A High-Order Level-Set Method Coupled with an Extended Discontinuous Galerkin Method for Simulating Moving Interface Problems

David Henneaux*
von Karman Institute for Fluid Dynamics, Rhode-Saint-Genèse, Belgium, 1640

Pierre Schrooyen †, Larbi Arbaoui ‡
Cenaero, Gosselies, Belgium, 6041

Philippe Chatelain§
Université catholique de Louvain, Louvain-la-Neuve, Belgium, 1348

Thierry Magin¶
von Karman Institute for Fluid Dynamics, Rhode-Saint-Genèse, Belgium, 1640

The application of emerging sharp interface high-order methods to complex flow problems is strongly linked with the ability of representing interfaces dynamics with high accuracy. In this work, we propose to develop efficient and high-order reinitialization and extension strategies for the level-set method in the framework of the discontinuous Galerkin method (DGM). The methodology is combined with a state-of-the-art extended DGM (XDGM), which requires special considerations to be consistently applied to moving interface problems. Several verification test cases demonstrate the potential of the proposed approach while identifying future developments in order to benefit from its high resolution features in complex multiphase flow situations.

Nomenclature

DGM	=	Discontinous Galerkin Method
XDGM	=	eXtended Discontinuous Galerkin Method
Γ	=	Phase interface / iso-0 of the level-set
φ	=	Level-set function
Ω_m	=	m^{th}-element in the computational domain Ω
Ω_{NB}	=	set of elements belonging to the narrow-band
I_f	=	f^{th}-edge in the computational domain
$\mathbf{v}_\Gamma$	=	Interface velocity
$\mathbf{v}_{\text{ext}}$	=	Extended velocity
$\mathbf{n}$	=	Normal vector
$\mathbf{x}_{\Gamma,\text{cp}}$	=	Closest point on the interface
$\mathbf{M}$	=	Mass matrix
$\mathcal{F}$	=	Numerical flux function
$\phi_{m,n}$	=	Shape function associated to the n^{th}-degree of freedom on the m^{th}-element
$\mathcal{R}^{\text{temp}}$, $\mathcal{R}^{\text{sp}}$	=	Temporal and spatial residuals
$\mathbf{x}$, $\boldsymbol{\xi}$	=	Physical and parametric coordinates
$d(\mathbf{x})$	=	Distance function

*Ph.D. Student, Aeronautics and Aerospace Department, Chaussée de Waterloo 72, AIAA Student Member.
†Senior Research Engineer, Cenaero, Rue des Frères Wright 29, AIAA Member.
‡Research Engineer, Cenaero, Rue des Frères Wright 29.
§Professor, Institute of Mechanical, Materials and Civil Engineering, Place du Levant 2, AIAA Member.
¶Professor, Aeronautics and Aerospace Department, Chaussée de Waterloo 72, AIAA Member.

$\boldsymbol{U}$ = Fluid nodal solutions
$\mathbf{Q}$ = Coupling matrix for cell-agglomeration

I. Context and motivations

Recently, various unstructured high-order methods have emerged. These methods offer the possibility of combining the geometrical flexibility of finite volume or finite element methods with the high precision of finite difference or spectral methods. The algorithms and the theoretical foundation behind the most mature methods of this kind, like the discontinuous Galerkin methods, are now well-established. In the context of computational fluid dynamics, active research is however still conducted in order to extend and apply those promising methods to complex flow situations. This paper is part of a more global project focused on the application of one of these methods to immiscible two-phase flows separated by a phase interface. More specifically, the goal is to design and combine numerical techniques to predict the dynamics of phase changing materials in a strong shear environment. The main challenge associated to such situations is to be able to preserve the high resolution of high-order methods in the vicinity of the interface where decisive physical phenomena take place. We propose to achieve this by means of a so-called extended discontinuous Galerkin method (XDGM). The focus of the present work is not to describe how the solution jumps at the phase interface are tackled with the XDGM. This was introduced previously in [1]. To simulate complex multiphase flow situations with high-fidelity, accurate methods to capture the interface and its evolution should also be developed. This paper addresses two fundamental aspects in this regard:

- the development of an efficient high-order level-set methodology on unstructured meshes (Sec. II);
- the establishment of coupling strategies between this level-set method and the XDGM in the context of moving interface problems (Sec. III).

The content of each part is briefly introduced below as well as the links between them.

Extended methods should not only enable to capture discontinuous solutions across a phase interface but also handle the evolution of this interface. Since the fluid governing equations do not provide directly any information about the interface location, an associated method representing the interface is needed. As the interface position will be inferred from this method, this will influence the order of convergence of the global approach. It is therefore important to make use of an accurate method supporting the high-order features of the XDGM. The choice here is the level-set method discretized by a classical DGM. The first part of this work is dedicated to the development of high-order reinitialization (Sec. II.D) and velocity extension (Sec. II.E) procedures.

The coupling between the level-set and the XDGM is then described in the second part. One of the main challenges of multiphase flow simulations is that the interface dynamics is determined itself by the solution of the equations for the driving physics. An appropriate coupling between the method representing the interface and the one solving the flow needs to be developed. Two main aspects are specifically investigated: (i) the use of cell-agglomeration technique to build a consistent time integration framework (Sec. III.B), (ii) the exchange of information between the two methods considered (Sec. III.C). This first topic goes slightly beyond the level-set method and its coupling with the XDGM. However, it seemed important to present it in the context of this paper dealing with general moving interface problems. It concerns an essential ingredient of the XDGM to be applied to such problems.

For the sake of clarity, the two main topics addressed here are described in two separated parts. Each of them is organized around (i) an introductory section giving some background about noticeable work from the literature, (ii) a part describing the equations and the methods involved complemented by some algorithms, and (iii) at the end some verification test cases. The presentation of the various algorithms is incremental. The techniques associated to the level-set method are first detailed before being inserted in the more general coupling methodology, making the link between both parts. The last section of this manuscript summarizes the outcomes of the present work and highlights its perspectives that will be addressed in future publications.

II. High-order single-step reinitialization and extension procedures for the level-set

This first part aims at presenting first a general introduction of the level-set method, its implementation within a DG scheme and two key elements of the method: reinitialization and extension. Narrow band strategy to reduce computational cost is also detailed. Finally, the various methods are organized inside a global algorithm.

A. The representation of moving interfaces and the choice of the level-set method

A large collection of flow applications involve the presence of moving interfaces, the most common situation being the interaction between two fluid phases. For those applications, one needs to accurately capture the dynamics within each phase, their coupling and the evolution of the interface itself. Two different approaches can be identified to account for interface evolution: interface tracking and interface capturing methods (see [2, §1.3] for a comprehensive summary). In the first set of methods, the interface is explicitly represented by a function (e.g. marker particles method) or by deforming the grid such that it conforms with the interface position. In the class of interface capturing methods, the interface is represented implicitly by a function defined on all the domain (see [3] for a review about recent interface-capturing methods applied to two-phase flows). To this class belong the volume-of-fluid (VOF) method and the level-set (LS) method. First, for the applications targeted in this work that might involve important interface distortions, a pure Eulerian method is desirable in order to use a fixed grid. Second, an interface capturing method is better adapted to deal with complex interfacial motions like merger, folding and break-up. The choice was therefore made to use the level-set method because it can handle topology changes of the interface automatically and it allows for a straightforward calculation of interfacial geometrical quantities [2]. In addition, the level-set method was the best candidate to develop an unfitted/ extended discontinous Galerkin method as it does not need any interface reconstruction and provides a direct implicit representation of the interface which can be used to build quadrature rules. One can note that methods combining advantages of both the volume-of-fluid and the level-set methods have been recently developed and would be perfectly suited as well.

B. Level-set properties and governing equation

The level-set method first proposed in [4] relies on two central embeddings: (i) the embedding of the $d-1$ dimensional interface Γ as the zero level-set of a d dimensional function φ defined over the entire domain Ω, (ii) the embedding (or extension) of the interface velocity $\mathbf{v}_\Gamma$ to this higher dimensional level-set function [5].

By definition, the interface is implicitly identified by the zero iso-contour of the level-set function $\varphi : \mathbb{R}^d \rightarrow \mathbb{R}$ for all time t, i.e.

$$\Gamma(\mathbf{x}, t) = \left\{ \mathbf{x} \in \mathbb{R}^d \,|\, \varphi(\mathbf{x}, t) = 0 \right\} \ , \quad \forall t \ . \tag{1}$$

The normal vector $\mathbf{n}_\Gamma$ to the interface and the interface curvature κ_Γ can then be obtained as

$$\mathbf{n}_\Gamma(\mathbf{x}, t) = \frac{\nabla\varphi(\mathbf{x}, t)}{\|\nabla\varphi(\mathbf{x}, t)\|} \ , \quad \kappa_\Gamma(\mathbf{x}, t) = \nabla \cdot \left(\frac{\nabla\varphi(\mathbf{x}, t)}{\|\nabla\varphi(\mathbf{x}, t)\|} \right) \ , \quad \forall \mathbf{x} \in \Gamma(\mathbf{x}, t) \ . \tag{2}$$

Away from the interface front, the level-set function needs also to be specified and it is usually defined as the signed distance to the interface, i.e.

$$\begin{aligned} \varphi(\mathbf{x}, t) &= \begin{cases} \text{dist}(\mathbf{x}, \Gamma(\varphi)) \ , & \mathbf{x} \in \Omega_1 \ , \\ 0 \ , & \mathbf{x} \in \Gamma \ , \\ -\text{dist}(\mathbf{x}, \Gamma(\varphi)) \ , & \mathbf{x} \in \Omega_2 \ , \end{cases} \\ \text{with:} \quad \text{dist}(\mathbf{x}, \Gamma(\varphi)) &= \min_{\mathbf{x}_{\Gamma,\text{cp}} \,:\, \varphi(\mathbf{x}_{\Gamma,\text{cp}})=0} \left\| \mathbf{x} - \mathbf{x}_{\Gamma,\text{cp}} \right\| \ , \end{aligned} \tag{3}$$

where $\mathbf{x}_{\Gamma,cp}(\mathbf{x}, t) \in \Gamma(\mathbf{x}, t)$ is the closest point belonging to the interface from the point $\mathbf{x}$. Defining the level-set as a signed distance function has interesting properties, namely that it is everywhere differentiable (except at some singular points) and that its gradients satisfy the eikonal equation,

$$\|\nabla\varphi(\mathbf{x})\| = 1 \ , \quad \mathbf{x} \in \mathbb{R}^d \ . \tag{4}$$

The level-set function, initially defined as a signed distance function, may loose its distance property due to advection. This can be troublesome for two main reasons: (i) the gradient of the level-set may become very steep in case the level-set function becomes disturbed and the computation of the normal may lead to numerical instabilities, (ii) the level-set function may become too flat to accurately determine the position of the iso-0. To overcome those challenges, reinitialization is used to recover the signed distance property. This procedure is detailed in Sec. II.D.

In order to update the position of the interface at every time step, the level-set advection equation needs to be solved:

$$\begin{aligned} \frac{\partial\varphi(\mathbf{x}, t)}{\partial t} + \mathbf{v}_{\text{ext}}(\mathbf{x}, t) \cdot \nabla\varphi(\mathbf{x}, t) = 0 \ , \quad \forall \mathbf{x} \in \mathbb{R}^d \ , \\ \varphi(\mathbf{x}, t = 0) = \varphi^0(\mathbf{x}) \ , \end{aligned} \tag{5}$$

where $\mathbf{v}_{\text{ext}}$ is the extended advection velocity which should, in the limit as one approaches the zero level-set, smoothly yield the velocity of the interface $\mathbf{v}_\Gamma$, i.e.

$$\lim_{\mathbf{x}\to\mathbf{x}_\Gamma} \mathbf{v}_{\text{ext}}(\mathbf{x}) = \mathbf{v}_\Gamma(\mathbf{x}_\Gamma) \ , \quad \mathbf{x} \in \Omega \ , \ \mathbf{x}_\Gamma \in \Gamma \ . \tag{6}$$

Beyond that requirement, different approaches can be considered to extend the velocity to every points away from the interface. The approach used in this work is described in Sec. II.E.

C. Spatial and temporal discretizations of the level-set advection equation

As mentioned in the introductory section, the level-set advection equation is spatially discretized by means of a classical DGM. Equation (5) is first rewritten in conservative form:

$$\frac{\partial \varphi}{\partial t} + \nabla \cdot (\mathbf{v}_{\text{ext}}\varphi) = \varphi \nabla \cdot \mathbf{v}_{\text{ext}} \ . \tag{7}$$

Since an extended velocity is used in the domain, this velocity field will not necessarily be divergence free and therefore the source term on the right hand side of Eq. (7) has to be taken into account. The DG discretization of this equation on a discretized domain $\Omega = \cup \Omega_m$ with the set of element interfaces $I = \cup I_f$ is then formulated as:

$$\begin{aligned} &\text{Solve for } \boldsymbol{\Phi} \text{ such that} \\ &\sum_{\Omega_m \in \Omega} \mathbf{M}_m \frac{\mathrm{d}\boldsymbol{\Phi}_m}{\mathrm{d}t} = \sum_{\Omega_m \in \Omega} \int_{\Omega_m} \nabla^h \boldsymbol{\phi}_m \cdot \left(\boldsymbol{\varphi}_m^h \mathbf{v}_{\text{ext}}^h \right) \mathrm{d}V \\ &\qquad - \sum_{I_f \in I} \int_{I_f} [\boldsymbol{\phi}_f] \mathcal{F}(\boldsymbol{\varphi}_m^{h,+}, \boldsymbol{\varphi}_m^{h,-}, \mathbf{n}_f) \, \mathrm{d}S \\ &\qquad + \sum_{\Omega_m \in \Omega} \int_{\Omega_m} \boldsymbol{\phi}_m \left(\boldsymbol{\varphi}_m^h \nabla^h \cdot \mathbf{v}_{\text{ext}}^h \right) \mathrm{d}V \ , \ \forall \boldsymbol{\phi}_m \end{aligned} \tag{8}$$

in which:

- The subscripts $\bullet_m$ and $\bullet_f$ refer to the m^{th}-element and the f^{th}-interface on the computational grid and $[\bullet]$ is the jump operator;
- $\mathbf{M}_m = \int_{\Omega_m} \boldsymbol{\phi}_m^T \boldsymbol{\phi}_m \, \mathrm{d}V$ is the element mass matrix;
- $\boldsymbol{\phi}_m$ are elementwise defined Lagrange shape functions spanning the polynomial space of degree p $\mathcal{P}_p(\Omega) = \bigcup_m \mathcal{P}_p(\Omega_m)$ to which the approximated finite dimensional level-set solution belongs: $\varphi \approx \varphi^h \in \mathcal{P}_p(\Omega)$;
- $\boldsymbol{\Phi}_m$ are the solved-for nodal degrees of freedom and are linearly combined with the shape function to interpolate the solution on each element:

$$\boldsymbol{\varphi}_m^h(\mathbf{x}, t) = \sum_{n=1}^{N_m} \phi_{m,n}(\mathbf{x}) \Phi_{m,n}(t) \ , \ \mathbf{x} \in \Omega_m \ . \tag{9}$$

- $\mathcal{F}$ is the convective numerical flux function that depends on the solutions from both sides of an element interface and is taken as an upwind flux:

$$\mathcal{F}(\varphi^+, \varphi^-, \mathbf{n}) = \begin{cases} \varphi^- \, v_n & \text{if } v_n > 0 \ , \\ \varphi^+ \, v_n & \text{if } v_n \leq 0 \ , \end{cases} \qquad \text{where } v_n = \mathbf{v}_{\text{ext}} \cdot \mathbf{n} \ . \tag{10}$$

The resulting system of ODE's reads under a compact form as

$$\mathbf{M} \frac{\mathrm{d}\boldsymbol{\Phi}}{\mathrm{d}t} = \mathcal{R}_{\text{LS}}^{\text{sp}}(\boldsymbol{\Phi}, \mathbf{v}_{\text{ext}}) \ , \tag{11}$$

where $\mathcal{R}_{\text{LS}}^{\text{sp}}$ is the residual associated to the right-hand side spatial terms of Eq. (8).

