SPACE-TIME DISCONTINUOUS GALERKIN FINITE ELEMENT METHOD FOR TWO-FLUID FLOWS

Henk Sollie

Colophon

The research presented in this dissertation was carried out at the Numerical Analysis and Computational Mechanics (NACM) group, Department of Applied Mathematics, Faculty of Electrical Engineering, Mathematics and Computer Science of the University of Twente, The Netherlands.

This work has been part of the research program "Dispersed multiphase flows" of the Institute for Mechanics, Process and Control, Twente; at the University of Twente. Financial support of the Department of Applied Mathematics, University of Twente, The Netherlands is gratefully acknowledged.

This thesis was typeset in LaTeXby the author and printed by Wöhrmann Printing Service, Zutphen, The Netherlands.

© W.E.H. Sollie, 2010.

All right reserved. No part of this work may be reproduced, stored in a retrieval system, or transmitted in any form or by any means, electronic, mechanical, photocopying, recording, or otherwise, without prior permission from the copyright owner.

ISBN 978-90-365-3008-8

SPACE-TIME DISCONTINUOUS GALERKIN FINITE ELEMENT METHOD FOR TWO-FLUID FLOWS

PROEFSCHRIFT

ter verkrijging van
de graad van doctor aan de Universiteit Twente,
op gezag van de rector magnificus,
prof. dr. H. Brinksma,
volgens besluit van het College voor promoties
in het openbaar te verdedigen
op vrijdag 16 april 2010 om 13.15 uur

door

Warnerius Egbert Hendrikus Sollie

geboren op 16 juli 1974
te Zwolle

This dissertation has been approved by the promotor,
prof. dr. ir. J. J. W. van der Vegt
and the assistant promotor,
dr. ir. O. Bokhove.

Contents

Chapter 1

Introduction

Fluid flows with interfaces involve combinations of gasses, 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.

In this thesis 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.

The discontinuous Galerkin (DG) finite element method was first proposed by Reed and Hill for solving the neutron equation [69]. It was further developed for hyperbolic partial differential equations by Cockburn et al., who introduced the Runge-Kutta discontinuous Galerkin (RKDG) method [17, 18, 19, 20] and its generalization, the local discontinuous Galerkin (LDG) method [21]. See also [8, 9, 10, 11, 47, 59]. For a complete survey of DG methods and their applications, see [22]. In [23] post processing to enhance the accuracy of the solution was introduced. Recently, also space-time DG methods have been proposed which make use of advancing front strategies [62, 94].

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. 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. Recently, Luo et

al. [56] proposed the Hermite WENO limiter for DG methods, which uses Hermite reconstruction polynomials to maintain a small stencil even for higher order solutions. Krivodonova et al. [51] proposed a discontinuity detector for DG methods for hyperbolic conservation laws based on a result of strong superconvergence at the outflow boundary of each element. The discontinuity detector is used to prevent activation of the slope limiter in a smooth solution, which would otherwise reduce accuracy.

The space-time discontinuous Galerkin finite element method (STDG) introduced by van der Vegt and van der Ven ([98]) is a space-time variant of the DG method which is especially suited for handling dynamic mesh motions in space-time (See also [7, 49, 83, 100]). 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 DG discretization.

Many methods have been proposed for computing flows with interfaces or, to be more general, fronts [77]. 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 [26, 43]. 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 [44, 66, 76, 109] 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 $\bar{\nabla} = (\partial/\partial x_1, \cdots, \partial/\partial x_d)$ 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. Also, they can automatically handle reconnection and breakup. Also, current VoF methods can conserve mass. However, VoF methods 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 [40, 41], which alleviates some of these problems.

The Level Set Method (LSM) was introduced by Osher and Sethian in [60] and further developed in [1, 79, 84]. For a survey, see [80]. 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 [45, 61, 64]. 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 [85].

Front capturing methods have the advantage of a relatively simple formulation. The main drawback of these methods lies in the need for com-

plex 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 [73] and further developed in [35, 36, 37, 55, 58, 90, 95] and [96]. For a survey, see [46] and [74]. 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. [38] 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 [39].

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, 16, 24, 28, 48, 63, 68, 89, 91, 92, 93, 105, 106, 107] 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 [108].

In Lagrangian or moving mesh methods [25, 27, 32, 33, 34, 57, 75] 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, 54]). 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.

To meet the aim of the present research it was chosen to combine 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. In order to structure the investigation, three research objectives are defined:

1. Develop a space-time discontinuous Galerkin method in combination with an accurate interface treatment using cut-cells and a level set method. To ensure that the complete method will have good stability and accuracy properties the individual components of the method need to connect and interact with each other correctly. Furthermore, the issue of performance should also be taken into consideration.

2. Investigate the numerical properties and performance of the method for a number of test problems. Firstly, a number of benchmark tests will be considered with the purpose of validating various individual aspects the method. Secondly, a number of more challenging real life applications will be considered, to investigate the behavior of the complete method.

3. Investigate the design and implementation aspects of the method. Since the method is composed of many complex parts its implementation is expected to be non trivial.

This thesis contains and extends the material presented in [81] and [82]. The outline is as follows. In Chapter 2 the numerical technique for the

solution of two-fluid flows is presented, followed in Chapter 3 by a discussion of several numerical problems with growing complexity. In Chapter 4, design and implementation issues of the two-fluid method are discussed. Finally, in Chapter 5 conclusions and recommendations for further research are presented.

Chapter 2

Two-Fluid Space-Time Discontinuous Galerkin Finite Element Method. Part I: Numerical Algorithm

2.1 Introduction

In this chapter a novel numerical method is presented for solving two-fluid flows which combines aspects of front capturing and front tracking methods with a space-time discontinuous Galerkin (STDG) finite element discretization. This new approach provides an accurate and versatile scheme for dealing with interfaces in two-fluid flow problems which can alleviate some of the problems encountered in existing methods. In order to explain and motivate the choices made in this method, first some aspects of existing techniques for dealing with interfaces are discussed. This is followed by a discussion of the space-time discontinuous Galerkin finite element method.

In Front Tracking methods the front is tracked explicitly in the mesh, typically by either moving the nodes in the mesh (Lagrangian methods) or by means of local h-refinement (Cut-Cell method). Front tracking meth-

ods can reach high accuracy when the interface representation is detailed enough, even on coarse meshes. Also, due to the explicit representation of the interface front tracking methods are good candidates for solving problems that involve complex interface physics. A drawback of front tracking methods is that they require a significant effort to implement, especially in higher dimensions, due to the complexity of the geometric refinement. Also, interface topological changes due to breakup or merging typically cannot be handled easily. A problem which is most notable in Cut-Cell methods, is the occurrence of small elements which can result in stiffness of the equations and numerical instability. In Lagrangian front tracking methods frequent remeshing may be required when the mesh deformations are large which introduces additional interpolation errors and is computationally expensive.

In front capturing methods the interface is not explicitly represented in the mesh but instead a regular stationary mesh is used in combination with an alternative interface representation, most often by means of particles or functions. Examples of front capturing methods are the Marker And Cell (MAC), the Volume of Fluid (VoF) and the Level Set (LSM) methods. In general, front capturing methods have the benefit of a relatively simple formulation; hence, they are easy to extend to higher dimensions. However, these methods tend to have difficulties in maintaining a sharp interface between fluids and may require complex interface shape restoration techniques. Also, front capturing methods can typically handle interface topological changes well but spurious interfaces may spawn in cases.

In the numerical method presented in this thesis front capturing and front tracking techniques are combined. The method makes use of two meshes, a background mesh for level set computations and a two-fluid mesh for two-fluid flow computations, as illustrated in Figure 2.1. The refined mesh is constructed from the background mesh based on the 0-level set by means of cut-cell mesh refinement.

In this chapter the numerical method will be discussed. The numerical applications are relegated to Chapter 3. The outline of this chapter is as follows. In Section 2.2 the flow and level set equations are introduced. In Section 2.3 the background and refined meshes are discussed and the mesh

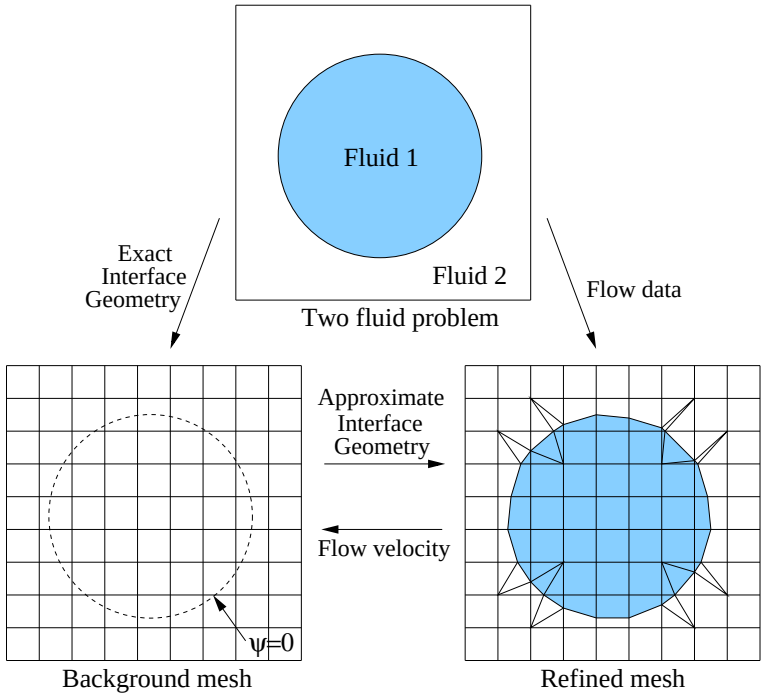

Figure 2.1: Representation of a flow problem in the two-fluid method. The background mesh is used for computing the interface dynamics by means of the level set method. The refined mesh is used for flow computations. The zero level set $\psi = 0$ provides the basis for the refinement of the background mesh into the refined mesh. The level set is advected with the flow velocity.

refinement procedure is presented. In Section 2.4 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 2.5 the two-fluid algorithm is presented which is followed in Section 2.6 by a final discussion and conclusions.

2.2 Equations

2.2.1 Two-fluid flow equations

Considered are flow problems involving two fluids as illustrated in Figure 2.2. The two fluids are separated in space-time by an interface S. Let $i = 1, 2$ denote the fluid index. Furthermore, let $\mathbf{x} = (t, \bar{\mathbf{x}}) = (x_0, \cdots, x_d)$ denote the space-time coordinates, with d the spatial dimension, $\bar{\mathbf{x}} = (x_1, \cdots, x_d)$ 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 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:

$$\frac{\partial \mathbf{w}^i}{\partial t} + \bar{\nabla} \cdot F^i(\mathbf{w}^i) = 0, \tag{2.1}$$

where $\bar{\nabla} = (\partial/\partial x_1, \ldots, \partial/\partial x_d)$ denotes the spatial gradient operator and $F^i(\mathbf{w}^i) = (F^i_1, \cdots, F^i_d)$ the spatial flux tensor for fluid i with F^i_j the j-th flux vector and $j = 1, \cdots d$. Reformulated in space-time (2.1) becomes:

$$\begin{aligned} &\nabla \cdot \mathcal{F}^i(\mathbf{w}^i) = 0, \text{ with} \\ &\mathcal{F}^i(\mathbf{w}^i) = (\mathbf{w}^i, F^i(\mathbf{w}^i)), \end{aligned} \tag{2.2}$$

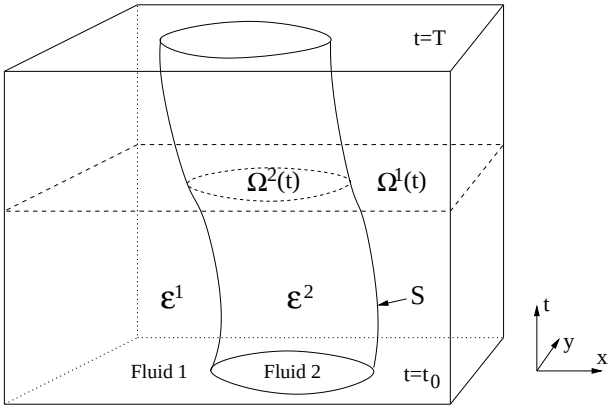

Figure 2.2: An example two-fluid flow problem in space-time. Here $\mathcal{E}^i$ and $\Omega^i(t)$ denote the space-time and space flow domains for fluids $i = 1, 2$; and, S denotes the interface between the two fluids in space-time.

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:

$$\mathbf{w}^i(0, \bar{\mathbf{x}}) = \mathbf{w}_0^i(\bar{\mathbf{x}}), \tag{2.3}$$

boundary conditions:

$$\mathbf{w}^i(t, \bar{\mathbf{x}}) = \mathcal{B}_B^i(\mathbf{w}^i, \mathbf{w}_b^i) \text{ on } \mathcal{Q}^i/S \tag{2.4}$$

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

$$\mathbf{w}^i(t, \bar{\mathbf{x}}) = \mathcal{B}_S^i(\mathbf{w}^1, \mathbf{w}^2) \text{ on } S. \tag{2.5}$$

Since the actual flow variables, fluxes and initial, boundary and interface conditions are problem specific they shall be provided in Chapter 3 where the test cases are discussed.

2.2.2 Level set equation

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

$$\psi(t,\bar{\mathbf{x}}) = \begin{cases} < 0 & \text{in Fluid 1} \\ > 0 & \text{in Fluid 2} \\ = 0 & \text{at the interface.} \end{cases} \tag{2.6}$$

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

$$\psi(t,\bar{\mathbf{x}}) = \alpha \inf_{\forall \bar{\mathbf{x}}_S \in S(t)} \|\bar{\mathbf{x}} - \bar{\mathbf{x}}_S\|, \tag{2.7}$$

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

$$\frac{\partial \psi}{\partial t} + \bar{\mathbf{a}} \cdot \bar{\nabla}\psi = 0, \tag{2.8}$$

where $\bar{\mathbf{a}} = (a_1, \cdots, a_d)$ 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:

$$\psi(0,\bar{\mathbf{x}}) = \psi_0(\bar{\mathbf{x}}), \text{ for } \bar{\mathbf{x}} \in \Omega(t_0). \tag{2.9}$$

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

$$\bar{\mathbf{a}}(t,\bar{\mathbf{x}}) \cdot \bar{\mathbf{n}} = 0, \text{ for } (t,\bar{\mathbf{x}}) \in \mathcal{Q}, \tag{2.10}$$

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

2.3 Meshes

2.3.1 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:

$$\mathcal{T}_h^{i,n} = \left\{ \mathcal{K}_j^{i,n} \subset \mathbb{R}^{d+1} | \bigcup_{j=1}^{N_h^i} \bar{\mathcal{K}}_j^{i,n} = \bar{\mathcal{I}}_n^i \right. \\ \left. \text{and } \mathcal{K}_j^{i,n} \bigcap \mathcal{K}_{j'}^{i,n} = \emptyset \text{ if } j \neq j', 1 \leq j, j' \leq N_h^{i,n} \right\} \quad (2.11)$$

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 $\bar{\mathcal{K}}_j^{i,n} = \mathcal{K}_j^{i,n} \cup \partial\mathcal{K}_j^{i,n}$ denotes the closure of the space-time element. The tessellations $\mathcal{T}_h^{i,n}$ will be referred to as the two-fluid or refined mesh (see Figure 2.3), 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 $S_h^{i,n}$ as a collection of finite element faces. The numerical interface is assumed to be geometrically identical in both tessellations, $S_h^{1,n} = S_h^{2,n}$. 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 $\mathcal{T}_h^{i,n}$

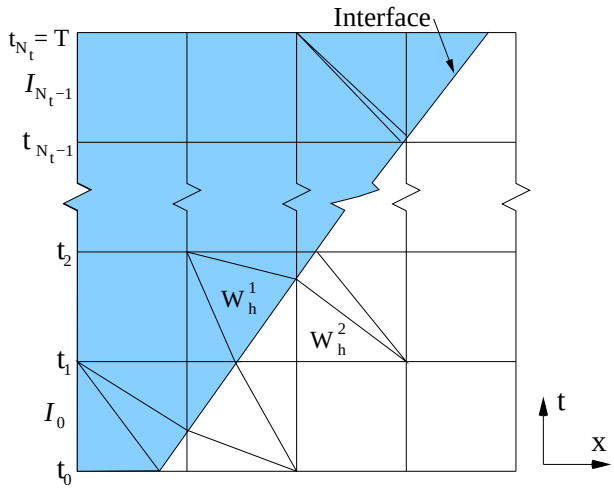

Figure 2.3: Two-fluid mesh.

is defined as:

$$B_h^k(\mathcal{T}_h^{i,n}) = \{\mathbf{w} \in L^2(\mathcal{E}_h^i) : \mathbf{w}|_{\mathcal{K}} \circ G_{\mathcal{K}} \in P^k(\hat{\mathcal{K}}), \forall \mathcal{K} \in \mathcal{T}_h^{i,n}\} \tag{2.12}$$

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

$$G_{\mathcal{K}_j^{i,n}} : \hat{\mathcal{K}} \to \mathcal{K}_j^{i,n} : \xi \mapsto \mathbf{x} = \sum_{k=1}^{N_{F,j}^{i,n}} x_k(\mathcal{K}_j^{i,n}) \chi_k(\xi) \tag{2.13}$$

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 $\hat{\mathcal{K}}$, with $\xi = (\xi_0, \cdots, \xi_d)$ the coordinates in the reference element. Given a set of basis functions $\hat{\phi}_m$ defined on the

reference element, the basis functions $\phi_m : \mathcal{K}_j^{i,n} \rightarrow \mathbb{R}$ are defined on the space-time elements $\mathcal{K}_j^{i,n} \in \mathcal{T}_h^{i,n}$ by means of the mapping $G_{\mathcal{K}_j^{i,n}}$:

$$\phi_m = \hat{\phi}_m \circ G_{\mathcal{K}_j^{i,n}}^{-1}. \tag{2.14}$$

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

$$\mathbf{w}_h^i(t, \bar{\mathbf{x}})|_{\mathcal{K}_j^{i,n}} = \sum_m \hat{\mathbf{W}}_m^i(\mathcal{K}_j^{i,n}) \phi_m(t, \bar{\mathbf{x}}) \tag{2.15}$$

with $\hat{\mathbf{W}}_m^i$ 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.

2.3.2 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:

$$\mathcal{T}_b^n = \left\{ \mathcal{K}_{b,\tilde{j}}^n \subset \mathbb{R}^{d+1} | \bigcup_{\tilde{j}=1}^{N_b} \bar{\mathcal{K}}_{b,\tilde{j}}^n = \bar{\mathcal{I}}_n \right.$$
$$\left. \text{and } \mathcal{K}_{b,\tilde{j}}^n \bigcap \mathcal{K}_{b,\tilde{j}'}^n = \emptyset \text{ if } \tilde{j} \neq \tilde{j}', 1 \leq \tilde{j}, \tilde{j}' \leq N_b \right\} \tag{2.16}$$

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 2.3.1 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:

$$\psi_h(t, \bar{\mathbf{x}})|_{\mathcal{K}^n_{b,\tilde{j}}} = \sum_m \hat{\mathbf{\Psi}}_m(\mathcal{K}^n_{b,\tilde{j}})\phi_m(t, \bar{\mathbf{x}}), \tag{2.17}$$

with $\hat{\mathbf{\Psi}}_m$ 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:

$$\bar{\mathbf{a}}_h(t, \bar{\mathbf{x}})|_{\mathcal{K}^n_{b,\tilde{j}}} = \sum_m \hat{\mathbf{A}}_m(\mathcal{K}^n_{b,\tilde{j}})\phi_m(t, \bar{\mathbf{x}}), \tag{2.18}$$

2.3.3 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

Algorithm 1 Mesh refinement algorithm.

FOR every element $\mathcal{K}^n_{b,\tilde{j}}$ *in* $\mathcal{T}^n_b$ *DO*
  Calculate intersection of 0*-level set* $\psi_c = 0$ *with* $\mathcal{K}^n_{b,\tilde{j}}$
  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 $\mathcal{K}^{i,n}_{h,\hat{j}}$ *and store in* $\mathcal{T}^{i,n}_h$
  END DO
END DO
Generate faces for $\mathcal{T}^{i,n}_h$
FOR every element $\mathcal{K}^{i,n}_{h,j}$ *in* $\mathcal{T}^{i,n}_h$
  Initialize data on $\mathcal{K}^{i,n}_{h,j}$
END DO

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. The refinement rules define how a single element will be refined given an intersection with a 0-level set. The global refinement step is followed by a face generation step to create the connectivity between the refined elements. The face generation is straightforward and will not be discussed.

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 can contain only one fluid and it is copied directly to the refined mesh $\mathcal{T}^n_h$. 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 and the actual 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:

$$\mathbf{x}_I = \frac{\mathbf{x}_A \psi_h(\mathbf{x}_B) - \mathbf{x}_B \psi_h(\mathbf{x}_A)}{\psi_h(\mathbf{x}_A) - \psi_h(\mathbf{x}_B)}, \tag{2.19}$$

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, to avoid difficulties with face integration the refined mesh should have full connectivity. Element refinement rules have been developed for two and three dimensions, similar to [38], which will be discussed now.

2.3.4 2D Refinement

Considered is a 2D background mesh containing only square elements. In order to define the 2D mesh refinement, first the symmetries of the square are introduced, followed by a discussion of all the relevant types of cuts in 2D and the introduction of a set of base types. Next, the square permutations are applied to the base types to find for each cut type the base type and the permutation which maps the base type to the cut type. Finally, the actual refinement rules are defined for each of the base types. In the mesh refinement algorithm, the refinement rule for a given cut type is obtained

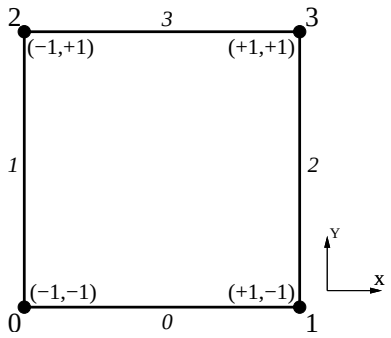

Figure 2.4: Vertex and edge numbering and nodal coordinates for the reference square.

by permuting the element refinement rule of the base type to the given cut type.

Square symmetries

The vertices and edges of the reference square are numbered using Local Node Indices (LNI) as shown in Figure 2.4. A square has a total of 8 symmetries usually referred to as the dihedral group D_4. Of these 4 are rotational symmetries and 4 are reflection symmetries. To describe the permutations there are two main notations, firstly as a decomposition in a product of disjoint cycles and secondly in relation notation. For example, in performing a counter clockwise rotation by 90 degrees, vertex 0 will move to vertex 1, vertex 1 to vertex 3, vertex 3 to vertex 2 and vertex 2 to vertex 0. In a decomposition in a product of disjoint cycles this is denoted as (0132). In relation notation it is denoted as $\{1, 3, 0, 2\}$, where the index into the array gives the 'from' vertex and the value gives the 'to' vertex. The square symmetries are defined in Table 2.1. Here, permutation 0 describes the identity, permutations $1 - 3$ describe rotations and permutations $4 - 7$ describe reflections.

In the refinement algorithm permutations are needed not only of the vertices but also of nodes lying on edges. For this purpose the edge mid-

Table 2.1: Square symmetries.

index	disjoint cycles	relation notation	Comment
0	(0)(1)(2)(3)	$\{0,1,2,3\}$	Identity
1	(0132)	$\{1,3,0,2\}$	90° right rotation
2	(03)(12)	$\{3,2,1,0\}$	180° right rotation
3	(0231)	$\{2,0,3,1\}$	270° right rotation
4	(02)(13)	$\{2,3,0,1\}$	Reflection x-axis
5	(01)(23)	$\{1,0,3,2\}$	Reflection y-axis
6	(03)(1)(2)	$\{3,1,2,0\}$	Reflection diagonal
7	(12)(0)(3)	$\{0,2,1,3\}$	Reflection diagonal

Algorithm 2 Algorithm to determine edge permutation.

Given an edge with index i on the reference square
Get the vertex indices of the two edge vertices
Determine the permutation of the vertex indices
Find the index of the permuted edge from the permuted vertex indices

points are numbered from 4 to 7, ordered in the same way as the edges. Given the index of the edge, the index of the edge midpoint is found by adding 4, the number of vertices of the square. Hence, the permutation of an edge midpoint is found directly from the permutation of the edge. The permutation of an edge is found by looking at the permutations of its vertices. The algorithm is given in Algorithm 2. As an example, when applying permutation 1 to the edge 0, first the edge vertices are retrieved, in this case 0 and 1. Permuting these vertices gives permuted edge vertices 1 and 3; hence, the permuted edge is 2.

2D base types

The classification of the 2D cuts is based on the values of the level set in the four vertices of the square. Each 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 $-$

Table 2.2: 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

and + corresponding to 0 and 1, respectively, we can assign the number $0011 = 3$. Since a square has 4 vertices, there are $2^4 = 16$ possibilities. In two-dimensional space-time three refinement types have been defined as given in Table 2.2. In Figure 2.5 the signs of the level set in each vertex for every type are shown. In Figure 2.6 the corresponding cuts are shown, where for simplicity the interface cuts at the edges midpoints only. For Type 2 two types of interface cuts are possible. The refinement rule will be able to handle both possibilities.

2D base type permutations

The symmetries of the square are applied to the base cut types to find for each cut type the base type and the permutation from the base type to the cut type. For simplification, the sign of the level set in vertex 0 is assumed to be $-$ (0), meaning that the cut types need to be explicitly defined only for the indices $0 - 7$. To calculate the cut type for an index in the range $8 - 15$ the index value only has to be subtracted from the number 15. In Table 2.3 the base type is given for each cut type, based on the index of the cut type. The algorithm used to fill the table is given in Algorithm 3. Due to symmetries in the base types, different permutations can give equal results; hence, the permutation index is not necessarily uniquely defined.

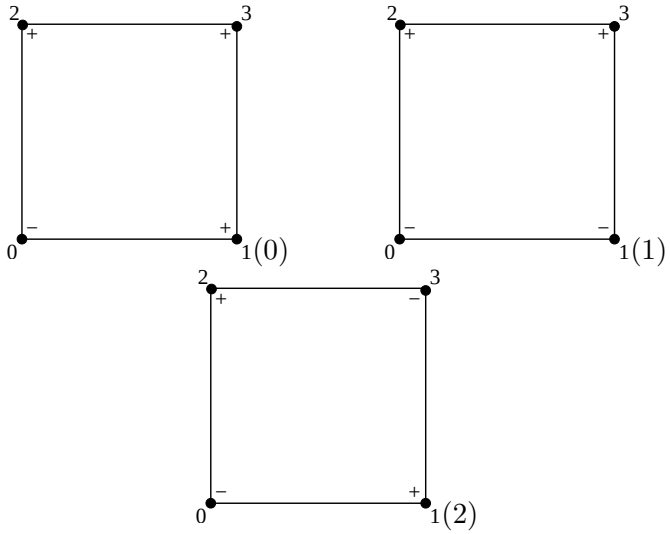

Figure 2.5: The vertex level set signs for the 2D base types.

Table 2.3: 2D base types corresponding to cut type indices $0 - 15$.

index	base type	index	base type
0	No cut	8	Type 0
1	Type 0	9	Type 2
2	Type 0	10	Type 1
3	Type 1	11	Type 0
4	Type 0	12	Type 1
5	Type 1	13	Type 0
6	Type 2	14	Type 0
7	Type 0	15	No cut

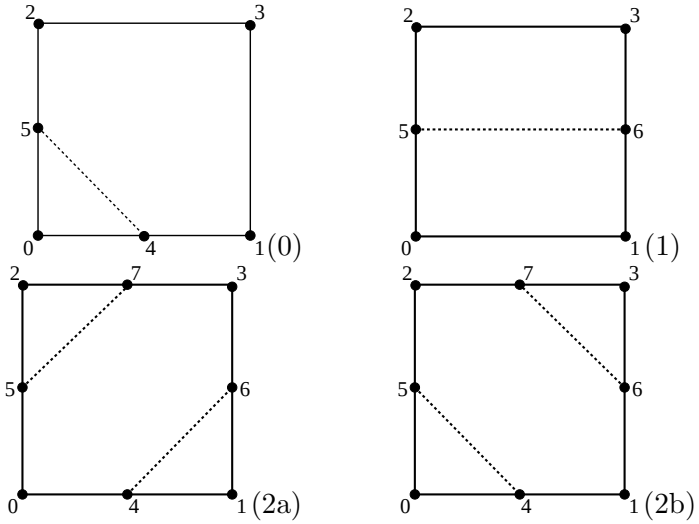

Figure 2.6: The interface cuts for the 2D base types. For type 2 two interface cuts are possible, which are both supported by the type 2 element refinement rule.

