107$. This manufactured solution does however not satisfy Equation 1 in the bulk, which is why we introduce a source term q on the right hand such that the resulting differential equation is fulfilled for the entire domain. That is, we obtain the equation

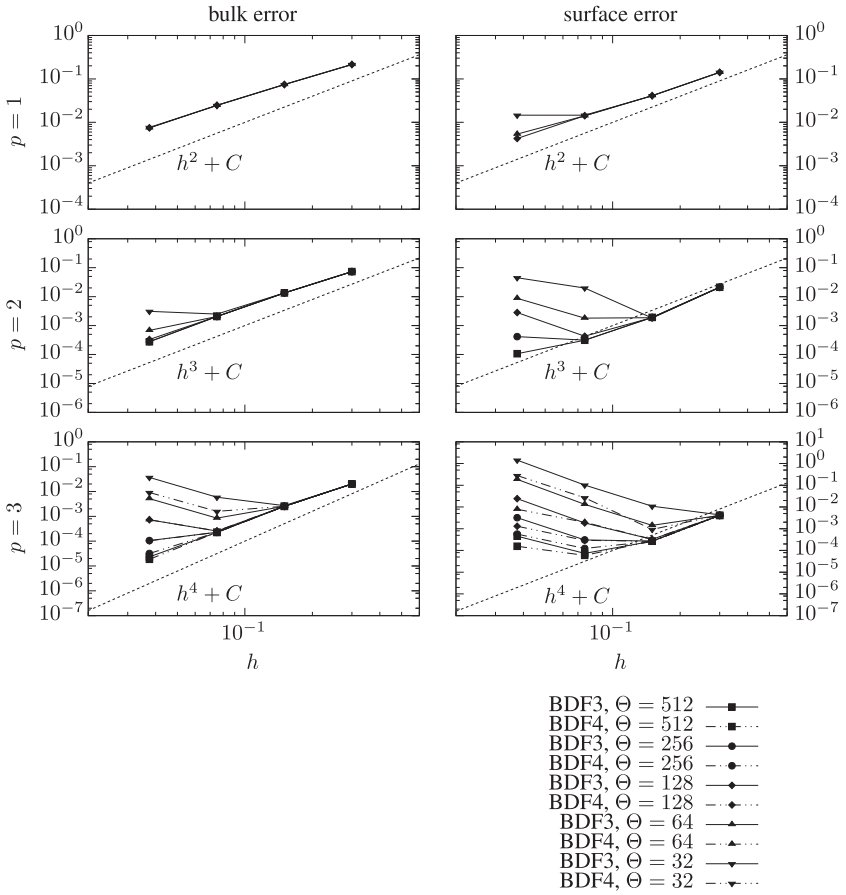

FIGURE 9 Spatial convergence study of the moving interface discretization applied to the scalar transport equation. Θ denotes the number of time steps; p denotes the polynomial degree of the discontinuous Galerkin Ansatz. We observe that the requirement for temporal refinement is over-proportional; eg, see curves for $p = 3$ and BDF3: An increase of the temporal resolution by a factor of 2 requires approximately an increase of the temporal resolution by a factor of 4 to regain the convergence rate of 4

$$\partial_t u + \operatorname{div}\left(f(\vec{u})\right) = q, \tag{46}$$

which we discretize instead of Equation 1.

Our coarsest grid consists of 18×2 equidistant, quadrilateral cells, and we simulate up to an end time of $t_{\text{end}} = 0.4$ using different BDF schemes and a DG degree of $p = 2$. For the discretization of the Laplace operator, we use a symmetric interior penalty formulation as introduced by Arnold.[22] At the moving interface, we correct the flux by the downwind value (see Equation 18).

Results and discussion: A plot of the mesh, initial value, and exemplary numerical solution is shown in Figure 10. We compare splitting to the moving interface approach using a temporal (Figure 11) as well as a spatial (Figure 12) convergence study.