In this work, time integration of Eq. (11) is performed with an implicit scheme. More precisely, the approach makes use of a dual-time stepping in which the pseudo-temporal term is discretized by a backward Euler scheme. An inexact

Newton's method (i.e. a single iteration is performed) is used to linearize the resulting non-linear system. This leads to the following linear system to be solved at each inner iteration:

$$\left[\frac{\mathbf{I}}{\Delta\tau} + \mathbf{M}^{-1}\frac{\partial \mathcal{R}^{\text{tot}}_{\text{LS}}}{\partial \mathbf{\Phi}}(\mathbf{\Phi}_k)\right](\mathbf{\Phi}_{k+1} - \mathbf{\Phi}_k) = -\mathbf{M}^{-1}\mathcal{R}^{\text{tot}}_{\text{LS}}(\mathbf{\Phi}_k, \mathbf{v}_{\text{ext}}) \,, \tag{12}$$

where τ is the pseudo-time and $\mathcal{R}^{\text{tot}}$ is the sum of the spatial and temporal residuals.

D. Reinitialization and computation of closest points on interface

1. Some background

Three main approaches exist in the literature to reinitialize the level-set function in order to recover its signed distance property. The first one is to solve the eikonal Eq. (4). Very efficient methods like the Fast Marching Method [6] and the Fast Sweeping Method [7] have been designed for that purpose. The second approach is to convert the static eikonal equation into a time-dependent hyperbolic PDE whose steady-state solution returns the signed distance function [8], or to reformulate the equation as a minimization problem and solve an elliptic equation [9]. The last approach is to stand at each computational mesh point, find the corresponding closest point on the interface and assign to the level-set the signed distance between the mesh point and the closest point on the front [10]. This geometrical approach directly uses the definition of the level-set function given by Eq. (3). This is the one adopted in this work for two main reasons: (i) the availability inside our solver of some building blocks that could be directly used, (ii) the simplicity associated to the non-iterative (i.e. single-step) and geometrical features on which the approach is based.

2. Implementation

The reinitialization strategy that has been implemented is largely based on the work of Saye [11] in which a procedure for high-order closest point calculations to implicitly defined surfaces is presented. This new implementation extends the original one available in the discontinuous Galerkin Argo platform* based on the work of Marchandise [2]. The different steps involved in the reinitialization procedure are described below.

1. A global *set of sampling points approximating the interface* identified by Eq. (1) is first generated. This is accomplished by using a *recursive contouring algorithm* [13]:
   - the elements crossed by the iso-0 of the level-set are divided recursively (with a chosen recursion level) into sub-elements and the edges of the subdivided elements are collected;
   - the values of the level-set at each vertex defining the edges are then obtained by interpolation;
   - if an edge is identified as being cut by the iso-0, i.e. if the level-set values at its vertices are of opposite sign, the approximate intersection point between the iso-0 and the edge is found by assuming a linear variation of the level-set solution in-between the two vertices:

   $$\mathbf{x}^*_\Gamma = \mathbf{x}^1_{\text{vertex}} - \varphi^1_{\text{vertex}}\frac{\mathbf{x}^2_{\text{vertex}} - \mathbf{x}^1_{\text{vertex}}}{\varphi^2_{\text{vertex}} - \varphi^1_{\text{vertex}}} \,. \tag{13}$$

   Starting from the approximated intersection point $\mathbf{x}^*_\Gamma$, a simple Newton's procedure is finally applied to project it onto the iso-0:

   $$\mathbf{x}_\Gamma \leftarrow \mathbf{x}_\Gamma - \frac{\varphi(\mathbf{x}_\Gamma)\nabla\varphi(\mathbf{x}_\Gamma)}{\|\nabla\varphi(\mathbf{x}_\Gamma)\|^2} \,. \tag{14}$$

   The sampling point lying exactly on the iso-0 is then added to the global set of sampling points: `cloudPts` = `cloudPts` $\cup\, \mathbf{x}_\Gamma$.
2. The sampling points are then stored in an Approximate Nearest Neighbor (ANN) *search tree* [14] denoted by `searchTree` after.
3. For each computational mesh point $\mathbf{x}_q$ (interpolation or quadrature point, see after), a *high-order computation of closest points* on the interface is finally applied:
   - first, the previously created search tree is used to find, in a efficient way, the closest point $\mathbf{x}^{\text{guess}}_{\Gamma,\text{cp}}$ among the set of sampling points;

*Argo is a proprietary code developed by the Belgian research center Cenaero. Details can be found in [12].

– then this point is used as a first guess to start a constrained optimization problem stated as

$$
\begin{aligned}
&\text{Starting from } \mathbf{x}_{\Gamma,\text{cp}}^{\text{guess}}\,,\\
&\text{Find } \mathbf{x}_{\Gamma,\text{cp}} \in \mathbb{R}^d \text{ such that:}\\
&\qquad \mathbf{x}_{\Gamma,\text{cp}} = \underset{\mathbf{x}\in\mathcal{F}}{\text{argmin}}\; d(\mathbf{x})\\
&\text{with:}\\
&\qquad d(\mathbf{x}) = \frac{1}{2}\left\|\mathbf{x}-\mathbf{x}_q\right\|^2\\
&\mathcal{F} = \{\, \mathbf{x}\in\mathbb{R}^d \;|(1)\; \varphi(\mathbf{x}) = 0\,,\\
&\qquad\qquad (2)\; \boldsymbol{\xi}(\mathbf{x}) \in \hat{\Omega}\,\}
\end{aligned}
\tag{15}
$$

which corresponds to finding the point lying on the iso-0 of the level-set function (first constraint in the feasible region set $\mathcal{F}$) minimizing the distance (the function $d(\mathbf{x})$) between this point and the target $\mathbf{x}_q$. The second constraint in the feasible region $\mathcal{F}$ deserves some more attention. It ensures that the solution expressed in parametric coordinates (obtained with the isomorphism $\boldsymbol{\xi}(\mathbf{x})$) stays within the reference parametric element $\hat{\Omega}$. First, as it will be discussed after, this minimization problem is reformulated in parametric coordinates as all the operations in a FEM context. Second, as the level-set function is represented by elemenwize polynomial functions, the process should ensure that the iterate stays within the element in which it started, otherwize the first constraint would not be valid anymore. Satisfying this second constraint, which translates into a set of inequality constraints, is not as straightforward as the first one. That is why two different approaches based on the method of Lagrange mulipliers were implemented. The first one, described below, does not directly incorporate the inequality constraints into its formulation. On the other hand, the augmented Lagrangian method allows to ensure the satisfaction of all the constraints for each new iterate. Numerical experiments showed no significant improvements using the second approach and since it is more computationally expensive, the first approach is used in this work.

Classical Lagrangian method with a single equality constraint: In this first method, the second constraint in Problem (15) is not directly incorporated into the formulation, only the first one is. Thus, by virtue of the method of Lagrange, it is recast as:

$$
\nabla L(\mathbf{x},\lambda) = 0\,, \quad \text{with: } L(\mathbf{x},\lambda) = \frac{1}{2}\left\|\mathbf{x}-\mathbf{x}_q\right\|^2 + \lambda\varphi(\mathbf{x})\,,
$$

with the Lagrangian L and the Lagrange coefficient λ. The resulting non-linear system is then solved with an iterative Newton's method, starting from $\mathbf{x}_{\Gamma,\text{cp}}^{\text{guess}}$. An important point to highlight here is the fact that all the operations are usually performed in the parametric coordinates in a FEM framework. The non-linear system to be solved is therefore expressed in terms of the derivatives with respect to the parametric coordinates $\boldsymbol{\xi}$ thanks to the mapping $\mathbf{x}(\boldsymbol{\xi})$:

$$
\nabla_{(\boldsymbol{\xi},\lambda)} L(\mathbf{x}(\boldsymbol{\xi}),\lambda) = 0\,, \tag{16}
$$

and its linearization around a certain point $(\boldsymbol{\xi}, \lambda)$ leads to the linear system giving the Newton's steps $(\Delta\boldsymbol{\xi}, \Delta\lambda)$ at each iteration:

$$
\begin{bmatrix} \nabla^2_{\boldsymbol{\xi}\boldsymbol{\xi}} L(\boldsymbol{\xi},\lambda) & \nabla_{\boldsymbol{\xi}}\varphi(\boldsymbol{\xi}) \\ \nabla^T_{\boldsymbol{\xi}}\varphi(\boldsymbol{\xi}) & 0 \end{bmatrix} \begin{bmatrix} \Delta\boldsymbol{\xi} \\ \Delta\lambda \end{bmatrix} = - \begin{bmatrix} \nabla_{\boldsymbol{\xi}} L(\boldsymbol{\xi},\lambda) \\ \varphi(\boldsymbol{\xi}) \end{bmatrix}\,, \tag{17}
$$

where the gradient of the Lagrangian with respect to $\boldsymbol{\xi}$ can be expressed based on the Jacobian matrix of the mapping between parametric and physical coordinates $\mathbf{J}_{\text{map}}$

$$
\nabla_{\boldsymbol{\xi}} L(\boldsymbol{\xi},\lambda) = \mathbf{J}^T_{\text{map}}(\mathbf{x}(\boldsymbol{\xi}) - \mathbf{x}_q) + \lambda\nabla_{\boldsymbol{\xi}}\varphi(\boldsymbol{\xi})\,, \tag{18}
$$

while the Hessian of the Lagrangian with respect to $\boldsymbol{\xi}$ is computed by central finite differences for simplicity

$$
\nabla^2_{\boldsymbol{\xi}\boldsymbol{\xi}} L(\boldsymbol{\xi},\lambda) = \frac{\nabla_{\boldsymbol{\xi}} L(\boldsymbol{\xi}+\epsilon,\lambda) - \nabla_{\boldsymbol{\xi}} L(\boldsymbol{\xi}-\epsilon,\lambda)}{2\epsilon}\,. \tag{19}
$$

Assuming that $\nabla_{\boldsymbol{\xi}} L(\boldsymbol{\xi}^0, \lambda^0) \approx 0$, the initial value for the Lagrange multiplier is specified as:

$$\lambda^0 = -\frac{\mathbf{J}_{\text{map}}^T(\mathbf{x}^0 - \mathbf{x}_q) \cdot \nabla_{\boldsymbol{\xi}}^T \varphi(\boldsymbol{\xi}^0)}{\left\|\nabla_{\boldsymbol{\xi}} \varphi(\boldsymbol{\xi}^0)\right\|^2} \ . \tag{20}$$

Depending on the local shape of the level-set at which a given iterate is computed, the gradient and Hessian of the Lagrangian may be such that the Netwon's step leads to the next iterate being outside of the reference element. This will of course be the case if the exact closest point can not be found inside the current element, in which case the process should be repeated on adjacent elements, as explained after. However, the exact solution could still be located inside the current element but be missed due to a too large Newton's step. Relaxation method is applied to limit the step in case the next iterate lies outside of the reference element. Performing such simple damping does however not guarantee that the next iterate will satisfy the equality constraint.

3. General algorithm and additional remarks

Algorithm 1 compiles the different steps of the reinitialization procedure. It is accompanied by the following additional remarks:

- *Considering level-set outside of its parent element*: instead of enforcing the iterates to remain strictly inside the reference element, we may set a non-zero margin allowing some overlapping between adjacent elements. This is particularly useful when the exact solution is located just on the other side of an element edge. However, in high curvature regions, considering an extension of the level-set in a neighbouring element, even small, may impact the accuracy;
- *Considering adjacent elements in case of non-convergence*: starting from a guess point inside a given element, the iterative process may not be able to find a solution inside this same element satisfying the prescribed tolerance. In this situation, the process is applied on the neighbouring elements. One first needs to retrieve the sampling points defined on each adjacent elements, then create a local search tree and find a new guess located inside the adjacent element. The minimization method is then called starting from the guess and the loop over the adjacent elements is stopped as soon as the exact point is found.
- *Sampling elements nastily cut by the iso-0*: on elements containing a very short segment of the level-set iso-0, it may be that the only sampling points available are the intersections points between the iso-0 and the element edges. As starting the iterative process with a guess point located on an element boundary is not optimal, subdivision with a higher recursion level is done on those elements.
- *Interpolation and quadrature points*: the mesh points for which the closest point computation is performed can be selected as either the interpolation points or the quadrature points. If the interpolation points are chosen, the reinitialized level-set field is directly obtained since it is defined by the values at the nodal degrees of freedom. On the other hand, when the quadrature points are selected, the reinitialized field is obtained by an L^2-projection of the quadrature points values.

4. Verification test case

A test case from [15] is selected to illustrate the convergence rate of the reinitialization procedure. It consists of two circles of different radii with iso-contours initially specified by an exponential function, as shown in Fig. 1a. The level-set iso-contour field after the application of a single reinitialization cycle is shown in Fig. 1b. The signed distance property is recovered on the narrow-band region while the level-set values are set to a constant value outside (the narrow-band concept is presented in Sec. II.F). The L_2 and L_∞ errors plotted in Fig. 2, whose definitions are given in Sec. II.E.4, confirm the high-order behavior of the procedure.

E. Velocity extension

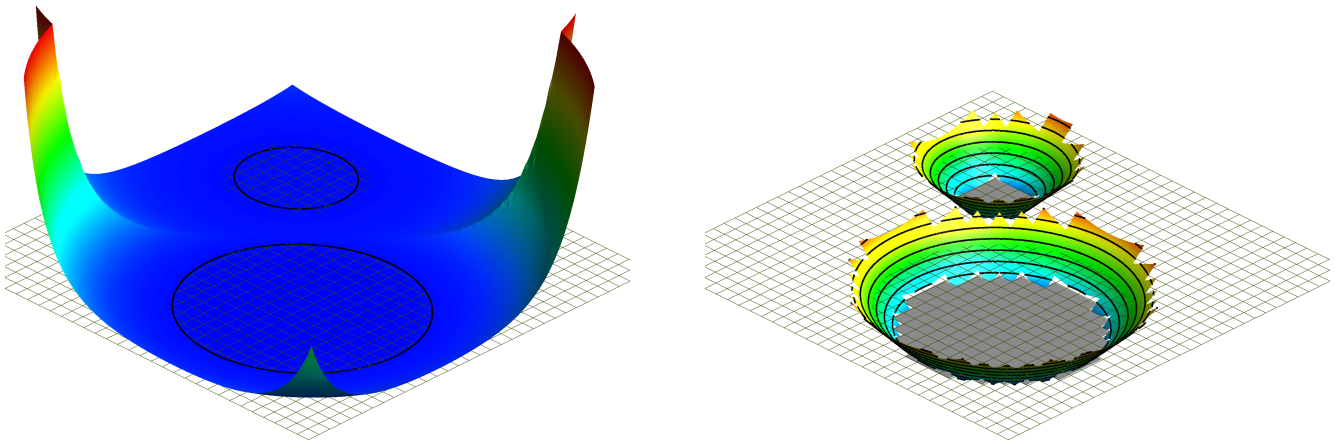
1. Some background

The most direct approach to define the velocity $\mathbf{v}_{\text{ext}}$ in Eq. (5) away from the interface is to use the underlying fluid velocity $\mathbf{u}_{\text{fluid}}$ itself:

$$\mathbf{v}_{\text{ext}}(\mathbf{x}, t) = \mathbf{u}_{\text{fluid}}(\mathbf{x}, t) \ . \tag{21}$$

