Journal of Computational Physics 230 (2011) 789–817

Contents lists available at ScienceDirect

Journal of Computational Physics

journal homepage: www.elsevier.com/locate/jcp

Space–time discontinuous Galerkin finite element method for two-fluid flows

W.E.H. Sollie, O. Bokhove, J.J.W. van der Vegt [*]

Department of Applied Mathematics, Institute of Mechanics, Processes and Control Twente, University of Twente, P.O. Box 217, 7500 AE, Enschede, The Netherlands

ARTICLE INFO

Article history:
Received 14 December 2009
Received in revised form 7 October 2010
Accepted 16 October 2010
Available online 26 October 2010

Keywords:
Cut-cell
Space–time discontinuous Galerkin
Front tracking
Level set
Two-fluid flow

ABSTRACT

A novel numerical method for two-fluid flow computations is presented, which combines the space–time discontinuous Galerkin finite element discretization with the level set method and cut-cell based interface tracking. The space–time discontinuous Galerkin (STDG) finite element method offers high accuracy, an inherent ability to handle discontinuities and a very local stencil, making it relatively easy to combine with local *hp*-refinement. The front tracking is incorporated via cut-cell mesh refinement to ensure a sharp interface between the fluids. To compute the interface dynamics the level set method (LSM) is used because of its ability to deal with merging and breakup. Also, the LSM is easy to extend to higher dimensions. Small cells arising from the cut-cell refinement are merged to improve the stability and performance. The interface conditions are incorporated in the numerical flux at the interface and the STDG discretization ensures that the scheme is conservative as long as the numerical fluxes are conservative. The numerical method is applied to one and two dimensional two-fluid test problems using the Euler equations.

1. Introduction

Fluid flows with interfaces involve combinations of gases, liquids and solids and have many applications in nature and industry. Examples include flows with bubbles, droplets or solid particles, wave–structure interactions, dam breaking, bed evolution, Rayleigh–Taylor and Kelvin–Helmholtz instabilities and industrial processes such as bubble columns, fluidized beds, granular flows and ink spraying. The flow patterns in these problems are complex and diverse and can be approached at various levels of complexity. Often the interface is not static but moves with the fluid flow velocity and in more complex cases interface topological changes due to breakup and coalescence processes may occur. Solutions often have a discontinuous character at the interface between different fluids, due to surface tension and other effects. In addition, the density and pressure differences across the interface can be very high, like in the case of liquid–gas flows. Also, the existence of shock or contact waves can introduce additional discontinuities into the problem. Because of the continuous advances in computer technology the numerical simulation of these problems is becoming increasingly affordable. However, there are several issues related to solving flows with interfaces numerically. These include issues regarding accuracy and conservation of the flow field quantities near the interface, robustness and stability of the interface coupling, complex geometries, unstructured mesh generation and motion, mesh topological changes and computational efficiency. A numerical method which has received much attention in recent years and which is especially suited for dealing with flows with strong discontinuities and unstructured meshes is the discontinuous Galerkin finite element method.

[*] Corresponding author. Tel.: +31 53 489 5628; fax: +31 53 489 4833.
E-mail addresses: w.e.h.sollie@math.utwente.nl (W.E.H. Sollie), o.bokhove@math.utwente.nl (O. Bokhove), j.j.w.vandervegt@math.utwente.nl (J.J.W. van der Vegt).

0021-9991/$ - see front matter © 2010 Elsevier Inc. All rights reserved.
doi:10.1016/j.jcp.2010.10.019

In this article a novel discontinuous Galerkin front tracking method for two-fluid flows is presented, which is accurate, versatile and can alleviate some of the problems commonly encountered with existing methods. In order to explain and motivate the choices made for the numerical method, first the most important aspects of the space–time discontinuous Galerkin finite element method are discussed. This is followed by a discussion of important existing techniques for dealing with interfaces. Based on this discussion the interface related choices in the method are explained. Finally, the research objectives are stated.

For a complete survey of discontinuous Galerkin (DG) methods and their applications, see [11]. The main feature of DG methods is that they allow solutions to be discontinuous over element faces. The basis functions are defined locally on each element with only a weak coupling to neighboring elements. The computational stencil is therefore very local; hence, DG methods are relatively easy to combine with parallel computation and also *hp*-refinement, where a combination of local mesh refinement (*h*-adaptation) and adjustment of polynomial order (*p*-adaptation) is used. Another important property is that DG discretizations are conservative. The DG method has order of accuracy $O(h^{p+1})$ for smooth solutions and order of accuracy $O(h^{1/2})$ for discontinuous solutions [31,55]. Front capturing and tracking techniques can help to improve the accuracy of the DG method around discontinuities. Near discontinuities higher order DG solutions will exhibit spurious oscillations. These oscillations may be removed by using slope limiting, shock fitting techniques or artificial dissipation in combination with discontinuity detection.

The space–time discontinuous Galerkin finite element method (STDG) introduced by van der Vegt and van der Ven [65] is a space–time variant of the DG method which is especially suited for handling dynamic mesh motions in space–time (see also [6,28,55,66]). It features a five-stage semi-implicit Runge–Kutta scheme with coefficients optimized for stability in combination with multigrid for accelerated convergence to solve the (non) linear algebraic equations resulting from the STDG discretization.

Many methods have been proposed for computing flows with interfaces or, to be more general, fronts. By looking at the front representation in the mesh one can distinguish between front capturing and front tracking methods. Other methods exist, such as particle methods and boundary integral methods, but these are not relevant for the current discussion.

In front capturing methods a regular stationary mesh is used and there is no explicit front representation. Instead, the front is either described by means of marker particles, like in the marker and cell method, or by use of functions, such as in the volume of fluid and level set methods. The earliest numerical method for time dependent free surface flow problems was the marker and cell (MAC) method [12,24]. Being a volume marker method it uses tracers or marker particles defined in a fixed mesh to locate the phases. However, the large number of markers required to obtain sufficient accuracy makes the method expensive.

In the Volume of Fluid (VoF) method [25,42,50,73] a fractional volume or color function is defined to indicate the fraction of a mesh element that covers a particular type of fluid. Algorithms for volume tracking are designed to solve the equation $\partial c/\partial t + \bar{\nabla}\cdot(c\mathbf{u}) = 0$, where c denotes the color function, $\mathbf{u}$ the local velocity at the front, t the time and $\bar{\nabla} = (\partial/\partial x_1, \ldots, \partial/\partial x_d)$ the spatial gradient operator in d-dimensional space. In the VoF method typically a reconstruction step is necessary to reproduce the interface geometry from the color function. More accurate VoF techniques like the Piecewise Linear Interface Construction (PLIC) method attempt to fit the interface by means of piecewise linear segments. VoF methods are easy to extend to higher dimensions and can be parallelized readily due to the local nature of the scheme. They can automatically handle reconnection and breakup. Current VoF methods can conserve mass but have difficulty in maintaining sharp boundaries between different fluids, and interfaces tend to smear. In addition, these methods can give inaccurate results when high interface curvatures occur. The computation of surface tension is not straightforward and in addition spurious bubbles and drops may be created. Recently, Greaves has combined the VoF method with Cartesian cut-cells with adapting hierarchical quadtree grids [22], which alleviates some of these problems.

The level set method (LSM) was introduced by Osher and Sethian in [36] and further developed in [1,52,56]. For a survey, see [53]. In the LSM an interface can be represented implicitly by means of the 0-level of a level set function $\psi(\mathbf{x}, t)$. The evolution of the interface is found by solving the level set equation $\partial\psi/\partial t + \mathbf{u}\cdot\bar{\nabla}\psi = 0$, with $\mathbf{u}$ the interface velocity. To reduce the computational costs a narrow band approach can be used, which limits the computations of the level set to a thin region around the interface. To enhance the level set accuracy it can be advected with the interface velocity, which for this purpose is extended from the interface into the domain. In case the level set becomes too distorted a reinitialization may be necessary. Various reinitialization algorithms are available based on solving a Hamilton–Jacobi partial differential equation [26,37,40]. Although the choice of the level set function is somewhat arbitrary, the signed distance to the interface tends to give the best accuracy in computing the curvature of the interface. Also, the LSM is easy to extend to higher dimensions and can automatically handle reconnection and breakup. The LSM, however, is not conservative in itself. Recent developments include the combination of the VoF method with the level set method [57].

Front capturing methods have the advantage of a relatively simple formulation. The main drawback of these methods lies in the need for complex interface shape restoration techniques, which often have problems in restoring the smooth and continuous interface shape, particularly in higher dimensions.

In front tracking and Lagrangian methods the front is tracked explicitly in the mesh. Front tracking was initially proposed in [47] and further developed in [19,33,63,64]. For a survey, see [27,48]. The evolution of the front is calculated by solving the equation $\partial\mathbf{x}/\partial t = \mathbf{u}$ at the front, where $\mathbf{x}$ is a point at the front and $\mathbf{u}$ its velocity. Glimm et al. [20] have combined front tracking with local grid based interface reconstruction using interface crossings with element edges. More recently they have proposed a fully conservative front tracking algorithm for systems of nonlinear conservation laws in [21].

Front tracking methods are often combined with either surface markers or cut-cells to define the location of the front. In the cut-cell method [4,13,39,60,61,71] a Cartesian mesh is used for all elements except those which are intersected by the front. These elements are refined in such a way that the front coincides with the mesh. At a distance from the front the mesh remains Cartesian and computations are less expensive. A common problem with cut-cell methods is the creation of very small elements which leads to problems with the stiffness of the equations and causes numerical instability. One way to solve this problem is by element merging as proposed in [72].

In Lagrangian or moving mesh methods [17,18,35,49] the mesh is modified to follow the fluid. In these methods the mesh can become considerably distorted, which gives problems with the mesh topology and stretched elements. In the worst case, frequent remeshing may be necessary [2,32]. In cases of breakup and coalescence, where the interface topology changes, these methods tend to fail.

Front tracking methods are good candidates for solving problems that involve complex interface physics. They are robust and can reach high accuracy when the interface is represented using higher order polynomials, even on coarse meshes. A drawback of front tracking methods is that they require a significant effort to implement, especially in higher dimensions.

The numerical algorithm for two-fluid flows presented here combines a space–time discontinuous Galerkin (STDG) discretization of the flow field with a cut-cell mesh refinement based interface tracking technique and a level set method (LSM) for computing the interface dynamics. The STDG discretization can handle interface discontinuities naturally, is conservative and has a very compact computational stencil. The level set method has the benefit of a simple formulation which makes it easier to extend the method to higher dimensions and also provides the ability to handle topological changes automatically. The interface tracking serves to maintain a sharp interface between the two fluids. This allows for different equations to be used for each fluid, which are coupled at the interface by a numerical interface flux, based on the interface condition. In addition, front tracking methods typically have high accuracy. Cut-cell refinement is used since it has the benefit of being local in nature and also is relatively easy to extend to higher dimensions.

An alternative approach to tracking singular surfaces with STDG meshes can be found in [38], where a space–time advancing front strategy ('tent pitching', [62]) is used to accomplish solution based tracking.

The outline of this article is as follows: In Section 2 the flow and level set equations are introduced. In Section 3 the background and refined meshes are discussed and the mesh refinement procedure is presented. In Section 4 the flow and level set discretizations, and the Runge–Kutta semi-implicit time integration method for the solution of the algebraic equations resulting from the numerical discretization are discussed. In Section 5 the two-fluid algorithm is presented. In Section 6 some test results are presented. Section 7 contains the final discussion and conclusions.

2. Equations

2.1. Two-fluid flow equations

Considered are flow problems involving two fluids as illustrated in Fig. 1. The two fluids are separated in space–time by an interface S. Let $i = 1, 2$ denote the fluid index. Furthermore, let $\mathbf{x} = (t, \bar{\mathbf{x}}) = (x_0, \ldots, x_d)$ denote the space–time coordinates, with d the spatial dimension, $\bar{\mathbf{x}} = (x_1, \ldots, x_d)$ the spatial coordinates and $t \in [t_0, T]$ the time coordinate, with t_0 the initial time and T the final time. The space–time flow domain for fluid i is defined as $\mathcal{E}^i \subset \mathbb{R}^{d+1}$. The (space) flow domain for fluid i at time t is defined as $\Omega^i(t) = \{\bar{\mathbf{x}} \in \mathbb{R}^d | (t, \bar{\mathbf{x}}) \in \mathcal{E}^i\}$. The space–time domain boundary for fluid i, $\partial\mathcal{E}^i$ is composed of the initial and final flow domains $\Omega^i(t_0)$ and $\Omega^i(T)$, the interface S and the space boundaries $\mathcal{Q}^i = \{\mathbf{x} \in \partial\mathcal{E}^i | t_0 < t < T\}$. The two-fluid space–time flow domain is defined as $\mathcal{E} = \cup_i \mathcal{E}^i$, the two-fluid (space) flow domain at time t as $\Omega(t) = \cup_i \Omega^i(t)$ and the two-fluid space–time

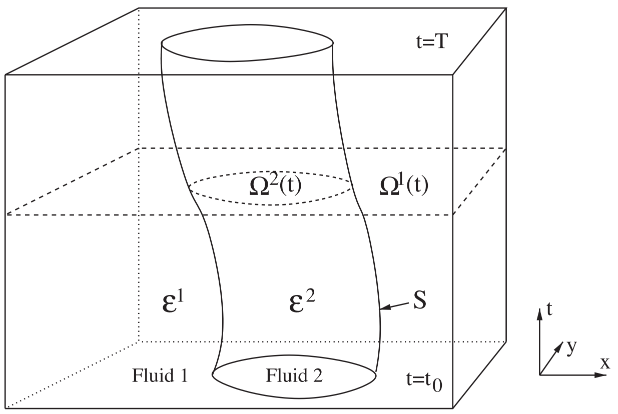

Fig. 1. An example two-fluid flow problem in space–time. Here $\mathcal{E}^i$ and $\Omega^i(t)$ denote the space–time and space flow domains for fluids $i = 1, 2$; and, S denotes the interface between the two fluids in space–time.

domain boundary as $\partial\mathcal{E} = \cup_i \partial\mathcal{E}^i$. Let $\mathbf{w}^i$ denote a vector of N_w flow variables for fluid i. The bulk fluid dynamics for fluid i are assumed to be given as a system of conservation laws:

$$\frac{\partial \mathbf{w}^i}{\partial t} + \bar{\nabla} \cdot F^i(\mathbf{w}^i) = 0, \tag{1}$$

where $\bar{\nabla} = (\partial/\partial x_1, \ldots, \partial/\partial x_d)$ denotes the spatial gradient operator and $F^i(\mathbf{w}^i) = \left(F_1^i, \ldots, F_d^i\right)$ the spatial flux tensor for fluid i with F_j^i the jth flux vector and $j = 1,\ldots,d$. Reformulated in space–time (1) becomes:

$$\begin{aligned}&\nabla \cdot \mathcal{F}^i(\mathbf{w}^i) = 0, \quad \text{with}\\ &\mathcal{F}^i(\mathbf{w}^i) = (\mathbf{w}^i, F^i(\mathbf{w}^i)),\end{aligned} \tag{2}$$

and $\nabla = (\partial/\partial t, \bar{\nabla})$ the space–time gradient operator and $\mathcal{F}^i(\mathbf{w}^i)$ the space–time flux tensor. The flow variables are subject to initial conditions:

$$\mathbf{w}^i(0, \bar{\mathbf{x}}) = \mathbf{w}_0^i(\bar{\mathbf{x}}), \tag{3}$$

boundary conditions:

$$\mathbf{w}^i(t, \bar{\mathbf{x}}) = \mathcal{B}_B^i(\mathbf{w}^i, \mathbf{w}_b^i) \quad \text{on } \mathcal{Q}^i/S \tag{4}$$

with $\mathbf{w}_b^i$ the prescribed boundary data at $\mathcal{Q}^i$, and interface conditions:

$$\mathbf{w}^i(t, \bar{\mathbf{x}}) = \mathcal{B}_S^i(\mathbf{w}^1, \mathbf{w}^2) \quad \text{on } S. \tag{5}$$

Since the actual flow variables, fluxes and initial, boundary and interface conditions are problem specific they shall be provided when the test cases are discussed.

2.2. Level set equation

To distinguish between the two fluids a level set function $\psi(\mathbf{x})$ is used:

$$\psi(t, \bar{\mathbf{x}}) = \begin{cases} < 0 & \text{in Fluid } 1, \\ > 0 & \text{in Fluid } 2, \\ = 0 & \text{at the interface.} \end{cases} \tag{6}$$

Initially, the level set function is defined as the minimum signed distance to the interface:

$$\psi(t, \bar{\mathbf{x}}) = \alpha \inf_{\forall \bar{\mathbf{x}}_S \in S(t)} \|\bar{\mathbf{x}} - \bar{\mathbf{x}}_S\|, \tag{7}$$

where $\alpha = -1$ in Fluid 1 and $\alpha = +1$ in Fluid 2, $\bar{\mathbf{x}}_S$ denotes a point at the interface $S(t)$ and $\|\cdot\|$ is the Euclidian distance. The evolution of the level set is determined by an advection equation:

$$\frac{\partial \psi}{\partial t} + \bar{\mathbf{a}} \cdot \bar{\nabla}\psi = 0, \tag{8}$$

where $\bar{\mathbf{a}} = (a_1, \ldots, a_d)$ is a vector containing the level set velocity, which will be taken equal to the flow velocity. The level set function is subject to initial conditions:

$$\psi(0, \bar{\mathbf{x}}) = \psi_0(\bar{\mathbf{x}}), \quad \text{for } \bar{\mathbf{x}} \in \Omega(t_0). \tag{9}$$

At the domain boundary the level set is subject to solid wall boundary conditions:

$$\bar{\mathbf{a}}(t, \bar{\mathbf{x}}) \cdot \bar{\mathbf{n}} = 0, \quad \text{for } (t, \bar{\mathbf{x}}) \in \mathcal{Q}, \tag{10}$$

where $\bar{\mathbf{n}}$ denotes the space outward unit normal vector at the domain boundary.