Regarding splitting, we observe that the result is slightly better than in the scalar transport case. Even though we are still not able to obtain consistent spatial convergence rates, Figure 11 indicates that temporal convergence can be restored using reasonable spatial resolutions, although far below the theoretical performance of the time integration scheme. This behaviour can be explained by the difference in the jump condition compared to the scalar transport equation: The splitting approach strongly benefits from the fact that Equation 43 enforces a continuous solution of the heat equation at all times. Still, splitting consistently delivers inferior results in almost all configurations.

For the moving interface approach, we obtain the expected spatial convergence rate of 3 (cf Figure 12) and the expected temporal convergence rate of 2 for the BDF2 scheme up to the point where the spatial error becomes dominating. Note that the error for BDF3 is solely determined by the spatial resolution, except at the coarsest resolution. This implies that the temporal error decays faster than the spatial error, which should require a BDF4 according to the arguments lined out in Section 2.2.1. A closer inspection reveals that this behaviour stems from the fact that the interface is paral-

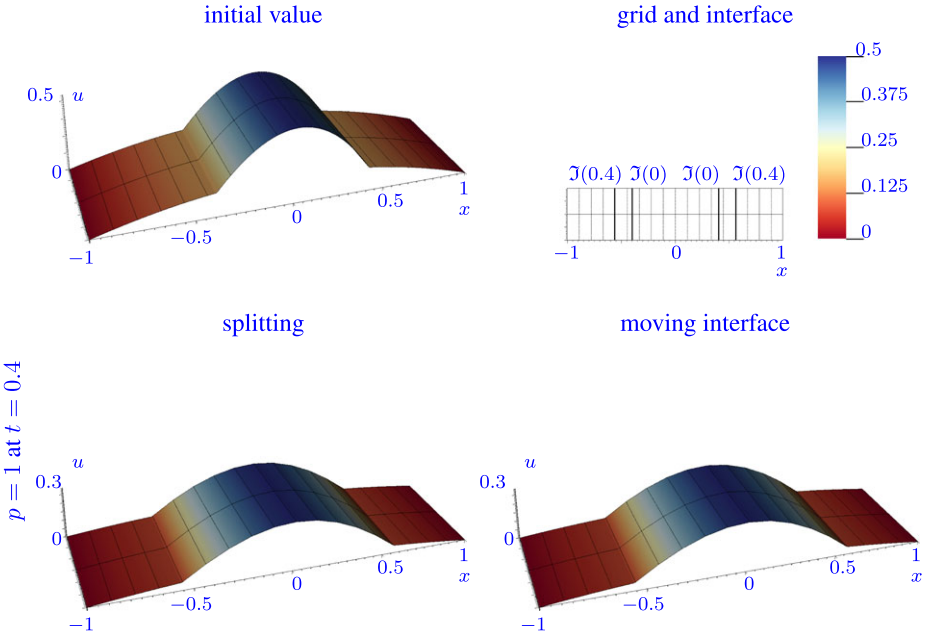

FIGURE 10 Plot of the initial value as well as the solution for $t = 0.4$ for the heat equation with the exact solution (45). For the heat equation, splitting obviously performs better as for the scalar transport or Burgers equation—compare Figures 4 and 13. The difference between $p = 1$ and $p = 2$ is almost invisible in the plot; therefore, the latter one is omitted