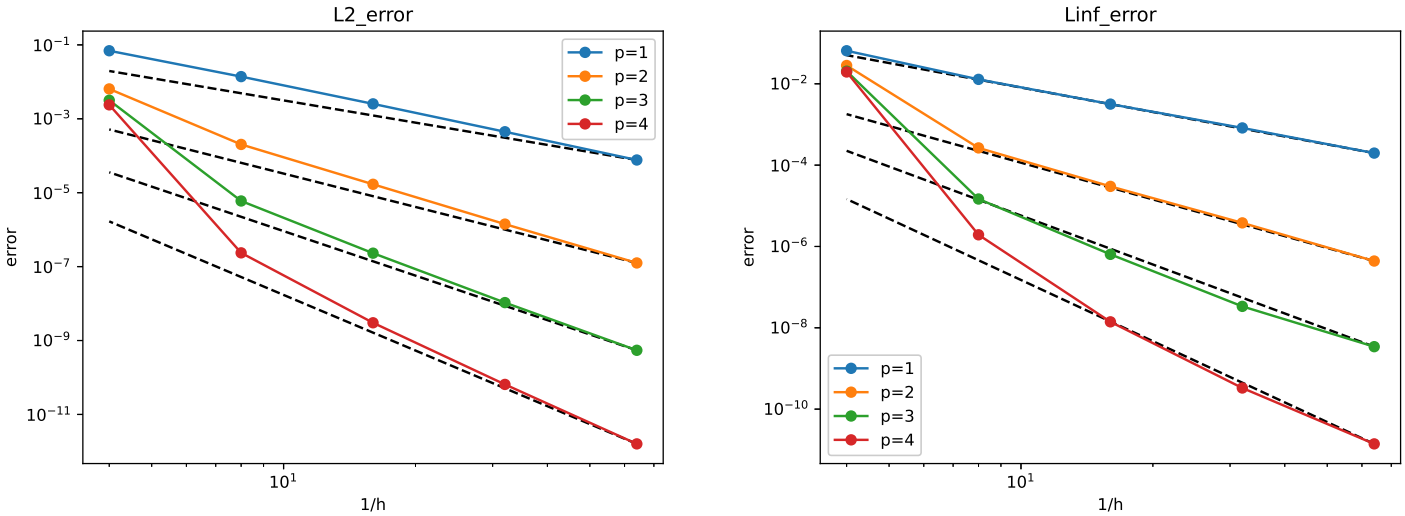
Algorithm 1 Reinitialization by high-order computation of the closest point $\mathbf{x}_{\Gamma,\mathrm{cp}}$ on the iso-0 of the level-set φ for each target mesh point $\mathbf{x}_q$. For each target point, the process is initialized with the closest point in the cloud of sampling points `cloudPts` that is found by a search tree `searchTree`. The reinitialized level-set field $\varphi^{\mathrm{reinit}}$ as well as the set of exact closest points are returned.

procedure ComputeClosestPoint(φ, `cloudPts`, `searchTree`)
  Loop on flagged elements Ω_m ▷ Full domain or narrow-band
    Loop on target points $\mathbf{x}_q$ ▷ Interpolation or quadrature points
      $\mathbf{x}_{\Gamma,\mathrm{cp}}^{\mathrm{guess}} \leftarrow$ `searchTree`$(\mathbf{x}_q,$ `cloudPts`$)$ ▷ Search tree algorithm from [14]
      $\mathbf{x}_{\Gamma,\mathrm{cp}}^{*}$, Success $\leftarrow$ `ComputeCP`$(\mathbf{x}_q, \mathbf{x}_{\Gamma,\mathrm{cp}}^{\mathrm{guess}}, \varphi)$ ▷ Eq. (17)
      If Success **do** ▷ Convergence of method & $\mathbf{x}_{\Gamma,\mathrm{cp}}^{*} \in \hat{\Omega}$
        $\mathbf{x}_{\Gamma,\mathrm{cp}} \leftarrow \mathbf{x}_{\Gamma,\mathrm{cp}}^{*}$
      Else do ▷ Non-convergence of method or $\mathbf{x}_{\Gamma,\mathrm{cp}}^{*} \notin \hat{\Omega}$
        Loop cut adjacent elements of $\Omega_m = \Omega_m^{\mathrm{adj}}$
          localTree $\leftarrow$ `CreateSearchTree(localCloudPts)` ▷ Sampling cloud points in Ω_m^{adj}
          $\mathbf{x}_{\Gamma,\mathrm{cp}}^{\mathrm{guess}} \leftarrow$ `searchTree`$(\mathbf{x}_q,$ `localCloudPts`$)$ ▷ Search tree algorithm from [14]
          $\mathbf{x}_{\Gamma,\mathrm{cp}}^{*}$, Success $\leftarrow$ `ComputeCP`$(\mathbf{x}_q, \mathbf{x}_{\Gamma,\mathrm{cp}}^{\mathrm{guess}}, \varphi^{\mathrm{adj}})$ ▷ Eq. (17)
          If Success **do** ▷ Convergence of method & $\mathbf{x}_{\Gamma,\mathrm{cp}}^{*} \in \hat{\Omega}$
            $\mathbf{x}_{\Gamma,\mathrm{cp}} \leftarrow \mathbf{x}_{\Gamma,\mathrm{cp}}^{*}$
            break
        end loop
      end do
      $\varphi^{\mathrm{reinit}} \leftarrow \mathrm{sign}(\varphi)\ \left\|\mathbf{x}_{\Gamma,\mathrm{cp}} - \mathbf{x}_q\right\|$ ▷ Eq. (3)
    end loop
  end loop
  return $\mathbf{x}_{\Gamma,\mathrm{cp}}$, $\varphi^{\mathrm{reinit}}$
end procedure



(a) Initial field with the iso-0 contours **(b) Reinitialied field with some iso-contours on the narrow-band**

Fig. 1 ***Dual-circle reinitialization test case*** **- Level-set fields with the underlying grid**



(a) L2-norm error measure **(b) L-inf norm error measure**

Fig. 2 ***Dual-circle reinitialization test case*** **- Spatial convergence study for different interpolation orders p and mesh sizes h. The error is computed from the analytical reinitialized level-set function. The rates of convergence tend to the theoretical rates $p+1$ (dashed lines).**

Doing so, a (very) frequent reinitialization is needed as the level-set function will quickly loose its signed-distance property. In many applications however, the velocity field $\mathbf{v}_{\text{ext}}$ with which the level-set is advected makes sense only at the interface. In this case, a dedicated technique to build the extended velocity in the domain from the interface is required. This technique should be designed such that the extended velocity matches the interface velocity (as specified by Eq. (6)). Another desirable feature is that it moves the neighbouring level-sets in such a way that the signed distance property is preserved. This can be achieved by extrapolating the interface velocity in the domain, i.e. for each grid point, the value of the interface velocity at the closest point on the front is used as the extension velocity at that point, as suggested in [16]:

$$\mathbf{v}_{\text{ext}}(\mathbf{x},t) = \mathbf{v}(\mathbf{x}_{\Gamma,\text{cp}},t)\ . \tag{22}$$

It can be shown that Eq. (22) is equivalent to the solution of the following equations:

$$\begin{aligned} \nabla \mathbf{v}_{\text{ext}}(\mathbf{x},t) \cdot \nabla \varphi(\mathbf{x},t) = 0 \qquad & \mathbf{x} \in \Omega \,, \\ \mathbf{v}_{\text{ext}}(\mathbf{x},t) = \mathbf{v}_\Gamma(\mathbf{x},t) \qquad & \mathbf{x} \in \Gamma \,, \end{aligned} \tag{23}$$

which ensures that the extended velocity, starting from the interface, remains constant along the normal direction. One the most common (and fastest) method to solve Eq. (23) directly is based on the Fast Marching Method and was proposed in [17, 18]. Finally, it is also possible to reformulate Eq. (23) as a hyperbolic ([19]) or elliptic ([20]) PDE. It can be shown that under the extended velocity field obtained either from Eq. (22) or Eq. (23), the level set function remains a signed distance function for all times, assuming that both $\mathbf{v}_{\text{ext}}$ and φ are smooth. Thus, at least theoretically, extending the velocity in this way would eliminate the need for reinitialization, and hence improve the accuracy of the resulting computation. In practice however, reinitizalization is still needed because numerical diffusion and approximated polynomial representation of the level-set function may degrade the signed-distance property, as pointed out in [21]. Finally, one can emphasize the importance of having an accurate and consistent velocity extension procedure in order to guarantee negligible mass loss and avoid any spurious interface deformations.
In this work, a direct geometrical approach based on Eq. (22) is used as it naturally fits into the framework developed for the reinitialization described in Sec. II.D.

2. Implementation

Since the velocity is extended into the computational domain based on Eq. (22), the position of the closest point on the interface computed during the reinitialization is first retrieved for each target mesh point. The normal to the interface at the closest point position is then determined by evaluating the gradient of the level-set, Eq. (2). The solution from the governing fluid equations is then interpolated onto the closest point on the interface with the help of the shape functions and the nodal values in the element to which this closest point belongs. From these quantities, the velocity on the iso-0 can be entirely determined and then extended to the target mesh point. More precisely, three different approaches are available to compute the interface velocity $\mathbf{v}_\Gamma \triangleq \mathbf{v}(\mathbf{x}_{\Gamma,\text{cp}})$ from the fluid states in case of two-phase flow configurations:

- *Using fluid velocities without phase transition*: in the absence of phase transition phenomena, the fluid normal velocity components must be continuous across the interface. As explained in [22], this condition is only weakly enforced in the fluid solver (through the use of the XDGM). Thus the two velocity components might not be perfectly equal, motivating the computation of the interface velocity from a density average:

$$\mathbf{v}_\Gamma = \frac{\rho_{1,\Gamma}\mathbf{v}_{1,\Gamma} + \rho_{2,\Gamma}\mathbf{v}_{2,\Gamma}}{\rho_{1,\Gamma} + \rho_{2,\Gamma}} \,, \tag{24}$$

  where $\rho_{i,\Gamma}$ and $\mathbf{v}_{i,\Gamma}$ are the density and the velocity vector of the i^{th}-phase at the interface, respectively.
- *Using fluid velocities with phase transition*: when phase transition takes place, there is a net mass flux across the interface and the velocities are not continuous anymore. Starting from the definition of the mass flux $\dot{m}_\Gamma$ across the interface and assuming the continuity of tangential velocities, the interface velocity computed from the i^{th}-phase is given by

$$\mathbf{v}^*_{i,\Gamma} = \mathbf{v}_{i,\Gamma} - \frac{\dot{m}_\Gamma}{\rho_i}\mathbf{n}_\Gamma \,, \tag{25}$$

  which leads to the definition of a density-averaged interface velocity, as done in [21]:

$$\mathbf{v}_\Gamma = \frac{\rho_{1,\Gamma}\mathbf{v}^*_{1,\Gamma} + \rho_{2,\Gamma}\mathbf{v}^*_{2,\Gamma}}{\rho_{1,\Gamma} + \rho_{2,\Gamma}} \,. \tag{26}$$

  In order to close the problem, a model for the mass flux needs to be provided and typically depends on the solutions in both phases $\mathbf{U}_{i,\Gamma}$:

$$\dot{m}_\Gamma = \dot{m}^{\text{model}}_\Gamma(\mathbf{U}_{1,\Gamma}, \mathbf{U}_{2,\Gamma}) \,. \tag{27}$$

- *Using a multiphase Riemann solver*: to impose the jumps conditions coupling the solutions from both phases at the interface, a multiphase Riemann solver, described in our previous work [1], is used in the context of the XDGM. This solver, in addition of providing the fluxes to be prescribed at the interface, also returns the interface velocity (with or without phase transition involved). Calling this solver also during the level-set velocity extension is another option, even if this is supposed to give a similar velocity as the one obtained from density averaging since the fluid states used in the averaging result from the fluxes obtained by the multiphase Riemann solver.

More information about when the fluid velocities are transferred to the level-set solver are given in Sec. III.C.2.

3. General algorithm and additional remarks

Algorithm 2 gives the main steps of the velocity extension procedure discussed before. It is preceded by the following remark:

- *Interpolation and quadrature points*: similarly to the reinitialization, the extended velocity can be computed either at the interpolation points or at the quadrature points inside the domain. For the quadrature points, the velocity can directly be injected into the advection equation since the DG discretization needs the values at the quadrature points to perform the integrals of the variational formulation. If the interpolation points are chosen, the velocity field is interpolated to the volume and face quadrature points in a second step.

Algorithm 2 Extension of the velocity of the iso-0 of the level-set φ to each target mesh point $\mathbf{x}_q$. The velocity of a point in the domain is taken equal to the velocity at the closest point on the interface $\mathbf{x}_{\Gamma,\text{cp}}$, determined from the provided fluid states $\mathbf{U}^{\text{fluid}}$. The extended velocity field over the domain $\mathbf{v}_{\text{ext}}$ is returned.

procedure ComputeExtendedVelocity($\varphi, \mathbf{x}_{\Gamma,cp}, \mathbf{U}^{\text{fluid}}$)
  Loop on flagged elements Ω_m ▷ Full domain or narrow-band
    Loop on target points $\mathbf{x}_q$ ▷ Interpolation or quadrature points
      $\mathbf{U}_{\Gamma,\text{cp}} = \sum_{i=1}^{\text{nbIP}} \phi_i(\mathbf{x}_{\Gamma,\text{cp}})\, \mathbf{U}_i^{\text{fluid}}$ ▷ Interpolate fluid states to CP on interface
      $\mathbf{v}_{q,\text{ext}} \leftarrow$ `getVelocity`($\mathbf{U}_{\Gamma,\text{cp}}$) ▷ Eqs. (24) or (26) or Riemann solver
    end loop
  end loop
  return $\mathbf{v}_{\text{ext}}$
end procedure

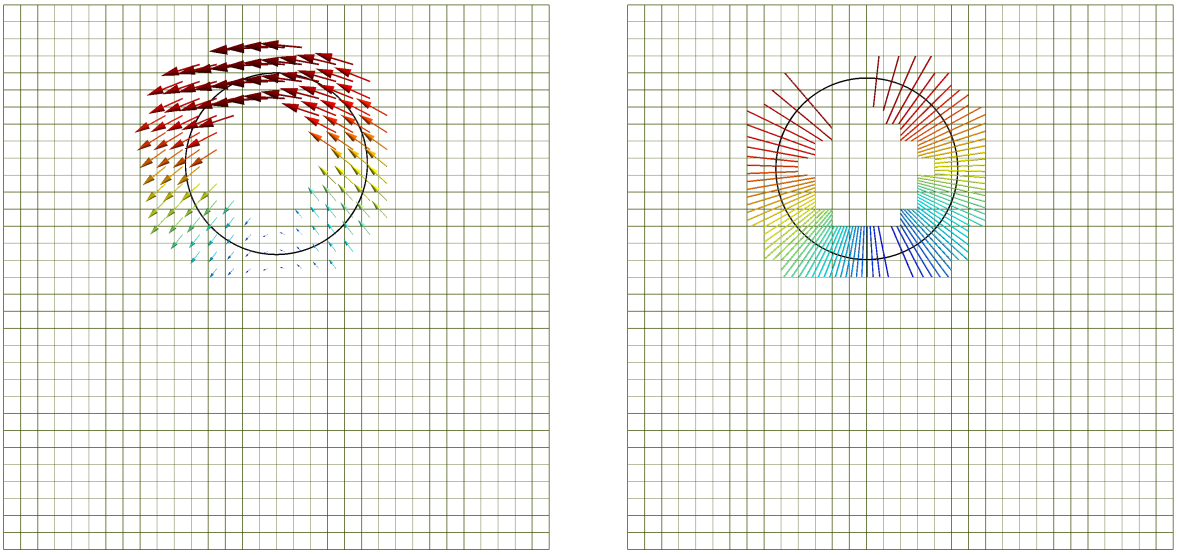
4. Verification test case

Four different error measures are used in this section:

- L_2 norm error: it measures the difference between the analytical and numerical solutions summing over the elements inside the narrow-band
$$E_{L^2} = \sum_{\Omega \in NB} \int_\Omega (\varphi^h - \varphi^{\text{exact}})^2 \, \mathrm{d}V \, .$$
- Signed distance error: it measures the degree to which the numerical solution satisfies the Eikonal equation, i.e. the signed distance property of the level-set field
$$E_{SD} = \sum_{\Omega \in NB} \int_\Omega (|\nabla \varphi^h| - 1)^2 \, \mathrm{d}V \, .$$
- Interface error: it measures the deviation of the numerical iso-0 level-set contour with respect to the analytical one and it is evaluated by integrating the computed solution along the analytical position of the iso-0
$$E_{Int} = \int_{\Gamma(\varphi^{\text{exact}})} (\varphi^h)^2 \, \mathrm{d}S \, .$$
- Volume error: it measures the volume loss, in percentage, of the numerical level-set field from an analytical solution
$$E_{Vol} = \frac{\left| \int_{\Gamma(\varphi^{\text{exact}})} \mathrm{d}V - \int_{\Gamma(\varphi^h)} \mathrm{d}V \right|}{\int_{\Gamma(\varphi^{\text{exact}})} \mathrm{d}V} \times 100 \, .$$

Rotating circle The first test case chosen here comes from [22]. It is about the motion of a circular contour of radius 1 centered initially at $(0.0, 1.25)$ in a flow field which is rotating around the point $(0.0, 0.0)$ outside the circle. The velocity field has an angular velocity equal to 1 and is given in Cartesian coordinates by
$$\mathbf{v}(\mathbf{x}) = (-y, x) \, .$$
The problem is discretized on the domain $\Omega = (-3, 3)^2$ on structured meshes with different mesh sizes. The implicit multi-step ESDIRK43 time-stepping scheme is chosen with $\Delta t = 5e-4$ and the simulations are run until $t = 0.3$. The

exact solution can be found using standard rotational transformation depending on time. No reinitialization is performed to test the ability of the extension procedure to preserve the initial signed distance property of the level-set function. In Fig. 3a, the initial velocity field and the position of the iso-0 are represented. In Fig. 3b, the iso-contours of the level-set over the narrow-band at the final time are drawn as well as the iso-contours of the extended velocity field. As it can be observed, the velocity iso-contours are straight and normal to the iso-0, indicating that the velocity of the iso-0 is correctly extended to the mesh points. This is confirmed on the convergence graphs in Fig. 4 where the L_2 and the signed distance errors between the analytical expression of the level-set at the final time and the computed solution follow the expected trend.