3. Meshes

3.1. Two-fluid mesh

To simplify computations, the two-fluid domain is subdivided into a number of space–time slabs on which the equations are solved consecutively. Interval (t_0, T) is subdivided into N_t intervals $I_n = (t_n, t_{n+1})$, with $t_0 < t_1 < \cdots < t_{N_t} = T$ and based on these intervals domains $\mathcal{E}^i$ are subdivided into space–time slabs $\mathcal{I}_n^i = \{\mathbf{x} \in \mathcal{E}^i | t \in I_n\}$. For every space–time slab $\mathcal{I}_n^i$ a tessellation $\mathcal{T}_h^{i,n}$ of non-overlapping space–time elements $\mathcal{K}_j^{i,n} \subset \mathbb{R}^{d+1}$ is defined:

$$\mathcal{T}_h^{i,n} = \left\{ \mathcal{K}_j^{i,n} \subset \mathbb{R}^{d+1} \,\middle|\, \bigcup_{j=1}^{N_h^i} \overline{\mathcal{K}}_j^{i,n} = \overline{\mathcal{I}}_n^i \text{ and } \mathcal{K}_j^{i,n} \bigcap \mathcal{K}_{j'}^{i,n} = \emptyset \text{ if } j \neq j',\ 1 \leqslant j, j' \leqslant N_h^{i,n} \right\} \tag{11}$$

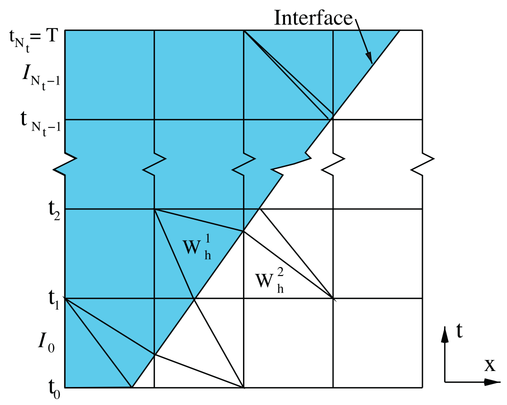

Fig. 2. Two-fluid mesh.

with $N_h^{i,n}$ the number of space–time elements in the space–time slab $\mathcal{I}_n^i$ for fluid i and where $\overline{\mathcal{K}}_j^{i,n} = \mathcal{K}_j^{i,n} \cup \partial\mathcal{K}_j^{i,n}$ denotes the closure of the space–time element. The tessellations $\mathcal{T}_h^{i,n}$, $i = 1,2$ will be referred to as the two-fluid or refined mesh $\mathcal{T}_h^n$ (see Fig. 2), since they will be constructed from a background mesh by performing local mesh refinement. The tessellations $\mathcal{T}_h^{i,n}$ define the numerical interface $\mathcal{S}_h^{i,n}$ as a collection of finite element faces. The numerical interface is assumed to be geometrically identical in both tessellations, $\mathcal{S}_h^{1,n} = \mathcal{S}_h^{2,n}$. Let $\Gamma^{i,n} = \Gamma_I^{i,n} \cup \Gamma_B^{i,n} \cup \Gamma_S^n$ denote the set of all fluid i faces $\mathcal{S}_m^{i,n}$, with $\Gamma_I^{i,n}$ the set of internal faces, $\Gamma_B^{i,n}$ the set of boundary faces, and Γ_S^n the set of interfaces. Every internal face connects to exactly two elements in $\mathcal{T}_h^{i,n}$, denoted as the left element $\mathcal{K}^l$ and the right element $\mathcal{K}^r$. Every boundary face connects to one element in $\mathcal{T}_h^{i,n}$, denoted as the element $\mathcal{K}^l$. Every interface connects to one element from $\mathcal{T}_h^{1,n}$ and also to one element from $\mathcal{T}_h^{2,n}$.

The finite element space $B_h^k(\mathcal{T}_h^{i,n})$ associated with the tessellation $\mathcal{T}_h^{i,n}$ is defined as:

$$B_h^k\left(\mathcal{T}_h^{i,n}\right) = \left\{\mathbf{w} \in L^2\left(\mathcal{E}_h^i\right) : \mathbf{w}|_{\mathcal{K}} \circ G_{\mathcal{K}} \in P^k(\widehat{\mathcal{K}}),\ \forall \mathcal{K} \in \mathcal{T}_h^{i,n}\right\} \tag{12}$$

with $\mathcal{E}_h^i$ the discrete flow domain, $L^2\left(\mathcal{E}_h^i\right)$ the space of square integrable functions on $\mathcal{E}_h^i$, and $P^k(\widehat{\mathcal{K}})$ the space of polynomials of degree at most k in the reference element $\widehat{\mathcal{K}}$. The mapping $G_{\mathcal{K}_j^{i,n}}$ relates every element $\mathcal{K}_j^{i,n}$ to a reference element $\widehat{\mathcal{K}} \subset \mathbb{R}^{d+1}$:

$$G_{\mathcal{K}_j^{i,n}} : \widehat{\mathcal{K}} \to \mathcal{K}_j^{i,n} : \xi \mapsto \mathbf{x} = \sum_{k=1}^{N_{F,j}^{i,n}} x_k(\mathcal{K}_j^{i,n})\chi_k(\xi) \tag{13}$$

with $N_{F,j}^{i,n}$ the number of vertices and $x_k(\mathcal{K}_j^{i,n})$ the coordinates of the vertices of space–time element $\mathcal{K}_j^{i,n}$. The finite element shape functions $\chi_k(\xi)$ are defined on the reference element $\widehat{\mathcal{K}}$, with $\xi = (\xi_0, \ldots, \xi_d)$ the coordinates in the reference element. Given a set of basis functions $\hat{\phi}_m$ defined on the reference element, the basis functions $\phi_m : \mathcal{K}_j^{i,n} \to \mathbb{R}$ are defined on the space–time elements $\mathcal{K}_j^{i,n} \in \mathcal{T}_h^{i,n}$ by means of the mapping $G_{\mathcal{K}_j^{i,n}}$:

$$\phi_m = \hat{\phi}_m \circ G_{\mathcal{K}_j^{i,n}}^{-1}. \tag{14}$$

On the two-fluid mesh the approximated flow variables are defined as:

$$\mathbf{w}_h^i(t,\bar{\mathbf{x}})|_{\mathcal{K}_j^{i,n}} = \sum_m \widehat{\mathbf{W}}_m^i(\mathcal{K}_j^{i,n})\phi_m(t,\bar{\mathbf{x}}) \tag{15}$$

with $\widehat{\mathbf{W}}_m^i$ the expansion coefficients of fluid i. Each element in the two-fluid mesh contains a single fluid. Therefore, in every element one set of flow variables is defined. Because the basis functions are defined locally in every element the space–time flow solution is discontinuous at the element faces.

3.2. Background mesh

In the construction of the two-fluid mesh $\mathcal{T}_h^n$ it was assumed that every element contains exactly one fluid or equivalently that the interface is represented by a set of finite element faces. In order to define a mesh which satisfies this requirement, a level set function ψ_h is defined on a space–time background mesh $\mathcal{T}_b^n$.

For every space–time slab $\mathcal{I}_n$ a tessellation $\mathcal{T}_b^n$ of space–time elements $\mathcal{K}_{b,\tilde{j}}^n \subset \mathbb{R}^{d+1}$ is defined:

$$\mathcal{T}_b^n = \left\{ \mathcal{K}_{b,\tilde{j}}^n \subset \mathbb{R}^{d+1} \left| \bigcup_{\tilde{j}=1}^{N_b} \overline{\mathcal{K}}_{b,\tilde{j}}^n = \overline{\mathcal{I}}_n \text{ and } \mathcal{K}_{b,\tilde{j}}^n \bigcap \mathcal{K}_{b,\tilde{j}'}^n = \emptyset \text{ if } \tilde{j} \neq \tilde{j}',\ 1 \leqslant \tilde{j},\ \tilde{j}' \leqslant N_b \right. \right\} \tag{16}$$

with N_b the number of space–time elements. The tessellation $\mathcal{T}_b^n$ will be referred to as the background mesh. In two and three space–time dimensions the background mesh is composed of square and cube shaped elements, respectively. The finite element space, mappings and basis functions are identical to those defined for the refined mesh in Section 3.1 except when dealing with the background mesh these will be denoted using a subscript b. On the background mesh a discontinuous Galerkin approximation of the level set is defined as:

$$\psi_h(t,\bar{\mathbf{x}})|_{\mathcal{K}_{b,\bar{j}}^n} = \sum_m \widehat{\mathbf{\Psi}}_m(\mathcal{K}_{b,\bar{j}}^n)\phi_m(t,\bar{\mathbf{x}}), \tag{17}$$

with $\widehat{\mathbf{\Psi}}_m$ the level set expansion coefficients. A discontinuous Galerkin discretization is used because the level set is advected with the flow velocity and will develop discontinuities in the vicinity of shock waves. In addition, a discontinuous Galerkin approximation of the level set velocity is defined as (later the flow velocity projected on the background mesh):

$$\bar{\mathbf{a}}_h(t,\bar{\mathbf{x}})|_{\mathcal{K}_{b,\bar{j}}^n} = \sum_m \widehat{\mathbf{A}}_m(\mathcal{K}_{b,\bar{j}}^n)\phi_m(t,\bar{\mathbf{x}}). \tag{18}$$

3.3. Mesh refinement

After solving the level set equation the interface shape and position are approximately known from the 0-level set. In order to define a mesh for two-fluid flow computations, the background mesh is refined by means of cut-cell mesh refinement. In the refined mesh the interface is represented by a set of faces on which the level set value is approximately zero.

The discontinuous nature of the level set approximation is not desirable for the mesh refinement, since it can result in hanging nodes. Hence the level set is smoothed before performing the mesh refinement. Assuming computations have reached time slab $\mathcal{I}_n$ the level set approximation ψ_h is smoothed by first looping over all elements in $\mathcal{I}_n$ and storing the multiplicity and the sum of the values of ψ_h in each vertex. For every vertex in $\mathcal{I}_n$ the continuous level set value ψ_h^c is calculated by dividing the sum of the ψ_h values by the vertex multiplicity. In every background element in $\mathcal{I}_n$, ψ_h is then reinitialized using the ψ_h^c values in the element vertices. To ensure continuity of the mesh only the values of the level set in the background elements belonging to the previous time slab $\mathcal{I}_{n-1}$ are used at the faces between the previous and the current time slab.

The mesh refinement algorithm is defined in Algorithm 1. The algorithm consists of a global element refinement step, in which all the elements of the background mesh are refined consecutively according to a set of refinement rules, followed by a face generation step to create the connectivity between the refined elements. The refinement rules define how a single element will be refined given an intersection with a 0-level set. The face generation is straightforward and will not be discussed.

Algorithm 1. Mesh refinement algorithm

```
FOR every element $\mathcal{K}_{b,\bar{j}}^n$ in $\mathcal{T}_b^n$ DO
  Calculate intersection of 0-level set $\psi_c = 0$ with $\mathcal{K}_{b,\bar{j}}^n$
  Select refinement rule
  Create and store interface physical nodes $\mathbf{x}_I$
  FOR all child elements $\hat{j}$ defined by the refinement rule DO
    Create $\mathcal{K}_{h,\hat{j}}^{i,n}$ and store in $\mathcal{T}_h^{i,n}$
  END DO
END DO
Generate faces for $\mathcal{T}_h^{i,n}$
FOR every element $\mathcal{K}_{h,j}^{i,n}$ in $\mathcal{T}_h^{i,n}$
  Initialize data on $\mathcal{K}_{h,j}^{i,n}$
END DO
```

Given a smoothed level set, the element refinement is executed separately for each background element. For a given background element, it is first checked if the element contains more than one fluid by evaluating the level set at each vertex of the element. If the level set has the same sign in every vertex, the element contains only one fluid and is copied directly to the refined mesh $\mathcal{T}_h^n$. Alternatively, the type of cut is determined from the level set signs. Depending on the cut type, the element is refined, based on a predefined element refinement rule for that type, see Sections 3.4 and 3.5, and the cut coordinates. The resulting elements are stored in $\mathcal{T}_h^n$. The element refinement rules have been designed such that for two neighboring elements the shared face is refined identically at both sides. Hence, no hanging nodes will occur in the refined mesh. The interface cut coordinates $\mathbf{x}_I$ for an edge cut by the interface are calculated as:

$$\mathbf{x}_I = \frac{\mathbf{x}_A\psi_h(\mathbf{x}_B) - \mathbf{x}_B\psi_h(\mathbf{x}_A)}{\psi_h(\mathbf{x}_A) - \psi_h(\mathbf{x}_B)}, \tag{19}$$

where $\mathbf{x}_A$ and $\mathbf{x}_B$ denote the coordinates of the edge vertices. For simplicity it is assumed that the level set is non-zero and can only be positive or negative in the vertices.

Because the refinement type is only based on the level set signs in the background element vertices, in cases where more than one interface intersects an element an ambiguity will occur where exactly the interface lies and the refinement rule will give rise to elements for which the fluid type is ambiguous. However, the fluid types of these elements can easily be found by computing the level set signs in the element midpoints.

The mesh refinement algorithm allows for freedom in choosing the element refinement rules. However, the refined mesh should have full connectivity to avoid difficulties with face integration. Element refinement rules have been developed for two and three dimensions, similar to [20], and these will be discussed next for a set of base types. In the implementation each cut is linked to one of these base types by means of the rotational and translational symmetries of the background element for which algorithmic details are available in [54].

3.4. 2D refinement

In 2D the background mesh consists of square elements. The classification of the 2D cuts is based on the values of the level set in the four vertices of the square. Each cut type is defined as a series of four signs corresponding to the level set signs in the four vertices. For example one type is defined by $--++$. Switching to a binary representation with $-$ and $+$ corresponding to 0 and 1, respectively, we can assign the number 0011 = 3. Since a square has four vertices, there are $2^4 = 16$ possibilities. In 2D three base types have been defined as given in Table 1. In Fig. 3 (top) the signs of the level set in each vertex for every type are shown. Level set configuration 2 allows for two possible linear interface cuts both of which are handled by the element refinement rule. The element refinements for the 2D base types are shown in Fig. 3 (bottom) and defined in Table 2.

3.5. 3D refinement

In 3D the background mesh consists of cubical elements. Like in the 2D refinement, the 3D types are classified based on the values of the level set in the vertices. Thirteen configurations were identified, and these are given in Table 3. In Fig. 4 the signs of the level set in each vertex for every base type are shown. It should be noted that level set configurations 6–12 allow for multiple interface cuts. This ambiguity is solved by making sure that for each level set configuration the element refinement rule is such that also multiple element cuts can be handled. The corresponding interfaces are shown in Fig. 5. In order to define the element refinement of the 13 base types, first a surface refinement is defined, which is based on the 2D refinements illustrated in Fig. 3. Element refinements have been manually devised based on the surface refinements. The element

Table 1
Binary codes of the 2D base types. Each code represents a combination of level set signs for each of the 4 background element vertices, where a negative (positive) level set sign is represented by a 0 (1).

Index	Binary code	Number
0	0111	7
1	0011	3
2	0110	6

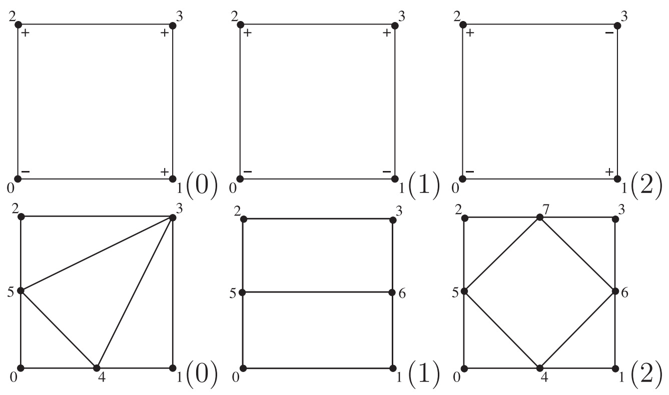

Fig. 3. The vertex level set signs (top) and the corresponding element refinements (bottom) for the 2D base types.

Table 2
2D base type element refinements.

Type index	Child index	Child LNI	Fluid type
0	0	{0,4,5}	0
	1	{4,1,3}	1
	2	{5,3,2}	1
	3	{5,4,3}	1
1	0	{0,1,5,6}	0
	1	{5,6,2,3}	1
2	0	{0,4,5}	0
	1	{4,1,6}	1
	2	{6,3,7}	1
	3	{7,2,5}	0
	4	{5,4,7,6}	0 or 1

Table 3
Binary codes of the 3D base types. Each code represents a combination of level set signs for each of the 8 background element vertices, where a negative (positive) level set sign is represented by a 0 (1).

Index	Binary code	Number	Index	Binary code	Number
0	00100000	32	7	00100100	36
1	00100010	34	8	01100100	100
2	10100010	162	9	10100101	165
3	10101010	170	10	00101101	45
4	10110010	178	11	00101001	41
5	10100011	163	12	01101001	105
6	00101000	40			

refinements for the 13 base types are given in Tables 4 and 5. In some of the refinements an additional node is used, which is located at the interface center and has local node index (LNI) 20. Due to the high complexity the 3D element refinements are not illustrated [54].

3.6. Merging

The occurrence of small elements in the refined mesh tends to cause numerical stability and performance problems. To solve these problems an element merging procedure was developed.