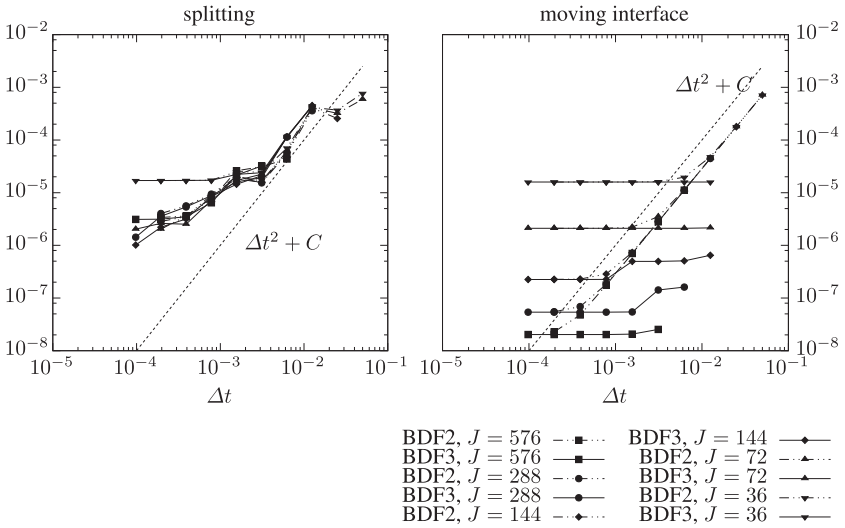

FIGURE 11 Comparison of the temporal convergence of the bulk error for the splitting and the moving interface approach applied to the heat equation for $p = 2$. J denotes the number of background cells. For the moving interface approach, the temporal convergence order of 2 is obtained up to the point where spatial errors become dominant. In this very special configuration (interface is always parallel to cell boundaries), BDF3 provides exact temporal integration; ie, only the spatial error is relevant. For large time steps and high spatial resolution, agglomeration decreases the effective h

lel to the cell boundaries of our Cartesian mesh, which causes different components errors to cancel out. Indeed, this super-convergence property is lost as soon when switching to a non-Cartesian grid.

5.3 | Pseudo-1D Burgers

Problem statement: The 1D Burgers equation is embedded in a 2D domain $\Omega = (-2, 2)^2$ by using the flux

$$\vec{f} := \frac{1}{2}\vec{V}u^2 \tag{47}$$

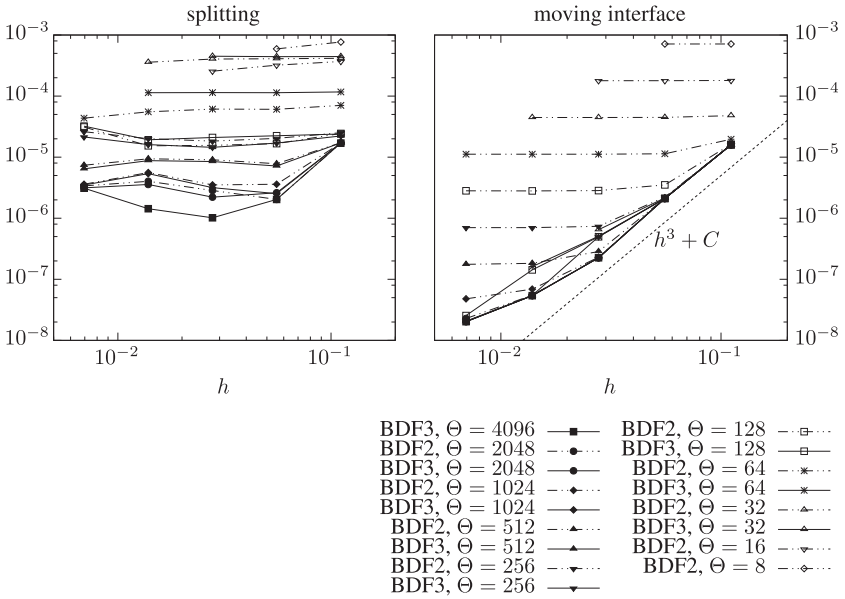

FIGURE 12 Spatial convergence study of the bulk error for both discretization approaches applied to the heat equation for $p = 2$. Θ denotes the number of time steps. Using sufficiently small time step size, we observe that the moving interface approach in combination with BDF2 and BDF3 is able to deliver the expected spatial convergence rate of 3

with

$$\vec{V} := \frac{1}{\sqrt{2}} \begin{bmatrix} 1 \\ 1 \end{bmatrix}. \tag{48}$$

We use the initial value

$$u_0(\vec{x}) := \begin{cases} 1 & \vec{V} \cdot \vec{x} > 0 \\ \exp(-\vec{V} \cdot \vec{x}) + 1 & \vec{V} \cdot \vec{x} < 0 \end{cases}. \tag{49}$$