(a) Initial configuration with the prescribed velocity field on the narrow-band

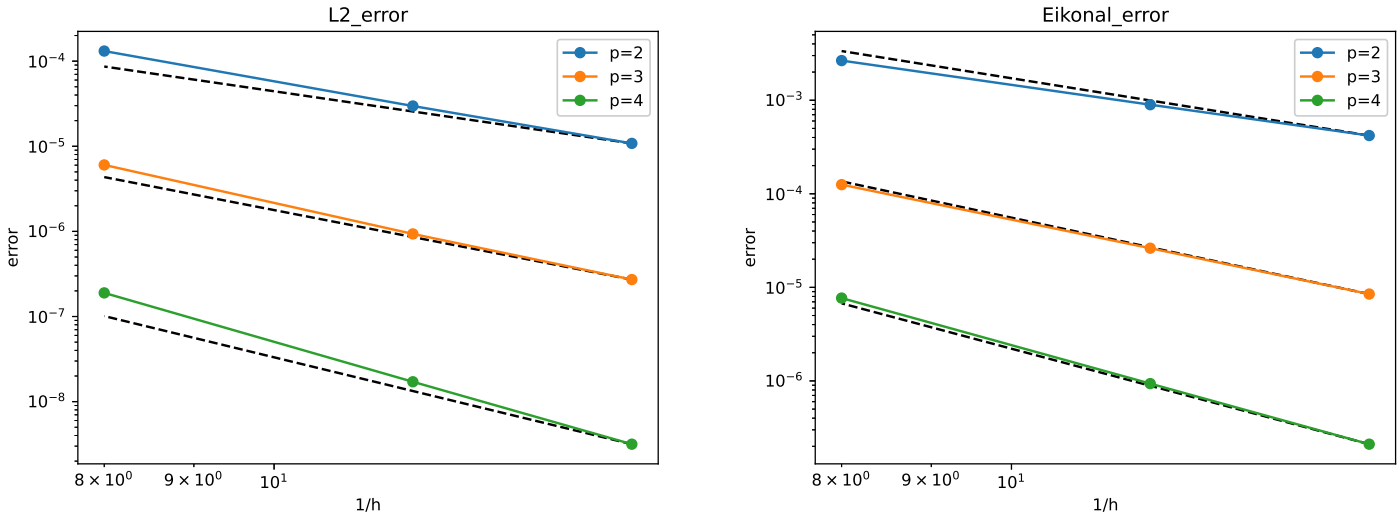
(b) Final configuration with the extended velocity iso-lines, perpendicular to the iso-0

Fig. 3 ***Rotating circle test case*** **- Extended velocity field, position of the iso-0 contour and underlying structured mesh.**

Circle sheared by a channel flow This test case is inspired from [23]. It involves an initially circular interface placed in a channel. Under the effect of a Poiseuille flow velocity profile, the interface is convected and deformed by the external velocity field. The value of this velocity field is extended from the iso-0 to the mesh points such that the iso-contours maintain a signed distance arrangement. An analytical expression of the level-set as a function of time can be found. The rectangular domain is discretized by an unstructured mesh containing 3808 quadrilateral elements. The implicit multi-step ESDIRK43 time-stepping scheme is chosen with $\Delta t = 1.25e-3$ and the simulations is run until $t = 1.25$. Reinitialization is applied every 100 time steps. An interpolation order of $p = 3$ is selected. The iso-0 level-set contours at different times along the simulation are drawn in Fig. 5. One can observe that the iso-0 contours are globally undergoing the expected deformation. However, one can notice some small undesired instabilities appearing in the elongated regions. This was concluded that they result from the inaccuracies of the closest point computation procedure for some of the points in this area. The high curvature of the level-set function there is indeed making the process more prone to failure. This also impacts the L_2 norm and the signed distance error measures listed in Table 1. The volume conservation remains however acceptably low.

Circle deformed by a vortex in a box This last test case, which is a standard benchmark, involves the deformation of an initially circular interface due to to a prescribed velocity field corresponding to a periodic swirling flow. The divergence free initial velocity field is defined as

$$u_x(x,y) = -\sin^2(\pi x)\sin(2\pi y)$$
$$u_y(x,y) = \sin^2(\pi y)\sin(2\pi x)$$



(a) L2-norm error measure

(b) Signed distance error measure

Fig. 4 ***Rotating circle test case*** **- Spatial convergence study for different interpolation orders p and mesh sizes h. The error is computed from the analytical level-set function at the final time $t = 0.3$ after 600 iterations. The rates of convergence tend to the theoretical rates $p + 1$ (dashed lines).**

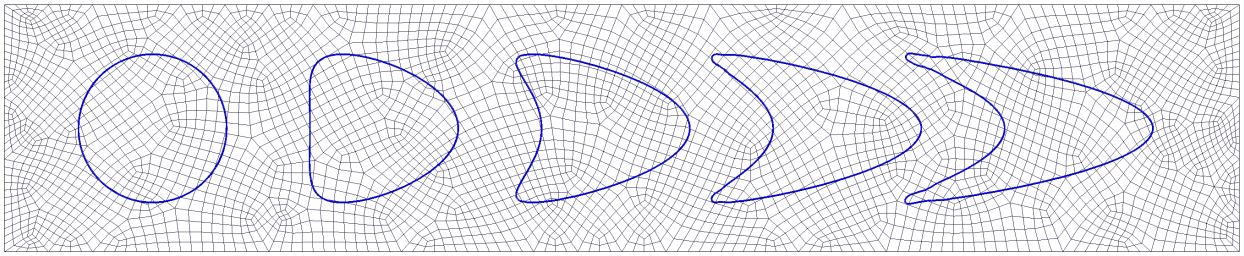


Fig. 5 ***Channel flow test case*** **- snapshots of the iso-0 of the level-set at time $t = 0, 0.5, 0.75, 1, 1.25$, being advected and deformed by a fully developed channel flow velocity profile.**

The interface is stretched by the vortical velocity field and develops filaments around the vortex center. The velocity field is reversed at $t = 1$ and it should recover its original shape at the final time $t = 2$. The $[0, 1] \times [0, 1]$ square domain is discretized by an unstructured mesh containing 1560 quadrilateral elements. The implicit multi-step ESDIRK43 time-stepping scheme is chosen with $\Delta t = 1.0e - 3$. Reinitialization is applied every 100 time steps. An interpolation order of $p = 4$ is selected. The iso-0 level-set contours at different times along the simulation are drawn in Fig. 6. The interface stretches and goes back to its initial shape as expected. However, some small undesired instabilities can be first observed in the filament section at maximum deformation just before the velocity field reverses. Those instabilities are then transported during the reversal phase and are still present on the final interface shape. The same conclusion as for the previous test case can be drawn, i.e. the failure of the closest point computation introduces inaccuracies which manifest in spurious interface oscillations. A possible remedy to this problem is discussed in the perspectives of this work. Looking at the error measures in Table 1, one can say that they remain relatively low, even if the volume conservation is not as good as in the previous test case.

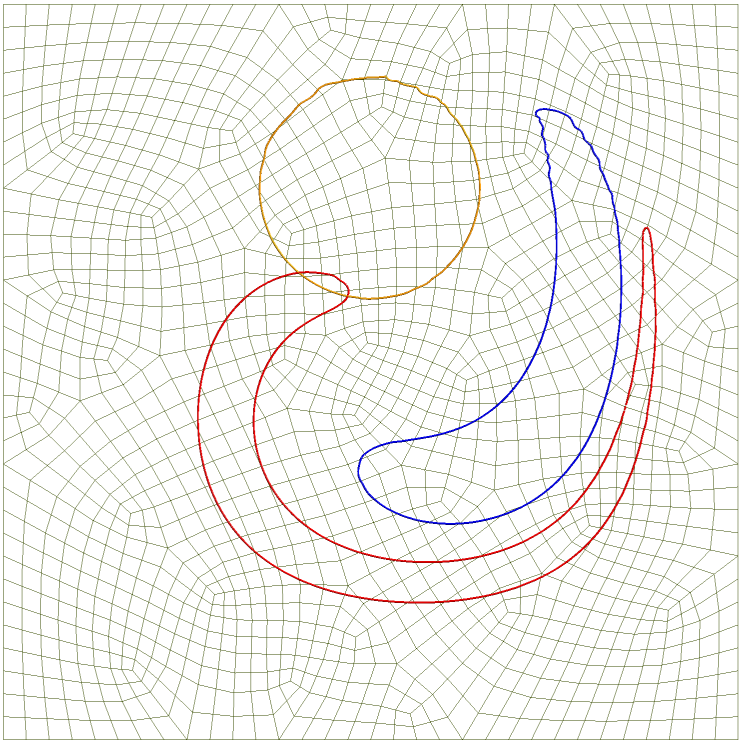


Fig. 6 ***Vortex-in-a-box test case*** **- snapshots of the iso-0 of the level-set at time** $t = 1.0$ **(red),** 1.5 **(blue), and** 2.0 **(orange), being stretched by a reversing vortical velocity field.**

	E_{L^2}	E_{SD}	E_{Int}	E_{Vol} (%)
Channel flow	$1.2 \cdot 10^{-2}$	$3.4 \cdot 10^{-2}$	$9.6 \cdot 10^{-7}$	0.69
Vortex-in-a-box	$4.0 \cdot 10^{-4}$	$7.6 \cdot 10^{-3}$	$1.5 \cdot 10^{-6}$	3.52

Table 1 Values of the different error measures considered for the two last extension velocity test cases

F. Narrow-band strategy

1. Some background

So far its was implicitly assumed that the advection, reinitialization and extension operations associated to the level-set were performed on the entire computational domain, in the same way as the fluid equations are solved. Even if by construction the level-set function is defined over the d-dimensional domain, its sole purpose is to provide information about the $d-1$-dimensional interface to the fluid solver, namely its position, its speed and its geometrical quantities. Thus advecting the level-set iso-contours in the entire domain represents an unnecessary effort, only those near the zero iso-contour corresponding to the interface need to be updated. This observation led to the development of narrow-band level-set methods, first introduced in [24] and studied extensively in [25]. The width of the narrow-band is problem-dependent but the following remark formulated in [26, §3.2.2] holds in general. Whether for reasons of accuracy, stability or both, the interface motion during one time step is expected to remain confined in the elements cut by the interface at the current time and their close neighbours. Thus, the operations that determine the level-set evolution can be limited to these elements, forming a so-called narrow-band around the iso-0 contour. As indicated also in [26], computational efficiency is not the only reason to adopt narrow-banding. Because of the signed-distance property of the level-set that needs to be maintained, there will always be some areas in the domain where the level-set function is singular. In such regions, the solution to the closest point problem seen before is not unique. Moreover, evaluating level-set derivatives (to obtain the normal e.g.) will produce oscillatory results that may lead to the failure of the method. If the mesh is sufficiently refined, it is possible to keep those singularities regions outside of the narrow-band and

therefore avoid any issues.

2. Implementation

Figure 7 illustrates the narrow-band concept and the notations used in the following. During the initialization and at the end of each time step, the building of the narrow-band is carried out as follows:

1) *Flagging of cut elements* $\Omega_{\text{NB, cut}}$: a loop on all the elements is done and on each on them, the nodal degrees of freedom of the level-set solution are collected. If the minimum and maximum values among the interpolation points values are of opposite sign, the element is flagged as cut;
2) *Flagging cut-neighbours elements* $\Omega_{\text{NB, neigh}}$: four kinds of neighbours elements can be flagged: (a) the elements sharing a vertex with one of the cut elements (single layer), (b) the ones sharing an edge with one the cut elements (single layer), (c) the ones sharing a vertex with one of the elements belonging to category (a) (double layer), and (d) the ones sharing an edge with one of the elements belonging to category (b) (double layer). For categories (a) and (b), a loop on the elements flagged as cut is done and on each of them, either the vertices (a) or the edges (b) defining the element are collected. Based on that, the elements can be flagged. For the double layer categories (c) and (d), the elements satisfying the criteria of categories (a) or (b), respectively, are first collected. Then a loop is done on them and the same flagging process is applied (checking for each element that it has not been added yet to the list of course);
3) *Assembling of narrow-band elements* Ω_{NB}: the narrow-band elements are simply flagged by combining the cut elements and their neighbours: $\Omega_{\text{NB}} = \Omega_{\text{NB, cut}} \cup \Omega_{\text{NB, neigh}}$. Those elements will be the ones on which all the previously explained operations will be performed;
4) *Flagging of far elements* $\Omega_{\text{NB, far}}$: the elements outside of the narrow-band are directly obtained by subtracting the narrow-band elements from the global element set: $\Omega_{\text{NB, far}} = \Omega \setminus \Omega_{\text{NB}}$;
5) *Flagging of appearing elements in the narrow-band* $\Omega_{\text{NB, app}}$: the elements that appear in the narrow-band at the new time step are obtained by storing the narrow-band elements set over two time levels. They can then be readily determined by combining the two sets together: $\Omega_{\text{NB, app}} = \Omega^{t}_{\text{NB}} \setminus \Omega^{t-1}_{\text{NB}}$. As the level-set in those elements was kept unmodified during at least one time step, reinitialization must be applied on them to generate a level-set field consistent with the other elements inside the narrow-band;
6) *Flagging of internal elements interfaces* $I_{\text{NB, int}}$: a loop on the narrow-band elements interfaces is done. For each interface, the element indices on the right and the left are retrieved. If both elements belong to the previously built narrow-band region, the interface is added to the set. It is then on this set of internal interfaces that the interfacial fluxes of the DGM will be integrated;
7) *Flagging of boundary elements interfaces* $I_{\text{NB, bnd}}$: the same approach as for internal interfaces is followed, except that for each interface, we check if one of the two elements on each side does not belong to the narrow-band. If it is the case, the interface is added to the list. It is on this boundaries set that the homogeneous Neumann boundary conditions for the level-set, normally prescribed on the domain boundaries, will be imposed.

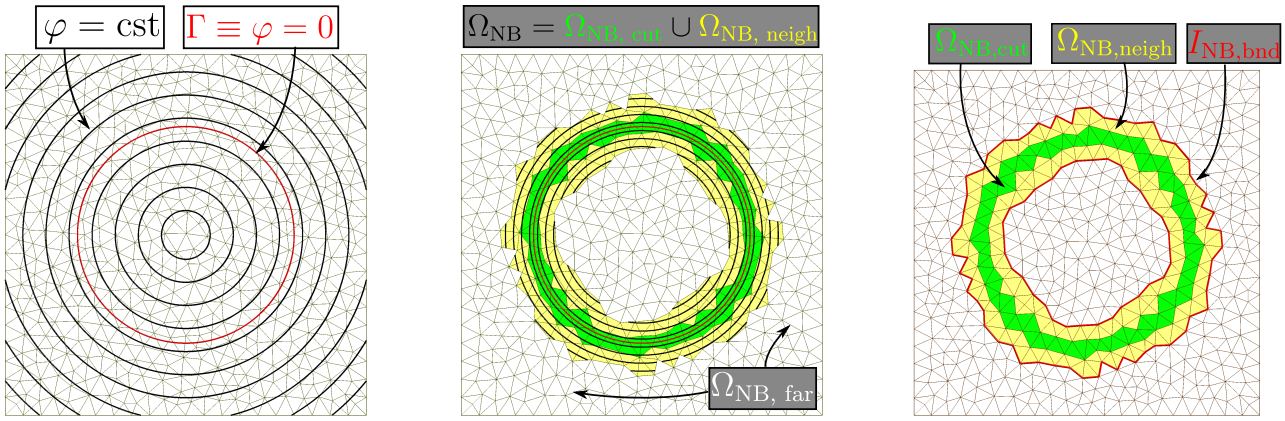
The building and update of this narrow-band along a simulation requires therefore efficient dynamic data structures and the possibility of accessing those data anywhere in the code.