Let $\mathcal{K}_k^{i,n},\ k = 0,\dots,N_{\hat{j}}$ denote a collection of elements which need be merged, determined by means of a merging strategy to be discussed later. The merged element $\mathcal{K}_{m,\hat{j}}^{i,n}$ is defined as:

$$\mathcal{K}_{m,\hat{j}}^{i,n} = \bigcup_{k=0}^{N_{\hat{j}}} \mathcal{K}_k^{i,n}. \tag{20}$$

For each merged element $\mathcal{K}_{m,\hat{j}}^{i,n}$ the minimum and maximum bounding points $\mathbf{x}_{\hat{j}}^{min}$ and $\mathbf{x}_{\hat{j}}^{max}$ are defined componentwise as:

$$x_{\hat{j},l}^{min} = \min_{\forall \mathbf{x} \in \mathcal{K}_{m,\hat{j}}^{i,n}} x_l, \quad x_{\hat{j},l}^{max} = \max_{\forall \mathbf{x} \in \mathcal{K}_{m,\hat{j}}^{i,n}} x_l, \ \ l = 0,\dots,d, \tag{21}$$

with d the space dimension. Let $\mathbf{x}_{\bar{j}}^{min}$ and $\mathbf{x}_{\bar{j}}^{max}$ denote the minimum and maximum bounding points of background element $\mathcal{K}_{b,\bar{j}}^{n}$. It is assumed that all background mesh elements are of equal size and shape; hence, $\mathbf{x}_{\bar{j}}^{max} - \mathbf{x}_{\bar{j}}^{min} = \mathbf{h}_{b,\bar{j}} = \mathbf{h}_b = constant$. For each merged element the minimum and maximum lengths relative to the background element are defined as:

$$\epsilon_{\hat{j}}^{min} = \min_{l=0,\dots,d} \frac{x_{\hat{j},l}^{max} - x_{\hat{j},l}^{min}}{h_{b,l}}, \ \epsilon_{\hat{j}}^{max} = \min_{l=0,\dots,d} \frac{x_{\hat{j},l}^{max} - x_{\hat{j},l}^{min}}{h_{b,l}}. \tag{22}$$

In addition two predefined parameters, $\epsilon^{MIN} = 0.9$ and $\epsilon^{MAX} = 1.9$, are introduced. The merging strategy is defined for each fluid i individually as follows:

- Step 1: For each background element $\mathcal{K}_{b,\bar{j}}^{n}$ retrieve the collection of all child elements that contain fluid i. For this collection of elements compute ϵ^{min} and ϵ^{max} and store these values on the background element. If the background element does not contain fluid i elements it is unavailable for merging and $\epsilon^{min} = \epsilon^{max} = 0.0$. If $\epsilon^{min} < \epsilon^{MIN}$ the collection defines a small or

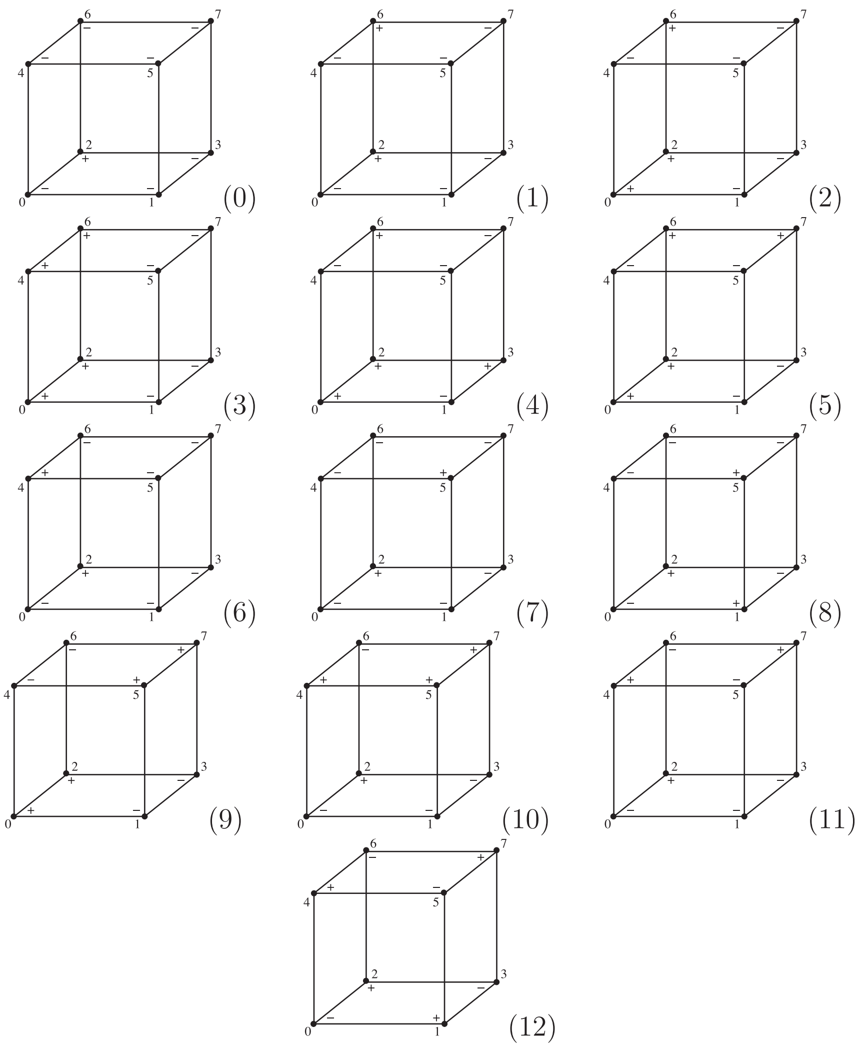

Fig. 4. The vertex level set signs for the 3D base types.

thin merged element and requires merging involving one or more neighboring background elements. If $\epsilon^{min} > \epsilon^{MIN}$ the collection itself defines a valid merged element. Step 1 is illustrated in Fig. 6.

- Step 2: Using a loop over the faces in the background mesh, it is determined for each background element $\mathcal{K}^n_{b,\tilde{j}}$ which neighboring elements $\mathcal{K}^n_{b,k},\ k = 0,\ldots,N_{\tilde{j}}$ are usable for merging, which is the case if the neighboring element contains a collection of fluid i elements with $\epsilon^{min} > \epsilon^{MIN}$. Step 2 is illustrated in Fig. 7.
- Step 3: The merged elements are determined in three steps. Each step corresponds to a different type of merging, and these are illustrated in Fig. 8. After a background element has been used in merging it is marked as UNAVAILABLE.
  - Type 1: For each available individual background element $\mathcal{K}^n_{b,\tilde{j}}$ check if $\epsilon^{min} > \epsilon^{MIN}$ and if it has at least two available neighboring elements for which $\epsilon^{min} < \epsilon^{MIN}$. If so, merge all refined elements $\mathcal{K}^{i,n}_j$ with the correct fluid type i contained in these background elements.
  - Type 2: For each available individual background element $\mathcal{K}^n_{b,\tilde{j}}$ check if $\epsilon^{min} < \epsilon^{MIN}$. If so loop over all available neighboring elements $\mathcal{K}^n_{b,k},\ k = 0,\ldots,N_{\tilde{j}}$ with $N_{\tilde{j}}$ the number of available neighboring elements. For each combination of the background element $\mathcal{K}^n_{b,\tilde{j}}$ and a neighboring element $\mathcal{K}^n_{b,k}$ determine ϵ^{min}_k. Find the $\tilde{k}$ for the combination which has the largest size, $\epsilon^{min}_{\tilde{k}} > \epsilon^{min}_k,\ k = 0,\ldots,N_{\tilde{j}}$. Merge all refined elements $\mathcal{K}^{i,n}_j$ with the correct fluid type i contained in the background elements $\mathcal{K}^n_{b,\tilde{j}}$ and $\mathcal{K}^n_{b,\tilde{k}}$.
  - Type 3: For each available individual background element $\mathcal{K}^n_{b,\tilde{j}}$ check if $\epsilon^{min} > \epsilon^{MIN}$. If so check if it contains more than one element $\mathcal{K}^{i,n}_j$ with the correct fluid type i and if so merge these elements.

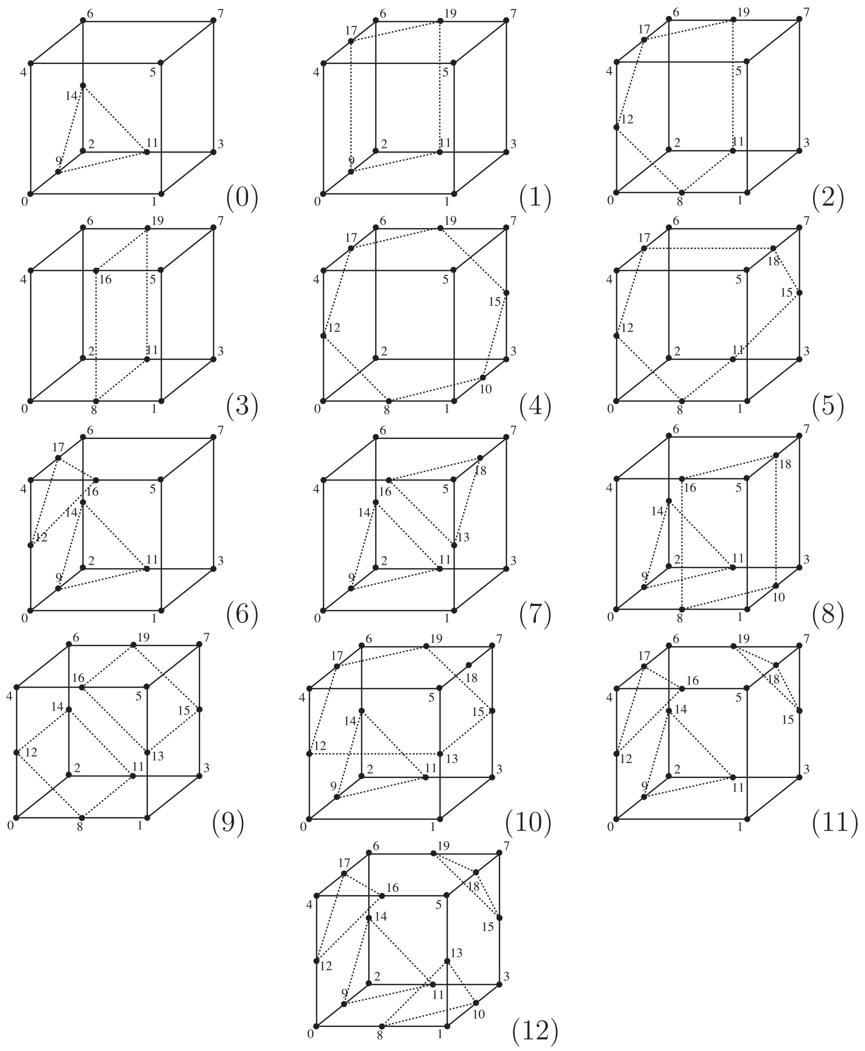

Fig. 5. The interface cuts for the 3D base types. For types 6–12 the level set configuration allows for alternative cuts not shown here, which are supported by the element refinement rule for that type.

The merged elements tend to have complex shapes which makes it difficult to find suitable reference elements and basis functions. To alleviate this problem a bounding box element is introduced [16], which is simple shaped and contains the merged element. This merging procedure is illustrated for two dimensions in Fig. 9 and an example of a mesh with merged elements in two dimensions is shown in Fig. 10.

Let $\mathcal{K}^{i,n}_{M,\hat{j}}$ denote the bounding box of the merging element $\mathcal{K}^{i,n}_{m,\hat{j}}$. The finite element space, mappings and basis functions used for the bounding box elements are identical to those defined for the refined mesh but will be denoted using a subscript M. On the bounding box element the approximated flow variables are defined as:

$$\mathbf{w}^i_h(t,\bar{\mathbf{x}})|_{\mathcal{K}^{i,n}_{M,\hat{j}}} = \sum_m \widehat{\mathbf{W}}^i_m\left(\mathcal{K}^{i,n}_{M,\hat{j}}\right)\phi_m(t,\bar{\mathbf{x}}) \tag{23}$$

with $\widehat{\mathbf{W}}^i_m$ the flow coefficients of fluid i. Each merged element contains exactly one fluid. For all elements $\mathcal{K}^{i,n}_k \subset \mathcal{K}^{i,n}_{m,\hat{j}}$ the flow evaluation is redefined as an evaluation in the bounding box element:

$$\mathbf{w}^i_h(\mathbf{x})|_{\mathcal{K}^{i,n}_k} = \mathbf{w}^i_h(\mathbf{x})|_{\mathcal{K}^{i,n}_{M,\hat{j}}}. \tag{24}$$

Integration of a function $f(\mathbf{w}^i_h)$ over a merged element $\mathcal{K}^{i,n}_{m,\hat{j}}$ is performed by integrating over all the individual elements and summing the contributions:

Table 4
Element refinements for 3D base types.

Type index	Child index	Child LNI	Fluid type
0	0	{11,1,3,5,7}	0
	1	{9,0,1,4,5}	0
	2	{1,5,9,11}	0
	3	{2,9,11,14}	1
	4	{14,4,5,6,7}	0
	5	{4,5,9,14}	0
	6	{5,7,11,14}	0
	7	{5,9,11,14}	0
1	0	{0,1,9,4,5,17}	0
	1	{1,3,11,5,7,19}	0
	2	{1,11,9,5,19,17}	0
	3	{2,9,11,6,17,19}	1
2	0	{20,8,1,11,3}	0
	1	{20,0,8,2,11}	1
	2	{4,12,17,20}	0
	3	{12,0,2,20}	1
	4	{6,17,2,20}	1
	5	{12,2,17,20}	1
	6	{12,8,0,20}	1
	7	{8,5,1,20}	0
	8	{12,5,8,20}	0
	9	{4,5,12,20}	0
	10	{20,1,5,3,7}	0
	11	{20,2,11,6,19}	1
	12	{20,11,3,19,7}	0
	13	{17,5,4,20}	0
	14	{19,7,5,20}	0
	15	{6,19,17,20}	1
	16	{17,19,5,20}	0
3	0	{0,8,2,11,4,16,6,19}	1
	1	{8,1,11,3,16,5,19,7}	0
4	0	{0,8,2,12}	1
	1	{1,8,5,10}	0
	2	{2,10,3,15}	1
	3	{2,6,17,19}	1
	4	{2,19,15,8,10}	1
	5	{2,17,19,12,8}	1
	6	{4,5,12,17}	0
	7	{5,7,15,19}	0
	8	{5,8,10,19,15}	0
	9	{5,12,8,17,19}	0
5	0	{20,0,8,2,11}	1
	1	{20,8,1,11}	0
	2	{0,2,12,20}	1
	3	{6,17,2,20}	1
	4	{4,12,17,20}	0
	5	{12,2,17,20}	1
	6	{0,12,8,20}	1
	7	{1,8,5,20}	0
	8	{4,5,12,20}	0
	9	{8,12,5,20}	0
	10	{7,15,18,20}	1
	11	{3,1,15,20}	0
	12	{5,18,1,20}	0
	13	{18,15,1,20}	0
	14	{2,11,6,20}	1
	15	{3,15,11,20}	0
	16	{7,6,15,20}	1
	17	{11,15,6,20}	1
	18	{20,7,18,6,17}	1
	19	{20,18,5,17,4}	0
6	0	{1,11,9,20}	0
	1	{1,3,11,20}	0
	2	{20,9,14,12,17}	0 or 1
	3	{5,1,16,20}	0
	4	{12,16,1,20}	0
	5	{20,5,7,1,3}	0
	6	{3,7,11,20}	0
	7	{14,11,7,20}	0
	8	{5,16,7,20}	0
	9	{16,17,7,20}	0
	10	{9,14,11,2}	1
	11	{9,11,14,20}	0 or 1
	12	{12,16,17,4}	1
	13	{12,17,16,20}	0 or 1
	14	{7,14,17,6}	0
	15	{7,17,14,20}	0
	16	{1,12,9,0}	0
	17	{1,9,12,20}	0
7	0	{0,1,9,20}	0
	1	{1,11,9,20}	0
	2	{1,3,11,20}	0
	3	{0,9,4,20}	0
	4	{9,14,4,20}	0
	5	{14,6,4,20}	0
	6	{0,1,13,20}	0
	7	{13,16,0,20}	0
	8	{4,0,16,20}	0
	9	{1,13,3,20}	0
	10	{18,3,13,20}	0
	11	{7,3,18,20}	0
	12	{3,11,7,20}	0
	13	{6,7,14,20}	0
	14	{14,7,11,20}	0
	15	{4,6,16,20}	0
	16	{16,6,18,20}	0
	17	{18,6,7,20}	0
	18	{9,11,14,20}	0
	19	{9,14,11,2}	0 or 1
	20	{13,16,18,20}	1
	21	{13,18,16,5}	0 or 1

$$\int_{\mathcal{K}^{i,n}_{m,\hat{j}}} f(\mathbf{w}^i_h)\,d\mathcal{K} = \sum_{k=0}^{N_j} \int_{\mathcal{K}^{i,n}_{k}} f(\mathbf{w}^i_h)\,d\mathcal{K}. \tag{25}$$

Hence, there are no hanging nodes.

4. Space–time discontinuous Galerkin discretization

4.1. Flow discretization

The discontinuous Galerkin finite element approximation for two-fluid flows on the refined mesh $\mathcal{T}^{i,n}_h$ is found by multiplying (2) with an arbitrary test function $\mathbf{v} \in B^k_h(\mathcal{T}^{i,n}_h)$ and integrating over all elements in the domains $\mathcal{E}^1$ and $\mathcal{E}^2$; further application of Gauss' theorem results in:

Table 5
Element refinements for 3D base types (continued).

Type index	Child index	Child LNI	Fluid type	Type index	Child index	Child LNI	Fluid type
8	0	{1,10,8,5,18,16}	1		6	{0,12,1,20}	0
	1	{14,16,18,8,10}	0 or 1		7	{12,16,1,20}	0
	2	{16,18,14,6}	0		8	{16,5,1,20}	0
	3	{14,8,10,9,11}	0 or 1		9	{5,18,1,20}	0
	4	{4,16,14,6}	0		10	{18,15,1,20}	0
	5	{4,14,16,9}	0		11	{15,3,1,20}	0
	6	{14,8,16,9}	0 or 1		12	{3,15,11,20}	0
	7	{9,4,16,0,8}	0		13	{6,14,19,20}	0
	8	{18,7,14,6}	0		14	{20,11,15,14,19}	0 or 1
	9	{18,14,7,11}	0		15	{5,16,18,20}	0
	10	{14,10,11,18}	0 or 1		16	{6,19,17,20}	0
	11	{11,18,7,10,3}	0		17	{20,17,19,16,18}	0 or 1
	12	{2,9,11,14}	1		18	{9,11,14,20}	0 or 1
9	0	{2,11,14,0,8,12}	1		19	{12,17,16,20}	0 or 1
	1	{3,15,11,1,13,8}	0		20	{18,19,15,20}	0 or 1
	2	{7,19,15,5,16,13}	1		21	{9,14,11,2}	1
	3	{6,14,19,4,12,16}	0		22	{12,16,17,4}	1
	4	{11,15,14,19,8,13,12,16}	0 or 1		23	{18,15,19,7}	1
10	0	{0,1,9,20}	0	12	0	{0,8,9,20}	0
	1	{9,1,11,20}	0		1	{3,11,10,20}	0
	2	{3,11,1,20}	0		2	{20,8,10,9,11}	0 or 1
	3	{0,9,12,20}	0		3	{0,9,12,20}	0
	4	{6,17,14,20}	0		4	{6,17,14,20}	0
	5	{20,12,9,17,14}	0 or 1		5	{20,12,9,17,14}	0 or 1
	6	{20,12,13,0}	0		6	{0,12,8,20}	0
	7	{20,13,15,1,3}	0		7	{5,13,16,20}	0
	8	{3,15,11,20}	0		8	{20,16,13,12,8}	0 or 1
	9	{6,14,19,20}	0		9	{3,10,15,20}	0
	10	{20,11,15,14,19}	0 or 1		10	{5,18,13,20}	0
	11	{6,17,19,20}	0		11	{20,18,15,13,10}	0 or 1
	12	{9,11,14,20}	0 or 1		12	{3,15,11,20}	0
	13	{9,14,11,2}	1		13	{6,14,19,20}	0
	14	{12,17,13,20}	0 or 1		14	{20,11,15,14,19}	0 or 1
	15	{17,19,13,20}	0 or 1		15	{6,19,17,20}	0
	16	{19,15,13,20}	0 or 1		16	{5,16,18,20}	0
	17	{19,17,13,5}	1		17	{20,17,19,16,18}	0 or 1
	18	{17,4,5,12,13}	1		18	{9,11,14,20}	0 or 1
	19	{19,5,7,13,15}	1		19	{12,17,16,20}	0 or 1
11	0	{0,1,9,20}	0		20	{8,13,10,20}	0 or 1
	1	{9,1,11,20}	0		21	{18,19,15,20}	0 or 1
	2	{3,11,1,20}	0		22	{9,14,11,2}	1
	3	{0,9,12,20}	0		23	{12,16,17,4}	1
	4	{6,17,14,20}	0		24	{8,10,13,1}	1
	5	{20,12,9,17,14}	0 or 1		25	{18,15,19,7}	1

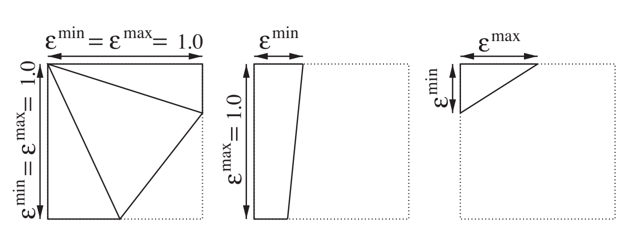

Fig. 6. Illustration of the first step in the merging strategy. The dotted lines represent the background element and the solid lines represent the collection of child elements of one of the fluid types. The collection on the left has an $\epsilon^{min} > \epsilon^{MIN}$ and hence is considered a valid merged element in itself. The collections in the middle and on the right are have a small ϵ^{min} and require merging with a neighboring element.

$$
\begin{aligned}
&-\sum_{\mathcal{K}_j^{i,n}\in\mathcal{T}_h^{i,n}}\int_{\mathcal{K}_j^{i,n}}\nabla\mathbf{v}\cdot\mathcal{F}^i(\mathbf{w}^i)d\mathcal{K}+\sum_{\mathcal{S}_m^{i,n}\in\Gamma_I^{i,n}}\int_{\mathcal{S}_m^{i,n}}\mathcal{F}^{i,l}(\mathbf{w}^{i,l})\cdot\mathbf{n}_{\mathcal{K}}^l\,\mathbf{v}^l+\mathcal{F}^{i,r}(\mathbf{w}^{i,r})\cdot\mathbf{n}_{\mathcal{K}}^r\mathbf{v}^r\,d\mathcal{S}+\sum_{\mathcal{S}_m^{i,n}\in\Gamma_B^{i,n}}\int_{\mathcal{S}_m^{i,n}}\mathcal{F}^{i,l}(\mathbf{w}^{i,l})\cdot\mathbf{n}_{\mathcal{K}}^l\,\mathbf{v}^l\,d\mathcal{S}\\
&+\sum_{\mathcal{S}_m^{i,n}\in\Gamma_S^{i,n}}\int_{\mathcal{S}_m^{i,n}}\mathcal{F}^{i,l}(\mathbf{w}^{i,l})\cdot\mathbf{n}_{\mathcal{K}}^l\,\mathbf{v}^l\,d\mathcal{S}=0, \qquad (26)
\end{aligned}
$$

where $\mathcal{F}^{i,K}$ and $\mathbf{w}^{i,K}$ are the limiting trace values at the face $\mathcal{S}$ of element $\mathcal{K}^{i,K}$, $K = l, r$.