Algorithm 3 Algorithm for filling the permutation lookup table.

Initialize permVec[8] of [type index, perm index] with [-1,-1]
FOR type index i from 0 to 3 DO
  FOR permutation index j from 0 to 7 DO
    Determine permutation j of cut type i
    Calculate index k of the permuted type
    Store [i,j] in permVec[k]
  END DO
END DO

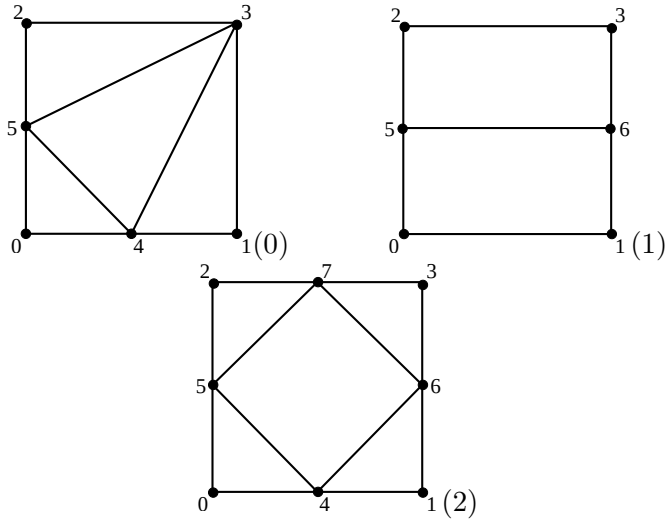

Figure 2.7: The element refinements for the 2D base types.

2D base type refinement

The element refinements for the 2D base types are shown in Figure 2.7 and the element refinements are given in Table 2.4. The algorithm to determine the element refinements given a level set configuration on a reference square is defined in Algorithm 4.

2.3.5 3D Refinement

Considered is a 3D background mesh containing only cubical elements. In order to define the 3D refinement, first the symmetries of the cube are introduced, followed by a discussion of all the relevant types of cuts in 3D and the introduction of the 3D base types. Next, the cube permutations are applied to the base types to find for each cut type the base type and the permutation which maps the base type to the cut type. Finally, the actual refinement rules are defined for each of the base types.

Table 2.4: 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

Algorithm 4 Algorithm for determining element refinements.

Calculate index i for level set configuration
get base type from permVec[i]
IF base type does not equal -1 *(unhandled type)*
  FOR all child elements j DO
    FOR all local node indices k of child element j DO
      IF ($k < 4$) (square vertex)
        Get permutation index l from permVec[i]
        Determine permuted local node index k' of node k
      ELSE IF ($4 < k < 8$) (edge midpoint)
        Calculate edge index $e = k - 4$
        Find permuted edge index e'
        Calculate permuted local node index $k' = e' + 4$
      END IF
      Store k' as local node index of permuted child element j
    END DO
  END DO
END IF
Return permuted child elements local node indices

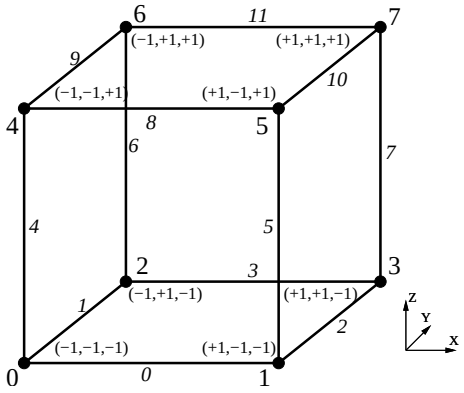

Figure 2.8: Vertex and edge numbering and nodal coordinates for the reference cube.

Cube symmetries

The vertices and edges of the reference cube are numbered as shown in Figure 2.8. A cube has a total of 48 symmetries which are usually referred to as octahedral symmetries, since the symmetries of the cube are the same as those of its dual, the octahedron. Of these 24 are rotational symmetries which are orientation preserving. The remaining 24 are combinations of rotations and reflections. The cube symmetries are defined in Table 2.5. Here, permutation 0 describes the identity permutation, permutations $1-6$ describe a 90 degree rotation around the axis from the face center to the opposite face center, permutations $7-9$ describe 180 degree rotation around the axis from the face center to the opposite face center, permutations $10-15$ describe 180 degree rotation around the axis from the edge center to the opposite edge center and permutations $16-23$ describe 120 degree rotation around a body diagonal. Permutations $24-47$ are defined by taking permutations $0-23$ and applying inversion $(07)(16)(25)(34)$. Similarly to what was done in the 2D refinement using Algorithm 2, the edge midpoints are numbered from 8 to 19 and the index of the edge midpoint is found by adding 8, the number of vertices of the cube, to the index of the edge.

Table 2.5: Octahedral symmetries.

index	disjoint cycles	relation notation	Comment
0	(0)(1)(2)(3)(4)(5)(6)(7)	$\{0,1,2,3,4,5,6,7\}$	Identity
1	(0132)(4576)	$\{1,3,0,2,5,7,4,6\}$	90° rotation z-axis
2	(0231)(4675)	$\{2,0,3,1,6,4,7,5\}$	90° rotation z-axis
3	(0462)(1573)	$\{4,5,0,1,6,7,2,3\}$	90° rotation x-axis
4	(0264)(1375)	$\{2,3,6,7,0,1,4,5\}$	90° rotation x-axis
5	(0154)(2376)	$\{1,5,3,7,0,4,2,6\}$	90° rotation y-axis
6	(0451)(2673)	$\{4,0,6,2,5,1,7,3\}$	90° rotation y-axis
7	(03)(12)(47)(56)	$\{3,2,1,0,7,6,5,4\}$	180° rotation z-axis
8	(06)(24)(17)(35)	$\{6,7,4,5,2,3,0,1\}$	180° rotation x-axis
9	(05)(14)(27)(36)	$\{5,4,7,6,1,0,3,2\}$	180° rotation y-axis
10	(01)(25)(34)(67)	$\{1,0,5,4,3,2,7,6\}$	180° rotation $\{0,1\}$, $\{6,7\}$ midpoints
11	(02)(16)(34)(57)	$\{2,6,0,4,3,7,1,5\}$	180° rotation $\{0,2\}$, $\{5,7\}$ midpoints
12	(07)(13)(25)(46)	$\{7,3,5,1,6,2,4,0\}$	180° rotation $\{1,3\}$, $\{4,6\}$ midpoints
13	(07)(16)(23)(45)	$\{7,6,3,2,5,4,1,0\}$	180° rotation $\{2,3\}$, $\{4,5\}$ midpoints
14	(04)(16)(25)(37)	$\{4,6,5,7,0,2,1,3\}$	180° rotation $\{0,4\}$, $\{3,7\}$ midpoints
15	(07)(15)(26)(34)	$\{7,5,6,4,3,1,2,0\}$	180° rotation $\{1,5\}$, $\{2,6\}$ midpoints
16	(0)(7)(142)(356)	$\{0,4,1,5,2,6,3,7\}$	120° rotation body diagonal $\{0,7\}$
17	(0)(7)(124)(365)	$\{0,2,4,6,1,3,5,7\}$	120° rotation body diagonal $\{0,7\}$
18	(1)(6)(053)(247)	$\{5,1,4,0,7,3,6,2\}$	120° rotation body diagonal $\{1,6\}$
19	(1)(6)(035)(274)	$\{3,1,7,5,2,0,6,4\}$	120° rotation body diagonal $\{1,6\}$
20	(2)(5)(063)(147)	$\{6,4,2,0,7,5,3,1\}$	120° rotation body diagonal $\{2,5\}$
21	(2)(5)(036)(174)	$\{3,7,2,6,1,5,0,4\}$	120° rotation body diagonal $\{2,5\}$
22	(3)(4)(056)(172)	$\{5,7,1,3,4,6,0,2\}$	120° rotation body diagonal $\{3,4\}$
23	(3)(4)(065)(127)	$\{6,2,7,3,4,0,5,1\}$	120° rotation body diagonal $\{3,4\}$
24	(07)(16)(25)(34)	$\{7,6,5,4,3,2,1,0\}$	Inversion ((07)(16)(25)(34))
25	(0635)(1427)	$\{6,4,7,5,2,0,3,1\}$	90° rotation + Inversion
26	(0536)(1724)	$\{5,7,4,6,1,3,0,2\}$	
27	(0365)(1274)	$\{3,2,7,6,1,0,5,4\}$	
28	(0563)(1472)	$\{5,4,1,0,7,6,3,2\}$	
29	(0653)(1247)	$\{6,2,4,0,7,3,5,1\}$	
30	(0356)(1742)	$\{3,7,1,5,2,6,0,4\}$	
31	(04)(15)(26)(37)	$\{4,5,6,7,0,1,2,3\}$	180° rotation + Inversion
32	(01)(23)(45)(67)	$\{1,0,3,2,5,4,7,6\}$	
33	(02)(13)(46)(57)	$\{2,3,0,1,6,7,4,5\}$	
34	(06)(17)(2)(3)(4)(5)	$\{6,7,2,3,4,5,0,1\}$	180° rotation edge + Inversion
35	(05)(1)(27)(3)(4)(6)	$\{5,1,7,3,4,0,6,2\}$	
36	(0)(14)(2)(36)(5)(7)	$\{0,4,2,6,1,5,3,7\}$	
37	(0)(1)(24)(35)(6)(7)	$\{0,1,4,5,2,3,6,7\}$	
38	(03)(1)(2)(47)(5)(6)	$\{3,1,2,0,7,5,6,4\}$	
39	(0)(12)(3)(4)(56)(7)	$\{0,2,1,3,4,6,5,7\}$	
40	(07)(132645)	$\{7,3,6,2,5,1,4,0\}$	120° rotation body diagonal
41	(07)(154623)	$\{7,5,3,1,6,4,2,0\}$	+ Inversion
42	(023754)(16)	$\{2,6,3,7,0,4,1,5\}$	
43	(045732)(16)	$\{4,6,0,2,5,7,1,3\}$	
44	(013764)(25)	$\{1,3,5,7,0,2,4,6\}$	
45	(046731)(25)	$\{4,0,5,1,6,2,7,3\}$	
46	(026751)(34)	$\{2,0,6,4,3,1,7,5\}$	
47	(015762)(34)	$\{1,5,0,4,3,7,2,6\}$	

Table 2.6: 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			

3D base types

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 2.6. In Figure 2.9 the signs of the level set in each vertex for every base type are shown. In Figure 2.10 the corresponding cuts are shown, where for simplicity the interface cuts at the edges midpoints only. 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.

3D base type permutations

The cube permutations are applied to the thirteen types of cuts to find the cut types permutations. In Figure 2.11 an example is shown of a permutation of the type 0 cut. To calculate the cut type for an index in the range $128 - 255$ the index value only has to be subtracted from the number 255. In the Table 2.7 the cut types for indices $0 - 127$ are given. The number of permuted cases for every type are given in Table 2.8. In the implementation a lookup table is used of size 256 which stores the type index $(0 - 12)$ of the cut and a permutation index $(0 - 47)$ from that base type. The algorithm

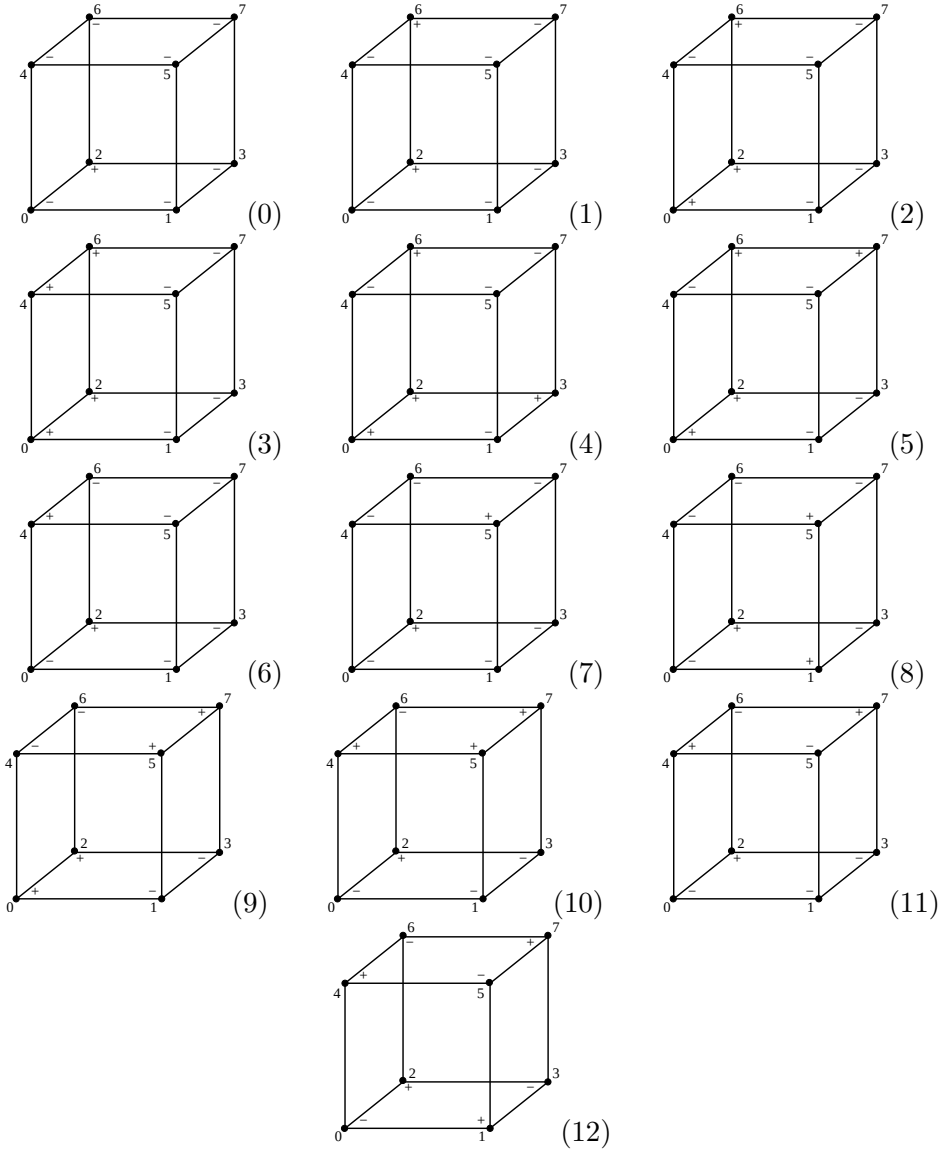

Figure 2.9: The vertex level set signs for the 3D base types.

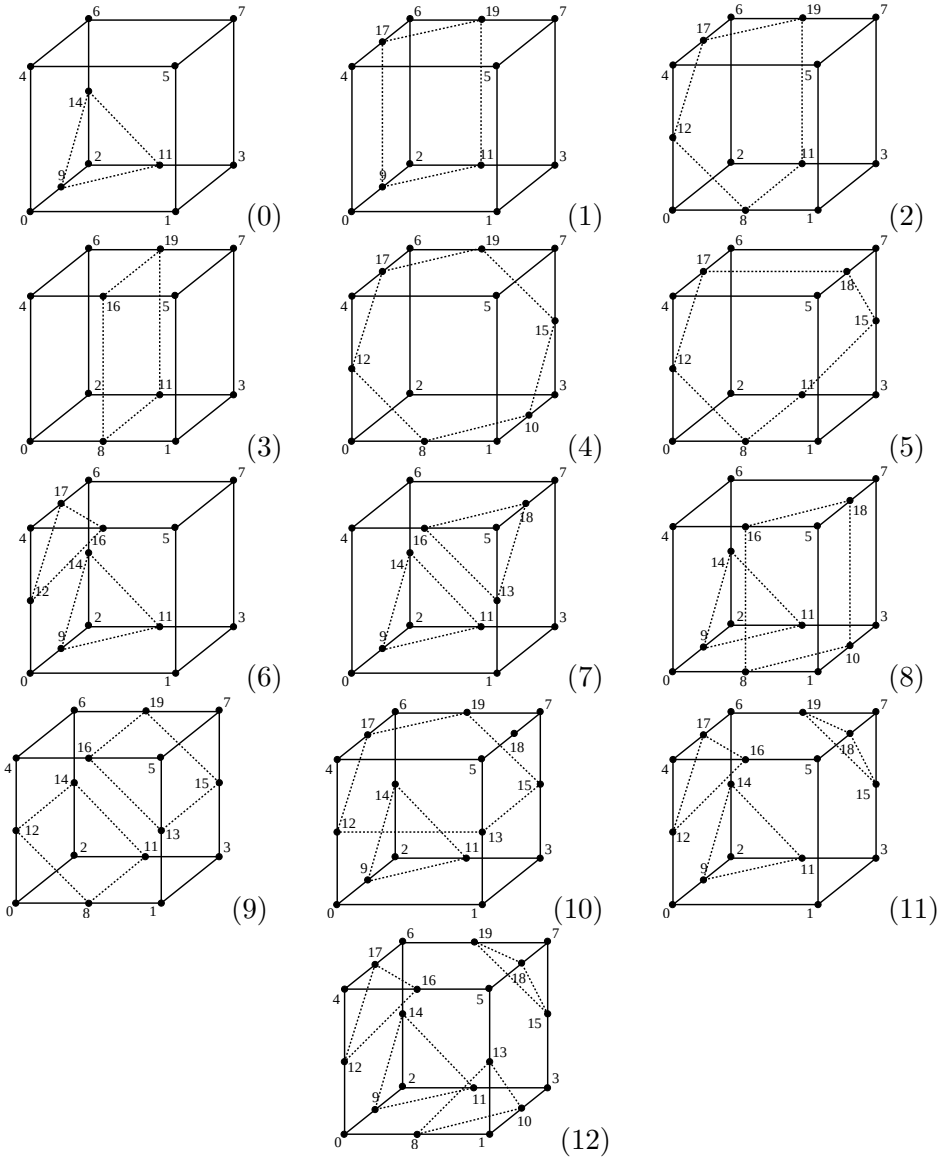

Figure 2.10: The interface cuts for the 3D base types. For types $6 - 12$ the level set configuration allows for alternative cuts not shown here, which are supported by the element refinement rule for that type.

Table 2.7: 3D base types corresponding to cut type indices $0-127$.

index	base type	index	base type	index	base type
0	No cut	43	Type 4	86	Type 10
1	Type 0	44	Type 8	87	Type 2
2	Type 0	45	Type 10	88	Type 8
3	Type 1	46	Type 5	89	Type 10
4	Type 0	47	Type 2	90	Type 9
5	Type 1	48	Type 1	91	Type 8
6	Type 6	49	Type 2	92	Type 5
7	Type 2	50	Type 2	93	Type 2
8	Type 0	51	Type 3	94	Type 8
9	Type 6	52	Type 8	95	Type 1
10	Type 1	53	Type 5	96	Type 6
11	Type 2	54	Type 10	97	Type 11
12	Type 1	55	Type 2	98	Type 8
13	Type 2	56	Type 8	99	Type 10
14	Type 2	57	Type 10	100	Type 8
15	Type 3	58	Type 5	101	Type 10
16	Type 0	59	Type 2	102	Type 9
17	Type 1	60	Type 9	103	Type 8
18	Type 6	61	Type 8	104	Type 11
19	Type 2	62	Type 8	105	Type 12
20	Type 6	63	Type 1	106	Type 10
21	Type 2	64	Type 0	107	Type 11
22	Type 11	65	Type 6	108	Type 10
23	Type 4	66	Type 7	109	Type 11
24	Type 7	67	Type 8	110	Type 8
25	Type 8	68	Type 1	111	Type 6
26	Type 8	69	Type 2	112	Type 2
27	Type 5	70	Type 8	113	Type 4
28	Type 8	71	Type 5	114	Type 5
29	Type 5	72	Type 6	115	Type 2
30	Type 10	73	Type 11	116	Type 5
31	Type 2	74	Type 8	117	Type 2
32	Type 0	75	Type 10	118	Type 8
33	Type 6	76	Type 2	119	Type 1
34	Type 1	77	Type 4	120	Type 10
35	Type 2	78	Type 5	121	Type 11
36	Type 7	79	Type 2	122	Type 8
37	Type 8	80	Type 1	123	Type 6
38	Type 8	81	Type 2	124	Type 8
39	Type 5	82	Type 8	125	Type 6
40	Type 6	83	Type 5	126	Type 7
41	Type 11	84	Type 2	127	Type 0
42	Type 2	85	Type 3		

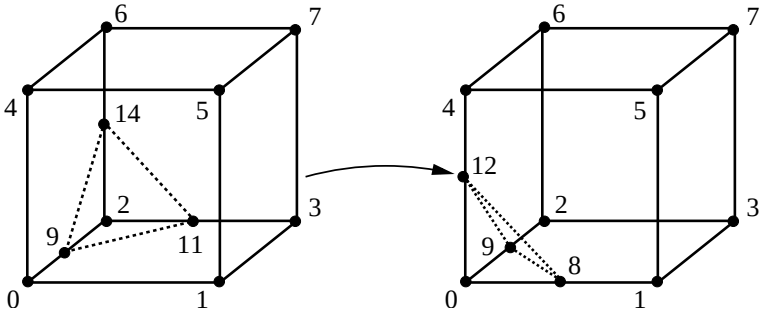

Figure 2.11: An example of a 3D permutation.

Table 2.8: Number of permutations for the 3D base types.

type	number of cases	type	number of cases
0	8	7	4
1	12	8	24
2	24	9	3
3	3	10	12
4	4	11	8
5	12	12	1
6	12		

used to fill the table is Algorithm 3, adapted to 3D.

3D base type refinement

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 Figure 2.7. The surface refinements are shown in Figure 2.12. Element refinements have been manually devised based on the surface refinements. The element refinements for the 13 base types are given in Tables 2.9 and 2.10. In some of the refinements an additional node is used, which is located at the interface center and has LNI 20. To determine the element

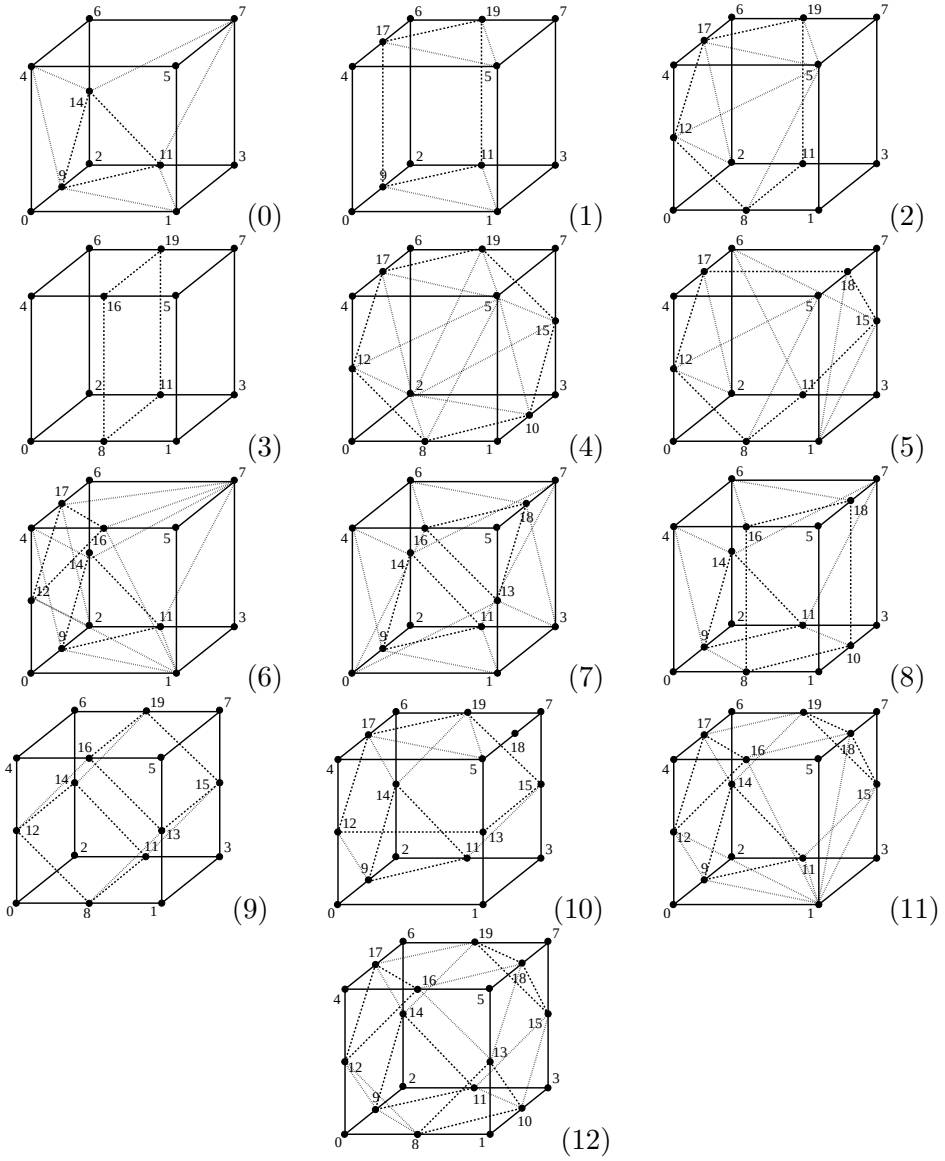

Figure 2.12: The surface refinements for the 3D base types.

Table 2.9: 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

Table 2.10: 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

refinements given a level set configuration on a reference cube Algorithm 4 is used, adapted to 3D.

2.3.6 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 $\mathcal{K}_k^{i,n}, k = 0, \cdots, N_{\hat{j}}$ 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:

$$\mathcal{K}_{m,\hat{j}}^{i,n} = \bigcup_{k=0}^{N_{\hat{j}}} \mathcal{K}_k^{i,n}. \tag{2.20}$$

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:

$$\begin{aligned} x_{\hat{j},l}^{min} &= \min_{\forall \mathbf{x} \in \mathcal{K}_{m,\hat{j}}^{i,n}} x_l, \ l = 0, \ldots, d \\ x_{\hat{j},l}^{max} &= \max_{\forall \mathbf{x} \in \mathcal{K}_{m,\hat{j}}^{i,n}} x_l, \ l = 0, \ldots, d, \end{aligned} \tag{2.21}$$

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

$$\begin{aligned} \epsilon_{\hat{j}}^{min} &= \min_{l=0,\cdots,d} \frac{x_{\hat{j},l}^{max} - x_{\hat{j},l}^{min}}{h_{b,l}} \\ \epsilon_{\hat{j}}^{max} &= \min_{l=0,\cdots,d} \frac{x_{\hat{j},l}^{max} - x_{\hat{j},l}^{min}}{h_{b,l}}. \end{aligned} \tag{2.22}$$

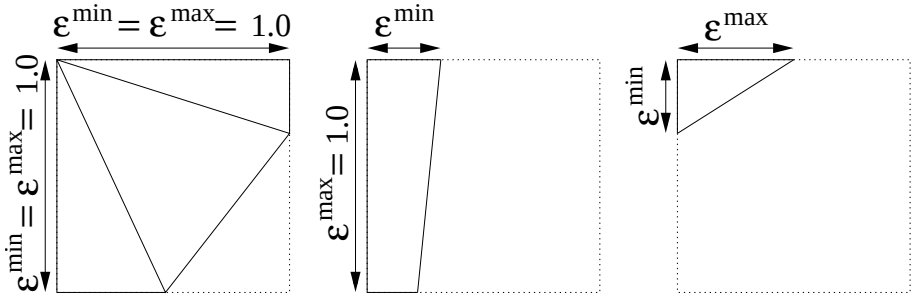

Figure 2.13: Illustration of the first step in the merging strategy. The dotted lines represent the background element and the solid lines represent the collection of child elements of one of the fluid types. The collection on the left has an $\epsilon^{min} > \epsilon^{MIN}$ and hence is considered 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.