G. General algorithm combining advection, reinitialization and extension on a narrow-band

In this last section, the various methods developed for the level-set are linked through two algorithms. Algorithm 3 summarizes the main steps of the reinitilization and extension procedures and is depicted by a sequence of sketches in Fig. 8. Algorithm 4 gives the complete overview of the steps involved in the advection of the level-set field over one time step and will be referred to in the second part.

Algorithm 3 Main steps for the reinitialization and extension of the velocity based on a direct geometric approach. It takes as inputs the initial level-set field φ and the solution from the fluid solver $\mathbf{U}^{\text{fluid}}$

procedure REINITEXTEND(φ, $\mathbf{U}^{\text{fluid}}$)
  cloudPts ← `SampleLevelSet`(φ) ▷ Generate cloud of points sampling the LS: Sec. II.F.2
  searchTree ← `CreateSearchTree`(cloudPts) ▷ Create search tree from the cloud of points
  $\mathbf{x}_{\Gamma,\text{cp}}, \varphi^{\text{reinit}}$ ← `ComputeClosestPoint`(φ, cloudPts, searchTree) ▷ Compute closest points ∀ nodes: Alg. 1
  $\mathbf{v}_{\text{ext}}$ ← `ComputeExtendedVelocity`($\varphi, \mathbf{x}_{\Gamma,\text{cp}}, \mathbf{U}^{\text{fluid}}$) ▷ Compute extended velocity ∀ nodes: Alg. 2
end procedure



(a) Level-set defined and computed on the entire domain.

(b) Restriction to a narrow-band region Ω_{NB} around the iso-0. Elements in the far region $\Omega_{\text{NB, far}}$ are not considered.

(c) Narrow-band composed of the elements cut by the iso-0 $\Omega_{\text{NB, cut}}$, their neighbhors $\Omega_{\text{NB, neigh}}$, and the elements interfaces I_{NB} at the border of Ω_{NB}.

Fig. 7 Illustration of the narrow-band concept.

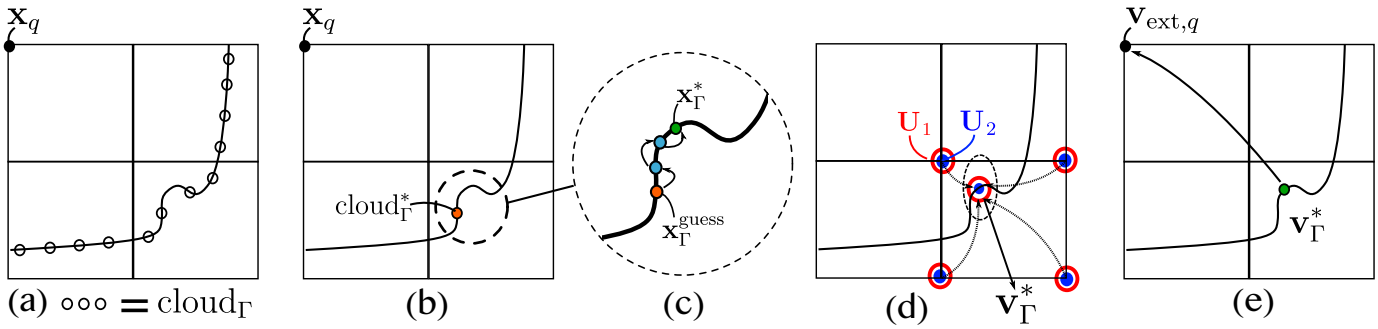


Fig. 8 Illustration of Algorithm 3. (a) sampling of level-set by cloud of points with recursive contouring algorithm, (b) closest point among the cloud w.r.t to a given mesh point x_q found by search tree algorithm, (c) point used as a guess for a non-linear constrained minimization problem leading to the exact closest point x_Γ^* on the interface, (d) interpolation of the fluid states from both sides of the interface to x_Γ^* and computation of the interface velocity $\mathbf{v}_\Gamma^*$, (e) extension of the velocity to the mesh point

III. Coupling between a DG level-set and an extended DG solvers

In this second part, the elements that are needed to apply the cut-cell XDG solver to moving interface problems are presented. The first section aims at introducing the method, in particular the assembly of the residuals, through an illustrative example. Starting from there, the use of the cell-agglomeration strategy is justified to handle the time integration on evolving cut-cell topologies. A test case is then used to show the performances but also limitations of the current approach. Finally, the coupling of the flow and level-set solvers is discussed and illustrated by a two-phase flow simulation.

A. The eXtended DGM and the residuals assembly

1. Some background

The eXtended Discontinuous Galerkin Method (XDGM) belongs to the class of unfitted / cut-cells sharp interface methods and allows to capture arbitrary discontinuities without suffering any degradation of the solution in the interface vicinity. Since it does not imply any mesh deformation, its application to transient interface problems exhibiting complex interface dynamics is particularly appealing. The purpose of this manuscript is not to describe in a detailed and rigorous manner the XDGM. The interested reader can refer to [27, 28] which gives a good overview of some of the work

Algorithm 4 Advection of the level-set field to the next time step based on the provided fluid states $\mathbf{U}^{\text{fluid}}$. It returns the advected level-set at the next time step φ^{t+1} and a continuous level-set field $\varphi^{\text{cont},t+1}$

procedure MoveLevelSet($\varphi^t, \mathbf{U}^{\text{fluid}}$)
  $t \leftarrow t + 1$
  On Ω^t_{NB} **do** ▷ Restrict the operations on NB elements
    $\mathbf{v}^t_{\text{ext}} \leftarrow$ `ReinitExtend`(φ^t, $\mathbf{U}^{\text{fluid}}$) ▷ Algorithm 3
    $k = 0$, $\varphi^*_k \leftarrow \varphi^t$
    While $\left(k \leq k_{\max} \text{ And } \left\|\mathbf{M}^{-1}\mathcal{R}^{\text{tot}}_{\text{LS}}(\varphi^*_k, \mathbf{v}^t_{\text{ext}})\right\| > \text{tol}\right)$ **do**
      $\varphi^*_{k+1} \leftarrow$ `SolveLinearSystem`($\varphi^*_k, \mathbf{v}^t_{\text{ext}}$) ▷ Solve Eq. (12)
      $k \leftarrow k + 1$
    end do
    $\varphi^{t+1}_{\text{NB}} \leftarrow \varphi^*_k$
  end do
  $\Omega^{t+1}_{\text{NB}} \leftarrow$ `UpdateNarrowBand`(φ^{t+1}_{NB}) ▷ Sec. II.F.2
  On Ω^{t+1}_{NB} **do**
    $\varphi^{t+1,\text{reinit}}_{\text{NB}} \leftarrow$ `ReinitExtend`(φ^{t+1},/) ▷ Algorithm 3
  end do
  $\varphi^{t+1}_{\text{NB, app}} \leftarrow \varphi^{t+1,\text{reinit}}_{\text{NB, app}}$ ▷ Assign reinitialized LS field to elements appearing in NB
  If Reinitialization **do**
    $\varphi^{t+1}_{\text{NB}} \leftarrow \varphi^{t+1,\text{reinit}}_{\text{NB}}$ ▷ Assign reinitialized LS field to NB elements, when specified
  end if
  $\varphi^{\text{cont},t+1}_{\text{NB}} \leftarrow \varphi^{t+1,\text{reinit}}_{\text{NB}}$ ▷ Make a copy of the reinitialized level-set field that will be used in XDG
  return φ^{t+1}, $\varphi^{\text{cont},t+1}$
end procedure

carried out by the research group at TU Darmstadt developing the BoSSS framework. Based on this research, the XDG framework has been implemented inside the Argo platform exploited in this work and first results were presented in [29, 30].

2. An illustrative example

To present some XDG concepts in an incremental and consistent way, the basic one-dimensional situation depicted in Fig. 9 will serve as a guideline. Extension to multiple dimension is straightforward but more complex to visualize. The one-dimensional convection equation is here considered as the model equation to be discretized:

$$\frac{\partial u}{\partial t} + \nabla \cdot f(u) = 0\,, \tag{28}$$

in which u is a scalar quantity being advected by the convective flux $f(u)$. If one considers that the initial condition of this problem is discontinuous, applying a classical DG discretization based on polynomial interpolation could lead to oscillatory and divergent results. To be able to cope with this situation, the nodal degrees of freedom on the element containing the discontinuity are doubled (hence the name of "extended" DG) such that the discontinuity can be perfectly captured by two separated polynomial solutions inside the same element, as shown in Fig. 9. From here, the following index notation will be used:

$$U_{m,s,n} \quad \text{with} \qquad m = 1, ..., M\,, \quad s = \{a, b\}\,, \quad n = 1, ..., N\,,$$

is the n^{th}-degree of freedom on the m^{th}-element associated to phase s. On the elements which are not cut by the interface, s will be either phase a or b. Similarly to the level-set method (see Eq. (12)), an implicit time discretization with a dual-time stepping strategy and a damped inexact Newton solver is used, leading to the linear system to be solved at the k^{th} inner iteration:

$$\left[\frac{\mathbf{I}}{\Delta\tau} + \mathbf{M}^{-1}\frac{\partial \boldsymbol{\mathcal{R}}^{\text{tot}}}{\partial \boldsymbol{U}}(\boldsymbol{U}_k)\right](\boldsymbol{U}_{k+1} - \boldsymbol{U}_k) = -\mathbf{M}^{-1}\boldsymbol{\mathcal{R}}^{\text{tot}}(\boldsymbol{U}_k)\,. \tag{29}$$

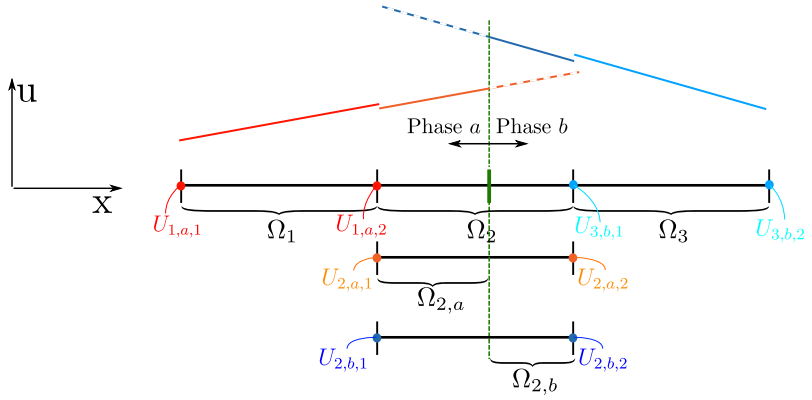


Fig. 9 1-D domain containing two phases separated by a discontinuity in the solution, here approximated by polynomials of degree one. The degrees of freedom are doubled on the element cut by the interface.

The degrees of freedom to be considered are for this example:

$$\boldsymbol{U} = \left[U_{1,a,1}\,,\ U_{1,a,2}\,,\ U_{2,a,1}\,,\ U_{2,a,2}\,,\ U_{2,b,1}\,,\ U_{2,b,2}\,,\ U_{3,b,1}\,,\ U_{3,b,2} \right]^T \,. \tag{30}$$

The most important point to discuss here is how the total global residual $\boldsymbol{\mathcal{R}}^{\text{tot}}$ is assembled. First, it is the sum of the temporal and the spatial residuals:

$$\boldsymbol{\mathcal{R}}^{\text{tot}} = \boldsymbol{\mathcal{R}}^{\text{temp}} + \boldsymbol{\mathcal{R}}^{\text{sp}} \,, \tag{31}$$

assuming that a single stage time stepping scheme for the physical time is considered[†]. The spatial residual can be further decomposed into its volume, interface and boundary components. An additional contribution coming from the residual computed on the iso-0 of the level-set is present in the cut elements:

$$\boldsymbol{\mathcal{R}}^{\text{sp}} = \begin{cases} \boldsymbol{\mathcal{R}}^{\text{vol}} + \boldsymbol{\mathcal{R}}^{\text{int}} + \boldsymbol{\mathcal{R}}^{\text{bnd}} & \text{on non-cut elements.} \\ \boldsymbol{\mathcal{R}}^{\text{vol}} + \boldsymbol{\mathcal{R}}^{\text{int}} + \boldsymbol{\mathcal{R}}^{\text{bnd}} + \boldsymbol{\mathcal{R}}^{\text{iso-0}} & \text{on cut elements.} \end{cases} \tag{32}$$

Those different residual terms can then be written by summing over the elements and the interfaces and the distinction is made between the cut and non-cut elements:

$$\boldsymbol{\mathcal{R}}^{\text{temp}},\ \boldsymbol{\mathcal{R}}^{\text{vol}} = \sum_{m=1}^{M} \begin{cases} \boldsymbol{\mathcal{R}}^{\text{temp}}_{m,s}\,,\ \boldsymbol{\mathcal{R}}^{\text{vol}}_{m,s} & \text{if } \Omega_m \text{ non-cut. } s = a \text{ or } s = b \\ \sum_{s\in\{a,b\}} \boldsymbol{\mathcal{R}}^{\text{temp}}_{m,s}\,,\ \boldsymbol{\mathcal{R}}^{\text{vol}}_{m,s} & \text{if } \Omega_m \text{ cut.} \end{cases} \qquad \Omega_m : \text{ element } m \tag{33}$$

$$\boldsymbol{\mathcal{R}}^{\text{int}} = \sum_{f=1}^{F} \begin{cases} \boldsymbol{\mathcal{R}}^{\text{int}}_{f,s} & \text{if } I_f \text{ non-cut. } s = a \text{ or } s = b \\ \sum_{s\in\{a,b\}} \boldsymbol{\mathcal{R}}^{\text{int}}_{f,s} & \text{if } I_f \text{ cut.} \end{cases} \qquad I_f : \text{ element interface } f \tag{34}$$

$$\boldsymbol{\mathcal{R}}^{\text{bnd}} = \sum_{g=1}^{G} \begin{cases} \boldsymbol{\mathcal{R}}^{\text{bnd}}_{g,s} & \text{if } B_g \text{ non-cut. } s = a \text{ or } s = b \\ \sum_{s\in\{a,b\}} \boldsymbol{\mathcal{R}}^{\text{bnd}}_{g,s} & \text{if } B_g \text{ cut.} \end{cases} \qquad B_g : \text{ domain boundary } g \tag{35}$$

$$\boldsymbol{\mathcal{R}}^{\text{iso-0}} = \sum_{m=1}^{M} \sum_{s\in\{a,b\}} \boldsymbol{\mathcal{R}}^{\text{iso-0}}_{m,s} \quad \text{if } \Omega_m \text{ cut.} \tag{36}$$

The expressions per element/interface/boundary for each contribution to the total residual are finally given for the model problem Eq. (28), assuming for simplicity a backward Euler scheme for the time discretization:

$$\boldsymbol{\mathcal{R}}^{\text{temp}}_{m,s} = \mathbf{M}_m \frac{\boldsymbol{U}^{t+1}_{m,s} - \boldsymbol{U}^{t}_{m,s}}{\Delta t} \,, \tag{37}$$

[†]For multi-stage time stepping schemes, the contributions coming from the spatial residuals of the previous stages should also be added.

$$\mathcal{R}_{m,s}^{\text{vol}} = -\int_{\Omega_{m,s}} \nabla \boldsymbol{\phi}_m \cdot f(u_{m,s}^h)\,\mathrm{d}V\,, \tag{38}$$

$$\mathcal{R}_{f,s}^{\text{int}} = \int_{I_{f,s}} [\boldsymbol{\phi}_f]\,\mathcal{F}(u_{m,s}^{h,+}, u_{m,s}^{h,-})\,\mathrm{d}S\,, \tag{39}$$

$$\mathcal{R}_{g,s}^{\text{bnd}} = \int_{B_{g,s}} \boldsymbol{\phi}_g\,\mathcal{F}(u_{m,s}^{h,+}, u^{\text{BC}})\,\mathrm{d}S\,, \tag{40}$$

$$\mathcal{R}_{m,s}^{\text{iso-0}} = \int_{\Gamma_m} \boldsymbol{\phi}_m\,\mathcal{F}_{\Gamma,s}(u_{m,s}^h, u_{m,s^*}^h)\,\mathrm{d}S\,, \quad s^* = b \text{ if } s = a\,,\ s^* = a \text{ if } s = b\,, \tag{41}$$

where $\mathcal{F}$ is a numerical flux function, single-valued across element edges and domain boundaries, double-valued across the phase interface.