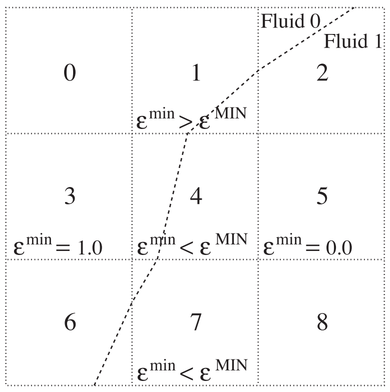

Fig. 7. Illustration of the second step in the merging strategy for fluid type 0 and background element 4. The background elements are shown in dotted lines and the 0-level set is shown as a dashed line. Background element 4 has an $\epsilon^{min} < \epsilon^{MIN}$ and hence requires merging with one or more of the neighboring elements 1, 3, 5 and 7. Elements 1 and 3 both contain enough fluid 0 ($\epsilon^{min} > \epsilon^{MIN}$) and hence are valid candidates for merging, while element 5 and element 7 do not contain enough fluid 0 ($\epsilon^{min} < \epsilon^{MIN}$) and hence are invalid candidates for merging.

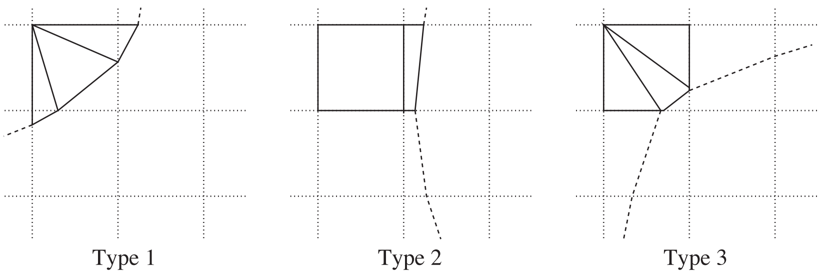

Fig. 8. The three types of merged elements. The solid lines represent the refined elements that will be combined into a single merged element. The dotted lines represent the background mesh and the dashed lines represent the interface at positions not occupied by the merged element.

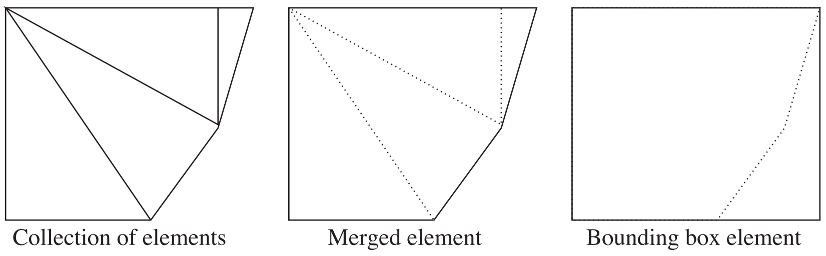

Fig. 9. A collection of elements, their merged element and its bounding box element, in physical space.

Let the trace v_h^K of a function v_h on a face $\mathcal{S}$ with respect to the element $\mathcal{K}^K$, $K = l, r$ be defined as $v_h^K = \lim_{\epsilon\downarrow 0} v_h(\mathbf{x} - \epsilon\mathbf{n}_{\mathcal{K}}^K)$, where $\mathbf{n}_{\mathcal{K}}^K = (n_0, \ldots, n_d)$ is the space–time outward unit normal vector at the face $\mathcal{S}$ with respect to element $\mathcal{K}^K$. Left and right normal vectors of a face are related as $\mathbf{n}_{\mathcal{K}}^l = -\mathbf{n}_{\mathcal{K}}^r$. The element local trace $v_h^{\pm}$ of a function v_h on a face $\mathcal{S}$ is defined as $v_h^{\pm} = \lim_{\epsilon\downarrow 0} v_h(\mathbf{x} \pm \epsilon\mathbf{n}_{\mathcal{K}})$. The average $\{\{F\}\}$ of a scalar or vector function F on the face $\mathcal{S}_m \in \Gamma_I$ is defined as $\{\{F\}\} := \frac{1}{2}(F^l + F^r)$, where l and r denote the traces at elements $\mathcal{K}^l$ and $\mathcal{K}^r$, respectively. The jump $[[F]]$ of a scalar function F on the face $\mathcal{S}_m \in \Gamma_I$ is defined as $[[F]] := F^l\mathbf{n}^l + F^r\mathbf{n}^r$ and the jump $[[\mathbf{G}]]$ of a vector function $\mathbf{G}$ on the face $\mathcal{S}_m \in \Gamma_I$ is defined as $[[\mathbf{G}]] := \mathbf{G}^l \cdot \mathbf{n}^l + \mathbf{G}^r \cdot \mathbf{n}^r$. The jump operator satisfies on Γ_I the product rule $[[F\mathbf{G}]] = \{\{F\}\}[[\mathbf{G}]] + [[F]]\{\{\mathbf{G}\}\}$.

By using a conservative flux, $\mathcal{F}^l(\mathbf{w}^l) \cdot \mathbf{n}_{\mathcal{K}}^l = -\mathcal{F}^r(\mathbf{w}^r) \cdot \mathbf{n}_{\mathcal{K}}^r$; hence, $[[\mathcal{F}(\mathbf{w})]] = 0$, the integration over the internal faces is rewritten as:

$$\sum_{\mathcal{S}_m^{i,n} \in \Gamma_I^{i,n}} \int_{\mathcal{S}_m^{i,n}} \mathcal{F}^{i,l}(\mathbf{w}^{i,l}) \cdot \mathbf{n}_{\mathcal{K}}^l\, \mathbf{v}^l + \mathcal{F}^{i,r}(\mathbf{w}^{i,r}) \cdot \mathbf{n}_{\mathcal{K}}^r\, \mathbf{v}^r d\mathcal{S} = \sum_{\mathcal{S}_m^{i,n} \in \Gamma_I^{i,n}} \int_{\mathcal{S}_m^{i,n}} \{\{\mathcal{F}^i(\mathbf{w}^i)\}\} \cdot [[\mathbf{v}]] d\mathcal{S}. \tag{27}$$

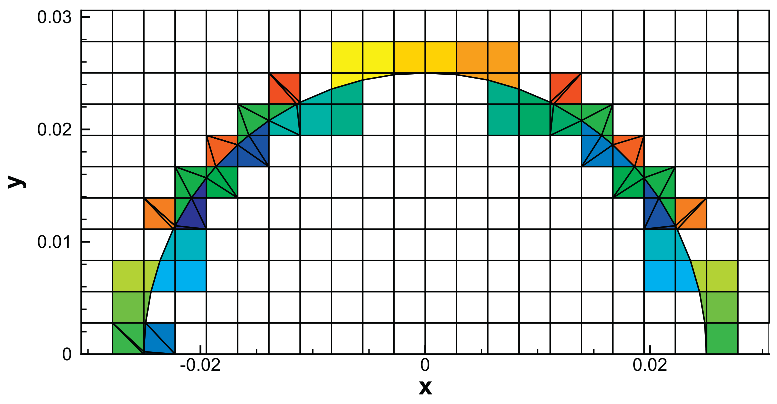

Fig. 10. Refined mesh showing the merged elements as colored collections of child elements.

So far the formulation (26) has been strictly local, in the sense that neighboring elements and also the initial, boundary and interface conditions are not incorporated. In order to do this, numerical fluxes are introduced. At internal faces the flux in (27) is replaced by a numerical flux $\mathcal{H}_I^i(\mathbf{w}^{i,l}, \mathbf{w}^{i,r}, \mathbf{n}_\mathcal{K})$, which is consistent: $\mathcal{H}(\mathbf{w}, \mathbf{w}, \mathbf{n}_\mathcal{K}) = \mathcal{F}(\mathbf{w}) \cdot \mathbf{n}_\mathcal{K}^l$, and conservative. Likewise at the boundary faces the flux is replaced by a numerical flux $\mathcal{H}_B^i(\mathbf{w}^{i,l}, \mathbf{w}_b^{i,r}, \mathbf{n}_\mathcal{K})$, which is also consistent. At the interface the flux is replaced by a numerical interface flux $\mathcal{H}_S^i(\mathbf{w}^{i,l}, \mathbf{w}_s^{i,r}, \mathbf{n}_\mathcal{K})$, with $\mathbf{w}_s^{i,r}$ the ghost state at the interface for fluid i. Using the fact that for a conservative flux $\{\{\mathcal{H}(\mathbf{w}^l, \mathbf{w}^r, \mathbf{n}_\mathcal{K})\}\} = \mathcal{H}(\mathbf{w}^l, \mathbf{w}^r, \mathbf{n}_\mathcal{K})$ and replacing the trial and test functions by their approximations in the finite element space $B_h^k(\mathcal{T}_h^{i,n})$, the weak formulation is defined as:

Find $\mathbf{w}_h^i \in B_h^k(\mathcal{T}_h^{i,n})$ such that for all $\mathbf{v}_h \in B_h^k(\mathcal{T}_h^{i,n})$:

$$\begin{aligned} &-\sum_{\mathcal{K}_j^{i,n} \in \mathcal{T}_h^{i,n}} \int_{\mathcal{K}_j^{i,n}} \nabla \mathbf{v}_h \cdot \mathcal{F}^i(\mathbf{w}_h^i) d\mathcal{K} + \sum_{\mathcal{S}_m^{i,n} \in \Gamma_I^{i,n}} \int_{\mathcal{S}_m^{i,n}} \mathcal{H}_I^i(\mathbf{w}_h^{i,l}, \mathbf{w}_h^{i,r}, \mathbf{n}_\mathcal{K})(\mathbf{v}_h^l - \mathbf{v}_h^r) d\mathcal{S} + \sum_{\mathcal{S}_m^{i,n} \in \Gamma_B^{i,n}} \int_{\mathcal{S}_m^{i,n}} \mathcal{H}_B^i(\mathbf{w}_h^{i,l}, \mathbf{w}_b^{i,r}, \mathbf{n}_\mathcal{K}) \mathbf{v}_h^l \, d\mathcal{S} \\ &+ \sum_{\mathcal{S}_m^{i,n} \in \Gamma_S^{i,n}} \int_{\mathcal{S}_m^{i,n}} \mathcal{H}_S^i(\mathbf{w}_h^{i,l}, \mathbf{w}_s^{i,r}, \mathbf{n}_\mathcal{K}) \mathbf{v}_h^l \, d\mathcal{S} = 0, \quad i = 1,2, \;\; n = 0, \ldots, N_t - 1. \end{aligned} \tag{28}$$

Introduction of the polynomial expansion (15) in (28) and using the basis functions ϕ_l for the test functions gives the following discretization in each space–time element $\mathcal{K}_j^{i,n}$:

$$\begin{aligned} \mathcal{L}_{kl}^{i,n}(\widehat{\mathbf{W}}^n; \widehat{\mathbf{W}}^{n-1}) = 0, \quad & i = 1,2, \;\; n = 0, \ldots, N_t - 1, \\ & k = 0, \ldots, N_w - 1, \;\; l = 0, \ldots, N_{B,j}^{i,n} - 1 \end{aligned} \tag{29}$$

with N_t the number of time slabs, N_w the number of flow variables and $N_{B,j}^{i,n}$ the number of basis functions. The nonlinear operator $\mathcal{L}_{kl}^{i,n}$ is defined as:

$$\begin{aligned} \mathcal{L}_{kl}^{i,n} = &-\int_{\mathcal{K}_j^{i,n}} (\nabla \phi_l)_j \cdot \mathcal{F}_{kj}^i(\mathbf{w}_h^i) d\mathcal{K} + \sum_{\mathcal{S}_m^{i,n} \in \partial\mathcal{K}_j^{i,n} \cap \Gamma_I^i} \int_{\mathcal{S}_m^{i,n}} \mathcal{H}_{I,k}(\mathbf{w}_h^{i,-}, \mathbf{w}_h^{i,+}, \mathbf{n}_\mathcal{K}) \phi_l \, d\mathcal{S} + \sum_{\mathcal{S}_m^{i,n} \in \partial\mathcal{K}_j^{i,n} \cap \Gamma_B^i} \int_{\mathcal{S}_m^{i,n}} \mathcal{H}_{B,k}(\mathbf{w}_h^{i,-}, \mathbf{w}_b^{i,+}, \mathbf{n}_\mathcal{K}) \phi_l \, d\mathcal{S} \\ &+ \sum_{\mathcal{S}_m^{i,n} \in \partial\mathcal{K}_j^{i,n} \cap \Gamma_S^i} \int_{\mathcal{S}_m^{i,n}} \mathcal{H}_{S,k}(\mathbf{w}_h^{i,-}, \mathbf{w}_s^{i,+}, \mathbf{n}_\mathcal{K}) \phi_l \, d\mathcal{S}. \end{aligned} \tag{30}$$

In Eq. (29) the dependency of $\mathcal{L}_{kl}^{i,n}$ on $\widehat{\mathbf{W}}^{n-1}$ stems from the integrals over the internal faces connecting the current and previous time slabs. The numerical fluxes are problem dependent and will be discussed later for each specific test problem.

4.2. Level set discretization

The level set equation can be characterized as a hyperbolic partial differential equation containing an intrinsic nonconservative product, meaning that it cannot be transformed into divergence form. This causes problems when the level set becomes discontinuous, because the weak solution in the classical sense of distributions does not exist. Thus, no classical Rankine–Hugoniot shock conditions can be defined. Although the level set is initially smooth, it can become discontinuous over time due to discontinuities in the global flow velocity advecting the level set. In order to find a discontinuous Galerkin discretization for the level set equation, valid even when level set solution and velocity become discontinuous, the theory presented in [45] is applied.

The nonconservative level set discretization is defined as:

$$\sum_{\mathcal{K}^n_{b,\bar{j}}\in\mathcal{T}^n_b}\int_{\mathcal{K}^n_{b,\bar{j}}} -\frac{\partial\phi_l}{\partial t}\psi_h + \phi_l\bar{\mathbf{a}}_h\cdot\bar{\nabla}\psi_h\, d\mathcal{K} + \sum_{\mathcal{K}^n_{b,\bar{j}}\in\mathcal{T}^n_b}\left(\int_{\mathcal{K}^n_{b,\bar{j}}(t_{n+1})}\phi^l_l\psi^l_h\, d\mathcal{S} - \int_{\mathcal{K}^n_{b,\bar{j}}(t_n)}\phi^l_l\psi^r_h\, d\mathcal{S}\right)$$
$$+\sum_{\mathcal{S}^n_{b,\bar{m}}\in\Gamma^n_b}\int_{\mathcal{S}^n_{b,\bar{m}}}\left(\phi^l_l-\phi^r_l\right)\widehat{P}^{nc}\, d\mathcal{S} - \sum_{\mathcal{S}^n_{b,\bar{m}}\in\Gamma^n_b}\int_{\mathcal{S}^n_{b,\bar{m}}}\{\{\phi_l\}\}[[\psi_h]]\{\{\bar{\mathbf{a}}_h\}\}d\mathcal{S} = 0, \tag{31}$$

with

$$\widehat{P}^{nc} = \begin{cases} +\frac{1}{2}[[\psi_h]]\{\{\bar{\mathbf{a}}_h\}\} & \text{if } S_L > 0,\\ +\frac{1}{2}\left(S_R\left(\psi^*_h-\psi^R_h\right)+S_L\left(\psi^*_h-\psi^L_h\right)\right) & \text{if } S_L<0<S_R,\\ -\frac{1}{2}[[\psi_h]]\{\{\bar{\mathbf{a}}_h\}\} & \text{if } S_R<0, \end{cases} \tag{32}$$

where $S_L = \min\{\bar{\mathbf{a}}^L_h\cdot\bar{\mathbf{n}}^L_K, \bar{\mathbf{a}}^R_h\cdot\bar{\mathbf{n}}^L_K\}$ and $S_R = \max\{\bar{\mathbf{a}}^L_h\cdot\bar{\mathbf{n}}^L_K, \bar{\mathbf{a}}^R_h\cdot\bar{\mathbf{n}}^L_K\}$ the minimum and maximum wavespeeds and where the star state level set value is defined as:

$$\psi^*_h = \begin{cases}\psi_L & \text{if } (S_L+S_R)/2>0,\\ \psi_R & \text{if } (S_L+S_R)/2<0.\end{cases} \tag{33}$$

At (solid wall) boundary faces the level set boundary conditions (10) are enforced by specifying the right state as:

$$\begin{aligned}&\psi^r(t,\bar{\mathbf{x}}) = \psi^l(t,\bar{\mathbf{x}}),\\ &\bar{\mathbf{a}}^r(t,\bar{\mathbf{x}}) = \bar{\mathbf{a}}^l(t,\bar{\mathbf{x}}) - 2(\bar{\mathbf{a}}^l(t,\bar{\mathbf{x}})\cdot\mathbf{n}_{\mathcal{K}})\mathbf{n}_{\mathcal{K}}, \quad \text{for } (t,\bar{\mathbf{x}})\in\mathcal{Q}.\end{aligned} \tag{34}$$

4.3. Pseudo-time integration

By augmenting the flow equations with a pseudo-time derivative, the discretized equations (29) are extended into pseudo-time, resulting in:

$$M^{i,n}_{ml}\frac{\partial\widehat{W}^{i,n}_{km}}{\partial\tau} + \mathcal{L}^{i,n}_{kl}\left(\widehat{\mathbf{W}}^n;\widehat{\mathbf{W}}^{n-1}\right) = 0, \tag{35}$$

using the summation convention on repeated indices, and with

$$M^{i,n}_{ml} = \int_{\mathcal{K}^{i,n}_j}\phi_l\phi_m\, d\mathcal{K} \tag{36}$$

the mass matrix. To solve (35) a five stage semi-implicit Runge–Kutta iterative scheme is used [29,65] as defined in Algorithm 2. Starting from a guess for the initial solution, the solution is iterated in pseudo-time until a steady state is reached, which is the real time solution of the space–time discretization.