Then, the Rankine-Hugoniot condition implies that the shock is travelling with a constant speed of $\frac{3}{2}$ in direction $\vec{V}$. The exact solution for this configuration can be found by the method of characteristics. We obtain

$$u_{\text{ex}}(t, \vec{x}) := \begin{cases} 1 & \vec{V} \cdot \vec{x} > \frac{3}{2}t \\ u_0(\vec{x}_0) & \vec{V} \cdot \vec{x} < \frac{3}{2}t \end{cases}, \tag{50}$$

where $\vec{x}_0$ can be obtained for any $\vec{x}$ by solving

$$\vec{V} \cdot \vec{x}_0 + (\exp(\vec{V} \cdot \vec{x}_0) + 1)t = \vec{V} \cdot \vec{x}. \tag{51}$$

The Burgers equation is especially difficult to analyse since the shock velocity depends on the shock height. We thus move the interface with the exact shock speed to avoid any influence of fluctuations the numerical shock speed. As a consequence, this study is limited to small end times. We use a fourth-order Runge-Kutta scheme and DG polynomial degrees of $p = 1$ and $p = 2$.

Results and discussion: A plot of the mesh, initial value, and exemplary numerical solution is shown in Figure 13. Once again, we compare the performance of splitting and the moving interface approach by means of a temporal convergence (Figure 14) and a spatial convergence (Figure 15) study. Just as in the previous cases, the moving interface approach outperforms splitting by orders of magnitude.

Regarding the moving interface approach, the temporal convergence results summarized in the right of Figure 14 show that the expected rate of the fourth-order scheme can be recovered for sufficient spatial resolution. On the other hand, the corresponding spatial convergence results (cf Figure 15, right) reconfirm the above-mentioned necessity for temporal accuracy: Second-order spatial convergence ($p = 1$) can readily be achieved using a fourth-order time integration, but insufficient temporal resolution spoils the spatial convergence for $p = 2$.