B. Cell-agglomeration to the rescue of time integration for moving interface problems

1. Some background

As explained in [22, §3.5.2], although the underlying mesh remains fixed in the XDGM, the domains occupied by each phase are time-dependent. This poses some conceptual issues for the time discretization of the spatially discretized equations with classical time-stepping schemes. This observation holds true for any unfitted methods applied to transient problems. Indeed, classical time-stepping schemes aim at calculating the solution at the next time step based on the solution(s) previously computed. Because the topology of the cut-mesh is evolving with time and that the spatial approximation is adapted on cut elements, combining the solutions from different time levels may become problematic. This is exemplified in the next section. In this context, several approaches have been designed:

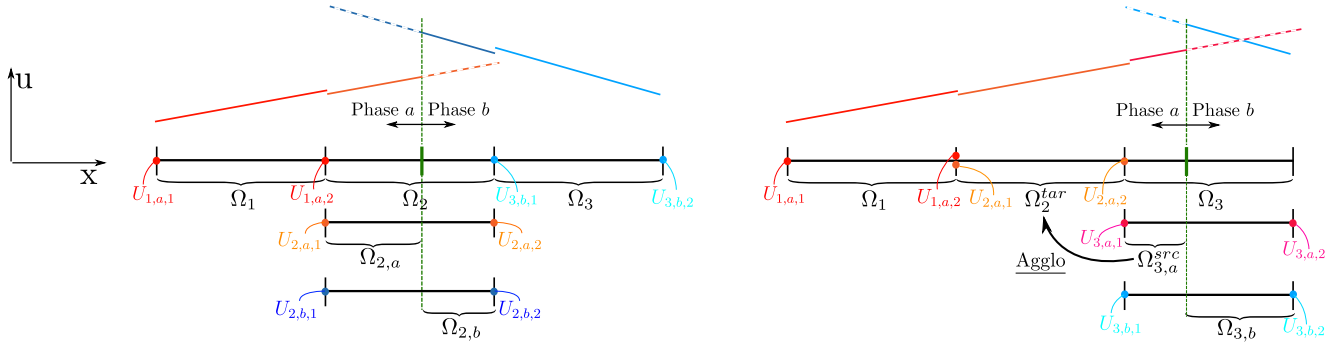
- *Space-time domain*: instead of applying separated space and time discretizations, time is considered as an additional dimension. A simultaneous discretization of spatial and temporal variables is done using basis functions depending on both space and time. This approach provides the most accurate results at the cost of involving much larger systems and a complex implementation in an existing FE or DG code framework;
- *Rothe method*: this technique consists in discretizing the equations first in time and then in space. When solving for a discrete approximation to the new time step, all terms are then evaluated at the corresponding new time level [31, §3.6.3], which avoids the compatibility issues when combining solutions on different cut-cell topologies. This is however not easy to apply because the integration of functions defined at different time levels must be handled, as explained in [32];
- *Splitting approach*: sticking to the more traditional approach of first discretizing in space and then using a common time-stepping scheme, the easiest approach is to split the interface evolution from the governing equations. One can therefore integrate in time the equations considering a fixed interface and then update the interface position in a second step. This however does not solve the issue of combining solutions on evolving cut-mesh topologies. That is why interpolation between time levels is needed [33, §6.2]. In this work, we follow the approach proposed in [34] that rely on a cell-agglomeration technique, explained just after;
- *Moving interface approach*: because of its simplicity, the splitting approach is appealing. However it limits the time accuracy of the global coupled problem, as discussed in [34]. A method that allows to regain the theoretical temporal convergence rate without the difficulties of the two first approaches is proposed in [34] and will be the subject of future investigations, as discussed in the perspectives of this work.

2. The issue of appearing elements in the splitting approach

The previous section emphasized the difficulties associated to the temporal integration in unfitted / cut-cell methods applied to moving interface problems. In the context of the splitting approach adopted in this work the guideline test case is used to illustrate the issues one may face. One considers the situation in Fig. 10 where the cut-cell topology changes between two time levels. Following the steps presented in Sec. III.A, the assembly of the temporal residual at time $t+1$ for the solution of phase a in the appearing element Ω_3 would be computed by:

$$\mathcal{R}_{3,a}^{temp,t+1} = \mathbf{M}_{3,a}\frac{\boldsymbol{U}_{3,a}^{t+1} - \overbrace{\boldsymbol{U}_{3,a}^{t}}^{?}}{\Delta t}\,. \tag{42}$$

However, assembling the temporal residual in this way would lead to an inconsistency because there exists no solution associated to phase a in Ω_3 at the previous time t. The solution $\boldsymbol{U}_{3,a}^{t+1}$ should therefore be combined with an adjacent element containing the solution associated to phase a at the previous time. That is the idea behind the cell-agglomeration procedure, explained in the next section.



(a) Time step t: element 2 is cut by the interface

(b) Time step t + 1: element 3 is cut by the interface at the new time level

Fig. 10 Because of the change in cut-cell topology between two consecutive time levels, the solution of phase a that appears in element 3, which is not present at the previous time, needs to be agglomerated to element 2.

3. Cell agglomeration technique

The cell agglomeration technique considered in this work is based on the work of [28] and [35] and its implementation in our solver was presented in [29, 30]. A step-by-step description of its application to the original XDG discretization of Sec. III.A is given below.

1) *Flagging of target-source pairs*: the first step is to identify the elements that need to be agglomerated (the "sources") and the candidates (the "targets") to perform the agglomeration. Since one considers here the agglomeration applied to temporal integration for the splitting approach[‡], the source elements are identified as the ones containing a given phase at the current time level but not at the previous one. The associated target element is then found by identifying the edge-neighbour element containing the highest volume fraction of the same phase under consideration. In the example (Fig. 10b), $\Omega_{3,a}$ is then flagged as "source" associated to the target $\Omega_{2,a}$.

2) *Coupling and agglomerated mass matrix*: next, a coupling matrix for every source-target pair is computed. It is defined as

$$\begin{aligned} Q_{i,j}^{*} &= \int_{\Omega^{\text{src}}} \phi_{\text{ext},j}^{\text{tar}} \phi_i^{\text{src}} \, \mathrm{d}V \,, \\ \mathbf{Q} &= (\mathbf{M}^{\text{src}})^{-1} \mathbf{Q}^{*} \,, \end{aligned} \tag{43}$$

where $\boldsymbol{\phi}^{\text{src}}$ are the shape functions on the source element while $\boldsymbol{\phi}_{\text{ext}}^{\text{tar}}$ is the smooth extension of the target shape functions into the source element. An example of such extension is shown in Fig. 11. By construction $\mathbf{Q}$ contains the projection of the extended target basis onto the source cell, which is very convenient because all the agglomerated operations can be computed from it. In particular, the basis of the agglomerated cell is easily obtained by:

$$\boldsymbol{\phi}^{\text{agg}} = \boldsymbol{\phi}^{\text{tar}} + \boldsymbol{\phi}^{\text{src}} \mathbf{Q} \,, \tag{44}$$

as well as the agglomerated mass matrix:

$$\mathbf{M}^{\text{agg}} = \mathbf{M}^{\text{tar}} + \mathbf{Q}^{T} \mathbf{M}^{\text{src}} \mathbf{Q} \,. \tag{45}$$

3) *Injection of agglomerated solution into original mesh*: the agglomerated solution can be obtained from the solution on the original mesh using the agglomerated coupling and mass matrices:

$$\boldsymbol{U}^{\text{agg}} = (\mathbf{M}^{\text{agg}})^{-1} \left(\mathbf{M}^{\text{tar}} \boldsymbol{U}^{\text{tar}} + \mathbf{Q}^{T} \mathbf{M}^{\text{src}} \boldsymbol{U}^{\text{src}} \right) \,. \tag{46}$$

[‡]When agglomeration is applied in the context of the moving interface approach, additional criteria must be taken into account

Since one only considers appearing elements to be agglomerated here and since the solution in the appearing element is not defined at the beginning of the new time step, Eq. (46) reduces to:

$$\boldsymbol{U}^{\text{agg}} = (\mathbf{M}^{\text{agg}})^{-1} \left(\mathbf{M}^{\text{tar}} \boldsymbol{U}^{\text{tar}}\right) \ . \tag{47}$$

By construction, the solution on the target and source elements can then be obtained as

$$\boldsymbol{U}^{\text{tar}} = \boldsymbol{U}^{\text{agg}} \ , \tag{48}$$

$$\boldsymbol{U}^{\text{src}} = \mathbf{Q}\boldsymbol{U}^{\text{agg}} \ , \tag{49}$$

which allows to initialize the solutions in those two elements before starting the iterations on the residual.

4) *Assembly of agglomerated residual*: in order to avoid facing the issue described in Sec. III.B.2, the equations must be solved for the agglomerated solution directly and not the original one. Going back to the example, this would imply that at first glance one should remove the degrees of freedom $\boldsymbol{U}_{3,a}$ from the global solution vector $\boldsymbol{U}$ and perform the spatial integrations of the variational formulation on $\Omega_2^{\text{agg}} = \Omega_2^{\text{tar}} \cup \Omega_{3,a}^{\text{src}}$ [§]. This would require to adapt the data structures quite significantly, especially for moving interface problems, as well as building new quadrature rules. This apparent additional difficulty can be circumvented by computing the *spatial* agglomerated residual from operations on the original mesh:

$$\mathcal{R}^{\text{agg, sp}} = \begin{cases} \mathcal{R}^{\text{tar, sp}} + \mathbf{Q}^T \mathcal{R}^{\text{src, sp}} & \text{in } \Omega^{\text{tar}} \\ 0 & \text{in } \Omega^{\text{src}} \\ \mathcal{R}^{\text{sp}} & \text{otherwize} \end{cases} \tag{50}$$

Using Eq. (50) is needed for the assembly of spatial residuals involving spatial integrations that must be kept unchanged. However, if the temporal residual is assembled in the same way, the exactly same issue as before would be faced since $\mathcal{R}^{\text{src, temp, t+1}}$ is not defined (see Eq. 42). That is why its is proposed in this work to assemble the temporal residual directly from the agglomerated solution while keeping the assembly of the spatial residuals based on non-agglomerated operations:

$$\mathcal{R}^{\text{agg, temp}} = \begin{cases} \mathbf{M}^{\text{agg}}(\boldsymbol{U}^{\text{agg, t+1}} - \boldsymbol{U}^{\text{t}})/\Delta t & \text{in } \Omega^{\text{tar}} \\ 0 & \text{in } \Omega^{\text{src}} \\ \mathbf{M}(\boldsymbol{U}^{\text{t+1}} - \boldsymbol{U}^{\text{t}})/\Delta t & \text{otherwize} \end{cases} \tag{51}$$

Applied to the example in Fig. 10b, this leads to:

$$\begin{aligned} \mathcal{R}_{2,a,1} = \ & \mathcal{R}^{\text{agg, temp}}_{2,a,1} + \mathcal{R}^{\text{agg, vol}}_{2,a,1} + \mathcal{R}^{\text{agg, int}}_{2,a,1} + \mathcal{R}^{\text{agg, iso-0}}_{2,a,1} \\ \mathcal{R}_{2,a,2} = \ & \mathcal{R}^{\text{agg, temp}}_{2,a,2} + \mathcal{R}^{\text{agg, vol}}_{2,a,2} + \mathcal{R}^{\text{agg, int}}_{2,a,2} + \mathcal{R}^{\text{agg, iso-0}}_{2,a,2} \\ \mathcal{R}_{3,a,1} = \ & 0 \\ \mathcal{R}_{3,a,2} = \ & 0 \end{aligned} \tag{52}$$

while the other residuals are assembled as described before.

The use of an implicit time stepping scheme in this work requires also the definition of the agglomerated Jacobian, i.e. the derivative of agglomerated residual w.r.t. the agglomerated solution. Without entering into the details that will be provided in future publications, it can just be mentioned that the Jacobian associated to the temporal residual is straightforward while the spatial Jacobian is more involved and it is based on chain rules derivation and the link between the agglomerated solution and the target and source solutions.

Steps 1 to 3 presented above are performed at the start of a new time step while step 4 is repeated during the iterations on the residual of the equations. This is presented in Algorithm 5.

As a final comment, one can note that the exact same cell-agglomeration procedure is applied to cut elements containing a small volume fraction of one of the phases. This is indeed necessary to avoid having to solve systems with very high condition numbers, which could compromise the convergence. The determination of the target-source pairs needs also to take into account this criterion. Except this aspect, the procedure described before remains the same. An example of such situation is sketched in Fig. 14.

[§]Note that the agglomerated cell is identified with the index of the target element since when one solves for the agglomerated solution, one solves for the degrees of freedom in the target element and then project the solution on the source

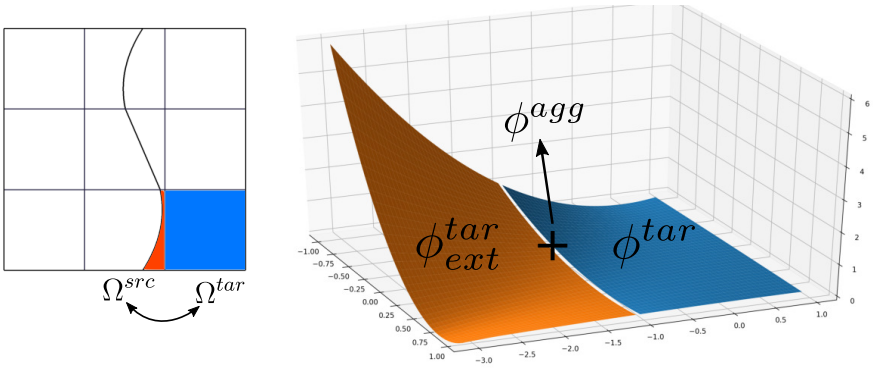


Fig. 11 The computation of the agglomeration coupling matrix Q requires the smooth extension of the target basis functions to the adjacent source element, which defines the agglomerated basis functions. Here, an example of the extension of one of the quadratic Lagrange shape function on a 2-D quadrilateral mesh.