Algorithm 2. Pseudo-time integration method for solving the non-linear algebraic equations in the space–time discretization

```
1. Initialize first Runge–Kutta stage: $\overline{\mathbf{W}}^{i,(0)} = \widehat{\mathbf{W}}^{i,n}$.
2. Calculate $\overline{\mathbf{W}}^{i,(s)}, s = 1,\cdots,5$:
       $(1+\alpha_s\lambda)\overline{\mathbf{W}}^{i,(s)} = \overline{\mathbf{W}}^{i,(0)} + \alpha_s\lambda\left(\overline{\mathbf{W}}^{i,(s-1)} - \Delta t(M^{i,n})^{-1}\mathcal{L}\left(\overline{\mathbf{W}}^{i,(s-1)},\overline{\mathbf{W}}^{i,n-1}\right)\right)$
3. Update solution: $\widehat{\mathbf{W}}^{i,n} = \overline{\mathbf{W}}^{i,(5)}$.
```

Here $\lambda = \Delta\tau/\Delta t$ denotes the ratio of pseudo-time and physical time step, and the coefficients α_s are defined as: $\alpha_1 = 0.0791451$, $\alpha_2 = 0.163551$, $\alpha_3 = 0.283663$, $\alpha_4 = 0.5$, $\alpha_5 = 1.0$. The physical time step Δt is defined globally by using a Courant–Friedrichs–Levy (*CFL*) condition:

$$\Delta t = \mathrm{CFL}_{\Delta t}h/|S_{max}|, \tag{37}$$

with $\mathrm{CFL}_{\Delta t}$ the physical CFL number, h the inradius of the space projection of the element and $|S_{max}|$ the maximum absolute value of the wave speed on the faces. The five stage semi-implicit Runge–Kutta iterative scheme is also used for solving the discretized level set equation.

5. Two-fluid algorithm

The two-fluid algorithm is defined in Algorithm 3. The operations at the initialization, in the inner iteration and at the time slab update are illustrated for two space–time dimensions in Figs. 11–13, respectively.

Algorithm 3. Computational steps in the two-fluid method. Lines 1–6 detail the initialization, lines 13–22 the inner iteration and lines 8–12 time slab update. The initialization, inner iteration and time slab update are illustrated for two space–time dimensions in Figs. 11–13

```
1.  $n = 0$
2.  Create background mesh $\mathcal{T}_b^{n-1}$
3.  Initialize level set $\psi_h^{n-1}(\mathbf{x})$ on $\mathcal{T}_b^{n-1}$
4.  Initialize level set velocity $\bar{\mathbf{a}}_h^{n-1}(\mathbf{x})$ on $\mathcal{T}_b^{n-1}$
5.  Create refined mesh $\mathcal{T}_h^{i,n-1}$ based on $\psi_h^{n-1} = 0$
6.  Initialize flow field $\mathbf{w}_h^{i,n-1}(\mathbf{x})$ on $\mathcal{T}_h^{i,n-1}$
7.  WHILE $n < N_t$ DO
8.    Create background mesh $\mathcal{T}_b^n$
9.    Initialize level set $\psi_h^n(\mathbf{x})$ on $\mathcal{T}_b^n$ as $\psi_h^{n-1}(t_n, \bar{\mathbf{x}})$ on $\mathcal{T}_b^{n-1}$ (41)
10.   Initialize level set velocity $\bar{\mathbf{a}}_h^n(\mathbf{x})$ on $\mathcal{T}_b^n$ as $\bar{\mathbf{a}}_h^{n-1}(t_n, \bar{\mathbf{x}})$ on $\mathcal{T}_b^{n-1}$ (42)
11.   Create refined mesh $\mathcal{T}_{h,0}^{i,n}$ based on $\psi_h^n = 0$
12.   Initialize flow field $\mathbf{w}_{h,0}^{i,n}(\mathbf{x})$ on $\mathcal{T}_{h,0}^{i,n}$ as $\mathbf{w}_{h,0}^{i,n-1}(t_n, \bar{\mathbf{x}})$ on $\mathcal{T}_h^{i,n-1}$ (38)
13.   $k = 0$
14.   WHILE two-fluid mesh has not converged:$|e_k - e_{k-1}| > \epsilon_{IF}$ DO
15.     Solve $\psi_h^n$ on $\mathcal{T}_b^n$
16.     Calculate level set interface error $e_k = (\sum_{S_h^{i,n} \in \Gamma_S^n} \int_{S_h^{i,n}} |\psi_h^n|^2 dS)^{1/2}$
17.     Create refined mesh $\mathcal{T}_{h,k}^{i,n}$ based on $\psi_h^n = 0$
18.     Initialize flow field $\mathbf{w}_{h,k}^{i,n}(\mathbf{x})$ on $\mathcal{T}_{h,k}^{i,n}$ as $\mathbf{w}_h^{i,n-1}(t_n, \bar{\mathbf{x}})$ on $\mathcal{T}_h^{i,n-1}$ (38)
19.     Solve $\mathbf{w}_{h,k}^{i,n}(t, \bar{\mathbf{x}})$ on $\mathcal{T}_{h,k}^{i,n}$
20.     Initialize level set velocity $\bar{\mathbf{a}}_h^n(\mathbf{x})$ on $\mathcal{T}_b^n$ as $\mathbf{u}_{h,k}^{i,n}(\mathbf{x})$ on $\mathcal{T}_h^{i,n}$ (39)
21.     $k = k + 1$
22.   END DO
23.   $n = n + 1$
24. END DO
```

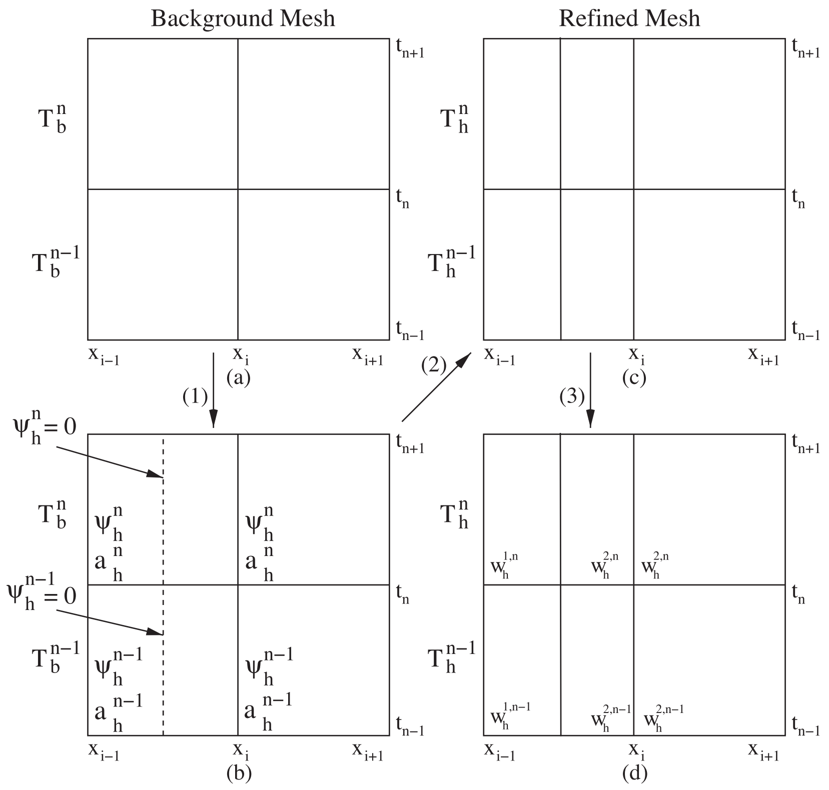

Fig. 11. At initialization, first the background mesh is created. Because the solution from the previous time step is required in the evaluation of the numerical flux at the time slab face, the background mesh is conveniently composed of a current (n) and a previous ($n-1$) time slab (a). Next the level set is initialized on the background mesh (b). Based on the 0-level set, the background mesh is refined to obtain the refined mesh (c). Finally, in all elements of the refined mesh the flow variables are initialized (d). The initialization is performed on the current as well as a previous time slab.

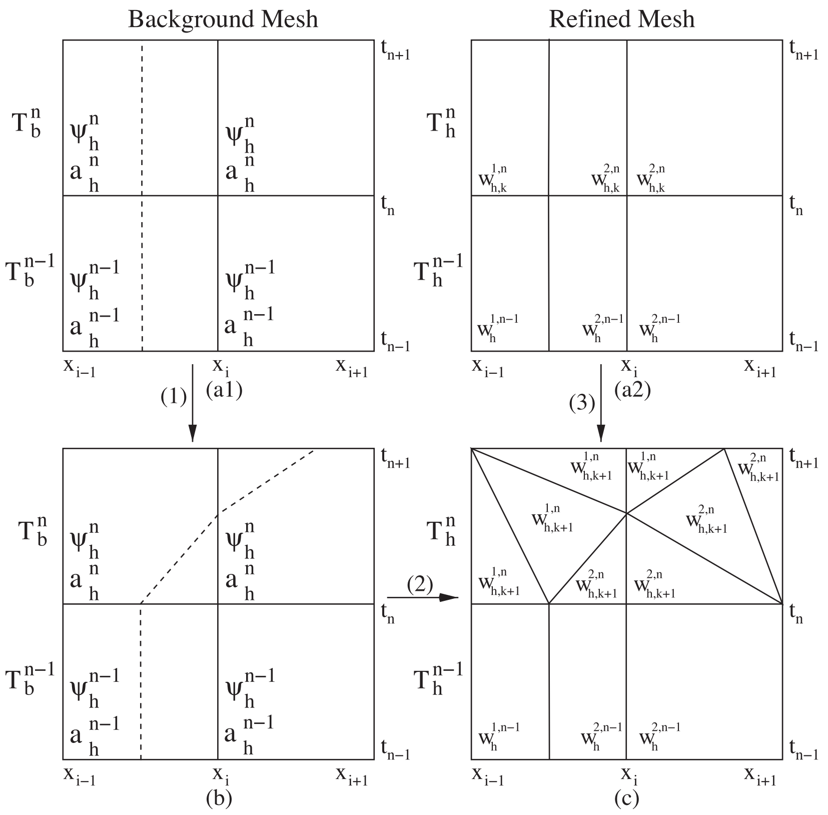

Fig. 12. In the inner iteration, given level set and flow solutions on the background and refined meshes (a1, a2), first the level set is solved on $\mathcal{T}_b^n$ (b). Based on the 0-level set the background mesh is refined to obtain a new two-fluid mesh $\mathcal{T}_h^n$, on which the flow field is reinitialized and solved (c). Finally, the level set velocity is reinitialized with the flow velocity.

In the inner iteration and at the time slab update the flow approximation $\mathbf{w}_h^{i,n}$ is reinitialized with the solution average from the previous time slab:

$$\mathbf{w}_h^{i,n}(t,\bar{\mathbf{x}}) = \bar{\mathbf{w}}_h^{i,n-1}(t_n,\bar{\mathbf{x}}). \tag{38}$$

When, for a fluid type, no solution exists in the previous time slab, the element is marked as such and is reinitialized at a later stage by using the reinitialized solution from a neighboring element in the new timeslab. To make the flow reinitialization compatible with the element merging it is preceded by a projection step, in which the solution in each merged element is projected onto the refined elements of which it is composed. After solving the flow equations the level set velocity $\mathbf{a}_h^n$ is reinitialized as:

$$\int_{\mathcal{K}_{b,\bar{j}}^n} \bar{\mathbf{a}}_h^n(\mathbf{x})\phi_l(\mathbf{x})d\mathcal{K} = \int_{\mathcal{K}_{b,\bar{j}}^n} \mathbf{u}_{h,k}^n(\mathbf{x})\phi_l(\mathbf{x})d\mathcal{K}. \tag{39}$$

In order to evaluate the flow velocity $\mathbf{u}_{h,k}^n$ on the background mesh, for every element $\mathcal{K}_j^{i,n}$ in the refined mesh $\mathcal{T}_h^n$, a child to parent mapping $H_{\mathcal{K}_j^{i,n}}$ is defined:

$$H_{\mathcal{K}_j^{i,n}} = G_{\mathcal{K}_j^{i,n}}^{-1} \circ G_{\mathcal{K}_{b,\bar{j}}^n}, \tag{40}$$

where $G_{\mathcal{K}_{b,\bar{j}}^n}$ and $G_{\mathcal{K}_j^{i,n}}$ are the mappings from the reference element to the physical space of the background and the child element, respectively. The mapping $H_{\mathcal{K}_j^{i,n}}$ maps the element $\mathcal{K}_j^{i,n}$ to its parent element $\mathcal{K}_{b,\bar{j}}^n$ in the background mesh $\mathcal{T}_b^n$. The inverse mappings $G_{\mathcal{K}_{b,\bar{j}}^n}^{-1}$ always exist, since the background elements are by construction never degenerate. The child to parent mapping is illustrated in Fig. 14. At the time slab update the level set approximation ψ_h^n is reinitialized as:

$$\psi_h^n(t,\bar{\mathbf{x}}) = \psi_h^{n-1}(t_n,\bar{\mathbf{x}}) \tag{41}$$

and the level set velocity approximation $\mathbf{a}_h^n$ is reinitialized as:

$$\bar{\mathbf{a}}_h^n(t,\bar{\mathbf{x}}) = \bar{\mathbf{a}}_h^{n-1}(t_n,\bar{\mathbf{x}}). \tag{42}$$

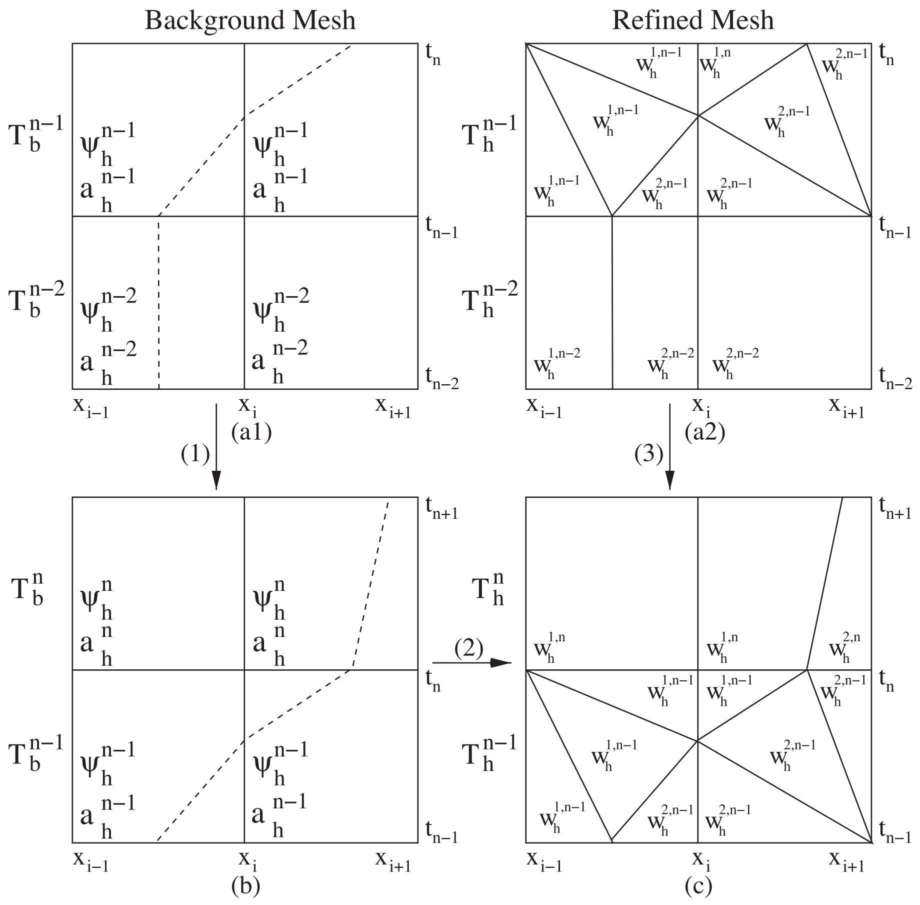

Fig. 13. When moving to the next time slab, given level set and flow solutions on the background and refined meshes (a1, a2), first a new background mesh $\mathcal{T}_b^n$ is created, on which a level set is initialized and solved (b). Based on the 0-level set, the background mesh is refined to obtain the two-fluid mesh $\mathcal{T}_h^n$, on which the flow field is initialized (c).

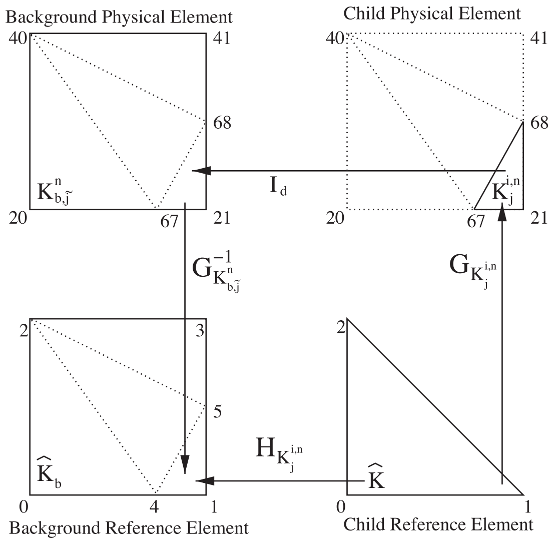

Fig. 14. The child to parent mapping $H_{K_j^{i,n}}$ is composed of the mapping $G_{K_j^{i,n}}$ from the child reference element to child physical element and the inverse mapping $G^{-1}_{K^n_{b,\tilde{j}}}$ from background physical element to the background reference element. The child physical element is connected to the background physical element through the identity mapping I_d.