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 $\mathcal{K}^n_{b,\tilde{j}}$ 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 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 Figure 2.13.

- 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 Figure 2.14.

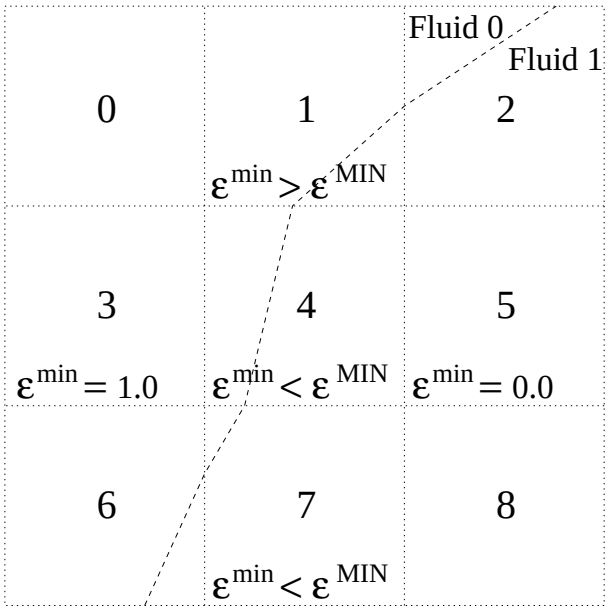

Figure 2.14: Illustration of the second step in the merging strategy for fluid type 0 and background element 4. The background elements are shown in dotted lines and the 0-level set is shown as a dashed line. Background element 4 has an $\epsilon^{min} < \epsilon^{MIN}$ and hence requires merging with 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.

- Step 3: The merged elements are determined in three steps. Each step corresponds to a different type of merging, and these are illustrated in Figure 2.15. 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 $\mathcal{K}^n_{b,k}, k = 0, \cdots, N_{\tilde{j}}$ 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, $\epsilon^{min}_{\tilde{k}} > \epsilon^{min}_k, k = 0, \cdots, N_{\tilde{j}}$. 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.

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 ([31]), which is simple shaped and contains the merged element. This merging procedure is illustrated for two dimensions in Figure 2.16 and an example of a mesh with merged elements in two dimensions is shown in Figure 2.17.

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

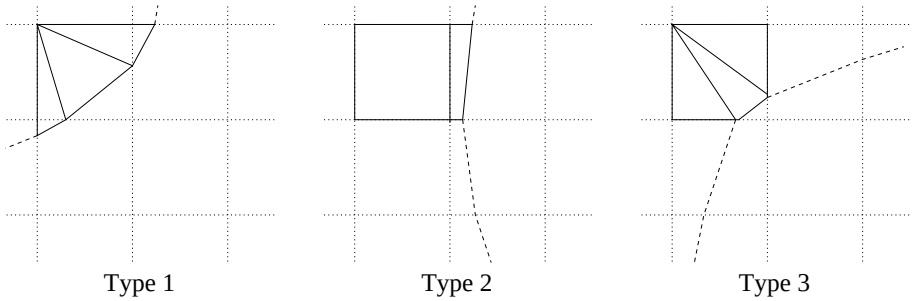

Figure 2.15: The three types of merged elements. The solid lines represent the refined elements that will be combined into a single merged element. The dotted lines represent the background mesh and the dashed lines represent the interface at positions not occupied by the merged element.

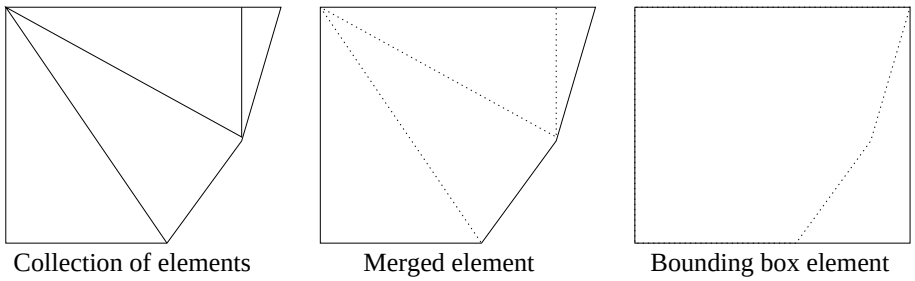

Figure 2.16: A collection of elements, their merged element and its bounding box element, in physical space.

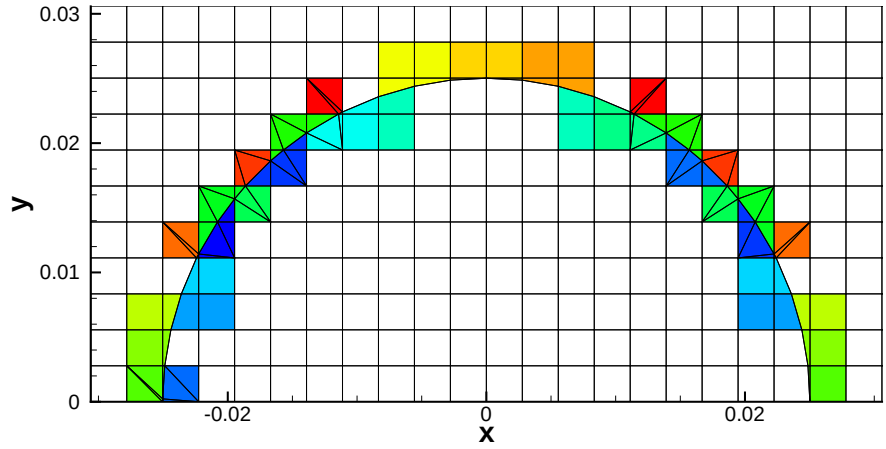

Figure 2.17: Refined mesh showing the merged elements as colored collections of child elements.

box elements are identical to those defined for the refined mesh. On the bounding box element the approximated flow variables are defined as:

$$\mathbf{w}_h^i(t,\bar{\mathbf{x}})|_{\mathcal{K}^{i,n}_{M,\hat{j}}} = \sum_m \hat{\mathbf{W}}_m^i(\mathcal{K}^{i,n}_{M,\hat{j}})\phi_m(t,\bar{\mathbf{x}}) \tag{2.23}$$

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

$$\mathbf{w}_h^i(\mathbf{x})|_{\mathcal{K}_k^{i,n}} = \mathbf{w}_h^i(\mathbf{x})|_{\mathcal{K}^{i,n}_{M,\hat{j}}}\,. \tag{2.24}$$

Integration of a function $f(\mathbf{w_h^i})$ over a merged element $\mathcal{K}^{i,n}_{m,\hat{j}}$ is performed by

integrating over all the individual elements and summing the contributions:

$$\int_{\mathcal{K}^{i,n}_{m,\hat{j}}} f(\mathbf{w}^i_h) d\mathcal{K} = \sum_{k=0}^{N_{\hat{j}}} \int_{\mathcal{K}^{i,n}_{k}} f(\mathbf{w}^i_h) d\mathcal{K}. \tag{2.25}$$

2.4 Space-time discontinuous Galerkin discretization

2.4.1 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 (2.2) 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 $\mathcal{E}^2$:

$$\sum_{\mathcal{K}^{i,n}_j \in \mathcal{T}^{i,n}_h} \int_{\mathcal{K}^{i,n}_j} \mathbf{v} \nabla \cdot \mathcal{F}^i(\mathbf{w}^i)\, d\mathcal{K} = 0. \tag{2.26}$$

Applying Gauss' theorem results in:

$$\begin{aligned}
&- \sum_{\mathcal{K}^{i,n}_j \in \mathcal{T}^{i,n}_h} \int_{\mathcal{K}^{i,n}_j} \nabla \mathbf{v} \cdot \mathcal{F}^i(\mathbf{w}^i)\, d\mathcal{K} \\
&+ \sum_{\mathcal{S}^{i,n}_m \in \Gamma^{i,n}_I} \int_{\mathcal{S}^{i,n}_m} \mathcal{F}^{i,l}(\mathbf{w}^{i,l}) \cdot \mathbf{n}^l_{\mathcal{K}}\, \mathbf{v}^l + \mathcal{F}^{i,r}(\mathbf{w}^{i,r}) \cdot \mathbf{n}^r_{\mathcal{K}}\, \mathbf{v}^r\, d\mathcal{S} \\
&+ \sum_{\mathcal{S}^{i,n}_m \in \Gamma^{i,n}_B} \int_{\mathcal{S}^{i,n}_m} \mathcal{F}^{i,l}(\mathbf{w}^{i,l}) \cdot \mathbf{n}^l_{\mathcal{K}}\, \mathbf{v}^l\, d\mathcal{S} \\
&+ \sum_{\mathcal{S}^{i,n}_m \in \Gamma^{i,n}_S} \int_{\mathcal{S}^{i,n}_m} \mathcal{F}^{i,l}(\mathbf{w}^{i,l}) \cdot \mathbf{n}^l_{\mathcal{K}}\, \mathbf{v}^l\, d\mathcal{S} = 0,
\end{aligned} \tag{2.27}$$

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

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:

$$\begin{aligned} &\sum_{\mathcal{S}_m^{i,n} \in \Gamma_I^{i,n}} \int_{\mathcal{S}_m^{i,n}} \mathcal{F}^{i,l}(\mathbf{w}^{i,l}) \cdot \mathbf{n}_{\mathcal{K}}^l \, \mathbf{v}^l + \mathcal{F}^{i,r}(\mathbf{w}^{i,r}) \cdot \mathbf{n}_{\mathcal{K}}^r \, \mathbf{v}^r \, d\mathcal{S} \\ &= \sum_{\mathcal{S}_m^{i,n} \in \Gamma_I^{i,n}} \int_{\mathcal{S}_m^{i,n}} \{\!\{\mathcal{F}^i(\mathbf{w}^i)\}\!\} \cdot [\![\mathbf{v}]\!] \, d\mathcal{S}. \end{aligned} \tag{2.28}$$

So far the formulation 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 is replaced by a numerical flux $\mathcal{H}_I^i(\mathbf{w}^l, \mathbf{w}^r, \mathbf{n}_{\mathcal{K}})$, 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 $\mathcal{H}_B^i(\mathbf{w}^l, \mathbf{w}^r, \mathbf{n}_{\mathcal{K}})$, which is also consistent. At the interface the flux is replaced by a numerical interface flux $\mathcal{H}_S^i(\mathbf{w}^{i,l}, \mathbf{w}_s^i, \mathbf{n}_{\mathcal{K}})$, with $\mathbf{w}_s^i$ 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 $\mathbf{v}_h \in B_h^k(\mathcal{T}_h^{i,n})$:

$$
\begin{aligned}
&- \sum_{\mathcal{K}_j^{i,n} \in \mathcal{T}_h^{i,n}} \int_{\mathcal{K}_j^{i,n}} \nabla \mathbf{v}_h \cdot \mathcal{F}^i(\mathbf{w}_h^i)\, d\mathcal{K} \\
&+ \sum_{\mathcal{S}_m^{i,n} \in \Gamma_I^{i,n}} \int_{\mathcal{S}_m^{i,n}} \mathcal{H}_I^i(\mathbf{w}_h^{i,l}, \mathbf{w}_h^{i,r}, \mathbf{n}_\mathcal{K})\, (\mathbf{v}_h^l - \mathbf{v}_h^r)\, d\mathcal{S} \\
&+ \sum_{\mathcal{S}_m^{i,n} \in \Gamma_B^{i,n}} \int_{\mathcal{S}_m^{i,n}} \mathcal{H}_B^i(\mathbf{w}_h^{i,l}, \mathbf{w}_b^i, \mathbf{n}_\mathcal{K})\, \mathbf{v}_h^l\, d\mathcal{S} \\
&+ \sum_{\mathcal{S}_m^{i,n} \in \Gamma_S^{i,n}} \int_{\mathcal{S}_m^{i,n}} \mathcal{H}_S^i(\mathbf{w}_h^i, \mathbf{w}_s^i, \mathbf{n}_\mathcal{K})\, \mathbf{v}_h^l\, d\mathcal{S} = 0, \\
&\qquad\qquad i = 1, 2,\ n = 0, \cdots, N_t - 1.
\end{aligned}
\tag{2.29}
$$

Introduction of the polynomial expansion (2.15) in (2.29) and using the basis functions ϕ_l for the test functions gives the following discretization in each space-time element $\mathcal{K}_j^{i,n}$:

$$
\begin{aligned}
&\mathcal{L}_{kl}^{i,n}(\hat{\mathbf{W}}^n, \hat{\mathbf{W}}^{n-1}) = 0,\ i = 1, 2,\ n = 0, \cdots, N_t - 1, \\
&\qquad\qquad k = 0, \cdots, N_w - 1,\ l = 0, \ldots, N_{B,j}^{i,n} - 1
\end{aligned}
\tag{2.30}
$$

with N_t the number of time slabs, N_w the number of flow variables, $N_{B,j}^{i,n}$

the number of basis functions. The nonlinear operator $\mathcal{L}_{kl}^{i,n}$ is defined as:

$$\begin{aligned}\mathcal{L}_{kl}^{i,n} = &- \int_{\mathcal{K}_j^{i,n}} (\nabla \phi_l)_j \cdot \mathcal{F}_{kj}^i(\mathbf{w}_h^i) d\mathcal{K} \\ &+ \sum_{\mathcal{S}_m^{i,n} \in \partial\mathcal{K}_j^{i,n} \cap \Gamma_I^i} \int_{\mathcal{S}_m^{i,n}} \mathcal{H}_{I,k}(\mathbf{w}_h^{i,-}, \mathbf{w}_h^{i,+}, \mathbf{n}_\mathcal{K})\, \phi_l \, d\mathcal{S} \\ &+ \sum_{\mathcal{S}_m^{i,n} \in \partial\mathcal{K}_j^{i,n} \cap \Gamma_B^i} \int_{\mathcal{S}_m^{i,n}} \mathcal{H}_{B,k}(\mathbf{w}_h^{i,-}, \mathbf{w}_b^{i,+}, \mathbf{n}_\mathcal{K})\, \phi_l \, d\mathcal{S} \\ &+ \sum_{\mathcal{S}_m^{i,n} \in \partial\mathcal{K}_j^{i,n} \cap \Gamma_S^i} \int_{\mathcal{S}_m^{i,n}} \mathcal{H}_{S,k}(\mathbf{w}_h^{i,-}, \mathbf{w}_s^{i,+}, \mathbf{n}_\mathcal{K})\, \phi_l \, d\mathcal{S}. \end{aligned} \tag{2.31}$$

In equation (2.30) the dependency of $\mathcal{L}_{kl}^{i,n}$ on $\hat{\mathbf{W}}^{n-1}$ 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 in Chapter 3 for specific test problems.

2.4.2 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 [71] is applied. For simplicity the same notation will be used as in [71].

In general, a hyperbolic system of m partial differential equations in

nonconservative form in q space dimensions can be defined as:

$$U_{i,0} + F_{ik,k} + G_{ikr}U_{r,k} = 0,\ i,r = 1,\cdots,m \tag{2.32}$$

with U the vector of variables, F the conservative spatial flux tensor, G the nonconservative spatial flux tensor and where $(.)_{,0}$ and $(.)_{,k}, k = 1,\cdots,q$ denote partial differentiation with respect to time and spatial coordinates, respectively. The space-time DGFEM weak nonconservative formulation of this system is defined as:

$$\begin{aligned}
0 = &\sum_{\mathcal{K}\in\mathcal{T}_h^n}\int_{\mathcal{K}}(-V_{i,0}U_i - V_{i,k}F_{ik} + V_iG_{ikr}U_{r,k})d\mathcal{K}\\
&+\sum_{\mathcal{K}\in\mathcal{T}_h^n}(\int_{K(t_{n+1}^-)}V_i^LU_i^LdK - \int_{K(t_n^+)}V_i^LU_i^RdK)\\
&+\sum_{\mathcal{S}\in\mathcal{S}^n}\int_{\mathcal{S}}(V_i^L - V_i^R)\hat{P}_i^{nc}d\mathcal{S}\\
&+\sum_{\mathcal{S}\in\mathcal{S}^n}\int_{\mathcal{S}}\{\!\{V_i\}\!\}(\int_0^1 G_{ikr}(\chi(\tau;U^L,U^R))\frac{d\chi_r}{d\tau}(\tau;U^L,U^R)d\tau\bar{n}_k^L)d\mathcal{S},
\end{aligned} \tag{2.33}$$

where V denotes the vector of trial functions and χ denotes the path function. The nonconservative flux is defined as:

$$\hat{P}^{nc}(U_L,U_R,v,\bar{n}^L) = \tag{2.34}$$
$$\begin{cases}
F_{ik}^L - \frac{1}{2}\int_0^1 G_{ikr}(\chi(\tau;U^L,U^R))\frac{d\chi_r}{d\tau}(\tau;U^L,U^R)d\tau\bar{n}_k^L, & \text{if } S_L > v\\
\{\!\{F_{ik}\}\!\}\bar{n}_k^L + \frac{1}{2}((S_R-v)\bar{U}_i^* + (S_L-v)\bar{U}_i^* - S_LU_i^L - S_RU_i^R), & \\
& \text{if } S_L < v < S_R\\
F_{ik}^L + \frac{1}{2}\int_0^1 G_{ikr}(\chi(\tau;U^L,U^R))\frac{d\chi_r}{d\tau}(\tau;U^L,U^R)d\tau\bar{n}_k^L, & \text{if } S_R < v,
\end{cases}$$

where S_L and S_R denote the minimum and maximum wavespeeds, v denotes the grid velocity and $\bar{U}_i^*$ denotes the average star state solution. When using a linear path $\chi = U_L + \tau(U_R - U_L)$, $\frac{d\chi_r}{d\tau}(\tau;U^L,U^R) = (U^R - U^L)$.

The level set equation can be considered a special case of (2.32), where the state and fluxes are defined as $U = \psi_h, F = 0, G = \bar{\mathbf{a}}_h$. The following

simplification can be made:

$$\begin{aligned}
&\int_0^1 G_{ikr}(\chi(\tau;U^L,U^R))\frac{d\chi_r}{d\tau}(\tau;U^L,U^R)d\tau\bar{n}_k^L \\
&= \int_0^1 \bar{\mathbf{a}}(\chi(\tau;\psi_h^L,\psi_h^R))(\psi_h^R-\psi_h^L)d\tau\bar{n}_k^L \\
&= -\{\!\{\bar{\mathbf{a}}_h\}\!\}[\![\psi_h]\!]. \qquad (2.35)
\end{aligned}$$

Hence, the nonconservative level set discretization becomes:

$$\begin{aligned}
&\sum_{\mathcal{K}_{b,\tilde{j}}^n\in\mathcal{T}_b^n}\int_{\mathcal{K}_{b,\tilde{j}}^n} -\frac{\partial\phi_l}{\partial t}\psi_h + \phi_l\,\bar{\mathbf{a}}_h\cdot\bar{\nabla}\psi_h\,d\mathcal{K} \\
&+\sum_{\mathcal{K}_{b,\tilde{j}}^n\in\mathcal{T}_b^n}\left(\int_{\mathcal{K}_{b,\tilde{j}}^n(t_{n+1})}\phi_l^l\,\psi_h^l\,d\mathcal{S}-\int_{\mathcal{K}_{b,\tilde{j}}^n(t_n)}\phi_l^l\,\psi_h^r\,d\mathcal{S}\right) \\
&+\sum_{\mathcal{S}_{b,\tilde{m}}^n\in\Gamma_b^n}\int_{\mathcal{S}_{b,\tilde{m}}^n}(\phi_l^l-\phi_l^r)\,\hat{P}^{nc}\,d\mathcal{S} \\
&-\sum_{\mathcal{S}_{b,\tilde{m}}^n\in\Gamma_b^n}\int_{\mathcal{S}_{b,\tilde{m}}^n}\{\!\{\phi_l\}\!\}\,[\![\psi_h]\!]\,\{\!\{\bar{\mathbf{a}}_h\}\!\}\,d\mathcal{S}=0, \qquad (2.36)
\end{aligned}$$

with

$$\hat{P}^{nc}=\begin{cases}+\frac{1}{2}[\![\psi_h]\!]\,\{\!\{\bar{\mathbf{a}}_h\}\!\} & \text{if } S_L>0\\ +\frac{1}{2}(S_R(\psi_h^*-\psi_h^R)+S_L(\psi_h^*-\psi_h^L)) & \text{if } S_L<0<S_R\\ -\frac{1}{2}[\![\psi_h]\!]\,\{\!\{\bar{\mathbf{a}}_h\}\!\} & \text{if } S_R<0\end{cases} \qquad (2.37)$$

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

$$\psi_h^*=\begin{cases}\psi_L & \text{if } (S_L+S_R)/2>0\\ \psi_R & \text{if } (S_L+S_R)/2<0.\end{cases} \qquad (2.38)$$

Algorithm 5 Pseudo-time integration method for solving the non-linear algebraic equations in the space-time discretization.

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

At boundary faces the level set boundary conditions (2.10) are enforced by specifying the right state as:

$$\begin{aligned} \psi^r(t, \bar{\mathbf{x}}) &= \psi^l(t, \bar{\mathbf{x}}) \\ \bar{\mathbf{a}}^r(t, \bar{\mathbf{x}}) &= \bar{\mathbf{a}}^l(t, \bar{\mathbf{x}}) - 2(\bar{\mathbf{a}}^l(t, \bar{\mathbf{x}}) \cdot \mathbf{n}_{\mathcal{K}})\mathbf{n}_{\mathcal{K}}, \text{ for } (t, \bar{\mathbf{x}}) \in \mathcal{Q}. \end{aligned} \tag{2.39}$$

2.4.3 Pseudo-time integration

By augmenting the flow equations with a pseudo-time derivative, the discretized equations (2.30) are extended into pseudo-time, resulting in:

$$M^{i,n}_{ml} \frac{\partial \hat{W}^{i,n}_{km}}{\partial \tau} + \mathcal{L}^{i,n}_{kl}(\hat{\mathbf{W}}^n, \hat{\mathbf{W}}^{n-1}) = 0, \tag{2.40}$$

using the summation convention on repeated indices, and with

$$M^{i,n}_{ml} = \int_{\mathcal{K}^{i,n}_j} \phi_l \phi_m \, d\mathcal{K} \tag{2.41}$$

the mass matrix. To solve (2.40) a five stage semi-implicit Runge-Kutta iterative scheme is used [50, 98] as defined in Algorithm 5. 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. 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:

$$\Delta t = \frac{CFL_{\Delta t}\, h}{S_{max}}, \tag{2.42}$$

with $CFL_{\Delta t}$ the physical CFL number, h the inradius of the space projection of the element and S_{max} the maximum 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.

2.5 Two-fluid algorithm

The two-fluid algorithm is defined in Algorithm 6. The operations at the initialization, in the inner iteration and at the time slab update are illustrated for two space-time dimensions in Figures 2.18, 2.19 and 2.20, respectively.

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:

$$\mathbf{w}_h^{i,n}(t, \bar{\mathbf{x}}) = \bar{\mathbf{w}}_h^{i,n-1}(t_n, \bar{\mathbf{x}}). \tag{2.43}$$

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:

$$\int_{\mathcal{K}_{b,\tilde{j}}^n} \bar{\mathbf{a}}_h^n(\mathbf{x})\phi_l(\mathbf{x})\, d\mathcal{K} = \int_{\mathcal{K}_{b,\tilde{j}}^n} \mathbf{u}_{h,k}^n(\mathbf{x})\phi_l(\mathbf{x})\, d\mathcal{K}. \tag{2.44}$$

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 $H_{\mathcal{K}_j^{i,n}}$

Algorithm 6 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 Figures 2.18, 2.19 and 2.20.

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* $\mathcal{T}_b^{n-1}$ *(2.46)*
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* $\mathcal{T}_b^{n-1}$ *(2.47)*
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* $\mathcal{T}_h^{i,n-1}$ *(2.43)*
13. $k = 0$
14. *WHILE two-fluid mesh has not converged:* $|e_k - e_{k-1}| > \epsilon_{IF}$ *DO*
15. *Solve* ψ_h^n *on* $\mathcal{T}_b^n$
16. *Calculate level set interface error* $e_k = \|\psi_h^n\|_2^{IF}$
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* $\mathcal{T}_h^{i,n-1}$ *(2.43)*
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* $\mathcal{T}_h^{i,n}$ *(2.44)*
21. $k = k + 1$
22. *END DO*
23. $n = n + 1$
24. *END DO*

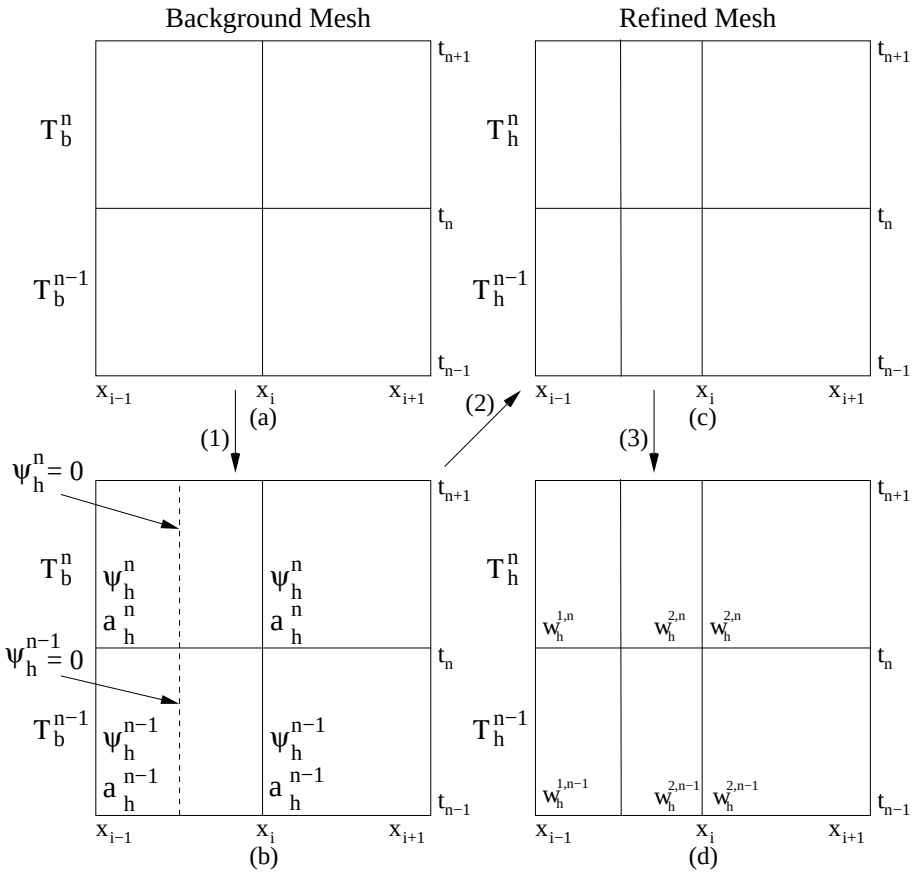

Figure 2.18: At initialization, first the background mesh is created. Because the solution from the previous time step is required in the evaluation of the numerical flux at the time slab face, the background mesh is conveniently composed of a current (n) and a previous ($n-1$) time slab (a). Next the 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.