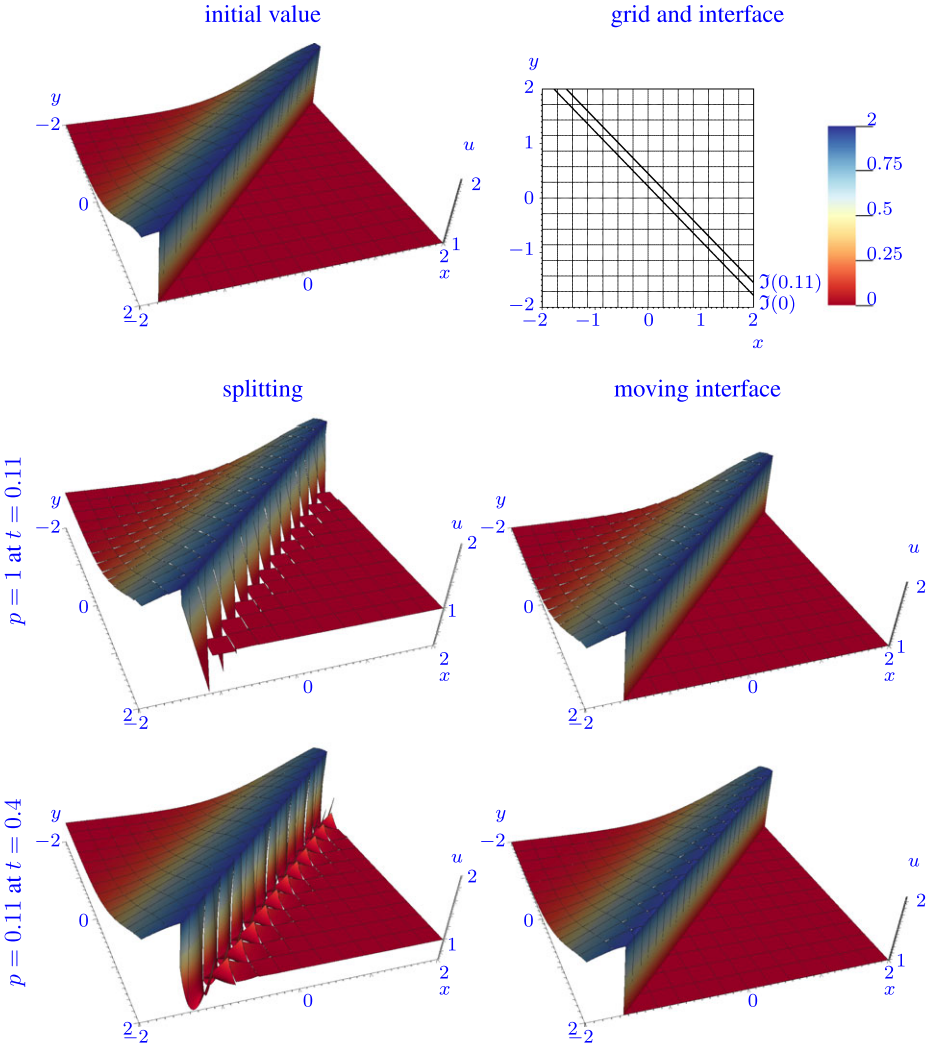

FIGURE 13 Plot of the initial value as well as the solution for $t = 0.11$ for the pseudo-1D Burgers Equation 47 and exact solution (50). The splitting discretization shows high oscillations—probably this could be fixed by a suitable limiter. The moving interface discretization, however, also works with a unlimited flux, as given in Equation 25. Since the interface is aligned in a 45 degree angle to the grid, the computation is essentially different than a pure 1D-calculation

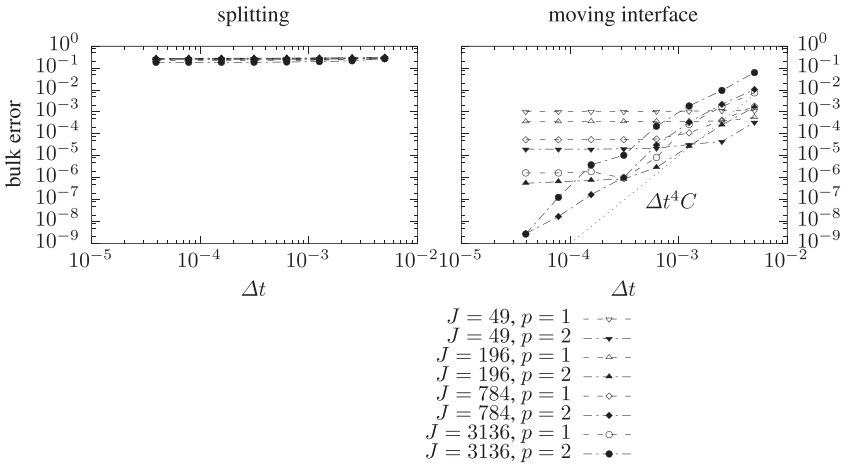

FIGURE 14 Comparison of the temporal convergence of the bulk error for the splitting and the moving interface approach applied to the Burgers equation using a fourth-order Runge-Kutta scheme. J denotes the number of background cells; p denotes the polynomial degree of the discontinuous Galerkin Ansatz