6. Test cases

The method is applied to model problems in two and three space–time dimensions. The interface is assumed to be without surface tension and therefore continuity of the normal velocity and pressure are imposed [14,51]. The simulations have been performed using three dimensional space–time codes based on the *hp*GEM software framework for Discontinuous Galerkin finite element methods [41]. More test cases are available in [54].

Let $w^i_h(t_{n+1},\mathbf{x})$ denote the approximate flow solution, $w^i(t_{n+1},\mathbf{x})$ the exact flow solution and $\Omega^i_h(t_{n+1})$ the spatial mesh for fluid i at time $t = t_{n+1}$. The L_2 flow error at time $t = t_{n+1}$ is defined as:

$$\left\|w^i_h(t_{n+1},\cdot) - w^i(t_{n+1},\cdot)\right\|_{L_2(\Omega^i_h(t_{n+1}))} = \left(\int_{\Omega^i_h(t_{n+1})} |w^i_h(t_{n+1},\mathbf{x}) - w^i(t_{n+1},\mathbf{x})|^2 \, d\mathbf{x}\right)^{1/2}. \tag{43}$$

The order of accuracy with respect to the norm $\|\cdot\|$ is defined as $\log(\|f_h - f\|/\|f_{h/2} - f\|)/\log(2)$, where f_h and $f_{h/2}$ denote numerical solutions on embedded meshes Ω^n_h and $\Omega^n_{h/2}$, with h the mesh width. It should be noted that the refined meshes are often only approximately embedded, hence a small error is introduced in the orders of accuracy for the flow solutions.

Solutions will be plotted as discontinuous data without any postprocessing to give a clear illustration of the behavior of the STDG numerical scheme in each individual element.

6.1. Isothermal magma–ideal gas shock tube

Considered is an isothermal magma–ideal gas shock tube problem. This test is motivated by the high speed geological event analyzed in [7–9,69,70] and it features very high density and pressure ratio's which cause strong oscillations around the interface between the gas and magma with standard shock capturing schemes. The purpose of this test is to investigate the performance of the method for a case where the interface moves with the flow velocity. To account for this, two solve steps are used for the flow and level set equations in each time step. The contact wave is considered an interface and is captured using the two-fluid method.

For the ideal gas the one dimensional Euler equations for mass, momentum and energy are used, which are defined as

$$\begin{aligned} &\frac{\partial \rho}{\partial t} + \frac{\partial(\rho u)}{\partial x} = 0,\\ &\frac{\partial(\rho u)}{\partial t} + \frac{\partial(\rho u^2 + p)}{\partial x} = 0,\\ &\frac{\partial(\rho E)}{\partial t} + \frac{\partial(u(\rho E + p))}{\partial x} = 0, \end{aligned} \tag{44}$$

with ρ the density, u the fluid velocity, p the pressure and $\rho E = \rho u^2/2 + \rho e$ the total energy, with ρe the internal energy. In addition to these equations an equation of state (EOS) is required to account for the thermodynamic properties of the ideal gas:

$$e = \frac{p}{\rho(\gamma - 1)}, \tag{45}$$

where $\gamma = 1.4$. The governing equations for an effectively compressible magma are the Euler equations for mass and momentum. The magma consists of a mixture of molten rock and 2 wt.% (weight percentage) H_2O. At high pressure, the H_2O only has a liquid form. When the pressure decreases water vapor is formed within the mixture due to decompression effects. In this situation the magma effectively is a pseudo one-phase mixture. In explosive eruptions starting with a high pressure difference viscosity effects are negligible at leading order relative to the nonlinear inertial effects driven by the high bubble content. The total mass fraction n_0 of H_2O in the magma consists of a fraction $n(p)$ which is exsolved in the magma as gas and a fraction $1 - n(p)$ which is dissolved in the magma as liquid.

The mixture of magma and liquid H_2O has a density $\sigma = 2500$ kg/m^3 and the water vapor has a density of ρ_g. The total void or bubble fraction of the mixture is given by $\alpha = n(p)\rho/\rho_g$. The density of the magma is defined as $\rho = \alpha\rho_g + (1 - \alpha)\sigma$. Using the relation for α and the ideal gas law $\rho_g = p/(RT)$ gives:

$$\rho = \left(\frac{n(p)R_m T}{p} + \frac{1 - n(p)}{\sigma}\right)^{-1}, \tag{46}$$

where $R_m = 462$ J/kg K is the mixtures gas constant. This relation is only valid when there are bubbles, i.e., $n(p) > 0$. The critical pressure p_c is reached when there are no longer any bubbles in the mixture. This is the case when $n(p = p_c) = 0$ which gives $p_c = (4/9) \times 10^8$ Pa. The magma considered will be assumed to be compressible; hence, $p < p_c$. For $p \geqslant p_c$ the following relation is used:

$$\rho = \sigma + c_m^{-2}(p - p_c), \tag{47}$$

with $c_m = 2000$ m/s the speed of sound in bubble free magma. The mass fraction $n(p)$ is assumed to satisfy Henry's law, which is valid when bubbles and melt are in equilibrium:

$$n(p) = n_0 - S_h p^{\beta}. \tag{48}$$

For basaltic high volatile magma, $n_0 = 0.02$, $\beta \approx 0.5$, $T = 1200$ K and $S_h = 3.0 \times 10^{-6}\ \mathrm{Pa}^{-\beta}$. The magma is assumed to be isothermal at a temperature of 1200 K. For isothermal magma the density depends only on the pressure, $\rho = \rho(p)$. The speed of sound a is defined for isothermal magma as:

$$1/a^2 \equiv \left(\frac{\partial \rho}{\partial p}\right)_T = -\rho^2 \frac{\partial(1/\rho)}{\partial p} = -\rho^2 \left[\frac{dn(p)}{dp}\left(\frac{R_m T}{p} + \frac{1}{\sigma}\right) - \frac{n(p) R_m T}{p^2}\right]. \tag{49}$$

The simulations are performed on a spatial domain $[-5\ \mathrm{m}, 5\ \mathrm{m}]$ from time $t = 0$ s to $t = 0.0075$ s. Initially the interface is located at $x = 0$ m, with the magma on the left and the ideal gas on the right, and both fluids are in constant states:

$$(\rho, u, p)(0, x) = \begin{cases} (\rho_L, u_L, p_L) = (535.195\ \mathrm{kg/m^3},\ 0\ \mathrm{m/s},\ 5 \times 10^6\ \mathrm{Pa}) & \text{for } x < 0\ \mathrm{m}, \\ (\rho_R, u_R, p_R) = (1.18902\ \mathrm{kg/m^3},\ 0\ \mathrm{m/s},\ 1.0 \times 10^5\ \mathrm{Pa}) & \text{for } x > 0\ \mathrm{m}. \end{cases} \tag{50}$$

At the boundaries solid wall conditions are imposed:

$$u \cdot \bar{n} = 0\ \mathrm{m/s} \quad \text{at } x = \pm 5\ \mathrm{m}. \tag{51}$$

At the magma–gas interface continuity of the velocity and pressure is assumed. The exact solution is calculated by solving the magma and ideal gas Riemann problem and consists of a left moving expansion wave with head and tail speeds of $S_{LH} = -97.2861$ m/s, $S_{LT} = 186.409$ m/s respectively, a contact wave which is identified with the magma–air interface and moves with speed $S_C = 286.329$ m/s; and, a right moving shock wave with speed $S_R = 555.540$ m/s. The left and right star states are defined as: $\rho_L^* = 28.0517\ \mathrm{kg/m^3},\ \rho_R^* = 2.45364\ \mathrm{kg/m^3},\ u^* = 286.329\ \mathrm{m/9}m,\ p^* = 2.89134 \times 10^5$ Pa. The solution structure is shown in Fig. 15.

Let $\mathbf{w} = (\rho, \rho u, \rho E)$ and $\mathbf{F} = (\rho u, \rho u^2 + p, u(\rho E + p))$ denote the conservative variables and flux vectors. The HLLC flux provides an accurate solution to the Riemann problem, which is an initial value problem for the Euler equations, where the initial condition consists of two constant states:

$$\mathbf{w}(x, 0) = \begin{cases} \mathbf{w}_L & \text{when } x < 0, \\ \mathbf{w}_R & \text{when } x > 0. \end{cases} \tag{52}$$

The HLLC flux extended to space–time meshes [5,65] is defined as:

$$\mathcal{H}_{\mathrm{HLLC}} = \frac{1}{2}\big(\mathbf{F}_L + \mathbf{F}_R - (|S_L - v| - |S_M - v|)\mathbf{w}_L^* + (|S_R - v| - |S_M - v|)\mathbf{w}_R^* + |S_L - v|\mathbf{w}_L - |S_R - v|\mathbf{w}_R - v(\mathbf{w}_L + \mathbf{w}_R)\big), \tag{53}$$

with v the interface velocity. It is assumed that the speeds are the same at both sides of the contact wave, so $S_M = u_L^* = u_R^* = u^*$. From the Rankine–Hugoniot relations $\mathbf{F}(\mathbf{w}_K) - \mathbf{F}(\mathbf{w}_K^*) = S_K(\mathbf{w}_K - \mathbf{w}_K^*)$ with $K = L$ or R for the left and the right waves, respectively, the following relations are found for the star state variables:

$$\begin{aligned} \rho_K^* &= \rho_K \frac{S_K - u_K}{S_K - u^*}, \\ \rho_K^* u^*(u^* - S_K) &= (p_K - p^*) + \rho_K u_K (u_K - S_K), \end{aligned} \tag{54}$$

and also an approximation for the speed $S_M = u^*$ of the contact wave is obtained:

$$S_M = \frac{\rho_R u_R (S_R - u_R) - \rho_L u_L (S_L - u_L) + p_L - p_R}{\rho_R (S_R - u_R) - \rho_L (S_L - u_L)}. \tag{55}$$

The wave speeds S_L and S_R are estimated as:

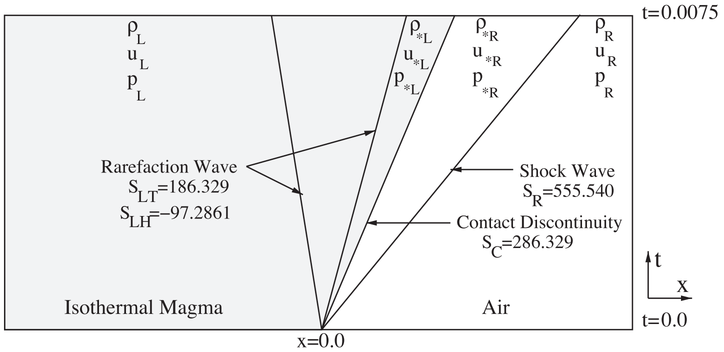

Fig. 15. The solution structure of the Euler magma–ideal gas shock tube.

$$S_L = \min(u_L - a_L, u_R - a_R), \quad S_R = \max(u_L + a_L, u_R + a_R). \tag{56}$$

By using the Rankine–Hugoniot relations of the left wave and substituting the left and right states and wave speeds, the values of $\mathbf{w}_L^*$ are calculated as:

$$\mathbf{w}_L^* = \frac{S_L - u_L}{S_L - S_M}\mathbf{w}_L + \frac{1}{S_L - S_M}\begin{pmatrix} 0 \\ p^* - p_L \\ p^* S_M - p_L u_L \end{pmatrix}, \tag{57}$$

and likewise for $\mathbf{w}_R^*$ by replacing L with R. By using the expression for ρ_K^* and u^* in the Rankine–Hugoniot relation for the momentum of the left and the right moving wave, the intermediate pressure is found:

$$p^* = \rho_L(S_L - u_L)(S_M - u_L) + p_L = \rho_R(S_R - u_R)(S_M - u_R) + p_R. \tag{58}$$

Assuming the interface coincides with the contact wave, $S_M = v$ and the corresponding HLLC flux defines the contact HLLC flux $\mathcal{H}^C_{\text{HLLC}}$:

$$\mathcal{H}^C_{\text{HLLC}} = (0, p^*, p^* u^*)^T \tag{59}$$

which shows that there is no mass flux through the contact interface.

For the interface an alternative interface flux is proposed, which is defined separately for the left and right sides of the interface:

$$\mathcal{H}^L_{\text{HLLC}} = \mathbf{w}_L^*(S_M - v) + \mathcal{H}^C_{\text{HLLC}} \quad \text{and} \quad \mathcal{H}^R_{\text{HLLC}} = \mathbf{w}_R^*(S_M - v) + \mathcal{H}^C_{\text{HLLC}}. \tag{60}$$

When the interface representation in the mesh is exact, $S_M = v$ and the interface flux is reduced to $\mathcal{H}^C_{\text{HLLC}}$. The interface numerical flux removes the small numerical oscillations caused by errors in the interface shape and position at the cost of mass conservation at the interface.

At the boundary faces the solid wall conditions are implemented in the HLLC flux by defining the right state as:

$$\rho_R = \rho_L, \quad u_R = -u_L, \quad p_R = p_L. \tag{61}$$

To account for the dependence of the level set on the flow velocity the flow and level set are updated twice each time step. The simulations are performed at $\text{CFL}_{\Delta t} \approx 0.56$, using interface flux (60) and primitive variable discretizations for both fluids.

The test results for the solution at time $t = 0.0075$ s are presented in Table 6 and convergence in the L_2 norm is observed. In Fig. 16 the interface evolution and the level set profile at the final time are shown. It is observed that the level set becomes

Table 6
Error and order of accuracy in the L_2 norm of the density for the isothermal magma and ideal gas Euler shock tube test.

$N_x \times N_t$	L_2 error	L_2 order
40 × 30	28.5747	–
80 × 60	16.7343	0.772
160 × 120	10.6157	0.657
320 × 240	5.95713	0.834

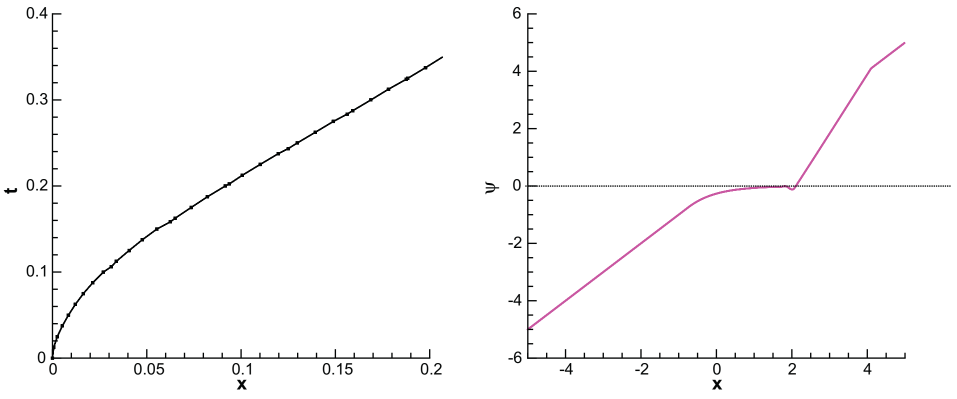

Fig. 16. The time evolution of the interface position and level set at time $t = 0.0075$ s for the Euler magma–ideal gas shock tube using 320 background elements.

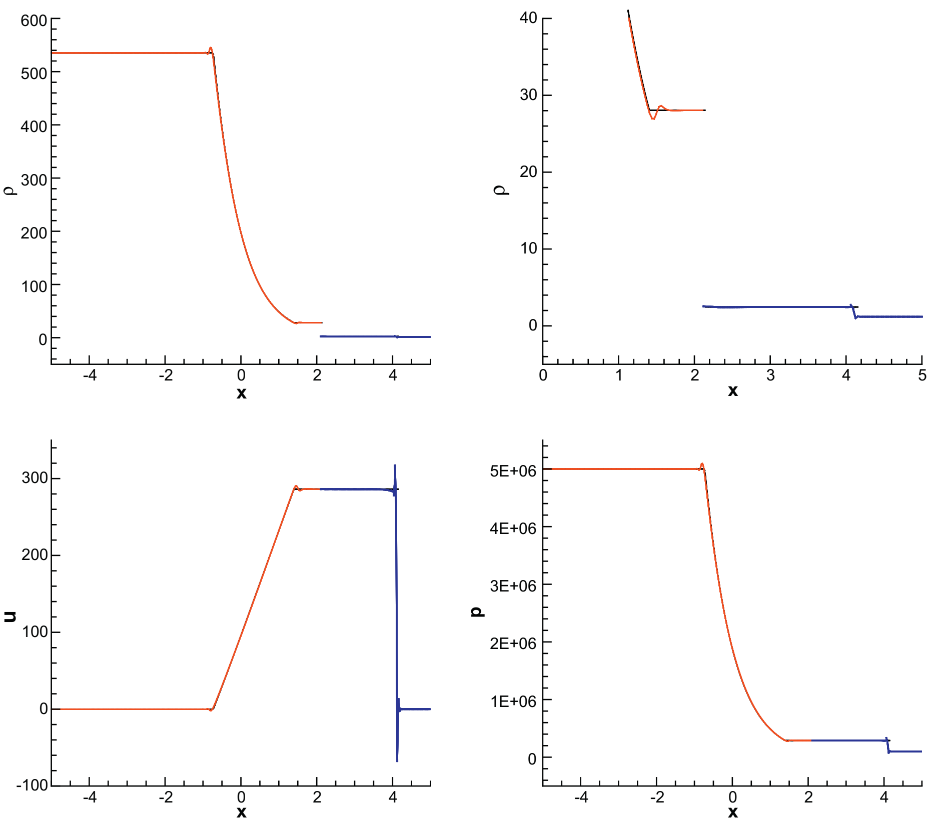

Fig. 17. The exact (black) and numerical (colored) density, density zoom, velocity and pressure at time $t = 0.0075$ s for the Euler magma–ideal gas shock tube using 320 background elements.

distorted over time. The reason for this behavior lies in the use of the global flow velocity for advecting the level set and the problem can be fixed by reinitializing the level set every few time steps. In Fig. 17 the density, density zoom, velocity and pressure at the final time are shown. It is observed that the solution shows significant over- and undershoots near the expansion and shock waves. To remove these spurious oscillations the HWENO slope limiter is used in combination with the Krivodonova discontinuity detector [30,34] and the results are shown in Fig. 18. The slope limiter significantly reduces the over- and undershoots, but also causes a small offset error in the star region and a decrease in the accuracy of the shock position. In Fig. 19 the mass evolution of the magma and the ideal gas is shown for the results without slope limiter. The mass loss is less than 1%.