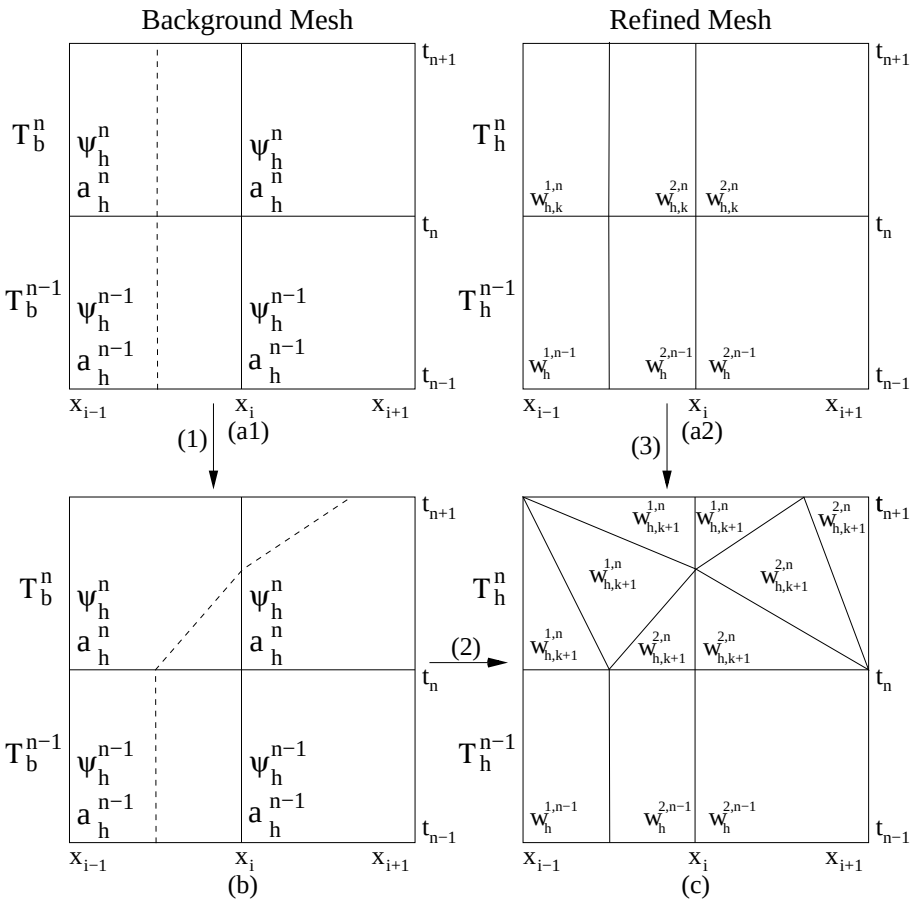

Figure 2.19: In the inner iteration, given level set and flow solutions on the background and refined meshes (a1, a2), first the level set is solved on $\mathcal{T}_b^n$ (b). Based on the 0-level set the background mesh is refined to obtain a new two-fluid mesh $\mathcal{T}_h^n$, on which the flow field is reinitialized and solved (c). Finally, the level set velocity is reinitialized with the flow velocity.

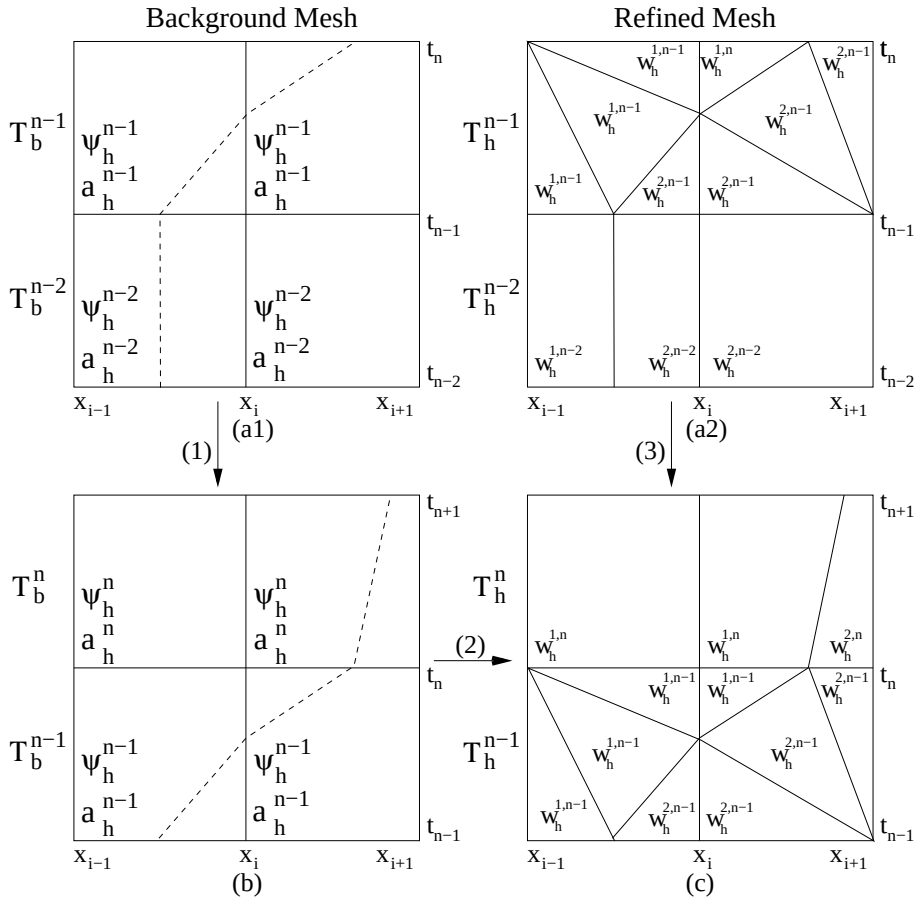

Figure 2.20: When moving to the next time slab, given level set and flow solutions on the background and refined meshes (a1, a2), first a new background mesh $\mathcal{T}_b^n$ is created, on which a level set is initialized and solved (b). Based on the 0-level set, the background mesh is refined to obtain the two-fluid mesh $\mathcal{T}_h^n$, on which the flow field is initialized (c).

is defined:

$$H_{\mathcal{K}_j^{i,n}} = G^{-1}_{\mathcal{K}_j^{i,n}} \circ G_{\mathcal{K}^n_{b,\tilde{j}}}, \tag{2.45}$$

where $G_{\mathcal{K}^n_{b,\tilde{j}}}$ 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 $\mathcal{K}^n_{b,\tilde{j}}$ in the background mesh $\mathcal{T}_b^n$. The inverse mappings $G^{-1}_{\mathcal{K}^n_{b,\tilde{j}}}$ always exists, since the background elements are by construction never degenerate. The child to parent mapping is illustrated in Figure 2.21. At the time slab update the level set approximation ψ_h^n is reinitialized as:

$$\psi_h^n(t, \bar{\mathbf{x}}) = \psi_h^{n-1}(t_n, \bar{\mathbf{x}}) \tag{2.46}$$

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

$$\bar{\mathbf{a}}_h^n(t, \bar{\mathbf{x}}) = \bar{\mathbf{a}}_h^{n-1}(t_n, \bar{\mathbf{x}}). \tag{2.47}$$

2.6 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 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. The problem with cut-cell mesh refinement with small cells is solved by using element merging. Topological changes such as merging and coalescence are handled in the method due to the level set method. Care must, however, be taken since topological changes may conflict with the conservativity of the

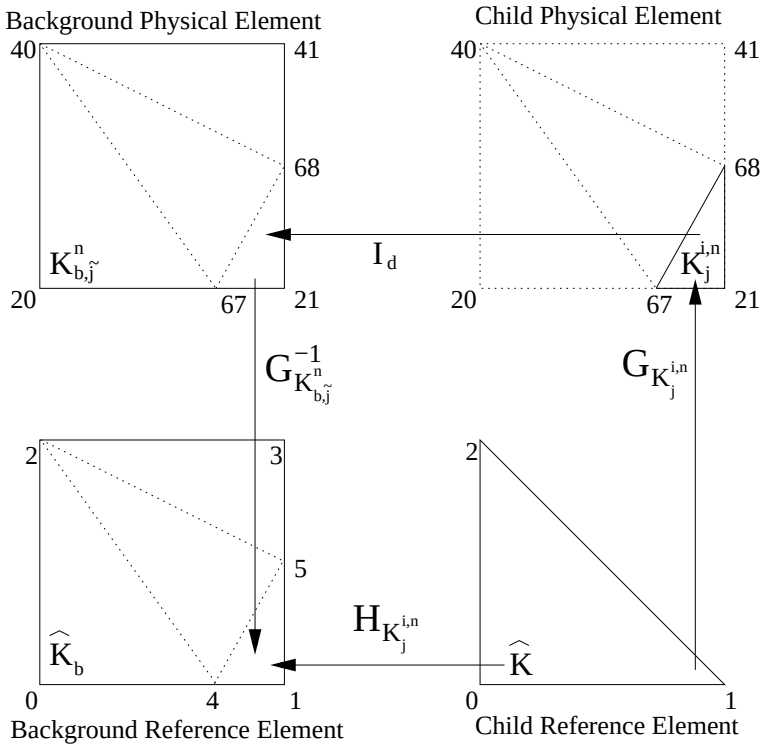

Figure 2.21: The child to parent mapping $H_{\mathcal{K}_j^{i,n}}$ is composed of the mapping $G_{\mathcal{K}_j^{i,n}}$ from the child reference element to child physical element and the inverse mapping $G^{-1}_{\mathcal{K}^n_{b,\tilde{j}}}$ from background physical element to the background reference element. The child physical element is connected to the background physical element through the identity mapping I_d.

scheme especially on coarse meshes and may cause non-convergence of the flow solution. In Chapter 3 the method will be tested on a number of test problems.

Chapter 3

Two-Fluid Space-Time Discontinuous Galerkin Finite Element Method. Part II: Applications

3.1 Introduction

In Chapter 2 a space-time discontinuous Galerkin (STDG) finite element method for two-fluid flows was presented. This 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. For the interface handling a front tracking approach is used because front tracking methods are capable of high accuracy. The front tracking is implemented using cut-cell mesh refinement because this type of refinement is very local in nature and hence combines well with the STGD. To compute the interface dynamics the level set method (LSM) is used, because of its ability to deal with merging and breakup, because it was expected that the LSM combines well with the cut-cell mesh refinement and also because the LSM is easy to extend to

higher dimensions. The small cell problem caused by the cut-cell refinement was solved by using a merging procedure involving bounding box elements, which improves stability and performance of the method. The interface conditions can be 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.

In this chapter the method is applied to a number of model problems in two and three space-time dimensions which range from one dimensional linear advection tests to complex two-fluid problems including a magma - ideal gas shock tube test and a shock wave - helium cylinder interaction test. The interface is assumed to be clean and without surface tension and therefore continuity of the normal velocity and pressure is imposed [29, 78]. The simulations have been performed using three dimensional space-time codes based on the *hp*GEM software framework for Discontinuous Galerkin finite element methods [65].

The outline of this chapter is as follows. First, in Section 3.2 the two-fluid flow error measurement is explained. In Section 3.3 the HWENO slope limiter is introduced. In Sections 3.4-3.9 the test results are discussed. Finally, in Section 3.10 a discussion and conclusions are presented.

3.2 Error measurement

Let $w_h^i(t_{n+1}, \mathbf{x})$ denotes the approximate flow solution, $w^i(t_{n+1}, \mathbf{x})$ the exact flow solution and $\Omega_h^i(t_{n+1})$ 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:

$$\|w_h^i(t_{n+1}, \cdot) - w^i(t_{n+1}, \cdot)\|_{L_2(\Omega_h^i(t_{n+1}))} = \left(\int_{\Omega_h^i(t_{n+1})} |w_h^i(t_{n+1},\mathbf{x}) - w^i(t_{n+1}, \mathbf{x})|^2 d\mathbf{x} \right)^{1/2}. \quad (3.1)$$

The order of accuracy with respect to the norm $\|.\|$ 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_h^n and $\Omega_{h/2}^n$, 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.

The STDG method has order of accuracy $O(h^{p+1})$ for smooth solutions and order of accuracy $O(h^{1/2})$ for discontinuous solutions ([53, 83]). Front capturing and tracking techniques can help to improve the accuracy of the STDG method around discontinuities.

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.

3.3 Slope limiter

Around strong discontinuities which are not captured or tracked DG solutions show spurious oscillations. To control these oscillations the Hermite WENO slope limiter introduced in [56] is used. The limiter is applied after every physical time step to the spatial solution at the most recent time level. Since a space-time mesh is used, this means the limiter is applied at time slab faces. Let $\mathcal{S}_e^{n+1}$ denote a time slab face on which the solution requires limiting. The solution after slope limiting is defined as the weighted sum of a number of reconstructed polynomials $P_i(u_h), i = 1, \cdots, N_P$:

$$\tilde{u}_h = \sum_{i=1}^{N_P} w_i P_i(u_h), \tag{3.2}$$

where u_h denotes the numerical solution on $\mathcal{S}_e^{n+1}$, $\tilde{u}_h$ the limited solution on $\mathcal{S}_e^{n+1}$ and N_P the number of reconstructed polynomials. The weights are defined as:

$$w_i = \frac{(\epsilon + o_i(P_i))^{-\gamma}}{\sum_{k=0}^{N_P-1}(\epsilon + o_k(P_k))^{-\gamma}}, \tag{3.3}$$

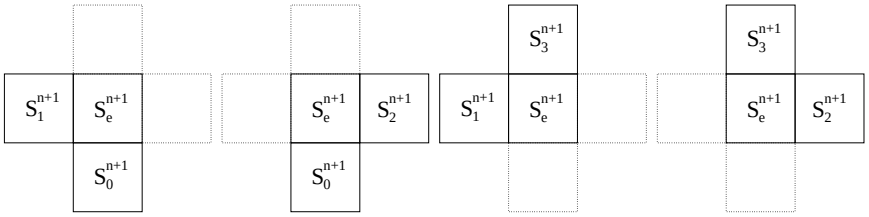

Figure 3.1: Lagrange stencils for quadrilateral shaped time slab faces.

with $\epsilon > 0$ and $\gamma > 0$ constants. Here $o_i(P_i)$ denotes the oscillation indicator for element $\mathcal{K}_e$:

$$o_i(P_i) = \|\bar{\nabla} P_i\|_{L_2(\mathcal{S}_i^{n+1})}, \tag{3.4}$$

with $\|.\|_{L_2(\mathcal{S}_i^{n+1})}$ the L_2 norm at time slab face $\mathcal{S}_i^{n+1}$ and $\bar{\nabla}$ the space gradient operator.

Each polynomial $P_i(u_h)$ is constructed from the numerical solution u_h on a specific stencil S_i^{n+1} of time slab faces, where the shape of the stencil depends on the type of reconstruction polynomial and the shape of the face $\mathcal{S}_e^{n+1}$. Three types of reconstruction polynomials are considered: Lagrange, Hermite, and, the linear projection of the original polynomial. The corresponding stencils are shown for quadrilateral shaped faces in Figures 3.1, 3.2 and 3.3, respectively. For the Lagrange polynomials, the stencil is composed of the face $\mathcal{S}_e^{n+1}$ and also d of the N_n neighboring faces $\mathcal{S}_{l_{i,j}}^{n+1}, j = 0, \cdots, d-1$, where d denotes the space dimension and $l_{i,j} \in \{0, \cdots, N_n\}$. For the Hermite polynomials, every stencil is composed of just two time slab faces, namely the face $\mathcal{S}_e^{n+1}$ and one neighboring face $\mathcal{S}_h^{n+1}$ where $h \in \{0, \cdots, N_n\}$. For the linear projection the stencil is composed only of the face $\mathcal{S}_e^{n+1}$.

Each Lagrange polynomial $P_{L,i}$ is constructed using only the solution averages for the time slab faces of the Lagrange stencil. Let x_e, x_a and x_b denote the face midpoints in physical coordinates for the time slab faces $\mathcal{S}_e^{n+1}, \mathcal{S}_{l_{i,0}}^{n+1}$ and $\mathcal{S}_{l_{i,1}}^{n+1}$. The reconstructed Lagrange polynomial $P_{L,i}$ is de-

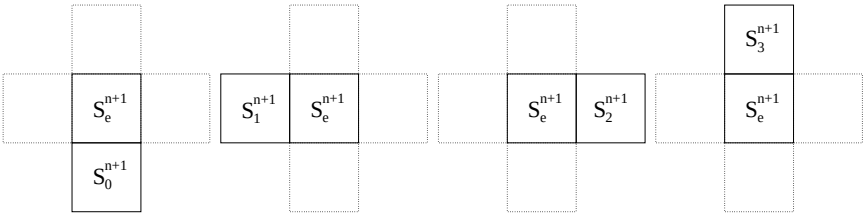

Figure 3.2: Hermite stencils for quadrilateral shaped time slab faces.

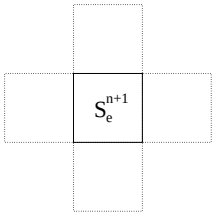

Figure 3.3: Stencil used for the restriction of the original polynomial for quadrilateral shaped time slab faces.

fined as:

$$\begin{aligned}
\frac{1}{|\mathcal{S}_e^{n+1}|}\int_{\mathcal{S}_e^{n+1}} P_{L,i}\,dK &= \frac{1}{|\mathcal{S}_e^{n+1}|}\int_{\mathcal{S}_e^{n+1}} u_h\,dK \\
\frac{1}{|\mathcal{S}_{l_{i,0}}^{n+1}|}\int_{\mathcal{S}_{l_{i,0}}^{n+1}} P_{L,i}\,dK &= \frac{1}{|\mathcal{S}_{l_{i,0}}^{n+1}|}\int_{\mathcal{S}_{l_{i,0}}^{n+1}} u_h\,dK \\
\frac{1}{|\mathcal{S}_{l_{i,1}}^{n+1}|}\int_{\mathcal{S}_{l_{i,1}}^{n+1}} P_{L,i}\,dK &= \frac{1}{|\mathcal{S}_{l_{i,1}}^{n+1}|}\int_{\mathcal{S}_{l_{i,1}}^{n+1}} u_h\,dK
\end{aligned} \tag{3.5}$$

with $i = 0, \cdots, 3$ and $l_i \in \{(0,1),(0,2),(1,3),(2,3)\}$ for quadrilateral shaped time slab faces. Each Hermite reconstruction polynomial $P_{H,i}$ is constructed from the solution average in the time slab face $\mathcal{S}_e^{n+1}$ and the average solution gradient from one neighbor time slab face and is defined

as:

$$\begin{aligned}
\frac{1}{|\mathcal{S}_e^{n+1}|}\int_{\mathcal{S}_e^{n+1}} P_{H,i}\,dK &= \frac{1}{|\mathcal{S}_e^{n+1}|}\int_{\mathcal{S}_e^{n+1}} u_h\,dK \\
\frac{1}{|\mathcal{S}_e^{n+1}|}\int_{\mathcal{S}_e^{n+1}} \frac{\partial P_{H,i}}{\partial x}\,dK &= \frac{1}{|\mathcal{S}_{h_i}^{n+1}|}\int_{\mathcal{S}_{h_i}^{n+1}} \frac{\partial u_h}{\partial x}\,dK \\
\frac{1}{|\mathcal{S}_e^{n+1}|}\int_{\mathcal{S}_e^{n+1}} \frac{\partial P_{H,i}}{\partial y}\,dK &= \frac{1}{|\mathcal{S}_{h_i}^{n+1}|}\int_{\mathcal{S}_{h_i}^{n+1}} \frac{\partial u_h}{\partial y}\,dK
\end{aligned} \tag{3.6}$$

with $h_i = i$ and $i = 0, \cdots, 3$ for quadrilateral shaped time slab faces. The linear projection of the original polynomial P_O is treated as a Hermite reconstruction polynomial, with $\mathcal{S}_{h_i}^{n+1} = \mathcal{S}_e^{n+1}$and is defined as:

$$\begin{aligned}
\frac{1}{|\mathcal{S}_e^{n+1}|}\int_{\mathcal{S}_e^{n+1}} P_O\,dK &= \frac{1}{|\mathcal{S}_e^{n+1}|}\int_{\mathcal{S}_e^{n+1}} u_h\,dK \\
\frac{1}{|\mathcal{S}_e^{n+1}|}\int_{\mathcal{S}_e^{n+1}} \frac{\partial P_O}{\partial x}\,dK &= \frac{1}{|\mathcal{S}_e^{n+1}|}\int_{\mathcal{S}_e^{n+1}} \frac{\partial u_h}{\partial x}\,dK \\
\frac{1}{|\mathcal{S}_e^{n+1}|}\int_{\mathcal{S}_e^{n+1}} \frac{\partial P_O}{\partial y}\,dK &= \frac{1}{|\mathcal{S}_e^{n+1}|}\int_{\mathcal{S}_e^{n+1}} \frac{\partial u_h}{\partial y}\,dK
\end{aligned} \tag{3.7}$$

The slope limiter only needs to be active at places where the solution displays strong discontinuities and for this purpose the discontinuity detector proposed by Krivodonova [51] is used. The discontinuity factor $\mathcal{I}$ of a numerical solution u_h at the time slab face $\mathcal{S}_e^{n+1}$ is defined as:

$$\mathcal{I} = \frac{|\int_{\partial\mathcal{S}_e^{n+1}}(u_h^- - u_h^+)de|}{h^{(p+1)/2}|\partial\mathcal{S}^{n+1}|\|u_h^-\|_{L_\infty}}, \tag{3.8}$$

where u_h^- and u_h^+ denote the solution traces from the inside and the outside of the time slab face, at time t_{n+1} and considering only space directions, $\partial\mathcal{S}_e^{n+1}$ denotes the time slab face boundary, which is composed of a number

of finite element edges, and h is the radius of the circle circumscribing $\mathcal{S}_e^{n+1}$. The solution is assumed to be smooth when $\mathcal{I} < \mathcal{I}_0$ and discontinuous when $\mathcal{I} > \mathcal{I}_0$, with $\mathcal{I}_0$ a constant parameter determining the amount of limiting.

3.4 Linear advection

Considered first are a number of single-fluid linear advection problems in one space dimension. The purpose of these tests is to check the accuracy of the STDG method without interface tracking, for continuous and discontinuous solutions, respectively, and also to investigate the effect of interface tracking on the accuracy for a discontinuous solution. In all test cases linear basis functions are used.

The linear advection equation:

$$\frac{\partial \rho}{\partial t} + a\frac{\partial \rho}{\partial x} = 0, \tag{3.9}$$

with ρ the advection variable and $a = 5\,m/s$ the advection velocity, is solved on a spatial domain $[-5\,m, 5\,m]$ from time $t = 0\,s$ to $1\,s$. Continuous and discontinuous initial conditions are defined as:

$$\rho(t,x) = \rho_0(x) = \begin{cases} 1.5 + 0.5\cos\left(\pi(x+2.5)\right) & \text{for } |x+2.5| \leq 1\ m \\ 1.0 & \text{for } |x+2.5| > 1\ m, \end{cases} \tag{3.10}$$

and

$$\rho(t,x) = \rho_0(x) = \begin{cases} 2.0 & \text{for } x < -2.5\ m \\ 1.0 & \text{for } x > -2.5\ m, \end{cases} \tag{3.11}$$

respectively. At the inflow boundary Dirichlet boundary conditions are used:

$$\rho(t,x) = \rho_0(x) \text{ at } x = -5\ m \tag{3.12}$$

Since this is a single-fluid test, an upwind flux is used everywhere. The exact solution to (3.9) is:

$$\rho(x,t) = \rho_0(x - at). \tag{3.13}$$

Table 3.1: Error and order of accuracy in the L_2 norm of the advection variable in the linear advection test with smooth initial conditions (3.10) and without interface tracking.

$N_x \times N_t$	L_2 error	L_2 order
20×10	0.287488	–
40×20	0.0986332	1.543
80×40	0.0212308	2.216
160×80	0.00459294	2.209

Table 3.2: Error and order of accuracy in the L_2 norm of the advection variable for the linear advection test with discontinuous initial conditions (3.11) and without interface tracking.

$N_x \times N_t$	L_2 error	L_2 order
20×10	0.327226	–
40×20	0.255301	0.358
80×40	0.198344	0.364
160×80	0.1537	0.368

The simulations are performed at $CFL_{\Delta t} = 1.0$.

First, the method is tested for the smooth initial solution (3.10) without mesh refinement. The solution at time $t = 1\,s$ is illustrated in Figure 3.4 (left). The results are presented in Table 3.1, where the L_2 errors and corresponding orders of accuracy are given for various mesh resolutions. Orders of accuracy of approximately 2 are observed, which is as expected since the STDG is of order $O(h^{p+1})$ for smooth solutions.

Second the method is tested for the discontinuous initial solution (3.11) without mesh refinement. The solution at time $t = 1\,s$ is illustrated in Figure 3.4 (right). Near the interface spurious oscillations are visible. The results are presented in Table 3.2, where the L_2 errors and the corresponding orders of accuracy are given for various mesh resolutions. In the L_2 norm the orders of accuracy are approximately 0.36, which is as expected since for discontinuous solutions computed on a static mesh the order of accuracy will typically not exceed $O(h^{1/2})$.

Third the method is tested for a discontinuous initial solution (3.11) with mesh refinement. The numerical solution at time $T = 1\,s$ and the refined space-time mesh using 20 elements are shown in Figure 3.5. The

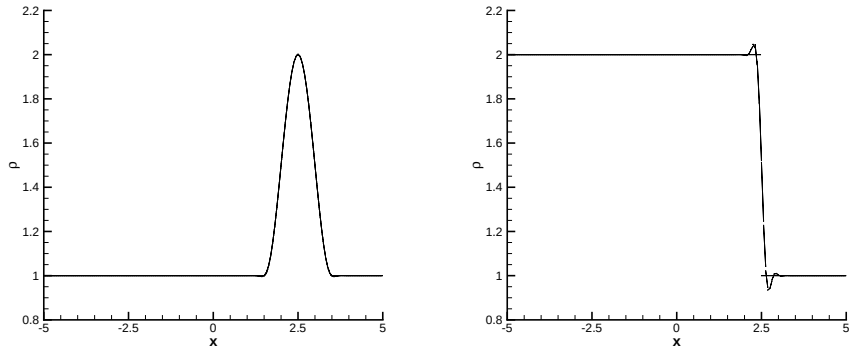

Figure 3.4: The exact (dotted) and numerical (solid) solutions at time $t = 1\,s$ of the linear advection tests without mesh refinement for continuous (left) and discontinuous initial conditions (right) using 160 elements.

performance of the two-fluid scheme is optimal for this test, with the error in the solution and the interface position both at machine precision. This is the case because the interface movement is linear in space-time; hence, the interface is represented exactly in the refined mesh.

Fourth, the method is tested for a non constant advection velocity $a = -x\,m/s$, a discontinuous initial solution (3.11) and with mesh refinement. The exact solution is given as:

$$\rho(t, x) = \rho_0(xe^t). \tag{3.14}$$

and the exact interface position at time t is $x_{IF}(t) = -2.5\,e^{-t}\,m$. In this test the discontinuity moves nonlinearly; hence, it cannot be represented exactly in the mesh. In Figure 3.6 the space-time mesh and solution are shown for a mesh of 20 elements and it is observed that the discontinuity is not resolved very well. The results are presented in Table 3.4.

Fifth, the same case as in the fourth test is considered, but at the discontinuity solid wall conditions are applied, implemented as a zero flux. By using solid wall conditions at the discontinuity, it is treated as an interface;

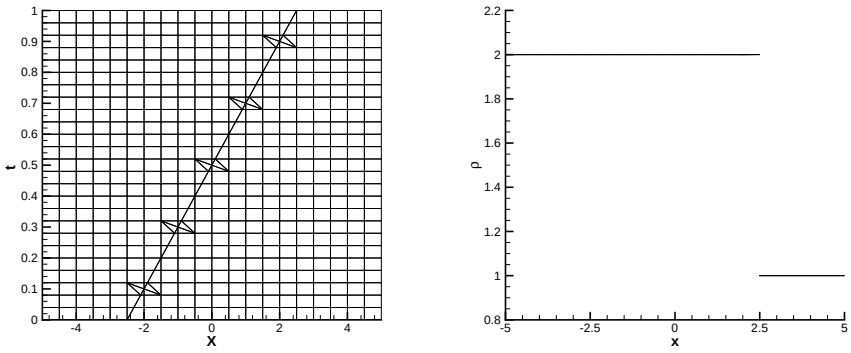

Figure 3.5: Space-time mesh and numerical solution at time $t = 1\,s$ of the linear advection test with mesh refinement for discontinuous initial conditions using 20 elements.

Table 3.3: Error and order of accuracy in the L_2 norm of the advection variable for the linear advection test with non constant velocity $a = -x\,m/s$ and upwind flux at the discontinuity.

$N_x \times N_t$	L_2 error	L_2 order
20×10	0.0408045	-
40×20	0.0254312	0.682
80×40	0.017552	0.535
160×80	0.0117917	0.5739

hence, the problem is considered as a two-fluid problem. In Figure 3.7 solution is shown for a mesh of 20 elements. The interface is captured much better. The results are presented in Table 3.4. In the L_2 norm the error converges to $O(h^{1/2})$.