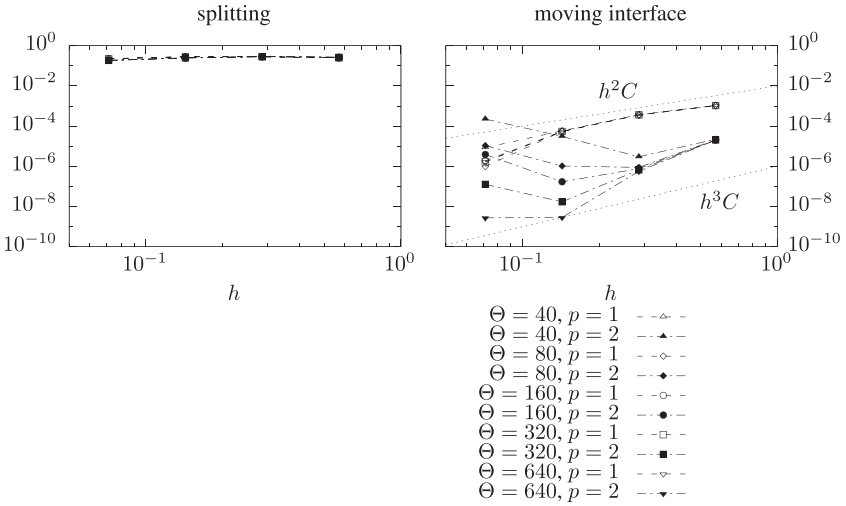

FIGURE 15 Spatial convergence study of the bulk error for both discretization approaches applied to the Burgers equation using a fourth-order Runge-Kutta scheme. Θ denotes the number of time steps; p denotes the polynomial degree of the discontinuous Galerkin Ansatz

6 | SUMMARY AND OUTLOOK

In this paper, we investigated 2 different time discretizations for XDG schemes involving moving interfaces that partition a domain into 2 moving subdomains. Both time discretizations are compatible with standard time integrators such as BDF or Runge-Kutta schemes. Topological changes in the mesh—ie, configurations where the interface enters or leaves a background cell during a time step—are handled with a cell-agglomeration procedure. This can be interpreted as a temporal re-meshing where cut cells are joined in a fashion so that the topology remains constant during one time step.

An important conclusion of our numerical experiments is that the splitting approach should not be used for problems with discontinuous solutions. In those cases, it is almost impossible to observe temporal or spatial convergence with splitting. As a result, extended higher order methods such as the XDG scheme considered in this work deliver no improvement to first-order methods on fixed grids. However, splitting proved less problematic for the heat equation, where only the gradient of the solution is discontinuous.

The moving interface discretization, on the other hand, is able to recover the expected convergence orders in time and space. However, one of the most important guidelines for the application of this method is that spatial and temporal resolution should not be chosen independently. Even for implicit schemes that have no intrinsic time step restriction, an increase in spatial resolution may *increase* the error if the precision of the temporal integration is insufficient. Therefore, spatial and temporal approximation orders be chosen consistently, which is particularly problematic for higher order discretizations: A spatial approximation of degree p theoretically requires a time integration scheme of at least order $2p$, but there is an obvious limit for the latter in implicit settings because, eg, BDF schemes above order 4 are known to be unstable.

Using lower time integration orders causes the moving interface approach to require quite small time steps to deliver the desired performance. Still, all our numerical experiments indicate that the moving interface discretization performs much better than the splitting method. Considering that both time discretization approaches have comparable computational costs, the moving interface approach seems to be the method of choice for problems where physical requirements demand relatively small time steps. An important application scenario is thus given, for example, by multiphase flows with surface tension effects, where the presence of capillary waves induces a strong time step restriction. As a result, the limitation to small time steps is not necessarily prohibitive, which is why the simplicity and efficiency of our moving interface approach features distinct advantages over space-time methods for this type of problems.

ACKNOWLEDGEMENTS

The work of F. Kummer was supported by the German Research Foundation (DFG) through the Collaborative Research Center 1194/B06. The work of B. Müller was supported by the German Research Foundation (DFG) through Research

Grant WA2610/2-1. The work of T. Utz was supported by the German Research Foundation (DFG) through the Priority Programme 1506.