6.2. Helium cylinder–ideal gas shock interaction

To test the algorithm in a more complex setting computations are performed on the interaction between a cylindrical helium volume in a tube filled with an ideal gas and a Mach 1.22 shock wave [15,23,43,68] as illustrated in Fig. 20. For the Euler equations this problem has no unique solution, because the shock induces a Rayleigh–Taylor instability at the interface, but it presents a challenging test case for the numerical algorithm. The adiabatic indices and the gas constants for an ideal gas and helium are given as $\gamma_I = 1.4$, $R_I = 287.0$ J/kg K and $\gamma_H = 1.67$, $R_H = 2080.0$ J/kg K. Initially the helium volume is a cylinder with a radius 0.025 m and is located at $(x,y) = (0\text{ m}, 0\text{ m})$ while the shock is located at $x = 0.055625$ m. The domain has dimensions $[-0.11125\text{ m}, 0.11125\text{ m}] \times [-0.0445\text{ m}, 0.0445\text{ m}]$. Both fluids are modelled using the two dimensional Euler equations. The initial state of the helium, and the ideal gas in front and behind of the shock are given as:

$$\begin{aligned}
(\rho_B, u_B, v_B, p_B) &= (0.164062\text{ kg/m}^3, 0\text{ m/s}, 0\text{ m/s}, 1.0 \times 10^5\text{ Pa}),\\
(\rho_L, u_L, v_L, p_L) &= (1.18902\text{ kg/m}^3, 0\text{ m/s}, 0\text{ m/s}, 1.0 \times 10^5\text{ Pa}),\\
(\rho_R, u_R, v_R, p_R) &= (1.63652\text{ kg/m}^3, -114.473\text{ m/s}, 0\text{ m/s}, 1.5698 \times 10^5\text{ Pa}),
\end{aligned} \tag{62}$$

where the density of the helium is related to the density of the air in front of the shock as $\rho_B = \rho_L R_I / R_H$. The shock velocity is $V_S = Ma_L = 418.628$ m/s, with $a_L = \sqrt{\gamma_I p_L / \rho_L} = 343.138$ m/s. The states on both sides of the shock wave are related through the Rankine–Hugoniot relations:

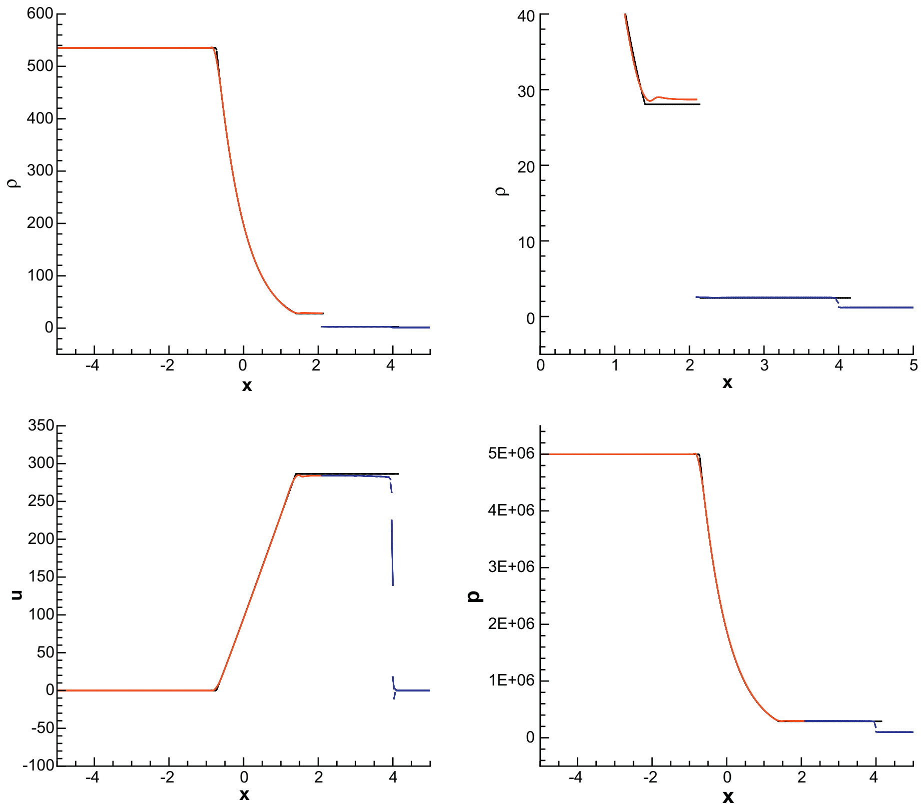

Fig. 18. The exact (black) and numerical (colored) density, density zoom, velocity and pressure at time $t = 0.0075$ s for the Euler magma – ideal gas shock tube using 320 background elements and with HWENO slope limiter.

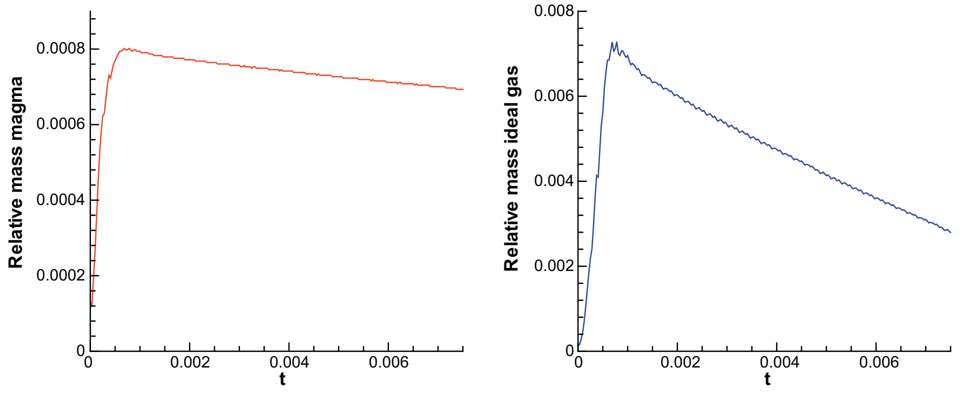

Fig. 19. Relative mass error over time of magma (left) and ideal gas (right) for the Euler magma–ideal gas shock tube using 320 background elements. The relative mass is defined as $|M_e - M_h|/M_e$, with M_e the exact and M_h the numerical amount of mass.

$$
\begin{aligned}
&(\rho_R - \rho_L)V_S = (\rho_R u_R - \rho_L u_L),\\
&(\rho_R u_R - \rho_L u_L)V_S = \left(\rho_R u_R^2 - \rho_L u_L^2\right) + (p_R - p_L),\\
&(\rho_R E_R - \rho_L E_L)V_S = u_R(\rho_R E_R + p_R) - u_L(\rho_L E_L + p_L).
\end{aligned}
\tag{63}
$$

Using the definition of the total energy, $\rho E = \rho(u^2 + v^2)/2 + \rho e$, and the EOS for an ideal gas, $\rho e = p/(\gamma_I - 1)$, the Rankine–Hugoniot conditions can be solved for ρ_R, u_R and p_R.

The cylindrical helium volume in this test acts as a divergent lens for the shock wave. When the initial shock wave incidents the upstream boundary of the helium volume, the shock is transmitted into the helium volume and accelerates due to the decrease in density, while the upstream boundary of the helium volume is set into downstream motion and an expansion

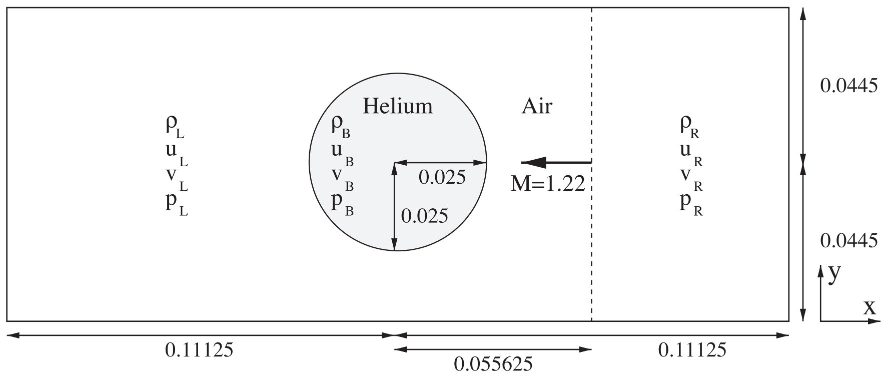

Fig. 20. Helium cylinder–shock interaction test.

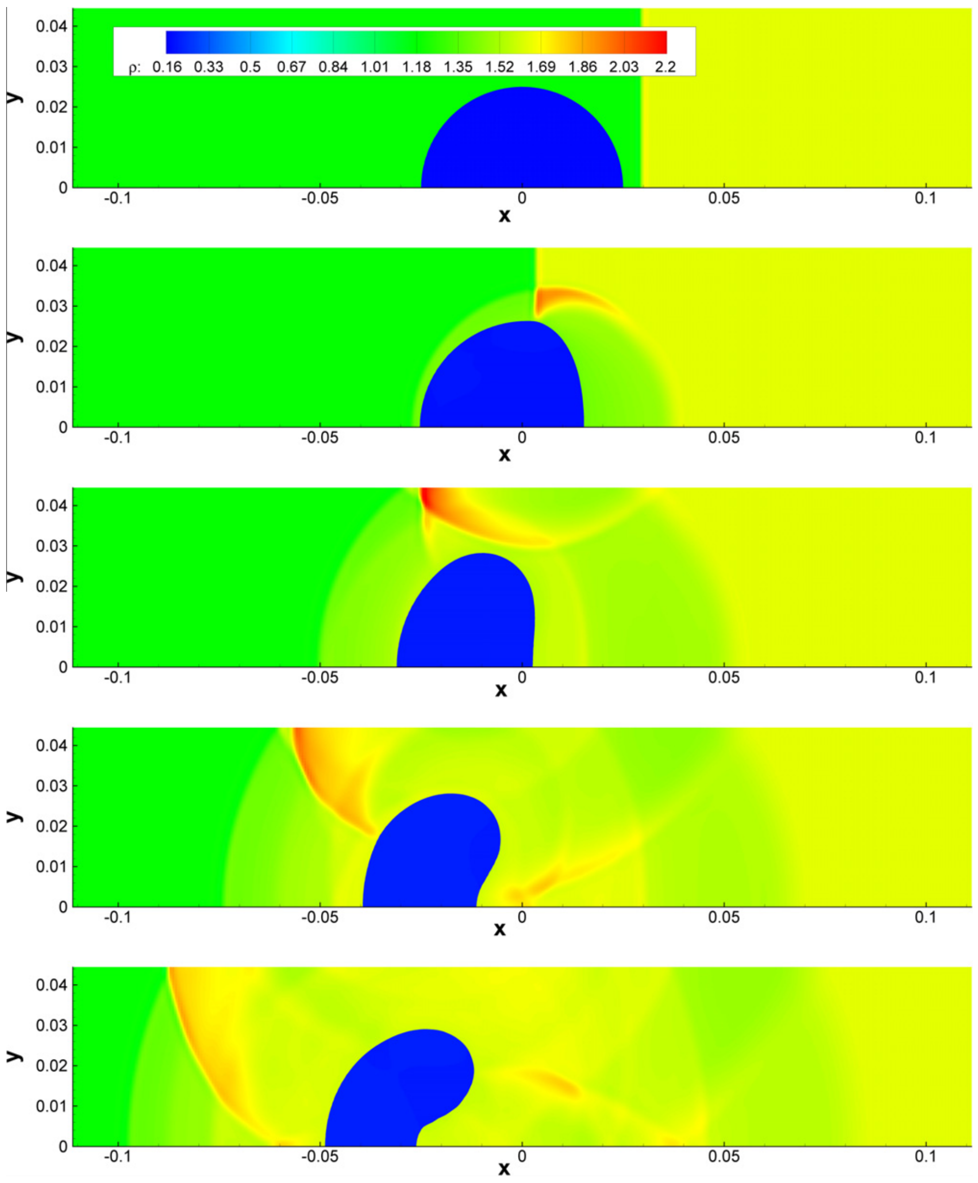

Fig. 21. Density contours at times $t = 0.625 \times 10^{-4}$ s, 1.25×10^{-4} s, 1.875×10^{-4} s, 2.5×10^{-4} s and 3.125×10^{-4} s for the helium cylinder–ideal gas shock interaction test using 320×64 elements.

wave is generated moving in the upstream direction. When the transmitted shock incidents the downstream boundary of the helium volume, the shock is transmitted and decelerates, while the downstream boundary of the helium volume is set into downstream motion and another expansion wave is generated moving in the upstream direction. Over time the helium volume flattens and is subsequently transformed into a vortex like structure. In addition, the top wall adds to the complexity of the solution through a number of wave reflections.

At the top, bottom and left boundaries solid wall boundary conditions are imposed. At the right boundary the ideal gas state behind the shock is imposed weakly by using it as the external state of the numerical flux. At the helium–ideal gas interface continuity of the normal velocity and the pressure is imposed and the numerical flux (60) is used. To account for the dependence of the level set on the flow velocity the flow and level set are updated twice during each time step. Because the solution is symmetric with respect to the x-axis, computations are performed on the half domain $[-0.11125\text{ m}, 0.11125\text{ m}] \times [0\text{ m}, 0.0445\text{ m}]$. The simulations are run using 40×8, 80×16, 160×32 and 320×64 elements from time $t = 0$ s to 3.125×10^{-4} s at CFL ≈ 1.0 using linear basis functions for the flow field and the level set, where the level set smoothing reconstructs a bilinear level set.

The density contours for subsequent times are shown in Fig. 21. The density at time $t = 3.4375 \times 10^{-4}$ s for different mesh resolutions are shown in Fig. 22. The evolution of helium mass over time for different mesh resolutions is shown in Fig. 23 and is relatively small, ranging from 0.5% to 2%. The evolution of the interface mesh is illustrated for 80×16 elements in Fig. 24. After the shock wave reflects off the left boundary it interacts for a second time with the helium volume, this time causing breakup, as illustrated in Fig. 25. At this point level set deformations become too large causing spurious bubbles to be generated. It is expected that this problem can be solved by a post-processing of the level set in which spurious and very small bubbles are detected and removed. The post-processing is a very crude subgrid scale model that would solely aim to keep the algorithm robust and stable. However, this concerns future work. Note that the Rayleigh–Taylor instability will move into finer and finer scales which ultimately cannot be resolved on a finite mesh.

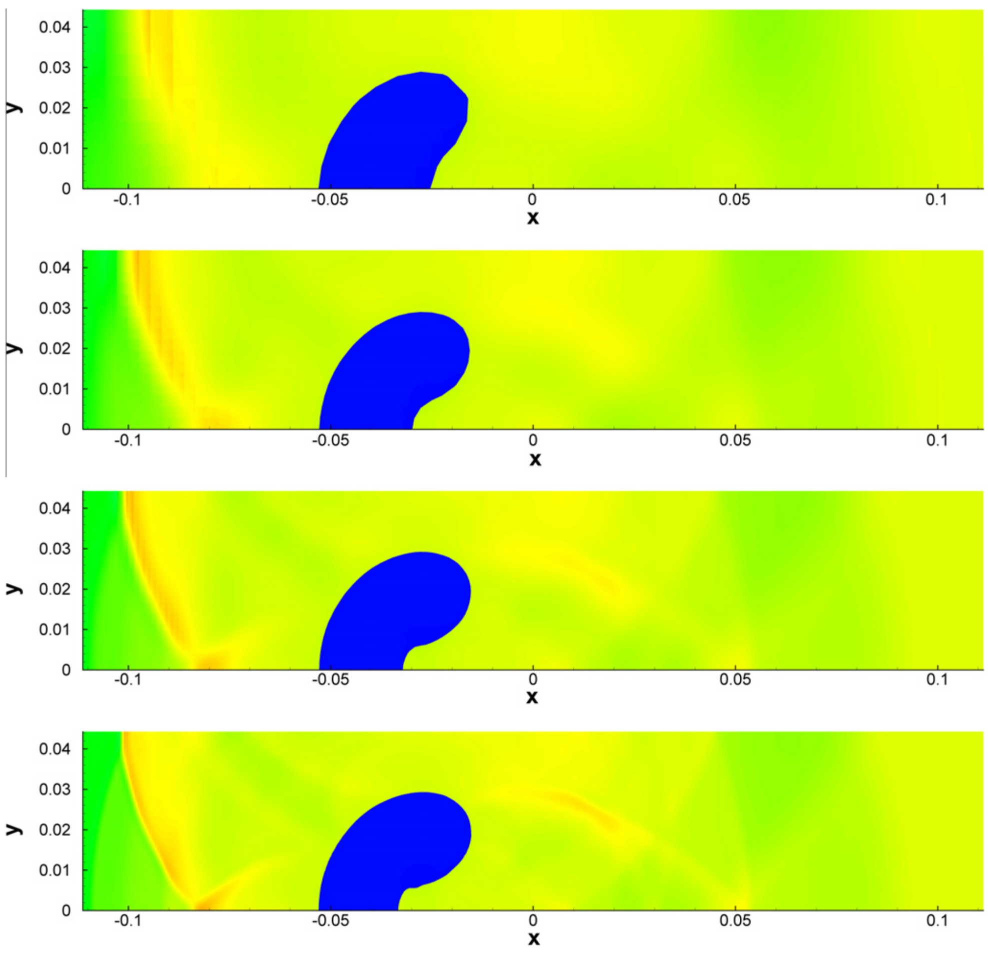

Fig. 22. Density contours at time $t = 3.4375 \times 10^{-4}$ s for the helium cylinder–ideal gas shock interaction test using 40×8, 80×16, 160×32 and 320×64 elements.

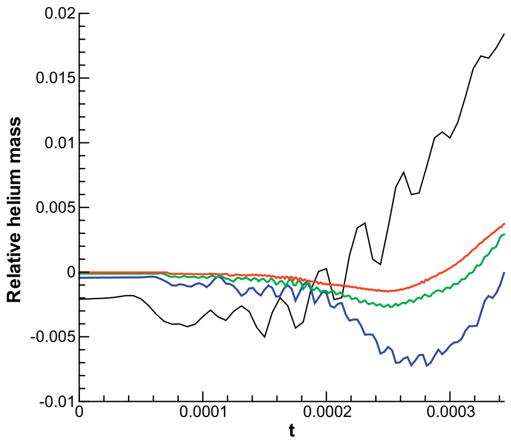

Fig. 23. Relative helium mass error over time for the helium cylinder–ideal gas shock interaction test using 40 × 8 (black), 80 × 16 (blue), 160 × 32 (green) and 320 × 64 (red) elements. The relative mass is defined as $(M_e - M_h)/M_e$, with M_e the exact and M_h the numerical amount of mass. (For interpretation of the references in colour in this figure legend, the reader is referred to the web version of this article.)