In conclusion, results were presented for a number of single-fluid linear advection tests in one space dimension. For a uniform mesh and continuous and discontinuous solutions respectively the theoretical L_2 orders of accuracy could be confirmed. The results for the discontinuous solution were improved greatly by applying mesh refinement to capture the discontinuity. In addition results were presented for a non constant advection

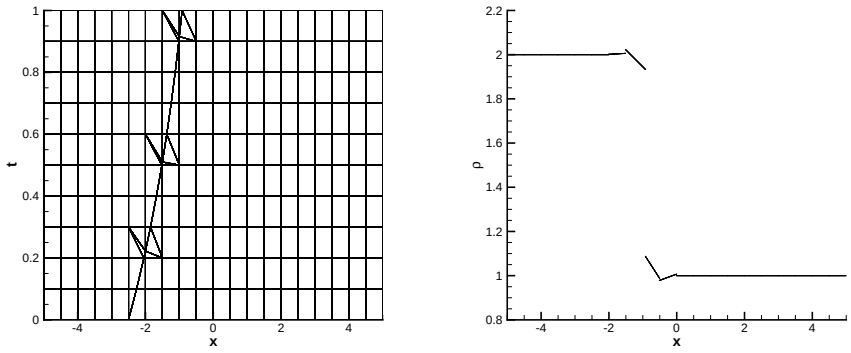

Figure 3.6: Space-time mesh and solution at time $t = 1\,s$ for the linear advection test with non constant velocity $a = -x$ using 20 elements and upwind flux at the discontinuity.

Table 3.4: Error and order of accuracy in the L_2 norm of the advection variable for the linear advection test with non constant velocity $a = -x\,m/s$ and solid wall flux at the discontinuity.

$N_x \times N_t$	L_2 error	L_2 order
20×10	0.00112646	-
40×20	0.000735784	0.614
80×40	0.000736948	-0.00228
160×80	0.000551662	0.4178

velocity $a = -x\,m/s$. When using an upwind flux at the discontinuity, it was observed that the discontinuity could not be captured very well. When using a solid wall flux at the interface, the discontinuity could be captured much better.

3.5 Zalesak disc

In order to investigate how well the method can handle moving interfaces in two space dimensions, the Zalesak disc test problem [110] is examined.

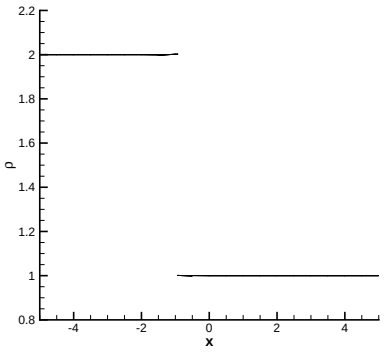

Figure 3.7: Solution at time $t = 1\,s$ for the linear advection test with non constant velocity $a = -x\,m/s$ using 20 elements and solid wall flux at the discontinuity.

A disc, initially as shown in Figure 3.8, is rotated counterclockwise one period around the domain midpoint $(x, y) = (0, 0)\,m$ with velocity $\mathbf{u} = (-2\pi y, 2\pi x)\,m/s$. The purpose of this test is to check the accuracy of the level set solution obtained with the discontinuous Galerkin method. The test also serves to illustrate the level set smoothing procedure and the two-fluid mesh refinement. The most difficult part in this test consists of capturing the sharp corners of the disc. The level set equation is solved in two space dimensions on the domain $[-4\,m, 4\,m] \times [-4\,m, 4\,m]$ from time $t = 0\,s$ to $1\,s$.

The simulations are performed at $CFL = 1.0$. In Figure 3.9 the approximate disc at the initial and the final time is shown for 80×80 and 160×160 elements, respectively. The interface evolution for 80×80 elements is shown in Figure 3.10. At the initial time the level set shows an error near the sharp edges of the disc, due to the fact that the nonlinear initial level set conditions cannot be represented exactly using piecewise linear polynomials. Also, after one rotation the level set solution shows a relatively large error near the sharp edges of the disc. It is expected that the results will improve by using a higher order level set approximation or

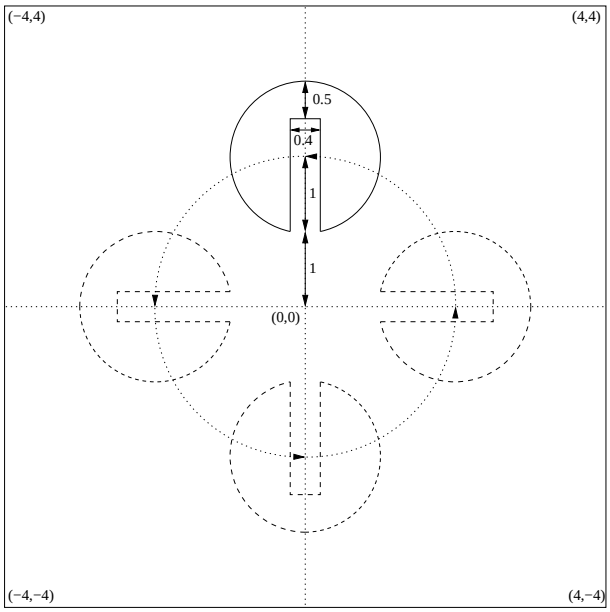

Figure 3.8: Zalesak disc test problem.

by applying h-refinement. The latter option is preferred since in general the level set velocity will be obtained from the flow velocity and may not be of high order.

From the test results it is concluded that when sharp corners are present in the problem, the accuracy of the method is expected to suffer quite severely. However, even for a smooth interface, small errors in the level set and interface position are likely to be present.

3.6 Sod's ideal gas shock tube

Considered is Sod's ideal gas shock tube test [88]. 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

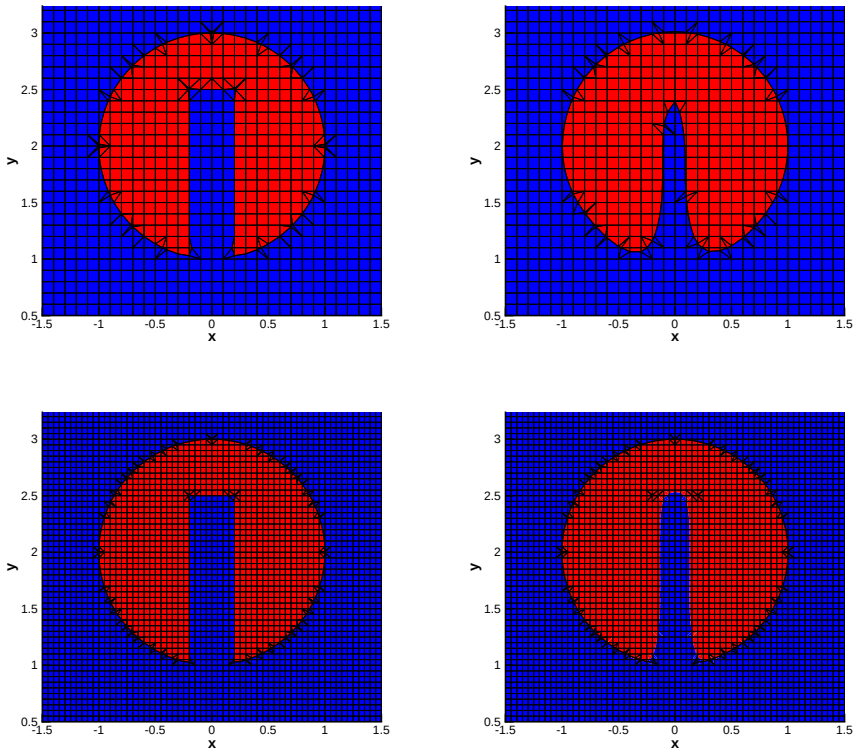

Figure 3.9: The refined mesh at the initial time (left) and after one rotation (right) for the Zalesak disc test problem for a mesh with 80×80 (top) 160×160 (bottom) elements.

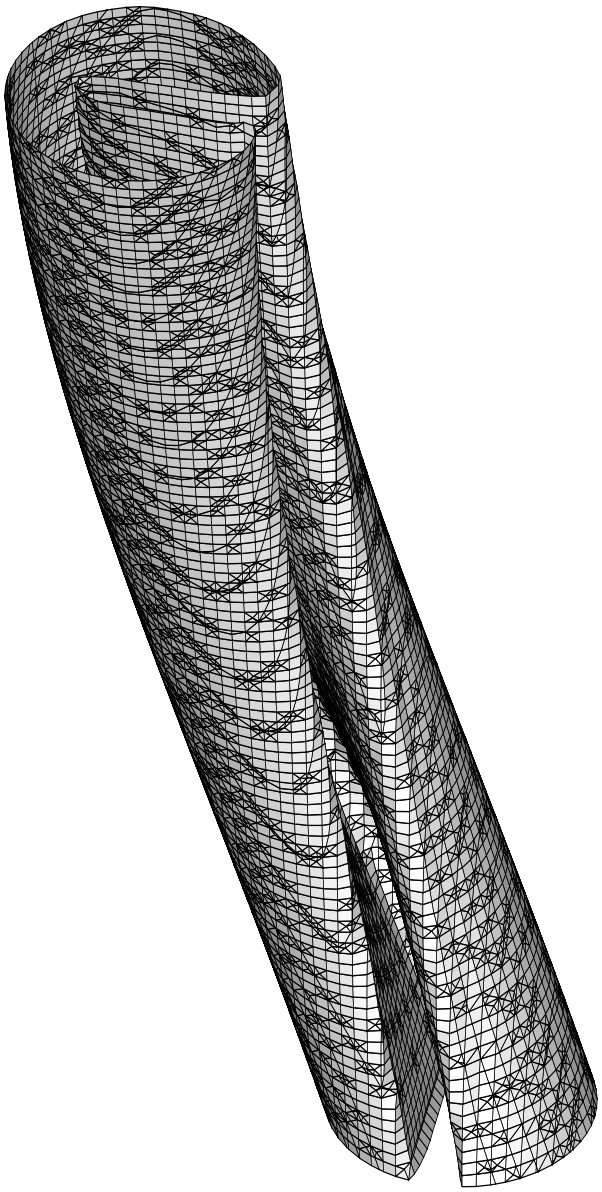

Figure 3.10: Interface evolution for the Zalesak disc test problem for the first 1/4 rotation for a mesh with 80×80 elements.

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.

The one dimensional Euler equations expressing conservation of mass, momentum and energy are defined as

$$\begin{aligned} \frac{\partial \rho}{\partial t} + \frac{\partial (\rho u)}{\partial x} &= 0 \\ \frac{\partial (\rho u)}{\partial t} + \frac{\partial (\rho u^2 + p)}{\partial x} &= 0 \\ \frac{\partial (\rho E)}{\partial t} + \frac{\partial (u(\rho E + p))}{\partial x} &= 0, \end{aligned} \tag{3.15}$$

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:

$$e = \frac{p}{\rho(\gamma - 1)}, \tag{3.16}$$

where $\gamma = 1.4$. The Euler equations are solved on a spatial domain $[-5\,m, 5\,m]$ from time $t = 0\,s$ to $0.01\,s$. Initially the interface is located at $x = 0\,m$ and both fluids are in constant states:

$$(\rho, u, p)(0, x) = \tag{3.17}$$
$$\begin{cases} (\rho_L, u_L, p_L) = (2.37804\ kg/m^3, 0\ m/s, 2.0 \times 10^5\ Pa) & \text{for } x < 0\ m \\ (\rho_R, u_R, p_R) = (1.18902\ kg/m^3, 0\ m/s, 1.0 \times 10^5\ Pa) & \text{for } x > 0\ m. \end{cases}$$

At the boundaries solid wall conditions are imposed:

$$u \cdot \bar{n} = 0 \text{ at } x = \pm 5\ m \tag{3.18}$$

At the interface the velocity and pressure are continuous.

The solution to (3.15), illustrated in Figure 3.11, features an expansion wave moving to the left with head speed $S_{LH} = -343.138\,m/s$ and tail speed $S_{LT} = -241.218\,m/s$, a contact wave moving to the right with

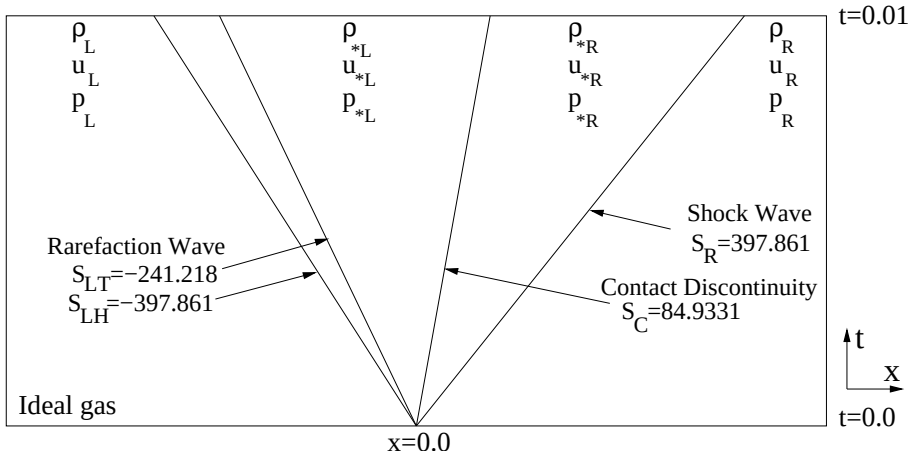

Figure 3.11: The solution structure of the ideal gas shock tube.

speed $S_C = 84.9331\,m/s$ and a shock wave also moving to the right with speed $S_R = 397.861\,m/s$. Between the expansion and the contact wave the solution is constant and equal to the left star state (ρ_L^*, u^*, p^*) and between the contact and the shock wave the solution is also constant and equal to the right star state (ρ_R^*, u^*, p^*), where $\rho_L^* = 1.84490\,kg/m^3, \rho_R^* = 1.51174\,kg/m^3, u^* = 84.9331\,m/s$ and $p^* = 1.40179 \times 10^5\,Pa$.

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 conditions consists of two constant states:

$$\mathbf{w}(x,0) = \begin{cases} \mathbf{w}_L & \text{when } x < 0 \\ \mathbf{w}_R & \text{when } x > 0. \end{cases} \tag{3.19}$$

The HLLC flux extended to space-time meshes [6, 98] is defined as:

$$\begin{aligned}\mathcal{H}_{HLLC} =& \frac{1}{2}\Big(\mathbf{F}_L + \mathbf{F}_R \\ &- (|S_L - v| - |S_M - v|)\mathbf{w}_L^* + (|S_R - v| - |S_M - v|)\mathbf{w}_R^* \\ &+ |S_L - v|\,\mathbf{w}_L - |S_R - v|\,\mathbf{w}_R - v(\mathbf{w}_L + \mathbf{w}_R)\Big),\end{aligned} \tag{3.20}$$

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:

$$\begin{aligned}\rho_K^* &= \rho_K \frac{S_K - u_K}{S_K - u^*} \\ \rho_K^* u^*(u^* - S_K) &= (p_K - p^*) + \rho_K u_K(u_K - S_K),\end{aligned} \tag{3.21}$$

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

$$S_M = \frac{\rho_R u_R(S_R - u_R) - \rho_L u_L(S_L - u_L) + p_L - p_R}{\rho_R(S_R - u_R) - \rho_L(S_L - u_L)}. \tag{3.22}$$

The wave speeds S_L and S_R are estimated as:

$$S_L = \min(u_L - a_L, u_R - a_R),\ S_R = \max(u_L + a_L, u_R + a_R). \tag{3.23}$$

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:

$$\mathbf{w}_L^* = \frac{S_L - u_L}{S_L - S_M}\mathbf{w}_L + \frac{1}{S_L - S_M}\begin{pmatrix} 0 \\ p^* - p_L \\ p^* S_M - p_L u_L \end{pmatrix}, \tag{3.24}$$

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:

$$p^* = \rho_L(S_L - u_L)(S_M - u_L) + p_L = \rho_R(S_R - u_R)(S_M - u_R) + p_R. \quad (3.25)$$

The Euler equations are discretized using the set of primitive variables $\mathbf{v} = (\rho, \mathbf{u}, p)$. This is motivated by the observation that in many two-fluid flow problems the velocity and often also the pressure are continuous across the interface while the momentum and energy are not. Since the conservative equations are used mass, momentum and energy are still conserved. The approximate primitive variables are defined as:

$$\mathbf{v}_h^{p,i}(t, \bar{\mathbf{x}})|_{\mathcal{K}_j^n} = \sum_m \hat{\mathbf{V}}_m^i(\mathcal{K}_j^{i,n})\phi_m(t, \bar{\mathbf{x}}) \quad (3.26)$$

with $\hat{\mathbf{V}}_m^i$ the primitive flow approximation coefficients. The discretized equations extended into pseudo-time become:

$$\tilde{M}_{mlkp}^{i,n} \frac{\partial \hat{V}_{pm}^{i,n}}{\partial \tau} + \mathcal{L}_{kl}^{i,n}(\hat{\mathbf{W}}^n(\mathbf{V}^n), \hat{\mathbf{W}}^{n-1}(\mathbf{V}^{n-1})) = 0 \quad (3.27)$$

with

$$\tilde{M}_{mlkp}^{i,n} = \int_{\mathcal{K}_j^{i,n}} \phi_l \phi_m \frac{\partial \mathbf{w}_k^i}{\partial \mathbf{v}_p^i} \, d\mathcal{K}. \quad (3.28)$$

The discretized equations are simplified by using evaluations in the element midpoints $\mathbf{x}_{mid}$ and replacing the $\phi_l\phi_m$ terms by the delta function δ_{lm} in (3.28) to obtain:

$$\tilde{N}_{kp}^{i,n} \frac{\partial \hat{V}_{pl}^{i,n}}{\partial \tau} + \mathcal{L}_{kl}^{i,n}(\hat{\mathbf{W}}^n(\mathbf{V}^n), \hat{\mathbf{W}}^{n-1}(\mathbf{V}^{n-1})) = 0 \quad (3.29)$$

with

$$\tilde{N}_{kp}^{i,n} = |\mathcal{K}_j^{i,n}| \, \frac{\partial \mathbf{w}_k^i}{\partial \mathbf{v}_p^i}(\mathbf{x}_{mid}). \quad (3.30)$$

Algorithm 7 Pseudo-time integration method for solving the non-linear algebraic equations in the space-time discretization.

1. *Initialize first Runge-Kutta stage:* $\bar{\mathbf{V}}^{i,(0)} = \hat{\mathbf{V}}^{i,n}$.
2. *Calculate* $\bar{\mathbf{V}}^{i,(s)}, s = 1, \cdots, 5$*:*

$$(1+\alpha_s\lambda)\bar{\mathbf{V}}^{i,(s)} = \bar{\mathbf{V}}^{i,(0)} + \alpha_s\lambda\Bigg(\bar{\mathbf{V}}^{i,(s-1)} - \Delta t\,(\tilde{N}^{i,n})^{-1}\,\mathcal{L}(\bar{\mathbf{W}}^{i,(s-1)}(\bar{\mathbf{V}}^{i,(s-1)}), \bar{\mathbf{W}}^{i,n-1}(\bar{\mathbf{V}}^{i,(n-1)}))\Bigg)$$

3. *Update solution:* $\hat{\mathbf{V}}^{i,n} = \bar{\mathbf{V}}^{i,(5)}$.

To account for the change in variables the Runge-Kutta pseudo time integration method is modified with $\tilde{\mathbf{N}}$ in the following way:

At the interface the HLLC flux for a contact discontinuity is used. Assuming the interface coincides with the contact wave, $S_M = v$ and the corresponding HLLC flux defines the contact HLLC flux $\mathcal{H}^C_{HLLC}$:

$$\mathcal{H}^C_{HLLC} = \frac{1}{2}\Bigg(\mathbf{F}_L + \mathbf{F}_R + (S_M - S_L)(\mathbf{w}_L - \mathbf{w}^*_L) + (S_M - S_R)(\mathbf{w}_R - \mathbf{w}^*_R) - S_M(\mathbf{w}_L + \mathbf{w}_R)\Bigg). \tag{3.31}$$

By inserting the expressions for $\mathbf{w}^*_K$, it follows that:

$$\mathcal{H}^C_{HLLC} = (0, p^*, p^*u^*)^T \tag{3.32}$$

which shows that there is no mass flux through the contact interface. At the domain boundary faces the solid wall conditions are implemented in the HLLC flux by defining the right state as:

$$\rho_R = \rho_L,\ u_R = -u_L,\ p_R = p_L. \tag{3.33}$$

The simulations are performed at $CFL_{\Delta t} \approx 0.4$.

The test results using the contact flux (3.32) are presented in Table 3.5. The solution converges in the L_2 norm. In Figure 3.12 the evolution of

Table 3.5: Error and order of accuracy in the L_2 norm of the density for the ideal gas Euler shock tube test using the contact interface flux.

$N_x \times N_t$	L_2 error	L_2 order
40×40	0.0708762	–
80×80	0.0484641	0.548
160×160	0.0296357	0.710
320×320	0.0213965	0.467

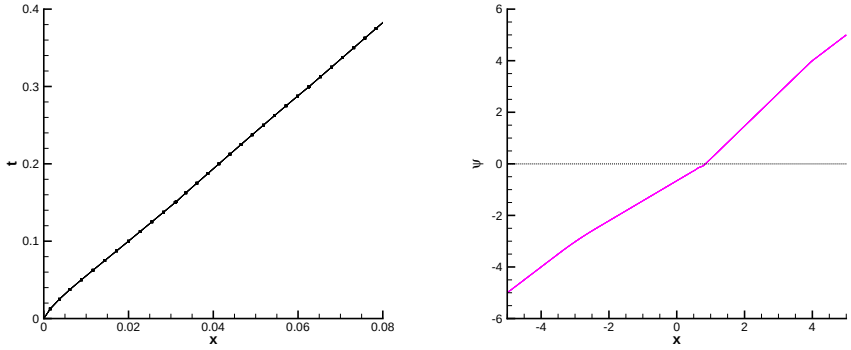

Figure 3.12: The time evolution of the interface and level set solution at time $t = 0.01\ s$ for the ideal gas shock tube using 320 background elements, the contact flux and no slope limiter.

the interface position for the first few time steps and the level set at the final time are shown. It is observed that in the first few time steps the interface moves too slow. The density, density zoom, velocity and pressure profiles at the final time are shown in Figure 3.13. Because the interface moves too slow initially, small undershoots are created in the density at the interface, which remain in the numerical solution until the final time. Small oscillations are also observed in the density, velocity and pressure profiles which radiate outwards from the interface.

In order to diminish the observed oscillations at the interface, an alternative interface flux is proposed, which is defined separately for the left and

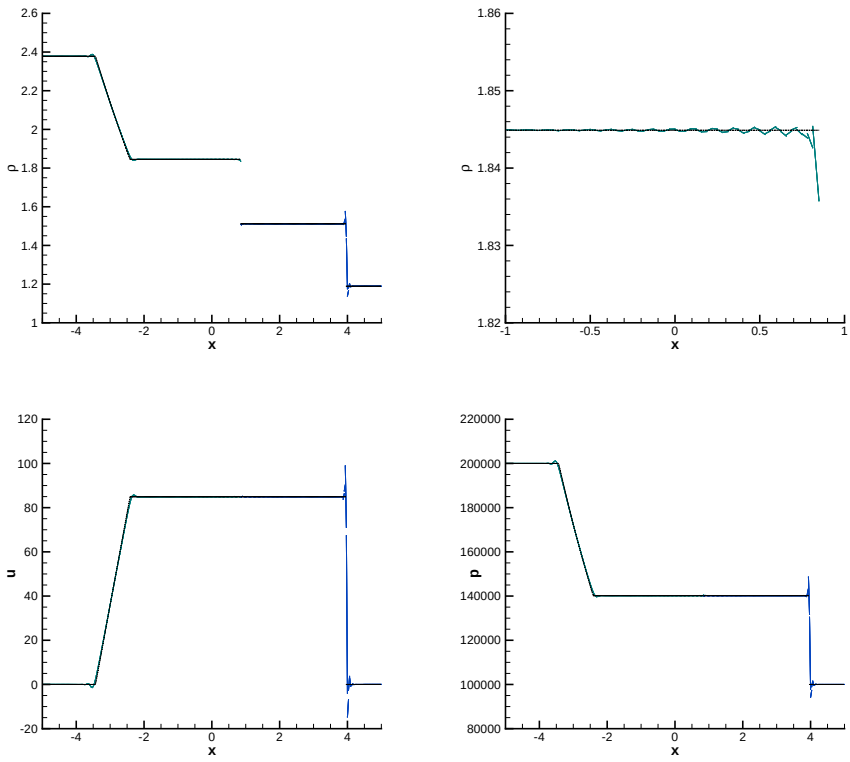

Figure 3.13: The exact (dotted) and numerical (solid) density, density zoom, velocity and pressure at time $t = 0.01\,s$ for the ideal gas shock tube using 320 background elements, the contact flux and no slope limiter.

Table 3.6: Error and order of accuracy in the L_2 norm of the density for the ideal gas Euler shock tube test with interface tracking and using the interface flux (3.34).

$N_x \times N_t$	L_2 error	L_2 order
40×40	0.0729742	–
80×80	0.0492437	0.567
160×160	0.0300191	0.714
320×320	0.0217169	0.467

right sides of the interface:

$$\begin{aligned} \mathcal{H}^L_{HLLC} =& \mathbf{w}^*_L(S_M - v) + \mathcal{H}^C_{HLLC} \\ \mathcal{H}^R_{HLLC} =& \mathbf{w}^*_R(S_M - v) + \mathcal{H}^C_{HLLC} \end{aligned} \tag{3.34}$$

When the interface representation in the mesh is exact, $S_M = v$ and the interface flux is reduced to $\mathcal{H}^C_{HLLC}$. The interface numerical flux now removes the small numerical oscillations caused by errors in the interface shape and position at the cost of mass conservation at the interface. The results with the interface flux (3.34) are presented in Table 3.6. The solution converges in the L_2 norm. In Figure 3.14 the density, density zoom, velocity and pressure profiles at the final time with the interface flux (3.34) are shown. Again, the interface moves too slow initially, causing undershoots in the density at the interface, which remains in the numerical solution until the final time. The density, velocity and pressure profiles do not show the oscillations observed before with the contact interface flux. In Figure 3.15 the mass evolution of the two fluids is shown. The mass loss is very small for this test.

In order to remove the spikes appearing near the expansion and shock waves in the solution with the interface flux (3.34) the HWENO slope limiter is used, and in Figure 3.16 the resulting density, density zoom, velocity and pressure profiles at the final time are shown. The slope limiter reduces the spikes at the expansion and shock waves. However, a small offset error is observed in the density, velocity and pressure profiles in the star region.

Finally, the simulation is run without the initial time steps, from $t = T/10$ to $t = T$. The resulting density profile is shown in Figure 3.17. The

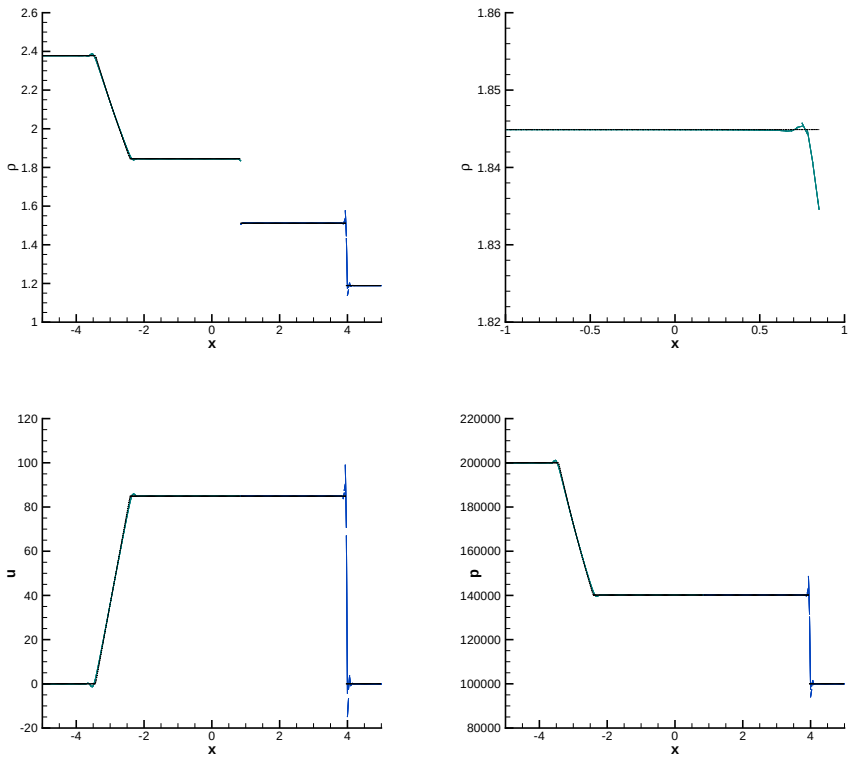

Figure 3.14: The exact (dotted) and numerical (solid) density, density zoom, velocity and pressure at time $t = 0.01\,s$ for the ideal gas shock tube using 320 background elements, interface flux (3.34) and no slope limiter.