4. Convergence studies

To close this section, a simple but instructive two-phase one-dimensional scalar advection-diffusion equation is considered as a model equation to perform a spatial and temporal convergence study:

$$\frac{\partial u}{\partial t} + c\frac{\partial u}{\partial x} = k\frac{\partial^2 u}{\partial x^2}\,, \tag{53}$$

with a uniform advection velocity $c = 1$ and diffusion coefficient $k = 1$. The domain is $[-1, 1]$. The interface is initially located at $x = 0$. The final time is $t_f = 0.75$. The jumps on the the solution and its gradient at the interface are imposed to be:

$$[u](x_\Gamma) = 1.0\,,$$
$$k[\frac{\partial u}{\partial x}](x_\Gamma) = 1.0\,,$$

where the interface location depends on time: $x_\Gamma(t) = c\,t$[¶]. On the left boundary, a Dirichlet boundary condition is imposed while on the right boundary, a Neumann boundary condition is used:

$$u(x = -1) = 0.0\,,$$
$$k\frac{\partial u}{\partial x}(x = 1) = 1.0\,.$$

The method of manufactured solution is followed to first determine the coefficients of the exact solution chosen to have the form:

$$u_{\text{left}}(x,t) = a_L(t) + b_L(t)\sin(x\pi/4) \qquad x < x_\Gamma(t)$$
$$u_{\text{right}}(x,t) = a_R(t) + b_R(t)\sin(x\pi/4) \qquad x > x_\Gamma(t)$$

Inserting this into Eq. (53), it is then possible to find a source term that will be added to the equation to be discretized. The accuracy with which the numerical method approximates the manufactured reference solution can finally be studied for different mesh sizes Δx and time step sizes Δt. The results of this investigation are summarized in Fig. 12. A second order BDF2 time stepping scheme is selected while in space, a first and second polynomial interpolation orders p are chosen. Some comments can be formulated:

- *Spatial convergence*: above a certain time step size, the temporal errors dominate and augmenting the number of elements lead to an increase in the final error. For a sufficiently low time resolution, the theoretical spatial convergence rates are recovered. It has to be mentioned here that since the transposed term in the interior penalty method has not been implemented yet for the XDG discretization, the spatial rate of convergence for even interpolation orders is limited to p instead of $p + 1$;

[¶]The strategy to enforce those interfacial jumps through integration of fluxes on the iso-0 level-set will be described in a future publication.

Algorithm 5 XDG fluid solver for one time step taking as input the topology $\texttt{Topo}^{t+1}$ and the quadrature $\texttt{Quad}^{t+1}$ at the next time step and computing the solution at the next time step $\boldsymbol{U}^{t+1}$ based on a spatial XDG discretization combined with cell-agglomeration and an implicit temporal integration

procedure XDGSOLVER($\texttt{Topo}^{t+1}, \texttt{Quad}^{t+1}$)
  $t \leftarrow t+1$, $k = 0$
  $\boldsymbol{U}_k^* \leftarrow \texttt{ResizeDataStructure}(\texttt{Topo}^{t+1})$ ▷ Resize the data structures based on the new topology
  $\boldsymbol{U}_k^* \leftarrow \boldsymbol{U}^t$ ▷ Initialize the iterate with the solution from the previous time
  $\{\text{src, tar}\}^{t+1} \leftarrow \texttt{FlagAgglom}(\texttt{Topo}^{t+1})$ ▷ Sec. III.B.3, step 1
  $\mathbf{Q}^{t+1}$, $\mathbf{M}^{\text{agg},t+1} \leftarrow \texttt{ComputeAgglomData}(\{\text{src, tar}\}^{t+1}, \texttt{Quad}^{t+1})$ ▷ Sec. III.B.3, step 2
  $\boldsymbol{U}_k^{\text{agg},*}$, $\boldsymbol{U}_k^{\text{tar},*}$, $\boldsymbol{U}_k^{\text{src},*} \leftarrow \texttt{InjectAgglomSolution}(\mathbf{Q}^{t+1}, \mathbf{M}^{\text{agg},t+1})$ ▷ Sec. III.B.3, step 3
  While $\left(k \leq k_{\max} \text{ And } \left\|(\mathbf{M}^{\text{agg},t+1})^{-1}\mathcal{R}_k^{\text{agg, tot}}\right\| > \text{tol}\right)$ **do**
    $\boldsymbol{U}_k^{\text{src}} = \mathbf{Q}^{t+1}\boldsymbol{U}_k^{\text{agg}}$ ▷ Eq. (49)
    $\mathcal{R}_k^{\text{agg, temp}} \leftarrow \texttt{AssembleAgglomTempRes}(\boldsymbol{U}_k^{\text{agg}}, \mathbf{M}^{\text{agg},t+1})$ ▷ Eq. (51)
    $\mathcal{R}_k^{\text{sp}} \leftarrow \texttt{AssembleSpatialRes}(\boldsymbol{U}_k, \texttt{Quad}^{t+1})$ ▷ Eq. (32)
    $\mathcal{R}_k^{\text{agg, sp}} \leftarrow \texttt{ProjectSpatialResToAgglom}(\mathcal{R}_k^{\text{sp}}, \mathbf{Q}^{t+1})$ ▷ Eq. (50)
    $\left[\frac{\mathbf{I}}{\Delta\tau} + (\mathbf{M}^{\text{agg},t+1})^{-1}\frac{\partial \mathcal{R}^{\text{agg, tot}}}{\partial \boldsymbol{U}^{\text{agg}}}(\boldsymbol{U}_k)\right]\Delta\boldsymbol{U}^{\text{agg}} = -(\mathbf{M}^{\text{agg},t+1})^{-1}\mathcal{R}_k^{\text{agg, tot}}$ ▷ Eq.(29)
    $\boldsymbol{U}_{k+1}^{\text{agg}} = \boldsymbol{U}_k^{\text{agg}} + \Delta\boldsymbol{U}^{\text{agg}}$
    $\boldsymbol{U}_{k+1}^{\text{src}} = \mathbf{Q}^{t+1}\boldsymbol{U}_{k+1}^{\text{agg}}$ ▷ Eq. (49)
    $k \leftarrow k+1$
  end do
    $\boldsymbol{U}^{t+1} \leftarrow \boldsymbol{U}_{k+1}^{\text{agg}}$, $\boldsymbol{U}^{\text{src},t+1} \leftarrow \boldsymbol{U}_{k+1}^{\text{src}}$
  return $\boldsymbol{U}^{t+1}$
end procedure

- *Temporal convergence*: the story is a bit different for the temporal convergence. Indeed, one observes that for spatial resolutions in the range considered for the spatial convergence study, the temporal convergence rate saturates. It is only at very small (impractical) spatial resolution that the convergence can be restored, given that the time step is small enough. For larger time steps, using a higher spatial resolution leads to a higher error. The same kind of observations was made in [34]. Using a higher order for the temporal scheme may change partially the picture. Further investigations would be needed here but it seems that the use of the splitting temporal integration approach introduces unavoidable dominating temporal errors. To overcome this limitation, the moving interface approach proposed in [34] will have to be tested. This is discussed in the perspectives.

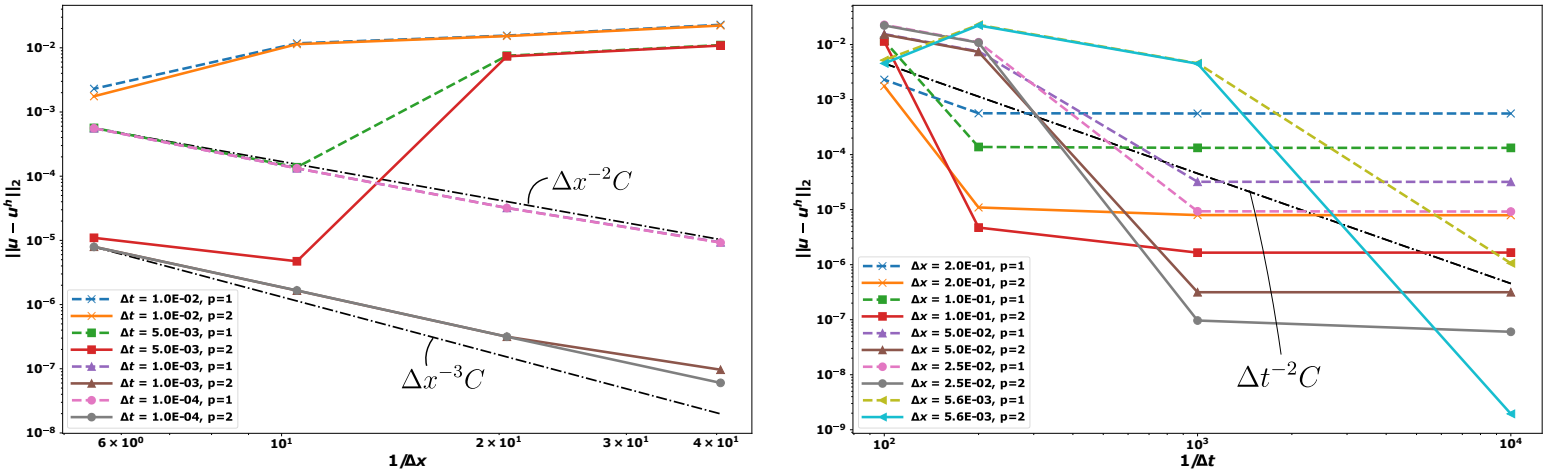
The spatial convergence study of the steady version of this test case is presented in Appendix V.B.

C. General algorithm combining level-set advection and sharp interface solver

1. Some background

In this last section, the coupling between the interface evolution and the flow solver is covered. Three main options are possible, from the simplest to the most difficult:

- *Operator splitting with explicit coupling*: the two main strategies in this category are the Lie and the Strang splittings. With the Lie splitting, the interface position is only updated once from time t to $t+1$ using the fluid velocity at time t before solving the governing equations considering the interface position at time $t+1$. In the case of the Strang splitting, one first performs a half-time step to advect the level-set to time $t+1/2$ with the fluid velocity at time t. One then solves the governing equations with a full time step from t to $t+1$ taking the interface position at $t+1/2$. Finally, one performs a second half-time step on the level-set to obtain the level-set at time $t+1$ from the fluid velocity at time $t+1$ [36, §3.4]. During the resolution of the governing equations, the interface can be assumed totally fixed, in which case a "splitting" approach for temporal integration is selected. The other possibility is to take into account the effect of the interface motion during the time integration of the fluid equations, without explicitly updating the interface position at every iteration of the fluid solver. This is done by incorporating into the convective fluxes the ones coming from the interface motion, in an Arbitrary



(a) Spatial convergence **(b) Temporal convergence**

Fig. 12 ***Two-phase advection-diffusion test case*: convergence study involving various mesh and time step sizes. Theoretical rates of convergence in dashed.**

Eulerian-Lagrangian (ALE) fashion. This method is referred as the moving interface approach [34].

- *Operator splitting with implicit coupling*: the second scenario consists in updating the interface position by the new fluid velocity field at time $t + 1$. This requires the update of the level-set during the iterations of the fluid solver. In [21, §3.9], the interface update is not performed in every iteration, but only when the fluid residual has converged, before starting a new iteration loop until the tolerances on both the fluid and the level-set residuals decrease below a prescribed threshold. Depending on the time step selected, this approach may produce more accurate results than the explicit coupling, with a higher computational cost.
- *Fully coupled fluid and interface dynamics*: this last option avoids any operator splitting by building a residual and a Jacobian matrix incorporating both the fluid and the interface dynamics variables. In this way, the interface dynamics and fluid motion are fully-non-linearly-coupled within a Newton iterator. This approach may be necessary to ensure convergence, maintain a strict conservation and meet the stringent resolution requirements when dealing with strongly coupled interfacial phenomena having very small time scales. It has been explored in [37] but seems complicated to extend to an existent XDG level-set framework.

In this work, the simplest approach among the ones discussed above is followed, namely an explicit operator splitting with a splitting approach for the time integration. Again, other more sophisticated approaches will probably be investigated in future work.

2. Global algorithm

As an introduction to the last Algorithm 6 combining the different methods covered in this manuscript, a short inventory of the information transferred between the level-set and the XDG solvers is given:

- *From the level-set to the XDG solver*: the level-set first determines the cut topology on which the fluid equations need to be solved, i.e. the cut elements and the cut edges. A fundamental point to mention here is that the discretized level-set field is in general discontinuous across the elements edges. However, the spatial discretization in the XDG space requires at least a C^0 interface representation. Therefore, two level-set fields are considered as in [22, §4]: the one which is advected, φ, and another one which is imposed to be continuous across the edges, φ^{cont}. In case the reinitialization is performed on the interpolation points, this continuous level-set field is directly available. If the procedure is done on the quadrature points, an additional step involving a L^2-projection of φ onto φ^{cont} with continuity constraints would be required [38], but it is not considered in this work. The field φ^{cont} is then used to build the cut quadrature rules from the implicit description of the interface it provides. Finally, geometrical quantities like the normal and the curvature of the interface are needed to compute the interfacial jump conditions.
- *From the XDG to the level-set solver*: The XDG solver on its side provides the fluid states on both sides of the interface which are then used to determine the velocity with which the level-set is advected (see Sec. II.E.2).

It is important to note that in the case of the moving interface approach mentioned before, the interface velocity must also be transferred from the level-set to the fluid solver such that the dynamic convective flux can be computed.

Algorithm 6 Level-set and XDG solvers over one time step coupled through an explicit operator splitting approach to update the level-set φ and fluid solutions $\mathbf{U}$ from time t to $t+1$

procedure SOLVEMOVINGINTERFACE($\varphi^t, \mathbf{U}^t$)
  $\varphi^{t+1}, \varphi^{\text{cont},t+1} \leftarrow$ `MoveLevelSet`$(\varphi^t, \mathbf{U}^t)$ ▷ Algorithm 4
  $\texttt{Topo}^{t+1} \leftarrow$ `UpdateTopology`$(\varphi^{\text{cont},t+1})$
  $\texttt{Quad}^{t+1} \leftarrow$ `UpdateQuadrature`$(\texttt{Topo}^{t+1}, \varphi^{\text{cont},t+1})$
  $\mathbf{U}^{t+1} \leftarrow$ `XDGSolver`$(\texttt{Topo}^{t+1}, \texttt{Quad}^{t+1})$ ▷ Algorithm 5
  return $\varphi^{t+1}, \mathbf{U}^{t+1}$
end procedure

3. The Rayleigh-Plesset test case

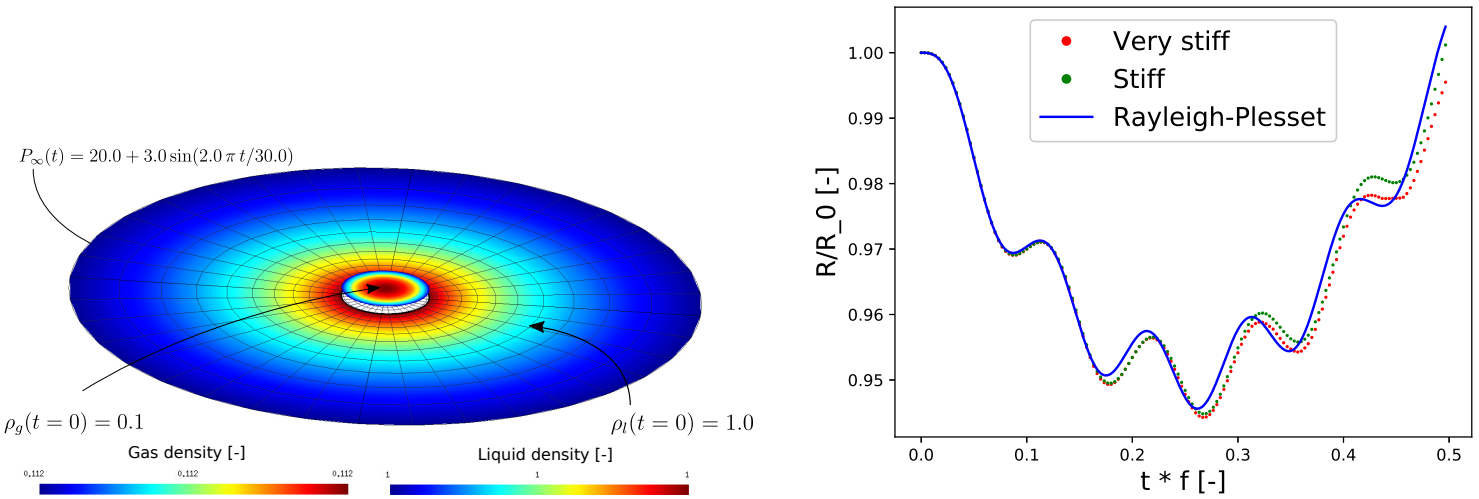
The test case is taken from [39] and is about the oscillation of a circular compressible gas bubble under the influence of pressure variations transmitted through an almost incompressible liquid. The situation can be fairly well represented by the Rayleigh-Plesset equation which governs the radial deformation of a spherical bubble in an infinite body of incompressible fluid. Adapted to a 2-D finite domain, the second order ODE governing the time evolution of the bubble radius becomes

$$\ddot{R} = \frac{1}{R\ln\left(\frac{R_\infty}{R}\right)}\left[\frac{P_g - P_\infty}{\rho_l} - \left(\frac{R^2}{R_\infty^2} - 1\right)\frac{\dot{R}^2}{2} - \dot{R}^2\ln\left(\frac{R_\infty}{R}\right)\right], \tag{54}$$

where $R(t)$ is the radius of the bubble, R_∞ is the radius of the domain, ρ_l is the density of the surrounding liquid (constant for an incompressible flow), $p_g(t)$ is the pressure of the gas inside the bubble (assumed uniform) and $p_\infty(t)$ is the pressure applied at the external boundary of the liquid, which is set to vary with time. To model the incompressible liquid behavior, a stiffened gas equation of state (EOS) is used. The gas phase is described by the perfect gas law. The enforcement of the jumps conditions across the gas-liquid interface is done via a multiphase Riemann solver which determines the fluxes to be integrated on the level-set iso-0. More information can be found in [1].
A snapshot of the discontinuous density field over the domain is shown in Fig. 13a along with the simulation set-up. The time evolution of the interface position plotted in Fig. 13b is in accordance with the solution of the Rayleigh-Plesset Eq. (54). The progressive shift that appears between the two curves is most probably due to the use of a weakly compressible model for the liquid, slightly delaying the pressure wave propagation compared to a pure incompressible model. Imputing this observation to the explicit operator splitting coupling adopted in this work is not possible at this stage and further investigations will be needed. This inviscid compressible two-phase flow situation demonstrates however the potential of the presented approach to handle moving interface problems involving jumps and advanced physical models.