ORCID

Florian Kummer http://orcid.org/0000-0002-2827-7576

REFERENCES

1. Kummer F. Extended discontinuous Galerkin methods for two-phase flows: the spatial discretization. *Int J Numer Methods Eng*. 2017;109(2):259-289. https://doi.org/10.1002/nme.5288
2. Bastian P, Engwer C. An unfitted finite element method using discontinuous Galerkin. *In J Numer Methods Eng*. 2009;79(12):1557-1576.
3. Lehrenfeld C. The Nitsche XFEM-DG space-time method and its implementation in three space dimensions. *SIAM J Sci Comput*. 2015;37(1):A245-A270.
4. Babuška I. The finite element method for elliptic equations with discontinuous coefficients. *Comput*. September 1970;5(3):207-213.
5. Barrett JW, Elliott CM. Fitted and unfitted finite-element methods for elliptic equations with smooth interfaces. *IMA J Numer Anal*. 1987;7(3):283-300.
6. Moës N, Dolbow J, Belytschko T. A finite element method for crack growth without remeshing. *Int J Numer Methods Eng*. 1999;46:131-150.
7. Zilian A, Legay A. The enriched space-time finite element method (EST) for simultaneous solution of fluid-structure interaction. *Int J Numer Methods Eng*. 2008;75(3):305-334.
8. Heimann F, Engwer C, Ippisch O, Bastian P. An unfitted interior penalty discontinuous Galerkin method for incompressible Navier-Stokes two-phase flow. *Int J Numer Methods Fluids*. 2013;71(3):269-293.
9. Merle R, Dolbow J. Solving thermal and phase change problems with the eXtended finite element method. *Comput Mech*. 2002;28(5):339-350.
10. Chessa J, Smolinski P, Belytschko T. The extended finite element method (XFEM) for solidification problems. *Int J Numer Methods Eng*. 2002;53(8):1959-1977.
11. Chessa J, Belytschko T. An extended finite element method for two-phase fluids. *J Appl Mech*. 2003;70(1):10-17.
12. Fries T-P, Zilian A. On time integration in the XFEM. *Int J Numer Methods Eng*. 2009;79(1):69-93.
13. Réthoré J, Gravouil A, Combescure A. A combined space-time extended finite element method. *Int J Numer Methods Eng*. 2005;64(2):260-284.
14. Chessa J, Belytschko T. Arbitrary discontinuities in space-time finite elements by level sets and X-FEM. *Int J Numer Methods Eng*. 2004;61(15):2595-2614.
15. Cheng KW, Fries T-P. Higher-order XFEM for curved strong and weak discontinuities. *Int J Numer Methods Eng*. 2010;82(5):564-590.
16. Hansbo P, Larson MG, Zahedi S. A cut finite element method for coupled bulk-surface problems on time-dependent domains. *Comput Methods Appl Mech Eng*. 2016;307:96-116.
17. Kramer RMJ, Noble DR. A conformal decomposition finite element method for arbitrary discontinuities on moving interfaces. *Int J Numer Methods Eng*. 2014;100(2):87-110.
18. Di Pietro DA, Ern A. *Mathematical Aspects of Discontinuous Galerkin Methods*, No. 69 in Mathématiques et Applications. Heidelberg: Springer; 2011.
19. Müller B, Kummer F, Oberlack M. Highly accurate surface and volume integration on implicit domains by means of moment-fitting. *Int J Numer Meth Eng*. 2013;96(8):512-528. https://doi.org/10.1002/nme.4569
20. Müller B, Krämer-Eis S, Kummer F. A high-order discontinuous Galerkin method for compressible flows with immersed boundaries. *Int J Numer Methods Eng*. 2017;110(1):3-30. https://doi.org/10.1002/nme.5343
21. Wang Y, Oberlack M. A thermodynamic model of multiphase flows with moving interfaces and contact line. *Continuum Mech Thermodyn*. 2011;23:409-433.
22. Arnold DN. An interior penalty finite element method with discontinuous elements. *SIAM J Numer Anal*. 1982;19(4):742.

How to cite this article: Kummer F, Müller B, Utz T. Time integration for extended discontinuous Galerkin methods with moving domains. *Int J Numer Meth Eng*. 2018;113:767–788. https://doi.org/10.1002/nme.5634