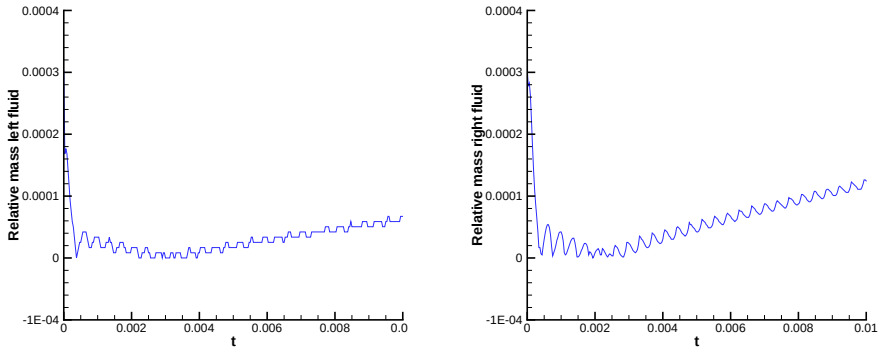

Figure 3.15: Relative mass over time of the left (left) and right (right) fluids for the ideal gas shock tube using 320 background elements, interface flux (3.34) and no slope limiter. The relative mass is defined as $|M_e - M_h|/M_e$, with M_e the exact and M_h the numerical amount of mass.

results are much better than those obtained previously, especially near the interface. This is because the error made in the first number of time steps, when the rarefaction, contact and shock waves are too close to each other to be resolved well numerically, remains in the simulation for all subsequent time.

In conclusion, the two-fluid method has been applied to Sod's shock tube test. Using a contact interface flux oscillations were observed at the interface. An alternative interface flux (3.34) was developed, reducing the oscillations at the interface at the cost of conservation. The interface flux (3.34) was tested with promising results. Using the slope limiter reduced the spikes near the expansion and shock waves, but introduced a small offset error in the star region. Starting the simulation at $t = T/10$ greatly improved the numerical results.

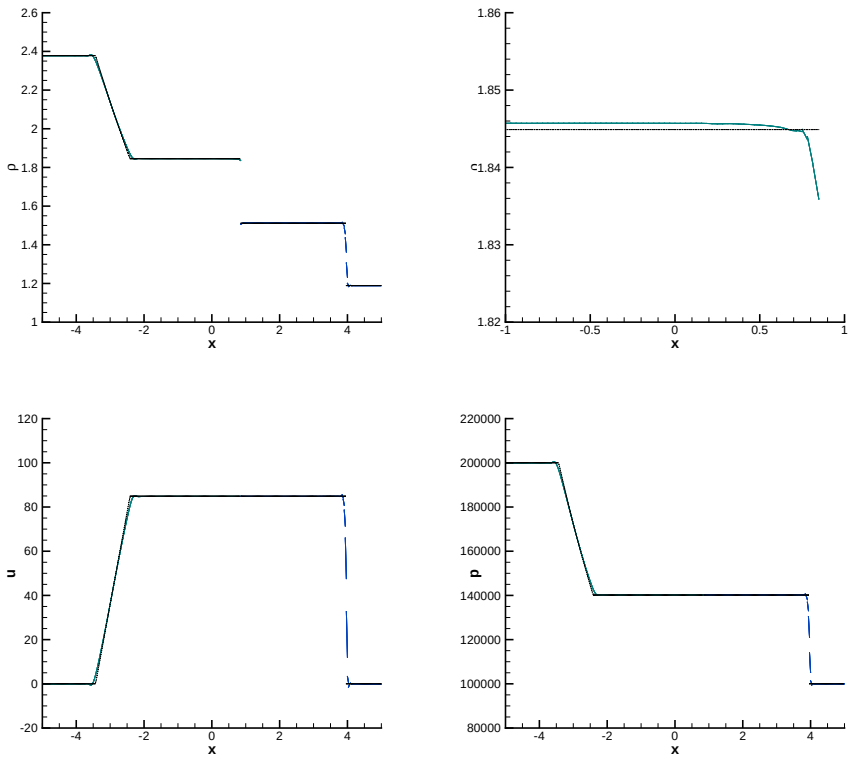

Figure 3.16: The exact (dotted) and numerical (solid) density, density zoom, velocity and pressure at time $t = 0.01\,s$ for the ideal gas shock tube using 320 background elements, interface flux (3.34) and slope limiter.

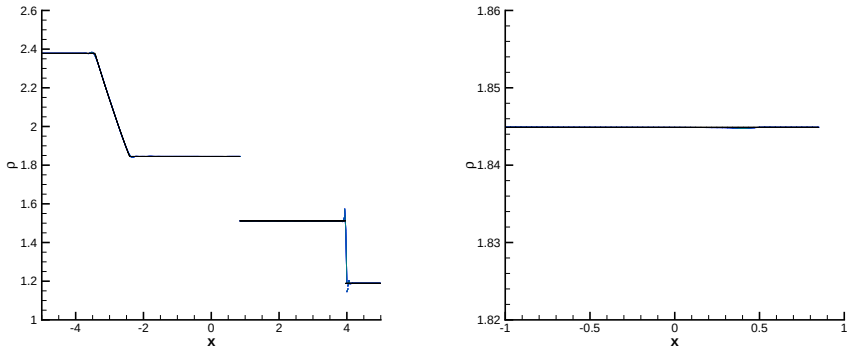

Figure 3.17: The exact (dotted) and numerical (solid) density and density zoom at time $t = 0.01\ s$ for the ideal gas shock tube using 320 background elements, interface flux (3.34), no slope limiter and starting at initial time $t = T/10$.

3.7 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 [12, 13, 14, 103] and [104] 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 governing equations for an effectively compressible magma are the Euler equations for mass and momentum:

$$\partial_t \mathbf{w} + \partial_x \mathbf{F}(\mathbf{w}) = 0, \tag{3.35}$$

with

$$\mathbf{w} = \begin{pmatrix} \rho \\ \rho u \end{pmatrix}, \ \mathbf{F} = \begin{pmatrix} \rho u \\ \rho u^2 + p \end{pmatrix}. \tag{3.36}$$

For the ideal gas the one dimensional Euler equations (3.15) are used. 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:

$$\rho = \left(\frac{n(p) R_m T}{p} + \frac{1 - n(p)}{\sigma} \right)^{-1}, \tag{3.37}$$

where $R_m = 462\, J/kgK$ 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 $p \geq p_c$ the following relation is used:

$$\rho = \sigma + c_m^{-2}(p - p_c), \tag{3.38}$$

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:

$$n(p) = n_0 - S_h p^{\beta}. \tag{3.39}$$

For basaltic high volatile magma, $n_0 = 0.02$, $\beta \approx 0.5$, $T = 1200\, K$ and $S_h = 3.0 \times 10^{-6}\, Pa^{-\beta}$. The magma is assumed to be isothermal at a temperature of $1200K$. For isothermal magma the density depends only

on the pressure, $\rho = \rho(p)$. The speed of sound a is defined for isothermal magma as:

$$\begin{aligned} 1/a^2 \equiv \left(\frac{\partial \rho}{\partial p}\right)_T &= -\rho^2 \frac{\partial (1/\rho)}{\partial p} \\ &= -\rho^2 \left[\frac{d\, n(p)}{dp}\left(\frac{R_m T}{p} + \frac{1}{\sigma}\right) - \frac{n(p) R_m T}{p^2}\right]. \end{aligned} \tag{3.40}$$

The simulations are performed on a spatial domain $[-5\,m, 5\,m]$ 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:

$$(\rho, u, p)(0, x) = \tag{3.41}$$
$$\begin{cases} (\rho_L, u_L, p_L) = (535.195\ kg/m^3, 0\ m/s, 5 \times 10^6\ Pa) & \text{for } x < 0\ m \\ (\rho_R, u_R, p_R) = (1.18902\ kg/m^3, 0\ m/s, 1.0 \times 10^5\ Pa) & \text{for } x > 0\ m. \end{cases}$$

At the boundaries solid wall conditions are imposed:

$$u \cdot \bar{n} = 0\ m/s \text{ at } x = \pm 5\ m. \tag{3.42}$$

At the interface continuity of the velocity and pressure is imposed. 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: $\rho_L^* = 28.0517\,kg/m^3, \rho_R^* = 2.45364\,kg/m^3, u^* = 286.329\,m/s, p^* = 2.89134 \times 10^5\,Pa$. The solution structure is shown in Figure 3.18. At the interface the interface flux (3.34) is used, adapted for use with isothermal magma. With the contact interface flux (3.32) the simulations were not stable enough. At the boundary faces the solid wall conditions are implemented in the HLLC flux by defining the right state as:

$$\rho_R = \rho_L,\ u_R = -u_L,\ p_R = p_L. \tag{3.43}$$

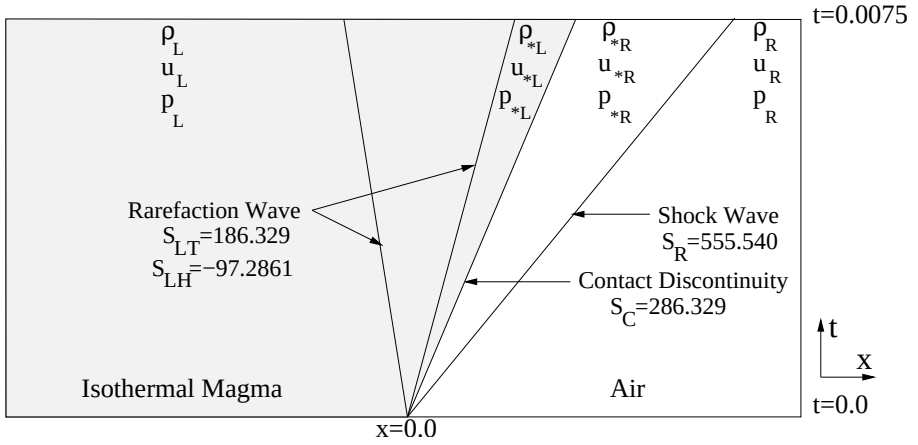

Figure 3.18: The solution structure of the Euler magma - ideal gas shock tube.

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 $CFL_{\Delta t} \approx 0.56$. Primitive variable discretizations are used for both fluids.

The test results for the solution at time $t = 0.0075\,s$ using the interface flux (3.34) are presented in Table 3.7 and convergence in the L_2 norm is observed. In Figure 3.19 the interface evolution over time and the level set profile at the final time are shown. Compared to the ideal gas shock tube test results, it takes much longer for the interface to reach the star velocity. Also, the level set becomes more distorted over time. The reason for this behavior lies in the use of the global flow velocity for advecting the level set. This problem can be fixed by reinitializing the level set every few time steps. In Figure 3.20 the density, density zoom, velocity and pressure at the final time using the interface flux (3.34) are shown. In Figure 3.21 the mass evolution for the magma and the ideal gas when using the interface flux (3.34) and without slope limiter is shown. The amount of mass loss is negligible. In Figure 3.22 the density, density zoom, velocity and pressure at the final time using the interface flux (3.34) and the slope limiter are

Table 3.7: Error and order of accuracy in the L_2 norm of the density for the isothermal magma and ideal gas Euler shock tube test using the interface flux (3.34).

$N_x \times N_t$	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

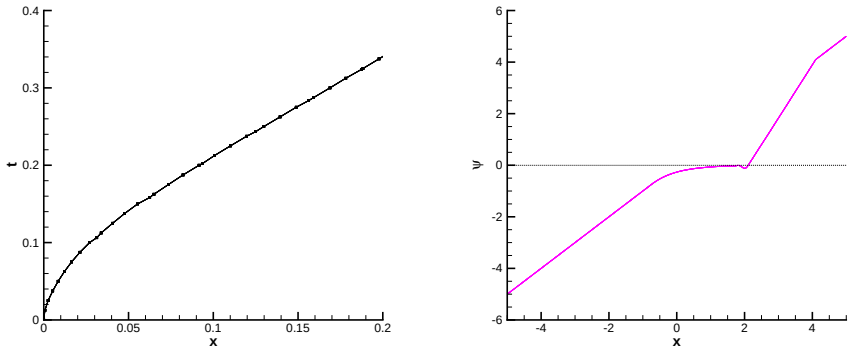

Figure 3.19: The time evolution of the interface position and level set at time $t = 0.0075\,s$ for the Euler magma - ideal gas shock tube using 320 background elements, interface flux (3.34) and no slope limiter.

shown. Like in the shock tube test with the ideal gas, the slope limiter reduces the spikes at the shock wave but introduces a small offset error in the density, velocity and pressure profiles in the star region. Also, in the solution with the slope limiter the error in the shock position is visibly larger, probably because of the numerical dissipation added by the slope limiter to the flow velocity near the shock.

Finally, the simulation is run without the initial time steps, from $t = T/10$ to $t = T$. The resulting density profile is shown in Figure 3.23. Because the error made in the first number of time steps remains is excluded in this simulation, the results are much better.

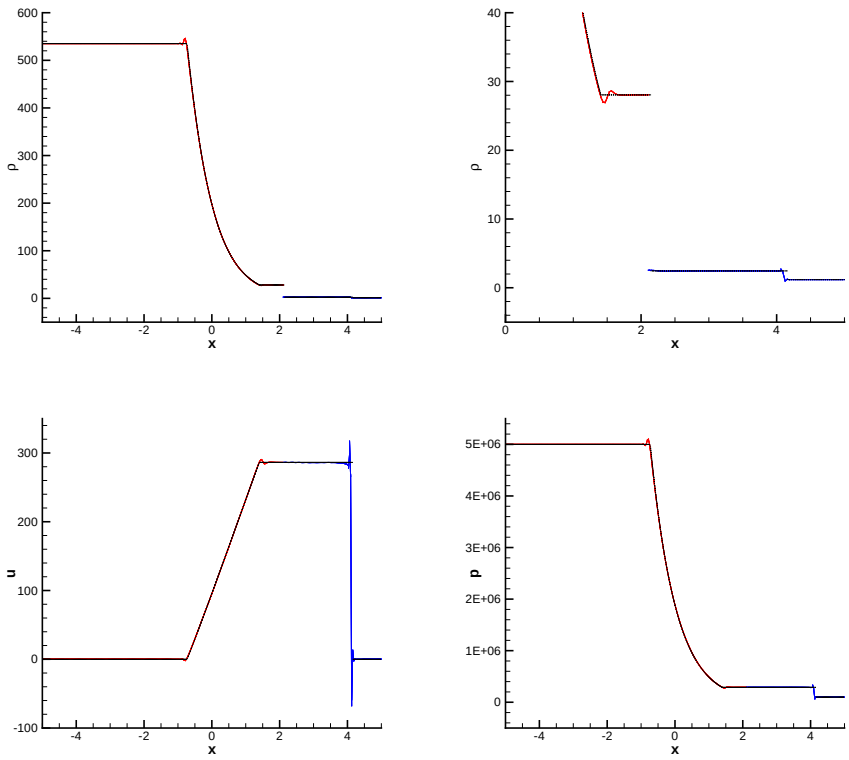

Figure 3.20: The exact (dotted) and numerical (solid) density, density zoom, velocity and pressure at time $t = 0.0075\,s$ for the Euler magma - ideal gas shock tube using 320 background elements, interface flux (3.34) and no slope limiter.

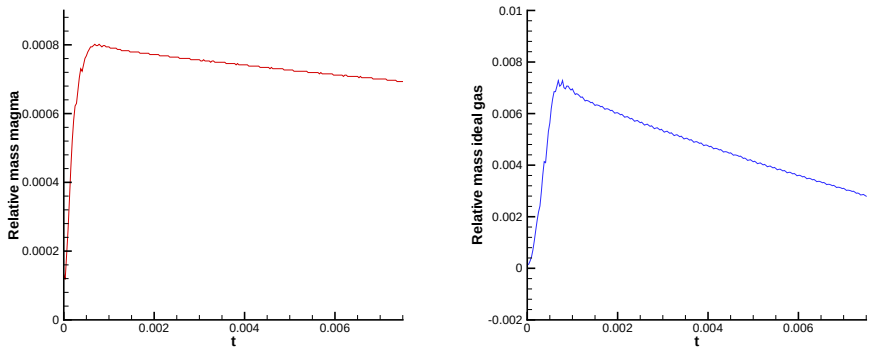

Figure 3.21: Relative mass over time of magma (left) and ideal gas (right) for the Euler magma - ideal gas shock tube using 320 background elements, interface flux (3.34) and no slope limiter. The relative mass is defined as $|M_e - M_h|/M_e$, with M_e the exact and M_h the numerical amount of mass.

In conclusion, the two-fluid method was used to solve a magma - ideal gas shock tube problem with the interface flux (3.34) with promising results. Using the slope limiter reduced the spikes near the expansion and shock waves, but introduced a small offset error in the star region and also decreased the accuracy of the shock position. Starting the simulation at $t = T/10$ greatly improved the numerical results. In this test the level set became very distorted, probably because of the advection with the global velocity. Periodic reinitialization of the level set can be used to solve this problem.

3.8 Cylinder flow

Considered is the subsonic flow of an ideal gas around a cylinder at Mach number 0.38 ([9, 52, 99]). The purpose of this test is to compare the performance of the method for cut-cell and boundary conforming meshes,

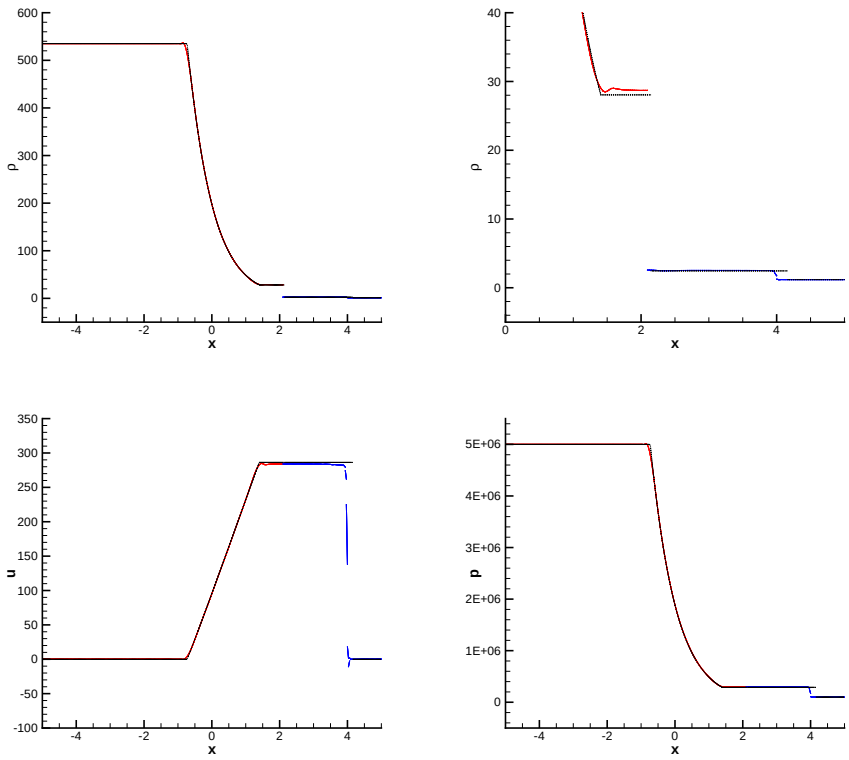

Figure 3.22: The exact (dotted) and numerical (solid) density, density zoom, velocity and pressure at time $t = 0.0075\,s$ for the Euler magma - ideal gas shock tube using 320 background elements and the interface flux (3.34) and slope limiter.

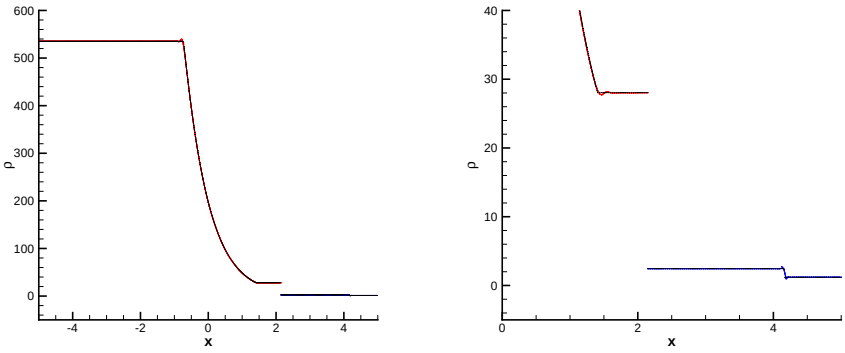

Figure 3.23: The exact (dotted) and numerical (solid) density and density zoom at time $t = 0.0075\,s$ for the Euler magma - ideal gas shock tube using 320 background elements and the interface flux (3.34), no slope limiter and starting at initial time $t = T/10$.

respectively. The two dimensional Euler equations are defined as

$$\begin{aligned}
\frac{\partial \rho}{\partial t} + \frac{\partial(\rho u)}{\partial x} + \frac{\partial(\rho v)}{\partial y} &= 0 \\
\frac{\partial(\rho u)}{\partial t} + \frac{\partial(\rho u^2 + p)}{\partial x} + \frac{\partial(\rho uv)}{\partial y} &= 0 \\
\frac{\partial(\rho v)}{\partial t} + \frac{\partial(\rho uv)}{\partial x} + \frac{\partial(\rho v^2 + p)}{\partial y} &= 0 \\
\frac{\partial(\rho E)}{\partial t} + \frac{\partial(u(\rho E + p))}{\partial x} + \frac{\partial(v(\rho E + p))}{\partial y} &= 0,
\end{aligned} \tag{3.44}$$

with ρ the density, u the velocity in the x direction, v the velocity in the y direction, 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 the equation of state (EOS) 3.16 is required to account for the thermodynamic properties of the ideal gas. At the cylinder surface solid wall boundary conditions

$$(u, v) \cdot \bar{\mathbf{n}} = 0 \tag{3.45}$$

are imposed. The far field state is defined as:

$$\begin{aligned} &\rho_\infty = 1.18902\ kg/m^3,\ u_\infty = 1.304 \times 10^2\ m/s, \\ &v_\infty = 0\ m/s,\ p_\infty = 1.0 \times 10^5\ Pa \end{aligned} \tag{3.46}$$

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

$$\begin{aligned} \rho_R &= \rho_L \\ u_R &= u_L - 2(u_L n_x + v_L n_y))n_x \\ v_R &= v_L - 2(u_L n_x + v_L n_y))n_y \\ p_R &= p_L. \end{aligned} \tag{3.47}$$

At the far field boundary the far field state is used.

The Mach number is defined as $M = u/a$ with $a = \sqrt{\lambda p/\rho}$ the speed of sound. Since the flow is subsonic, $M < 1$ everywhere. The total pressure loss is defined as

$$p_{loss} = 1 - \frac{p}{p_\infty}\left(\frac{1 + \frac{1}{2}(\gamma - 1)M^2}{1 + \frac{1}{2}(\gamma - 1)M_\infty^2}\right)^{\frac{\gamma}{\gamma - 1}}. \tag{3.48}$$

For subsonic inviscid flow the total pressure loss should be zero; hence, it is a good indicator for the accuracy of the numerical algorithm.

The test was performed both for a refined cut-cell mesh and a boundary conforming mesh around the cylinder. Computations were continued until steady state was reached. The cylinder has a radius of $1\,m$. The cut-cell simulations were performed using a 160×160 background mesh covering the area $[-10\,m, 10\,m] \times [-10\,m, 10\,m]$. The fitted mesh had 96×72 elements and an outer radius of 56.6. In the tests only linear basis function were used.

The results using a cut cell mesh with 160×160 elements are shown in Figures 3.24 and 3.25. The results using a boundary conforming mesh of 96×72 elements are shown in Figures 3.26 and 3.27.

For both types of meshes the results are of similar quality as those found in [9] and [99]. No convergence problems such as experienced in

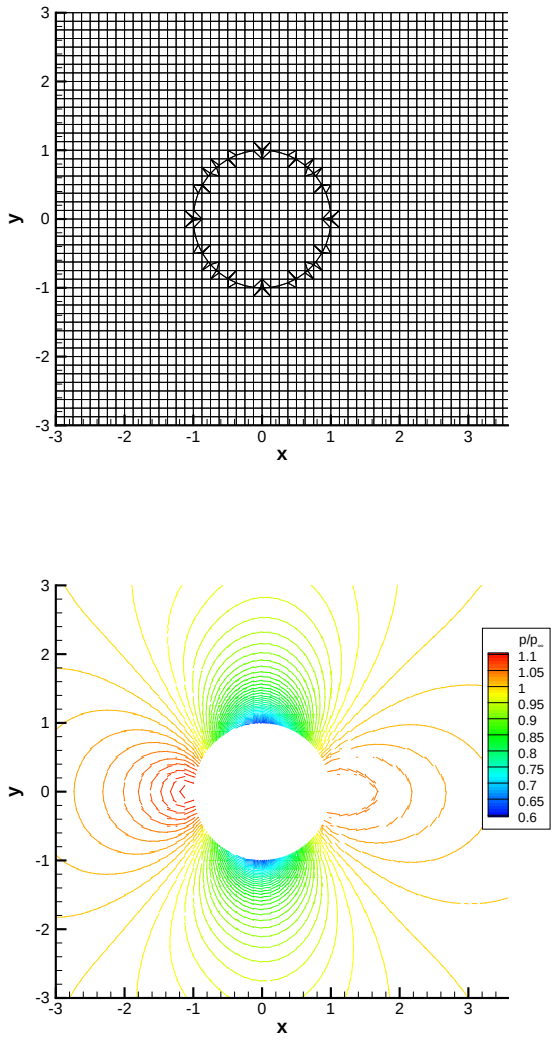

Figure 3.24: Mesh and normalized pressure for $M = 0.38$ subsonic flow around a cylinder of radius $1\,m$ and using a 160×160 cut-cell mesh of dimensions $[-10\,m, 10\,m] \times [-10\,m, 10\,m]$.

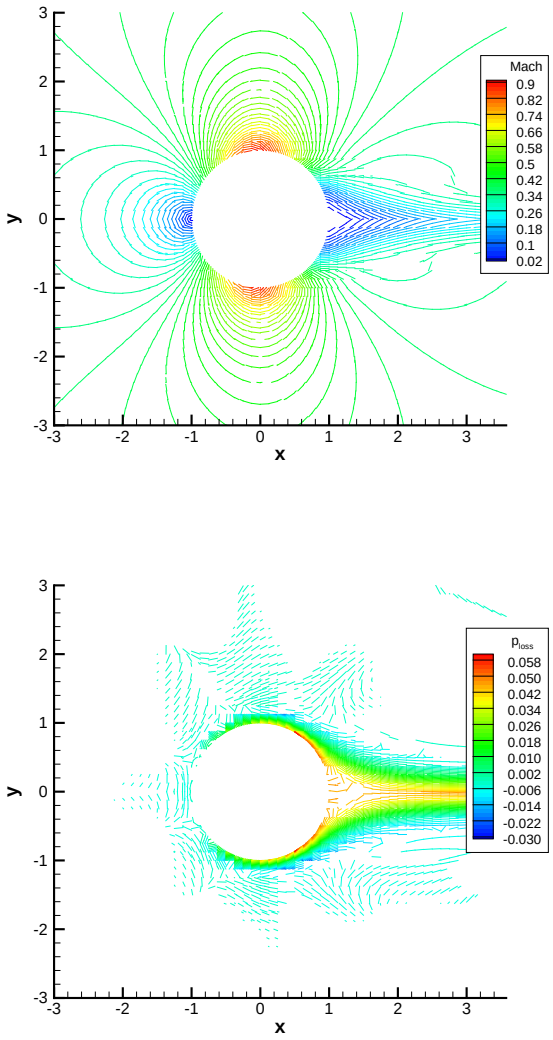

Figure 3.25: Mach number and pressure loss for $M = 0.38$ subsonic flow around a cylinder of radius $1\,m$ and using a 160×160 cut-cell mesh of dimensions $[-10\,m, 10\,m] \times [-10\,m, 10\,m]$.