IV. Conclusions and perspectives

In this paper, two central pieces of a high-order sharp interface multiphase solver relying on an extended discontinuous Galerkin method were analyzed.
Efficient and accurate reinitialization and velocity extension procedures were developed. They are based on a single-step geometric approach involving a constrained minimization problem for the exact closest point computation. They offer flexibility in the way they can be realized: on the quadrature or the interpolation points, the velocity obtained from fluid velocities, transition model or the jumps conditions, and the operations performed on different narrow-band regions. The high-order convergence of the proposed approach was demonstrated on two benchmark test cases. The application to more involved situations proved to be successful despite the apparition of some instabilities in less resolved regions. Capitalizing on different recent investigations from the literature, a consistent time integration strategy for unfitted method applied to moving interface problems was also designed. It uses cell-agglomeration technique to remove the inconsistency appearing in the assembly of the temporal residual. It has been implemented in a splitting time integration approach. A simple but still challenging transient test case showed the possible limitations in terms of temporal accuracy of this approach. The explicit operator splitting coupling between the level-set and the XDG fluid solvers was finally



(a) Snapshot of the gas and liquid density fields with the under-lying mesh. The initial conditions and the boundary condition are shown

(b) Non-dimensionalized evolution of the bubble radius - comparison between numerical and pseudo-analytical solutions

Fig. 13 ***Rayleigh-Plesset test case*** **- Oscillations of a compressible gas bubble surrounded by a weakly compressible liquid**

presented. Its application to a compressible gas-liquid flow suggested the potential of the developed framework to deal with more complex flow situations.

Through the state-of-the art review and the different test cases, some possible improvements to the methods developed so far were identified.
Concerning the reinitialization and extension, the improvement of the robustness of the closest point computation revealed necessary. The implementation of a dedicated Lagrange-Newton SQO method is foreseen to overcome this issue.
The simulation of more intricate interfacial dynamics like thin filaments or topology changes may require the coupling with an adaptative mesh refinement strategy.
In order to avoid the inaccuracies introduced by the use of a splitting approach for time integration, the moving interface approach will be implemented.
Strang splitting between the level-set and the XDG will also be considered in a first step and compared to the use of the current Lie splitting. A more elaborated implicit coupling may also be investigated in the future in an effort to guarantee strict conservation of the global approach.

Acknowledgments

The first author is supported by the Fonds de la Recherche Scientifique (FNRS) of Belgium, under a FRIA grant. Computational resources have been provided by the supercomputing facilities of the Université catholique de Louvain (CISM/UCL) and the Consortium des Équipements de Calcul Intensif en Fédération Wallonie Bruxelles (CÉCI) funded by the Fond de la Recherche Scientifique de Belgique (F.R.S.-FNRS) under convention 2.5020.11 and by the Walloon Region

V. Appendix

A. Cell-agglomeration technique to deal with small cut elements

The cell-agglomeration strategy applied to the appearing cut elements for the temporal integration in the XDGM is presented in Sec. III.B. The exact same procedure is applied to cut elements containing a volume fraction of one of the phase below a chosen threshold, typically around $0.2 - 0.3$. This is sketched in Fig. 14.

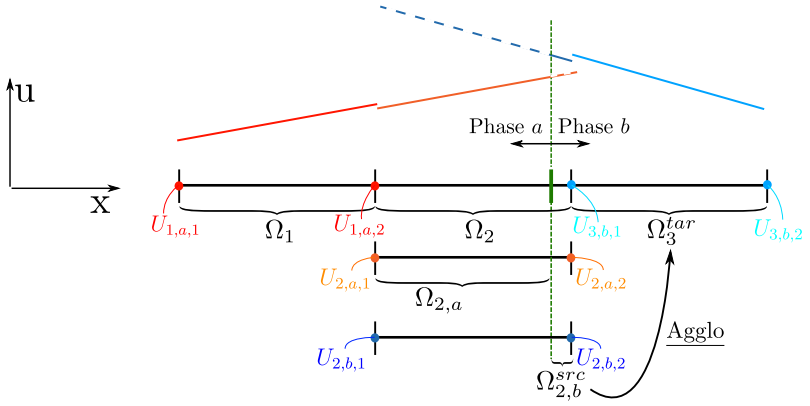


Fig. 14 The volume fraction of phase b being below a certain threshold in element 2, the solution is agglomerated with the adjacent element 3 containing the same phase b.

B. Additional spatial convergence results for the XDGM

The spatial convergence study applied to the steady version of the two-phase advection-diffusion test case of Sec. III.B.4 is presented in Fig. 15. Because the temporal errors are absent in this case, the theoretical spatial convergence rates are achieved for all interpolation orders and mesh sizes. It is recalled here that since the transposed term in the interior penalty method has not been implemented yet for the XDG discretization, the spatial rate of convergence for even interpolation orders is limited to p instead of $p + 1$;

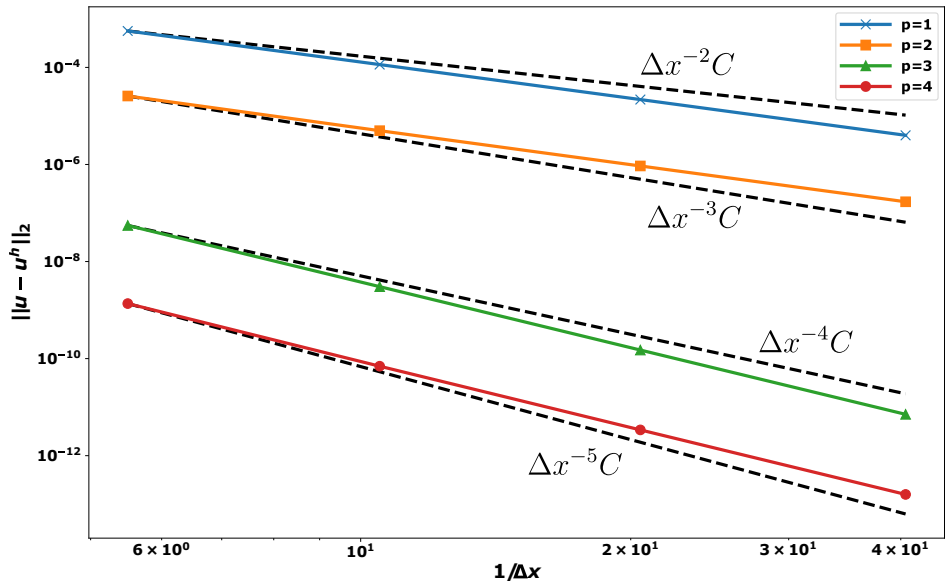


Fig. 15 *Steady two-phase advection-diffusion test case*: spatial convergence study for different interpolation orders p. Theoretical rates of convergence in dashed.

References

[1] Henneaux, D., Schrooyen, P., Ricardo Barros Dias, B., Turchi, A., Chatelain, P., and Magin, T., "Extended Discontinuous Galerkin Method for Solving Gas-Liquid Compressible Flows with Phase Transition," AIAA AVIATION 2020 FORUM, 2020, p. 2971.

[2] Marchandise, E., "Simulation of three-dimensional two-phase flows : coupling of a stabilized finite element method with a discontinuous level set approach," Ph.D. thesis, Universite catholique de Louvain, 2006.

[3] Mirjalili, S., Jain, S. S., and Dodd, M., "Interface-capturing methods for two-phase flows: An overview and recent developments," Center for Turbulence Research Annual Research Briefs, Vol. 2017, 2017, pp. 117–135.

[4] Osher, S., and Sethian, J. A., "Fronts propagating with curvature-dependent speed: algorithms based on Hamilton-Jacobi formulations," Journal of computational physics, Vol. 79, No. 1, 1988, pp. 12–49.

[5] Sethian, J. A., and Smereka, P., "Level set methods for fluid interfaces," Annual review of fluid mechanics, Vol. 35, No. 1, 2003, pp. 341–372.

[6] Sethian, J. A., "A fast marching level set method for monotonically advancing fronts," Proceedings of the National Academy of Sciences, Vol. 93, No. 4, 1996, pp. 1591–1595.

[7] Tsai, Y.-H. R., Cheng, L.-T., Osher, S., and Zhao, H.-K., "Fast sweeping algorithms for a class of Hamilton–Jacobi equations," SIAM journal on numerical analysis, Vol. 41, No. 2, 2003, pp. 673–694.

[8] Sussman, M., "A level set approach for computing solutions to incompressible two-phase flow," Ph.D. thesis, UCLA - Computational and Applied Mathematics, 1994.

[9] Utz, T., Kummer, F., and Oberlack, M., "Interface-preserving level-set reinitialization for DG-FEM," International Journal for Numerical Methods in Fluids, Vol. 84, No. 4, 2017, pp. 183–198.

[10] Chopp, D. L., "Some improvements of the fast marching method," SIAM Journal on Scientific Computing, Vol. 23, No. 1, 2001, pp. 230–244.

[11] Saye, R., "High-order methods for computing distances to implicitly defined surfaces," Communications in Applied Mathematics and Computational Science, Vol. 9, No. 1, 2014, pp. 107–141.

[12] Hillewaert, K., "Development of the discontinuous Galerkin method for high-resolution, large scale CFD and acoustics in industrial geometries," Ph.D. thesis, Université catholique de Louvain, 2013.

[13] Remacle, J.-F., Chevaugeon, N., Marchandise, E., and Geuzaine, C., "Efficient visualization of high-order finite elements," International Journal for Numerical Methods in Engineering, Vol. 69, No. 4, 2007, pp. 750–771.

[14] Arya, S., Mount, D. M., Netanyahu, N. S., Silverman, R., and Wu, A. Y., "An optimal algorithm for approximate nearest neighbor searching fixed dimensions," Journal of the ACM (JACM), Vol. 45, No. 6, 1998, pp. 891–923.

[15] Anumolu, L., and Trujillo, M. F., "Gradient augmented reinitialization scheme for the level set method," International Journal for Numerical Methods in Fluids, Vol. 73, No. 12, 2013, pp. 1011–1041.

[16] Malladi, R., Sethian, J. A., and Vemuri, B. C., "Shape modeling with front propagation: A level set approach," IEEE transactions on pattern analysis and machine intelligence, Vol. 17, No. 2, 1995, pp. 158–175.

[17] Adalsteinsson, D., and Sethian, J. A., "The fast construction of extension velocities in level set methods," Journal of Computational Physics, Vol. 148, No. 1, 1999, pp. 2–22.

[18] Chopp, D. L., "Another look at velocity extensions in the level set method," SIAM Journal on Scientific Computing, Vol. 31, No. 5, 2009, pp. 3255–3273.

[19] Peng, D., Merriman, B., Osher, S., Zhao, H., and Kang, M., "A PDE-based fast local level set method," Journal of computational physics, Vol. 155, No. 2, 1999, pp. 410–438.

[20] Utz, T., and Kummer, F., "A high-order discontinuous Galerkin method for extension problems," International Journal for Numerical Methods in Fluids, Vol. 86, No. 8, 2018, pp. 509–518.

[21] Smuda, M., "Direct Numerical Simulation of Multi-Phase Flows using Extended Discontinuous Galerkin Methods," Ph.D. thesis, Technische Universität, 2020.

[22] Utz, T., "Level set methods for high-order unfitted discontinuous Galerkin schemes," Ph.D. thesis, Technische Universität, 2018.

[23] Mousavi, R., "Level set method for simulating the dynamics of the fluid-fluid interfaces: Application of a Discontinuous Galerkin Method," 2014.

[24] Chopp, D. L., "Computing minimal surfaces via level set curvature flow," J. Comput. Phys., Vol. 106, 1991, pp. 77–91.

[25] Adalsteinsson, D., and Sethian, J. A., "A fast level set method for propagating interfaces," Journal of computational physics, Vol. 118, No. 2, 1995, pp. 269–277.

[26] Adams, T. D., "Discontinuous Galerkin discretised level set methods with applications to topology optimisation," Ph.D. thesis, Durham University, 2020.

[27] Kummer, F., Weber, J., and Smuda, M., "BoSSS: A package for multigrid extended discontinuous Galerkin methods," Computers & Mathematics with Applications, Vol. 81, 2021, pp. 237–257.

[28] Kummer, F., "Extended discontinuous Galerkin methods for two-phase flows: the spatial discretization," International Journal for Computational Methods in Engineering, Vol. 109, 2017, pp. 259–289.

[29] Schrooyen, P., Cagnogne, J.-S., Poletz, N., and Hillewaert, K., "A High-order Extended Discontinuous Galerkin Method for Moving Immersed Interface Problem," XDMS Conference, Umea, 2017.

[30] Schrooyen, P., Arbaoui, L., Cagnone, J.-S., Poletz, N., and Hillewaert, K., "A High-order Extended Discontinuous Galerkin Method to Treat Hydrodynamics Problems," ECCM-ECFD Conference, Glasgow (UK), 2018.

[31] Schott, B., "Stabilized cut finite element methods for complex interface coupled flow problems," Ph.D. thesis, Technische Universität München, 2017.

[32] Schrooyen, P., Hillewaert, K., Magin, T. E., and Chatelain, P., "Discontinuous Galerkin discretization coupled with sharp interface method for ablative materials," 21st AIAA Computational Fluid Dynamics Conference, 2013, p. 2457.

[33] Gürkan, C., "Extended hybridizable discontinuous Galerkin method," Ph.D. thesis, Universitat Politècnica de Catalunya, 2018.

[34] Kummer, F., Müller, B., and Utz, T., "Time integration for extended discontinuous Galerkin methods with moving domains," International Journal for Numerical Methods in Engineering, Vol. 113, No. 5, 2018, pp. 767–788.

[35] Müller, B., Krämer-Eis, S., Kummer, F., and Oberlack, M., "A high-order Discontinuous Galerkin method for compressible flows with immersed boundaries," Internal Journal For Numerical Methods in Engineering, Vol. 110, 2017, pp. 3–30.

[36] Heimann, F., "An Unfitted Higher-Order Discontinuous Galerkin Method for Incompressible Two-Phase Flow with Moving Contact Lines," Ph.D. thesis, Ruperto-Carola University of Heidelberg, 2013.

[37] Nourgaliev, R., Theofanous, T., Park, H., Mousseau, V., and Knoll, D., "Direct numerical simulation of interfacial flows: Implicit sharp-interface method (I-SIM)," 46th AIAA Aerospace Sciences Meeting and Exhibit, 2008, p. 1453.

[38] Smuda, M., and Kummer, F., "On a marching level-set method for extended discontinuous Galerkin methods for incompressible two-phase flows," arXiv preprint arXiv:2010.08417, 2020.

[39] Nair, P., and Tomar, G., "Multiphase flows with compressible and incompressible phases," arXiv preprint arXiv:1804.06329, 2018.