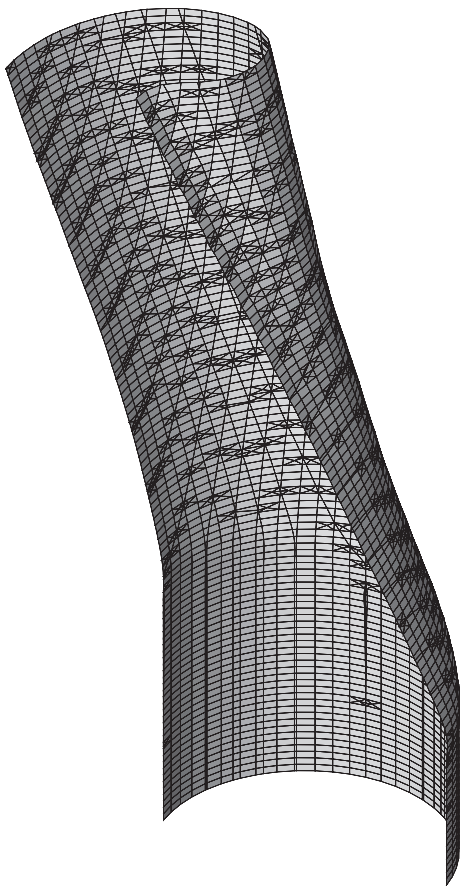

Fig. 24. Interface evolution for the helium cylinder–ideal gas shock interaction test using 80 × 16 elements.

7. Discussion

A space–time discontinuous Galerkin finite element method for two-fluid flows has been presented which combines aspects of front tracking and front capturing methods with cut-cell mesh refinement and a STDG discretization. It is anticipated

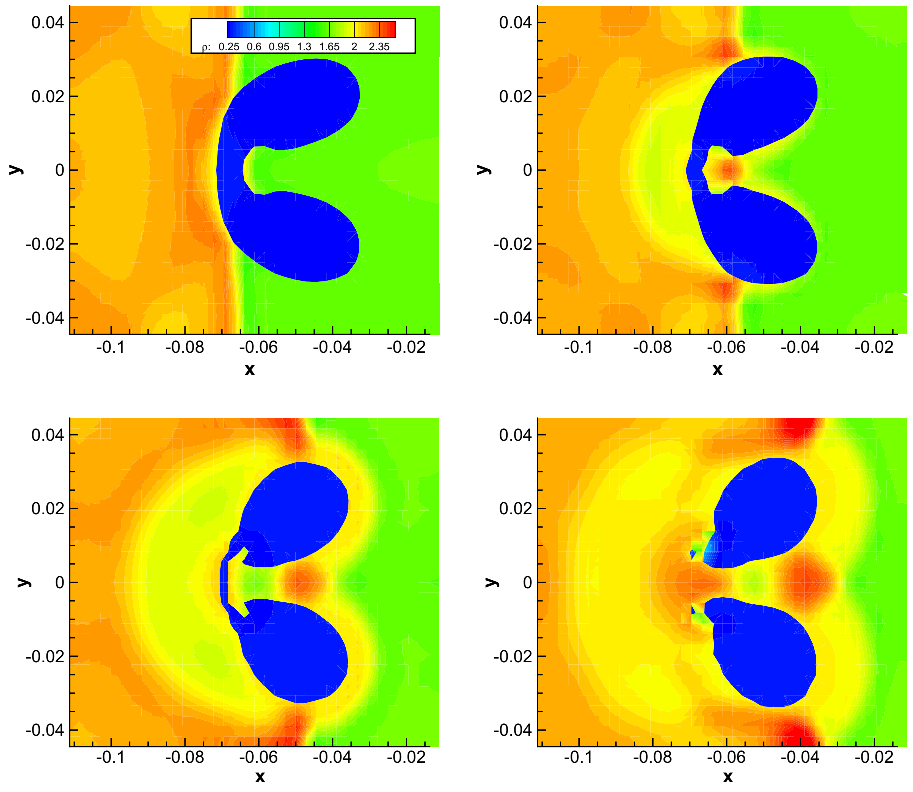

Fig. 25. Density contours at times $t = 5.0 \times 10^{-4}$ s, 5.25×10^{-4} s, 5.5×10^{-4} s and 5.75×10^{-4} s for the helium cylinder–ideal gas shock interaction test using 80×32 elements.

that this scheme can accurately solve smaller scale problems where the interface shape is of importance and where complex interface physics are involved. Special attention has been paid to making the scheme as generic as possible to allow for future implementations in higher dimensions. The STDG discretization ensures that the scheme is conservative as long as the numerical fluxes are conservative. A merging procedure is used to deal with small cells created by the cut-cell mesh refinement. Topological changes such as merging and coalescence can be handled in the method because of the level set.

The STDGFEM for two-fluid flows was applied to solve a magma–ideal gas shock tube problem and the interaction between a helium cylinder and a shock wave. An interface flux (60) was developed, which reduced oscillations at the interface at the cost of a small mass loss.

Candidates for further research are:

- Automation of the refinement procedure to allow 4D space–time applications. It is expected that such an automation will be valuable also outside of the current context.
- Incorporation of surface tension and curvature by means of level set methodology, including level set reinitialization.
- Incorporation of viscosity (Navier–Stokes equations).
- Improvement of performance using *h*-refinement and multigrid algorithms.
- Applications involving the shallow water equations to simulate flooding and drying [5,10,44,58], two-phase flows [46] and other applications [3,59,67]. Because of the methods' flexibility in defining flow domains with interfaces it is expected that valuable contributions are possible in these fields.

Acknowledgments

The authors are kindly indebted to V.R. Ambati, A. Bell, L. Pesch and S. Rhebergen for the valuable discussion, suggestions and support.

References

[1] D. Adalsteinsson, J.A. Sethian, A fast level set method for propagating interfaces, J. Comput. Phys. 118 (1995) 269–277.
[2] H.T. Ahn, M. Shashkov, Adaptive moment-of-fluid method, J. Comput. Phys. 228 (2009) 2792–2821.

[3] B. Akers, O. Bokhove, Hydraulic flow through a channel contraction: multiple steady states, Phys. Fluids 20 (2008) 056601.
[4] A.S. Almgren, J.B. Bell, P. Colella, T. Marthaler, A Cartesian grid projection method for the incompressible Euler equations in complex geometries, SIAM J. Sci. Comput. 18 (1997) 1289–1309.
[5] V.R. Ambati, O. Bokhove, Space–time discontinuous Galerkin discretization of rotating shallow water equations, J. Comput. Phys. 225 (2007) 1233–1261.
[6] V.R. Ambati, O. Bokhove, Space–time discontinuous Galerkin finite element method for shallow water flows, J. Comput. Appl. Math. 204 (2007) 452–462.
[7] O. Bokhove, Numerical modeling of magma-repository interactions, University of Twente, 2001, 97 pp. <http://eprints.eemcs.utwente.nl/>.
[8] O. Bokhove, Decompressie van magma in opslagtunnels, Ned. Tijdschr. Natuurk. 68 (2002) 232–235. English version: <http://eprints.eemcs.utwente.nl/>.
[9] O. Bokhove, A.W. Woods, A. de Boer, Magma flow through elastic-walled dikes, Theoret. Comput. Fluid Dyn. 19 (2005) 261–286.
[10] O. Bokhove, Flooding and drying in finite-element Galerkin discretizations of shallow-water equations. Part 1: One dimension, J. Sci. Comput. 22 (2005) 47–82.
[11] B. Cockburn, G.E. Karniadakis, C.W. Shu, Discontinuous Galerkin methods theory, computation and applications, Lecture Notes in Computational Science and Engineering, vol. 11, Springer, Berlin, 2000.
[12] B.J. Daly, Numerical study of the effect of surface tension on interface instability, Phys. Fluids 12 (1969) 1340–1354.
[13] D. de Zeeuw, K.G. Powell, An adaptively refined Cartesian mesh solver for the Euler equations, J. Comput. Phys. 104 (1993) 56–68.
[14] D.A. Edwards, H. Brenner, D.T. Wasan, Interfacial Processes and Rheology, Butterworth-Heineman, Stoneham, 1991.
[15] R.P. Fedkiw, T. Aslam, B. Merriman, S. Osher, A non-oscillatory Eulerian approach to interface in multimaterial flows (the ghost fluid method), J. Comput. Phys. 152 (1999) 457–492.
[16] K.J. Fidkowski, D.L. Darmofal, A triangular cut-cell adaptive method for high-order discretizations of the compressible Navier–Stokes equations, J. Comput. Phys. 225 (2007) 1653–1672.
[17] M.J. Fritts, W. Cowley, H.E. Trease (Eds.), The Free Lagrange Method, Lecture Notes on Physics, vol. 238, Springer-Verlag, New York, 1985.
[18] D.E. Fyfe, E.S. Oran, M.J. Fritts, Surface tension and viscosity with Lagrangian hydrodynamics on a triangular mesh, J. Comput. Phys. 76 (1988) 349–384.
[19] J. Glimm, J.W. Grove, X.-L. Li, K.-M. Shyue, Q. Zhang, Y. Zeng, Three-dimensional front tracking, SIAM J. Sci. Comput. 19 (1998) 703–727.
[20] J. Glimm, J.W. Grove, X.-L. Li, N. Zhao, Simple front tracking, Contemp. Math. 238 (1999) 133–149.
[21] J. Glimm, X.-L. Li, Y.-J. Liu, Z.-L. Xu, N. Zhao, Conservative front tracking with improved accuracy, SIAM J. Numer. Anal. 41-5 (2003) 1926–1947.
[22] D.M. Greaves, Simulation of viscous water column collapse using adapting hierarchial grids, Int. J. Numer. Methods Fluids 50 (2005) 693–711.
[23] J.-F. Haas, B. Sturtevant, Interaction of weak shock waves with cylindrical and spherical gas inhomogeneities, J. Fluid Mech. 181 (1987) 41–76.
[24] F.H. Harlow, J.E. Welch, Numerical calculation of time-dependent viscous incompressible flow of fluid with free surface, Phys. Fluids 8 (1965) 2182–2189.
[25] C.V. Hirt, B.D. Nichols, Volume of fluid (VOF) methods for the dynamics of free boundaries, J. Comput. Phys. 39 (1981) 201–255.
[26] C. Hu, C.W. Shu, A discontinuous Galerkin finite element method for Hamilton–Jacobi equations, SIAM J. Sci. Comput. 21 (1999) 666–690.
[27] J.M. Hyman, Numerical methods for tracking interfaces, Physica D 12 (1984) 396–407.
[28] C.M. Klaij, J.J.W. van der Vegt, H. van der Ven, Space–time discontinuous Galerkin method for the compressible Navier–Stokes equations, J. Comput. Phys. 217 (2006) 589–611.
[29] C.M. Klaij, J.J.W. van der Vegt, H. van der Ven, Pseudo-time stepping methods for space–time discontinuous Galerkin discretizations of the compressible Navier–Stokes equations, J. Comput. Phys. 219 (2006) 622–643.
[30] L. Krivodonova, J. Xin, J.-F. Remacle, N. Chevaugeon, J.E. Flaherty, Shock detection and limiting with discontinuous Galerkin methods for hyperbolic conservation laws, Appl. Numer. Math. 48 (2004) 323–338.
[31] D. Kröner, Numerical Schemes for Conservation Laws, Wiley und Teubner, Stuttgart, 1997.
[32] M. Kucharik, J. Limpouch, R. Liska, Laser plasma simulations by Arbitrary Lagrangian Eulerian method, J. de Phys. 133 (2006) 167–169.
[33] R.J. LeVeque, K.-M. Shyue, Two-dimensional front tracking based on high resolution wave propagation methods, J. Comput. Phys. 123 (1996) 354–368.
[34] H. Luo, J.D. Baum, R. Lohner, A Hermite WENO-based limiter for discontinuous Galerkin method on unstructured grids, J. Comput. Phys. 225 (2007) 686–713.
[35] J. Magnaudet, M. Rivero, J. Fabre, Accelerated flows around a rigid sphere or a spherical bubble. Part 1: Steady straining flow, J. Fluid Mech. 284 (1995) 97–135.
[36] S.J. Osher, J.A. Sethian, Fronts propagating with curvature dependent speed: algorithms based on Hamilton–Jacobi formulations, J. Comput. Phys. 79 (1988) 12–49.
[37] S.J. Osher, C.W. Shu, High-order essentially nonoscillatory schemes for Hamilton–Jacobi equations, SIAM J. Numer. Anal. 28 (1991) 907–922.
[38] J. Palaniappan, S.T. Miller, R.B. Haber, Sub-cell shock capturing and spacetime discontinuity tracking for nonlinear conservation laws, Int. J. Numer. Methods Fluids 57 (2008) 1115–1135.
[39] R.B. Pember, J.B. Bell, P. Colella, W.Y. Curtchfield, M.L. Welcome, An adaptive Cartesian mesh method for unsteady compressible flow in irregular regions, J. Comput. Phys. 120 (1995) 278–304.
[40] D. Peng, B. Merriman, S. Osher, H.K. Zhao, M. Kang, A PDE-based fast local level set method, J. Comput. Phys. 155 (1999) 410–438.
[41] L. Pesch, A. Bell, W.E.H. Sollie, V.R. Ambati, O. Bokhove, J.J.W. van der Vegt, hpGEM- A software framework for discontinuous Galerkin finite element methods, ACM Trans. Math. Software 33 (2007).
[42] E.G. Puckett, J.S. Saltzman, A 3D adaptive mesh refinement algorithm for interfacial gas dynamics, Physica D 60 (1992) 84–93.
[43] J.X. Qiu, T.G. Liu, B.C. Khoo, Simulations of compressible two-medium flow by Runge–Kutta discontinuous Galerkin methods with the ghost fluid method, Commun. Comput. Phys. 3 (2008) 479–504.
[44] J.-F. Remacle, S.S. Frazao, X. Li, M.S. Shephard, An adaptive discretization of shallow-water equations based on discontinuous Galerkin methods, Int. J. Numer. Methods Fluids 52 (2006) 903–923.
[45] S. Rhebergen, O. Bokhove, J.J.W. van der Vegt, Discontinuous Galerkin finite element methods for hyperbolic nonconservative partial differential equations, J. Comput. Phys. 227 (2008) 1887–1922.
[46] S. Rhebergen, O. Bokhove, J.J.W. van der Vegt, Discontinuous Galerkin finite element method for shallow two-phase flows, Comput. Methods Appl. Mech. Eng. 198 (2009) 819–830.
[47] R.D. Richtmyer, K.W. Morton, Difference Methods for Initial-Value Problems, Inter-science, New York, 1967.
[48] J.M. Rudman, Volume-tracking methods for interfacial flow calculations, Int. J. Numer. Methods Fluids 24 (1997) 671–691.
[49] G. Ryskin, L.G. Leal, Numerical solution of free-boundary problems in fluid mechanics, Part 2: Buoyancy-driven motion of a gas bubble through a quiescent liquid, J. Fluid Mech. 148 (1984) 19–35.
[50] R. Saurell, R. Abgrall, A simple method for compressible multifluid flows, SIAM J. Sci. Comput. 21 (1999) 1115–1145.
[51] L.E. Scriven, Dynamics of a fluid interface, Chem. Eng. Sci. 12 (1960) 98–108.
[52] J.A. Sethian, Level Set Methods, Cambridge University Press, 1996.
[53] J.A. Sethian, P. Smereka, Level set methods for fluid interfaces, Annu. Rev. Fluid Mech. 35 (2003) 341–372.
[54] W.E.H. Sollie, Space–time discontinuous Galerkin finite element method for two-fluid flows, Ph.D. Thesis, University of Twente, 2010. <http://eprints.eemcs.utwente.nl/>.
[55] J.J. Sudirham, J.J.W. van der Vegt, R.M.J. van Damme, Space–time discontinuous Galerkin method for advection–diffusion problems on time-dependent domains, Appl. Numer. Math. 56 (2006) 1491–1518.
[56] M. Sussman, P. Smereka, S. Osher, A level set approach for computing solutions to incompressible two-phase flow, J. Comput. Phys. 114 (1994) 146–159.

[57] M. Sussman, E.G. Puckett, A coupled level set and volume-of-fluid method for computing 3D and axisymmetric incompressible two-phase flows, J. Comput. Phys. 162 (2000) 301–337.
[58] P.A. Tassi, O. Bokhove, C.A. Vionnet, Space discontinuous Galerkin method for shallow water flows-kinetic and HLLC flux, and potential vorticity generation, Adv. Water Resour. 30 (2007) 998–1015.
[59] P.A. Tassi, S. Rhebergen, C.A. Vionnet, O. Bokhove, A discontinuous Galerkin finite element model for bed evolution under shallow flows, Comput. Methods Appl. Mech. Eng. 197 (2008) 2930–2947.
[60] P.G. Tucker, Z. Pan, A Cartesian cut element method for incompressible viscous flow, Appl. Math. Model. 24 (2000) 591–606.
[61] H.S. Udaykumar, R. Mittal, P. Rampunggoon, A. Khanna, A sharp interface Cartesian mesh method for simulating flows with complex moving boundaries, J. Comput. Phys. 174 (2001) 345–380.
[62] A. Ungor, A. Sheffer, Pitching tents in space–time: mesh generation for discontinuous Galerkin method, Int. J. Found. Comput. Sci. 13 (2002) 201–221.
[63] S.O. Unverdi, G. Tryggvason, Computations of multi-fluid flows, Physica D 60 (1992) 70–83.
[64] S.O. Unverdi, G. Tryggvason, A front-tracking method for viscous, incompressible multi-fluid flows, J. Comput. Phys. 100 (1992) 25–37.
[65] J.J.W. van der Vegt, H. van der Ven, Space–time discontinuous Galerkin finite element method with dynamic mesh motion for inviscid compressible flows, J. Comput. Phys. 182 (2002) 546–585.
[66] J.J.W. van der Vegt, Y. Xu, Space–time discontinuous Galerkin method for nonlinear water waves, J. Comput. Phys. 224 (2007) 17–39.
[67] A.W. Vreman, M. Al-Tarazi, J.A.M. Kuipers, M. van Sint Annaland, O. Bokhove, Supercritical shallow granular flow through a contraction: experiment, theory and simulation, J. Fluid Mech. 578 (2007) 233–269.
[68] J. Wackers, B. Koren, A fully conservative model for compressible two-fluid flow, Int. J. Numer. Methods Fluids 47 (2005) 1337–1343.
[69] A.W. Woods, S. Sparks, O. Bokhove, A.-M. Lejeune, C.B. Connor, B.E. Hill, Modeling magma-drift interaction at the proposed high-level radioactive waste repository at Yucca Mountain, Nevada, USA, Geophys. Res. Lett. 29 (2002) 1641.
[70] A.W. Woods, O. Bokhove, A. de Boer, B.E. Hill, Compressible magma flow in a two-dimensional elastic-walled dike, Earth Planet. Sci. Lett. 246 (2006) 241–250.
[71] G. Yang, D.M. Causon, D.M. Ingram, Calculation of compressible flows about complex moving geometries using a three-dimensional Cartesian cut element method, Int. J. Numer. Methods Fluids 33 (2000) 1121–1151.
[72] T. Ye, R. Mittal, H.S. Udaykumar, W. Shyy, An accurate Cartesian grid method for viscous incompressible flows with complex immersed boundaries, J. Comput. Phys. 156 (1999) 209–240.
[73] S.S. Young, J.L. White, E.S. Clark, Y. Oyanagi, A basic experimental study of sandwich injection moulding with sequential injection, Pol. Eng. Sci. 20 (1980) 798–804.