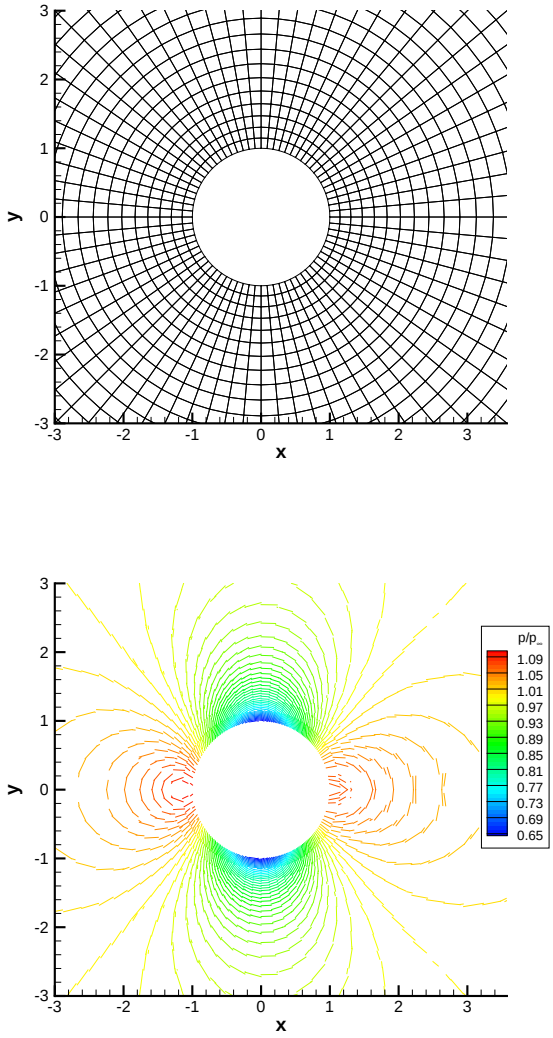

Figure 3.26: Mesh and normalized pressure for $M = 0.38$ subsonic flow around a cylinder of radius $1\,m$ and using a 96×72 boundary conforming mesh with outer radius $56.6\,m$.

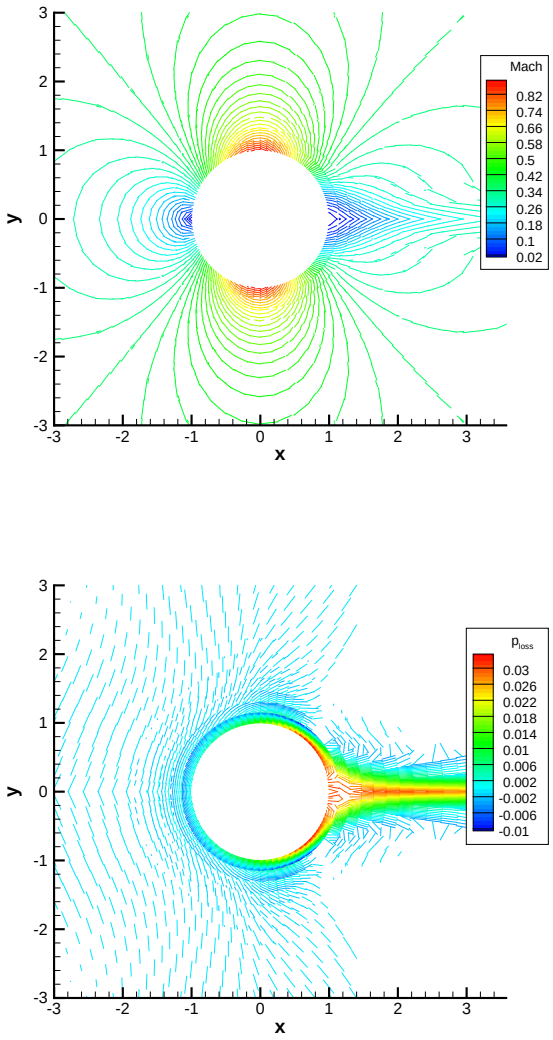

Figure 3.27: Mach number and pressure loss for $M = 0.38$ subsonic flow around a cylinder of radius $1\,m$ and using a 96×72 boundary conforming mesh with outer radius $56.6\,m$.

[9] are observed, most likely because of the difference in element shapes, triangles in [9] versus quadrilaterals here. The results using the boundary conforming mesh are somewhat better than those with the cut-cell mesh. The efficiency of the simulation with the cut-cell mesh can be improved by regular h-adaptation of the background mesh near the cylinder boundary.

3.9 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 [30, 42, 67, 102] as illustrated in Figure 3.28. 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 $\gamma_I = 1.4, R_I = 287.0\,J/kgK$ and $\gamma_H = 1.67, R_H = 2080.0\,J/kgK$. Initially the helium volume is a cylinder with a radius $0.025\,m$ and is located at $(x, y) = (0\,m, 0\,m)$ while the shock is located at $x = 0.055625\,m$. The domain has dimensions $[-0.11125\,m, 0.11125\,m] \times [-0.0445\,m, 0.0445\,m]$. 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:

$$\begin{aligned}(\rho_B, u_B, v_B, p_B) &= (0.164062\ kg/m^3, 0\ m/s, 0\ m/s, 1.0 \times 10^5\ Pa)\\ (\rho_L, u_L, v_L, p_L) &= (1.18902\ kg/m^3, 0\ m/s, 0\ m/s, 1.0 \times 10^5\ Pa)\\ (\rho_R, u_R, v_R, p_R) &= (1.63652\ kg/m^3, -114.473\ m/s, 0\ m/s, 1.5698 \times 10^5\ Pa),\end{aligned} \tag{3.49}$$

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

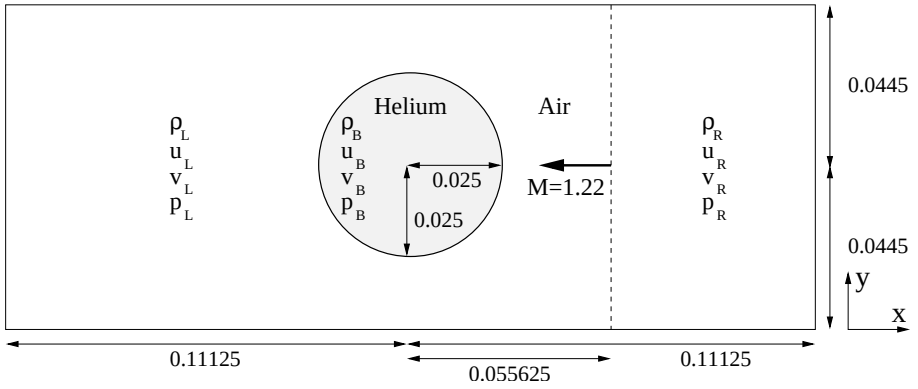

Figure 3.28: Helium cylinder - shock interaction test

sides of the shock wave are related through the Rankine-Hugoniot relations:

$$\begin{aligned}(\rho_R - \rho_L)V_S &= (\rho_R u_R - \rho_L u_L)\\ (\rho_R u_R - \rho_L u_L)V_S &= (\rho_R u_R^2 - \rho_L u_L^2) + (p_R - p_L)\\ (\rho_R E_R - \rho_L E_L)V_S &= u_R(\rho_R E_R + p_R) - u_L(\rho_L E_L + p_L).\end{aligned} \tag{3.50}$$

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.

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 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. Basically the cylindrical helium volume acts as a divergent lens for the shock

wave. 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 interface continuity of the normal velocity and the pressure is imposed and the numerical flux (3.34) 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 $[-0.11125\,m, 0.11125\,m] \times [0\,m, 0.0445\,m]$. The simulations are run using 40×8, 80×16, 160×32 and 320×64 elements from time $t = 0\,s$ to $3.125 \times 10^{-4}\,s$ at $CFL \approx 1.0$ using linear basis functions for the flow field and the level set, where the level set smoothing reconstructs a bilinear level set. By solving for a linear level set the Rayleigh-Taylor instability is effectively suppressed. Because the shock is not very strong the slope limiter is not used.

The density contours for subsequent times are shown in Figures 3.29 and 3.30. The mesh at time $t = 3.4375 \times 10^{-4}\,s$ for different mesh resolutions is shown in Figures 3.31 and 3.32. The evolution of helium mass over time for different mesh resolutions is shown in Figure 3.33 and is relatively small. The mesh evolution is illustrated for 80×16 elements in Figure 3.34.

In conclusion, the interaction between a helium cylinder and a shock wave was simulated using the interface flux (3.34). The mass loss was observed to be small.

3.10 Discussion

The space-time discontinuous Galerkin method with interface tracking has been applied to a number of one and two dimensional single-fluid and two-fluid test problems.

1. Using a one dimensional linear advection test it was observed that the flow solution has approxmate orders of accuracy of 2 for smooth initial conditions and 0.36 for discontinuous initial conditions, which

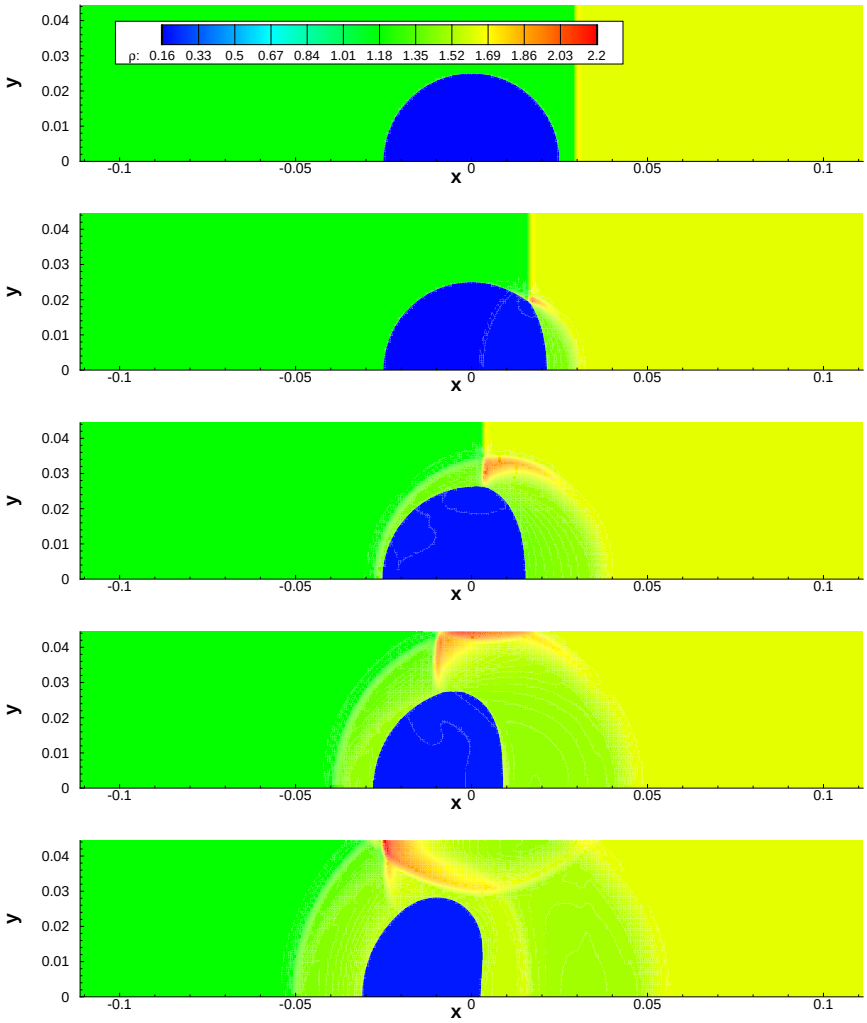

Figure 3.29: Density contours at times $t = 0.625 \times 10^{-4}\, s, 0.9375 \times 10^{-4}\, s, 1.25 \times 10^{-4}\, s, 1.5625 \times 10^{-4}\, s, 1.875 \times 10^{-4}\, s$ for the helium cylinder - ideal gas shock interaction test using 320×64 elements.

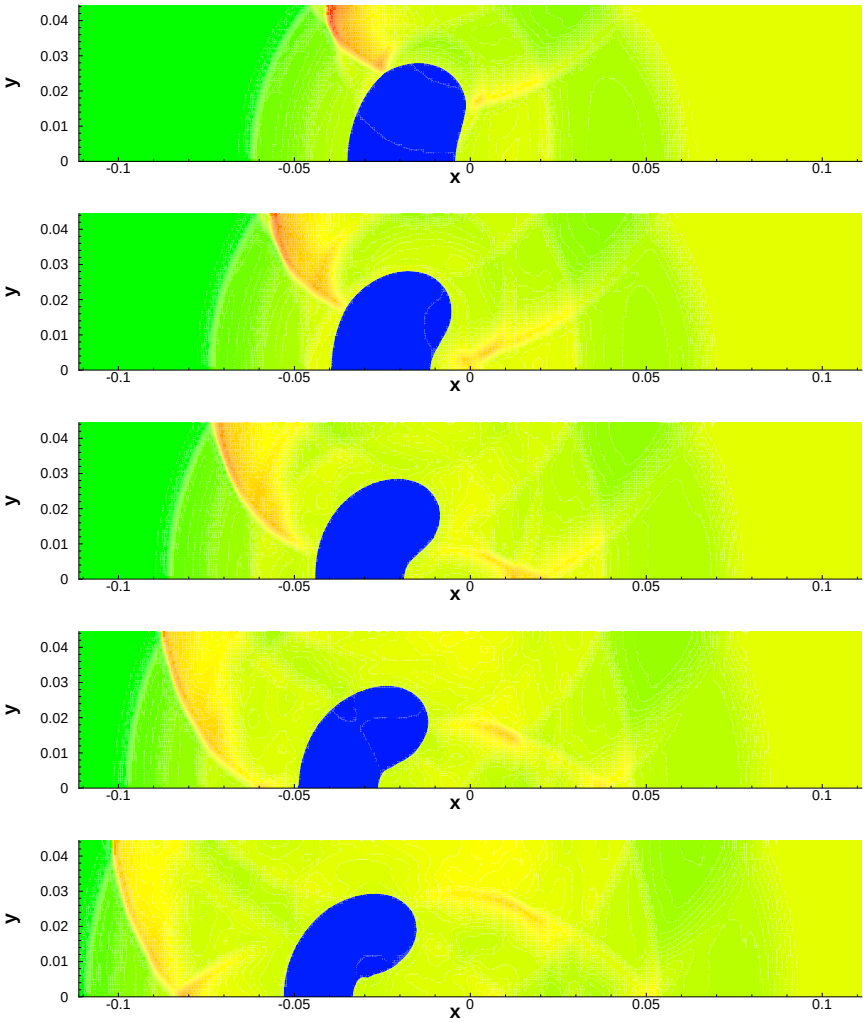

Figure 3.30: Density contours at times $t = 2.1875 \times 10^{-4}\,s, 2.5 \times 10^{-4}\,s, 2.8125 \times 10^{-4}\,s, 3.125 \times 10^{-4}\,s, 3.4375 \times 10^{-4}\,s$ for the helium cylinder - ideal gas shock interaction test using 320×64 elements.

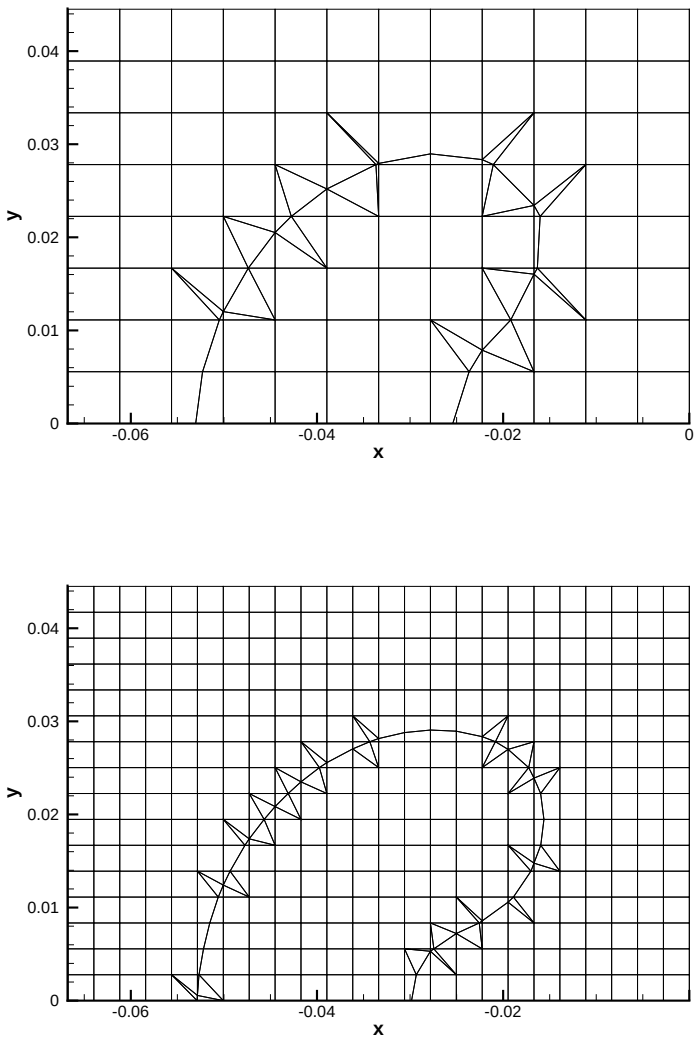

Figure 3.31: Mesh at time $3.4375 \times 10^{-4}\, s$ for the helium cylinder - ideal gas shock interaction test using 40×8 and 80×16 elements.

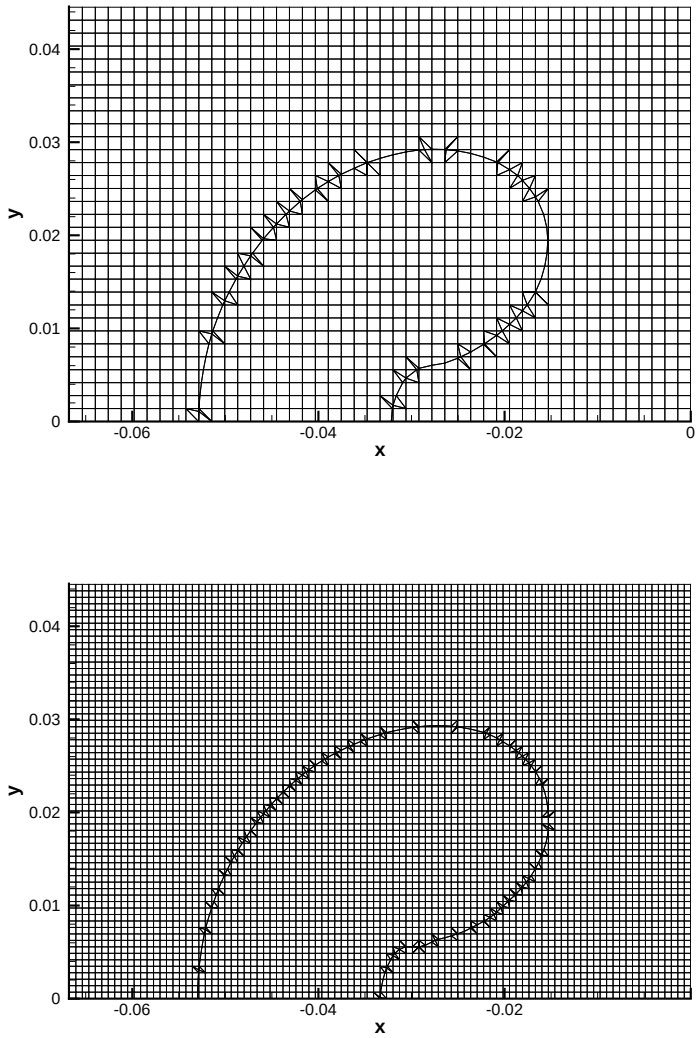

Figure 3.32: Mesh at time $3.4375 \times 10^{-4}\, s$ for the helium cylinder - ideal gas shock interaction test using 160×32 and 320×64 elements.

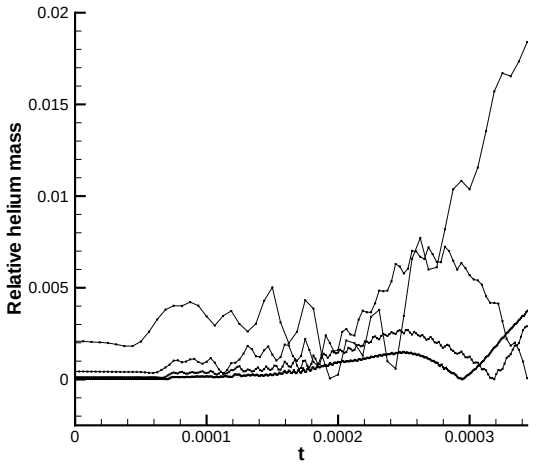

Figure 3.33: Relative helium mass over time for the helium cylinder - ideal gas shock interaction test using 40×8, 80×16, 160×32 and 320×64 elements. The relative mass is defined as $|M_e - M_h|/M_e$, with M_e the exact and M_h the numerical amount of mass.

matched theoretical orders of accuracy obtained in various studies. For the discontinuous solution, results improved when interface tracking was applied, because the interface could be captured exactly by mesh refinement. For a non-constant advection velocity it was observed that the interface tracking works quite well in combination with solid wall interface conditions.

2. The level set accuracy was tested using Zalesak's test. The shape of the disc after one rotation was preserved well at smooth area's of the disc, while in the neighborhood of the sharp corners the accuracy clearly suffered.

3. The method was applied to a one dimensional ideal gas single-fluid Euler shock tube problem. Using a contact interface flux, oscillations were observed at the interface. An alternative interface flux (3.34)

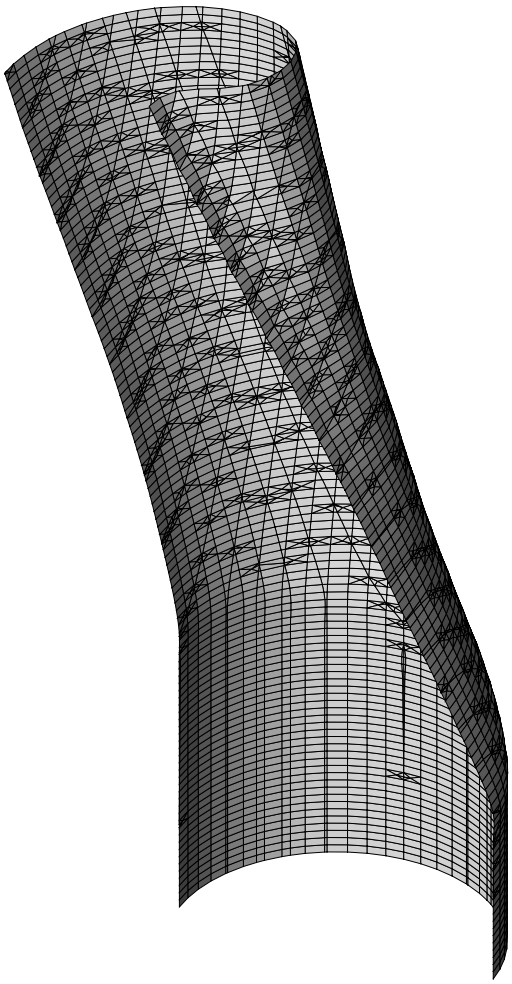

Figure 3.34: Interface evolution for the helium cylinder - ideal gas shock interaction test using 80×16 elements.

was developed, which reduces the oscillations at the interface at the cost of a very small mass conservation error. The interface flux (3.34) was tested with promising results. Slope limiting reduced the spikes in the solution but also caused a decrease in accuracy. Starting the simulation at $t = T/10$ greatly improved the results.

4. The method was applied to a magma - ideal gas shock tube. This test case featured two very different fluids and very high density and pressure ratio's. The method gave good results with the interface flux (3.34). Slope limiting reduced the spikes in the solution but also caused a decrease in accuracy. Starting the simulation at $t = T/10$ greatly improved the results.

5. The method was applied to subsonic flow around a cylinder at Mach 0.38. A comparison was made between the results using a boundary conforming mesh around the cylinder and a cut-cell mesh. The result matched those found in the literature. It was observed that for the cut-cell mesh *h*-refinement should be applied to the background mesh to increase performance.

6. The method was applied to calculate the interaction between a helium cylinder and a shock wave using the interface flux (3.34). The mass loss was observed to be small.

In general is was found that the level set deformation over time restricted the simulation lengths. The addition of a level set reinitialization procedure seems to be the most effective way to fix this problem. To improve the efficiency and stability of the method the incorporation of *hp*-refinement and a multigrid algorithm seems promising.

Chapter 4

Design and Implementation

4.1 Introduction

In Chapter 2 a new method for two-fluid flow computations was presented, which combines a space-time discontinuous Galerkin (STDG) finite element discretization with cut-cell based front tracking and a level set method. Since this method has a high degree of complexity its implementation is a non trivial task and therefore in this chapter issues related to the design and implementation of the two-fluid method will be addressed.

The outline of this chapter is as follows. In Section 4.2 aspects of the object oriented design of the method are presented. In Section 4.3 the mesh refinement implementation is discussed. In Section 4.4 the implementation of the method in *hp*GEM, a package for discontinuous Galerkin finite element methods, is discussed. Finally, in Section 4.5 some conclusions are presented.

4.2 Object oriented design

The two-fluid method has been designed and implemented using Object Oriented Programming (OOP). OOP is based on the use of self sufficient modules called objects and their interactions. Each object has a blueprint,

called a class, which defines the common attributes and methods of that type of object. The attributes are encapsulated in the object, which enhances the safety of the data. The methods define the operations performed by the object on its own data. Objects communicate with each other through the process of message passing. Abstraction allows for the definition of subclasses, which inherit the attributes and methods from their parent classes, and in addition can introduce their own. Polymorphism enables one common interface for many implementations, and for objects to act differently under different circumstances.

General advantages of OOP are reusability, reliability, robustness, extensibility and maintainability, each of which is of great importance when developing software with a high degree of complexity. For numerical methods OOP has the additional advantage that mathematical entities can be directly related to individual classes, which allows for a strong connection between the numerical method and its implementation. By using polymorphism a common interface can be defined for aspects such as problem dimension and geometry types.

The Unified Modeling Language (UML) is a standardized graphical modeling language which can be used for visualizing, specifying and constructing OOP software. The UML class diagram will be used to illustrate classes, their attributes and operations. The algorithms will be defined using pseudo code.

In the remainder of this section the OOP design for the two-fluid method will be discussed.

4.2.1 Mesh

The two-fluid method involves two meshes. The first mesh is a static background mesh for solving the level set equation. The second mesh is used for solving the two-fluid flow equations and is obtained by refining the background mesh where the level set is zero, to capture the interface between the two fluids. An instance of a refined mesh is shown in Figure 4.1. In the STDG numerical fluxes are usually conservative at each face and need to be computed only once. Hence, it is efficient to perform computations using

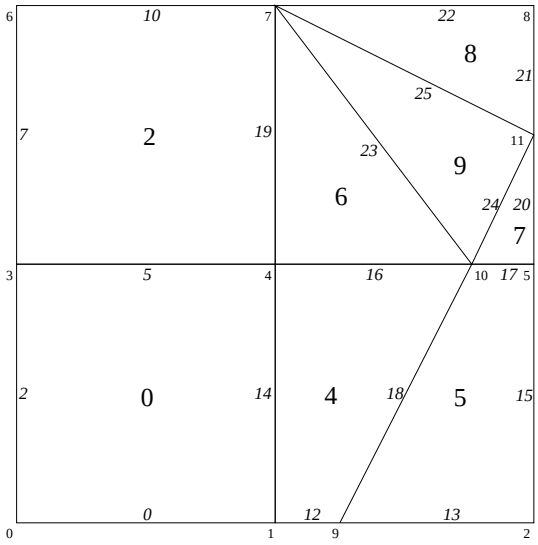

Figure 4.1: A two-fluid mesh showing the refinement around the interface. The numbering of the element, face and nodes is shown using larger to smaller font sizes.

two loops, one over elements and one over faces. Therefore, it is beneficial to use a mesh composed of elements, faces and nodes. Each mesh is stored in an object of class $\mathbf{Mesh^m}$, where the superscript $\mathbf{m} = \mathbf{b}, \mathbf{h}$ is used to denote background and refined meshes, respectively. The class $\mathbf{Mesh^m}$ holds containers for storing elements, faces and nodes where a container is defined as a holder object used to store collections of objects. The use of data containers allows for a separation of the various types of data related to a single object. The containers are defined by classes $\mathbf{ElementContainer^m}$, $\mathbf{FaceContainer^m}$ and $\mathbf{PhysicalNodeContainer^m}$.

The container class types are chosen based on the type of operations required. In addition to the type of operations it is also important at what position in the container the operations take place. The background and refined meshes are stored separately. An additional data structure stores the relation between the two meshes. This approach has the advantage

that the separate containers are kept simple. In addition, this separation will make it easier to incorporate h-refinement at a later stage.

The containers classes have been chosen as follows:

- The $\mathbf{ElementContainer^m}$ stores elements $\mathbf{Element^m[i], i = 0, \cdots, N_E^m}$, with $\mathbf{N_E^m}$ the number of elements. The required operations are (1) inserting, (2) deleting, (3) iterating over elements. The choice for the element container is a list.

- The $\mathbf{FaceContainer^m}$ stores faces $\mathbf{Face^m[j], j = 0, \cdots, N_F^m}$, with $\mathbf{N_F^m}$ the number of faces. The required operations are (1) inserting, (2) deleting and (3) iterating over faces. Hence, the choice for the face container is a list.

- The $\mathbf{PhysicalNodeContainer^m}$ stores physical nodes $\mathbf{PhysicalNode^m[k], k = 0, \cdots, N_N^m}$, with $\mathbf{N_N^m}$ the number of nodes. The required operations are (1) inserting, (2) deleting, (3) searching for nodes based on the nodal data (check if node exists) and (4) iterating over the nodes. However, since operations (1)-(3) are only required during mesh updates and are not performed very often, a simple list should be sufficient. Alternatively, a search optimized data structure like an ordered list may be used, with an ordering based on the nodal coordinates.

The relation between the background and refined meshes is implemented using a data structure $\mathbf{MeshHierarchy[N_E^b]}$. It connects the parent elements from the background mesh $\mathbf{ElementContainer^b[N_E^b]}$ to the respective child elements in the refined mesh $\mathbf{ElementContainer^h[N_E^h]}$. In Figure 4.2 the class diagrams for the mesh related classes are given.

4.2.2 Element

The class $\mathbf{Element^m[i]}$ defines a single element. It stores the following geometry related information:

- The geometry of the element in physical space, represented by an object of the class **PhysicalGeometry**.

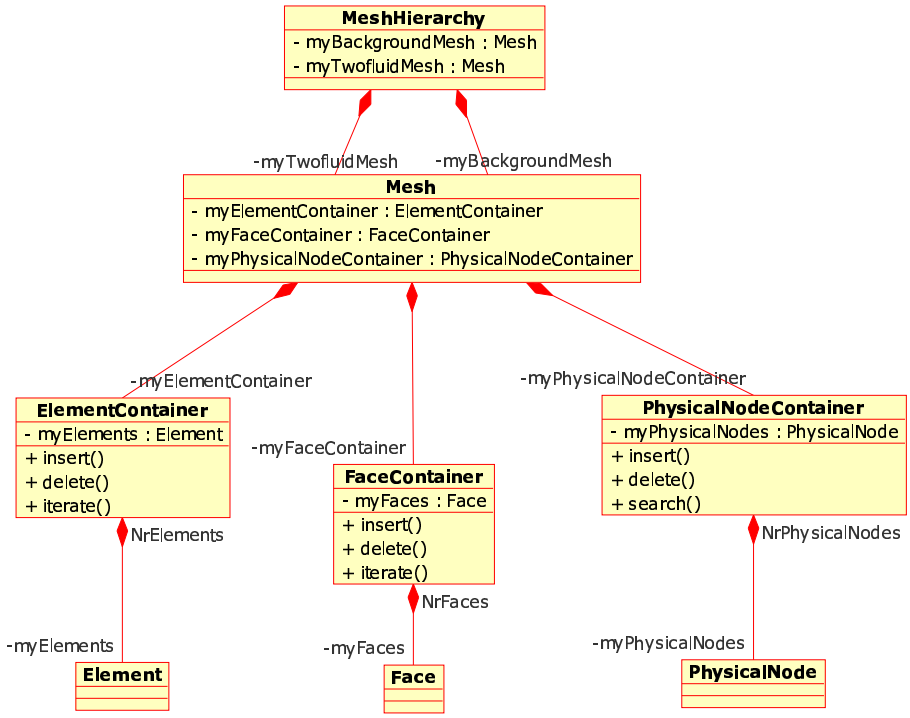

Figure 4.2: UML class diagram for the mesh classes. In the diagram every class is represented by a box that contains, from top to bottom, the class name (written in boldface), its attributes and its methods. The attributes are denoted by a - sign followed by the attribute's name, a : and the attribute's class. The methods are denoted by a + sign followed by the method's name. Method arguments have been omitted to simplify the diagram. The lines with diamonds denote aggregation relations between classes, which indicate that the class on the side of the diamond owns a number of objects of the other class type. The number of objects is either written at the right of the diamond or omitted, in the case when the number equals one.

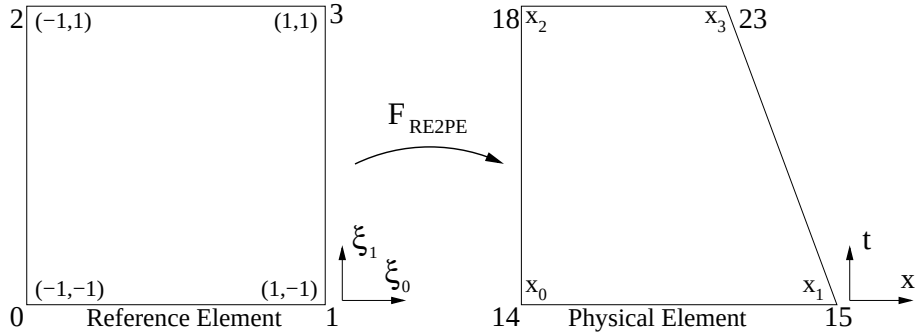

Figure 4.3: The reference and physical element geometries and the reference to physical element mapping F_{RE2PE}.

- The geometry of the element in reference space, represented by an object of the class **ReferenceGeometry**. The reference geometry is used to define data expansions on the element and is also used for integration.
- The mapping from the reference element onto the physical element, represented by an object of the class **ReferenceToPhysicalElementMapping**. This mapping is required for element and face integration and also for evaluations in physical space.
- A child to parent element mapping, represented by an object of the class **ReferenceChildToParentElementMapping**. This mapping is required for prolongating and restricting solutions between the background mesh and the refined mesh.

An instance of a physical geometry, reference geometry and reference element to physical element mapping are illustrated in Figure 4.3. The class **PhysicalGeometry** stores the following data:

- The global node indices (GNI) of vertices of the physical geometry, stored using an array $\mathbf{GNI}[\mathbf{N}_\mathbf{N}^{\mathbf{PG}}]$, with $\mathbf{N}_\mathbf{N}^{\mathbf{PG}}$ the number of nodes.

Each GNI corresponds to a physical point stored in the node container. It is efficient to use GNI's and a separate node container because nodes are typically shared by multiple elements.

The class **ReferenceGeometry** stores the following data:

- The local node indices (LNI) of the vertices of the reference geometry, stored using an array $\mathbf{LNI}[\mathbf{N}_{\mathbf{N}}^{\mathbf{RG}}]$, with $\mathbf{N}_{\mathbf{N}}^{\mathbf{RG}}$ the number of nodes of the reference geometry.

- The nodal data corresponding to each LNI, stored in a node container $\mathbf{ReferenceNodeContainer}[\mathbf{N}_{\mathbf{N}}^{\mathbf{ref}}]$.

Typically only a few reference geometries are defined, each of which is shared by a number of elements. Currently, in two space-time dimensions, the two-fluid method requires triangle and quadrilateral reference geometries and in three space-time dimensions, the two-fluid method requires simplex, pyramid, prism and hexahedron reference geometries. The class diagram for the element related classes is given in Figure 4.4. The mapping from the reference onto the physical element, as represented by the class **ReferenceToPhysicalElementMapping** is easily constructed from the coordinates of the vertices in reference and physical space. In Figure 4.5 the element mappings used in the two-fluid method are illustrated.

The class $\mathbf{Element}^{\mathbf{m}}[\mathbf{i}]$ also stores information related to the two-fluid refinement:

- Timeslab: $\mathbf{Timeslab} \in \{\mathbf{Old}, \mathbf{New}\}$. Often, the STDG is implemented using two timeslabs, one old and one new. The new timeslab contains all the elements currently used in computations, while the old timeslab contains the elements used in the previous time step.

- Fluid Type: $\mathbf{FluidType} \in \{\mathbf{Fluid1}, \mathbf{Fluid2}\}$. The fluid type is used to find out what variables are active and what equations need to be solved in the element and on the element faces. An interface is identified by unequal types of fluid in the left and right elements.

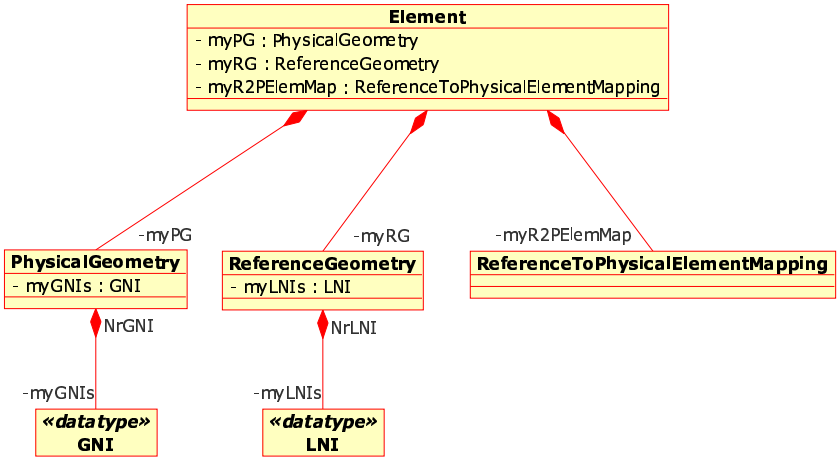

Figure 4.4: UML class diagram for the element class.

Finally, in addition to the information related to the element geometry and the two-fluid refinement, the class $\mathbf{Element^m[i]}$ stores a number of polynomial expansions for the test and trial functions. The expansions are defined on the reference elements and linked to basis functions which are also defined on the reference geometries:

- Basisfunctions: $\mathbf{BasisFunctions[N_{BF}]}$, with $\mathbf{N_{BF}}$ the number of basis functions.

The expansion coefficients are defined as:

- Flow Expansions: $\mathbf{FlowExpansionCoefs[N_{Flow}][N_{BF}]}$, with $\mathbf{N_{Flow}}$ the number of flow variables.

- Level Set Expansions: $\mathbf{LevelSetExpansionCoefs[N_{BF}]}$.

The expansions used in the method are shown in Figure 4.6.

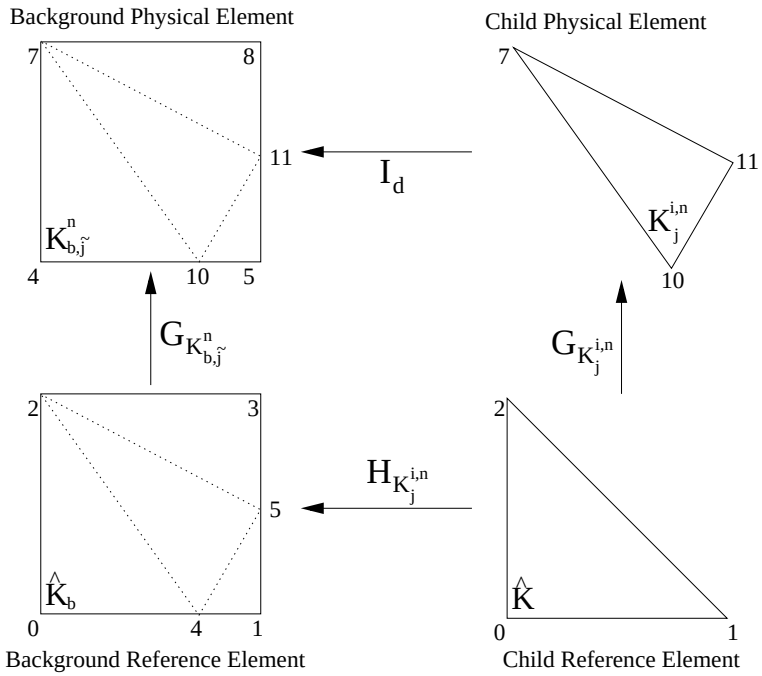

Figure 4.5: The element mappings in the two-fluid method. The mappings $G_{\mathcal{K}_{b,\tilde{j}}^n}$ and $G_{\mathcal{K}_j^{i,n}}$ map a reference element onto a physical element in the background and refined mesh, respectively. The mapping $H_{\mathcal{K}_j^{i,n}}$ maps the child reference element onto its parent reference element. The mapping may be defined either explicitly or otherwise as a composition $H_{\mathcal{K}_j^{i,n}} = G^{-1}_{\mathcal{K}_{b,\tilde{j}}^n} \circ G_{\mathcal{K}_j^{i,n}}$.

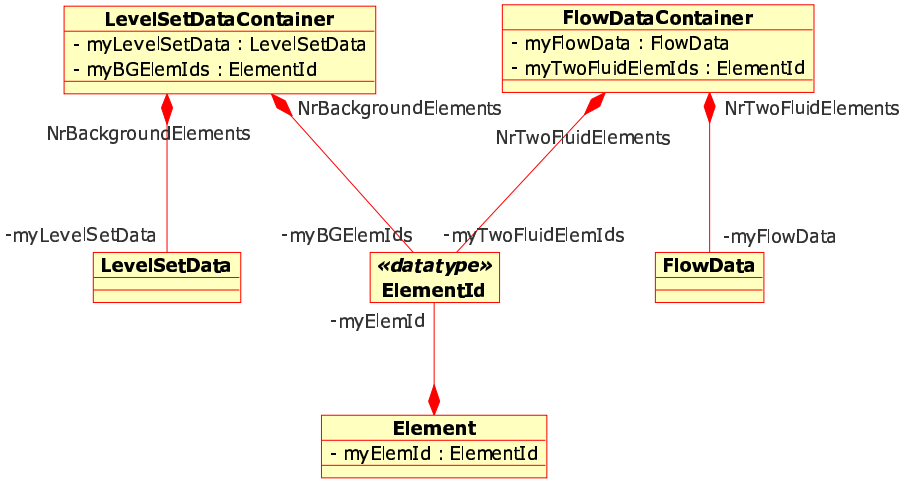

Figure 4.6: UML class diagram for the expansion classes. The level set and flow field expansions are defined on the reference background and refined elements, respectively.

4.2.3 Face

The class **Face**[**j**] defines a face which is either internal, connected to two neighboring elements, or a boundary face, connected to one element. It stores the following geometry related data:

- The geometry of the face in reference space, represented by an object of the class **FaceReferenceGeometry**. The face reference geometry is used in the face integration.
- The left and right element indices $\mathbf{Element_{Left}}, \mathbf{Element_{Right}}$ of the elements connected to the face.
- The left and right local face indices (LFI) $\mathbf{LFI_{Left}}, \mathbf{LFI_{Right}}$. The LFI indicates the location of the reference face in the reference element.
- Two reference face to reference element map-

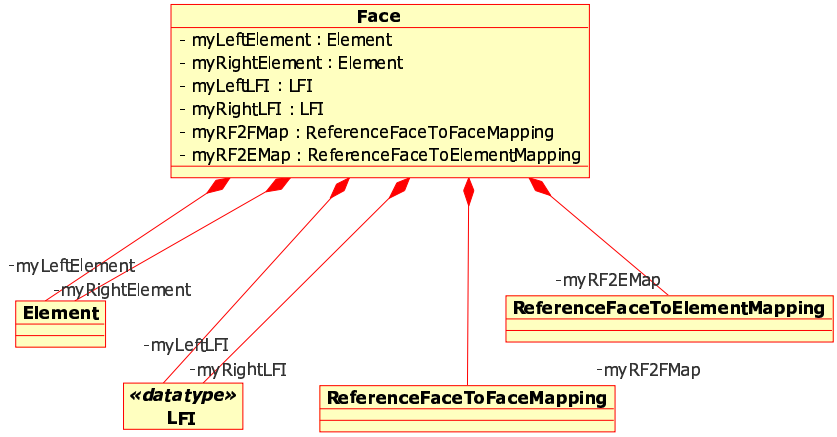

Figure 4.7: UML class diagram for the face class.

pings **ReferenceFaceToElementMapping**$_{Left}$ and **ReferenceFaceToElementMapping**$_{Right}$. These mappings are required for face integration when evaluations in the reference element are needed.

- The reference face to reference face mapping **ReferenceFaceToFaceMapping**. This mapping is required since the reference face orientation with respect to the left and the right elements can be different.

The class diagram for the face related classes is given in Figure 4.7 and the various face mappings are illustrated in Figure 4.8.

4.3 Mesh refinement

In this section the implementation of the two-fluid mesh refinement algorithm for two and three space-time dimensions presented in Section 2.3 will be discussed. Based on a continuous level set function ψ_c defined on

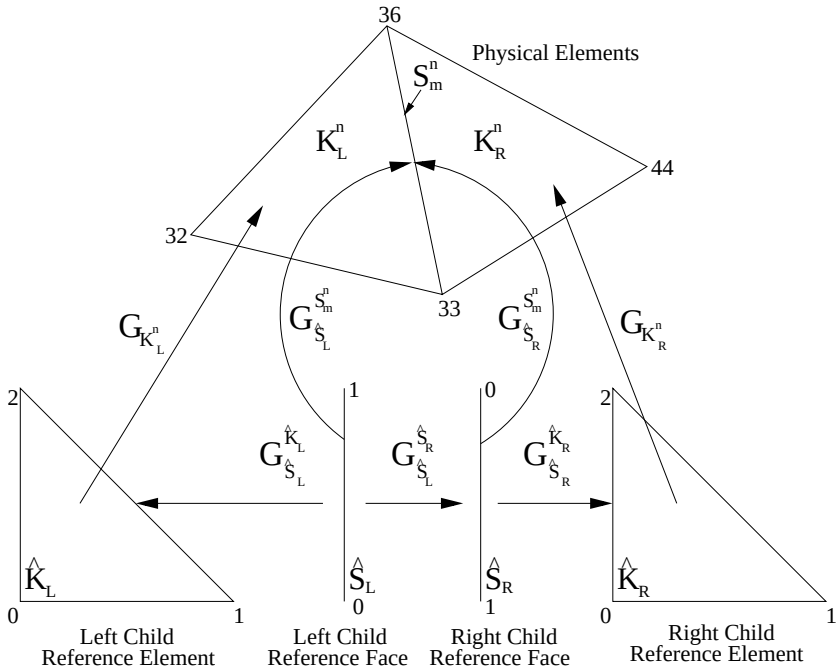

Figure 4.8: The face mappings in the STDG method. Here $K = L, R$ denote the left and right side with respect to the face. The mapping $G^{\mathcal{S}_m^n}_{\hat{\mathcal{S}}_K}$ reference face onto the physical face $\mathcal{S}_m^n$ with node indices $\{33, 36\}$. The mapping $G^{\hat{\mathcal{S}}_R}_{\hat{\mathcal{S}}_L}$ maps the left reference face $\hat{\mathcal{S}}_L$ onto the right reference face $\hat{\mathcal{S}}_R$. The mapping $G^{\hat{\mathcal{K}}_K}_{\hat{\mathcal{S}}}$ maps the reference face $\hat{\mathcal{S}}_K$ to the reference element $\hat{\mathcal{K}}_K$. The mapping $G_{\mathcal{K}_K^n}$ maps the reference element $\hat{\mathcal{K}}_K$ onto the physical element $\mathcal{K}_K^n$.

Algorithm 8 Mesh refinement algorithm.

1. FOR every $\mathbf{Element^b[i]}$ *in* $\mathbf{Mesh^b}$ *DO*
2. Calculate intersection of 0-level set ψ_c *with* $\mathbf{Element^b[i]}$*:*
1) interface reference nodes $\hat{\mathbf{N}}[\mathbf{N_N^{IF}}]$
2) local face indices $\mathbf{LFI}[\mathbf{N_N^{IF}}]$
3. Select $\mathbf{RefinementType}$ *based on* $\mathbf{LFI}[\mathbf{N_N^{IF}}]$
4. Create and store interface physical nodes $\mathbf{N_{IF}}[\mathbf{N_N^{IF}}]$
5. FOR all child elements $\mathbf{j}$ *defined by* $\mathbf{RefinementType}$ *DO*
6. Create $\mathbf{Element^h[j]}$
7. Store connectivity in $\mathbf{MeshHierarchy}[\mathbf{N_E^b}]$
8. Store $\mathbf{Element^h[j]}$ *in* $\mathbf{Mesh^h}$
9. END DO
10. END DO
11. Generate faces for $\mathbf{Mesh^h}$
12. FOR every element $\mathbf{Element^h[j]}$ *in* $\mathbf{Mesh^h}$ *DO*
13. Initialize data on $\mathbf{Element^h[j]}$
14. END DO

the background mesh, the algorithm creates a refined mesh on which the two-fluid flow computations can be performed.

In Algorithm 8 a reinterpretation of the mesh refinement algorithm defined in Algorithm 1 is given, in terms of the UML classes defined in Section 4.2.

The two-fluid element refinement is performed separately for each element. Given an element, first the intersection of the 0-level set with the element is calculated (Equation 2.19), from which is obtained a list of cut node coordinates defined in the elements reference space, and corresponding LFI's. The list of LFI's is used to find the type of the element refinement, and based on this type the element refinement rule is retrieved, which defines all the child elements in terms of the local node indices (LNI) of the parent element and of the cut nodes. To create the child elements only these LNI's and the corresponding nodal coordinates are required. The two-fluid

element refinement is illustrated in Figure 4.9. The use of refinement types and LNI's allows for a general definition of the two-fluid refinement, in the sense that given a background element only the cut node positions and refinement type have to be specified for the two-fluid refinement to be fully determined.

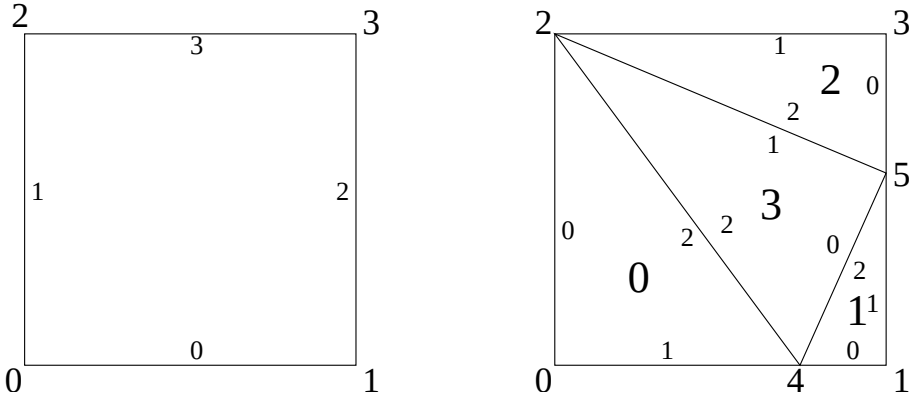

Figure 4.9: Two-fluid element refinement. On the left a background reference element is shown with nodes numbered with LNI's $0-3$ (normal fontsize), corresponding to the nodal coordinates $(-1,-1),(1,-1),(-1,1),(1,1)$, and faces numbered with LFI's $0-3$ (small fontsize). The background element is cut by an interface at the faces with LFI's 0 and 2 and the resulting refinement is shown on the right. By assigning the interface nodes $(0.4,-1.0),(1.0,0.1)$ the LNI's 4 and 5 the child elements $0-3$ (large fontsize) can be defined in terms of their LNI's as $\{0,4,2\},\{4,1,5\},\{5,3,2\}$ and $\{4,5,2\}$.

From the collection of child elements $\mathcal{K}_j^{i,n}$, the faces are generated as follows. First for every element a face descriptor is created of each face of that element, consisting of the element index, the local face index (LFI) and the GNI's of the face nodes, where the LFI is a number assigned to each face in the reference element. The face descriptors are stored in a container and sorted by the face node GNI's. The faces are constructed by looping over the ordered face descriptors, checking if subsequent descriptors describe the same face or not, and creating an internal or boundary face.

The two-fluid element refinement was developed for two and three space-

time dimensions using square and cube shaped background elements in Sections 2.3.4 and 2.3.5, respectively. The element refinement implementation uses of a classification of cut types based on the sign of the level set in the background element vertices. For this purpose the concept of a level set code was introduced, a binary number that stores the signs of the level set in all the vertices of the background element, where $-$ and $+$ correspond to 0 and 1, respectively. The implementation incorporates the data defined in these sections as follows:

1. Create the array **Perm**[**i**] which stores the permutations of the background element given in Tables 2.1 and 2.5 for two and three space-time dimensions, respectively. For each permutation the permuted LNI's of all the cube vertex LNI's are stored. Using **Perm**[**i**] an array of permutations of the edges **PermEdge**[**j**] is derived (Algorithm 2).

2. Create the array **BaseTypeBS**[**k**] which stores the base types in terms of their level set codes given in Tables 2.2 and 2.6 for two and three space-time dimensions, respectively.

3. Create for every base type **k** an array **elemRefType**[**k**] which stores for each element refinement rule the child elements in terms of their LNI's as defined in Tables 2.4 and 2.9, 2.10 for two and three space-time dimensions, respectively. The LNI's range from 0 to $N_v - 1$, corresponding to the N_v background element vertices, N_v to $N_v + N_e - 1$, corresponding to the interface cut nodes for each of the N_e edges of the cube, and $N_v + N_e$, for an additional node inside the element. The child element fluid types are stored in an array **elemRefTypeFluidSigns**[**k**], with values $0, 1$ denoting the two-fluid types and 2 denoting an ambiguous fluid type, which can occur when two interfaces are present in the element.

4. Create the array **permVec**[**l**] which stores for each level set code **l** the corresponding base type **k** and permutation **i** as defined in Tables 2.3 and 2.7 for two and three space-time dimensions, respectively. This array is created at creation of the refiner object, by looping over all

level set codes, applying each permutation and checking if it matches a base type using Algorithm 3. During computations Algorithm 4 is used. Given a level set code **l** the corresponding base type **k** and permutation **i** are given by **permVec**[**l**]. The LNI's of the child elements given by **elemRefType**[**k**] are permuted using **cubePerm**[**i**] for nodes 0 to N_v-1 and **cubePermEdge**[$\mathbf{i}-\mathbf{N_v}$] + $\mathbf{N_v}$ for nodes N_v to N_v+N_e-1. Node N_v+N_e is always permuted onto itself. The child element fluid type is directly found from **elemRefTypeFluidSigns**[**i**].

The two-fluid element refinement requires the following data:

- Interface cut face indices $\mathbf{LFI^{IF}}[\mathbf{N_N^{IF}}]$ and cut node positions $\mathbf{Nodes^{IF}}[\mathbf{N_N^{IF}}]$, where $\mathbf{N_N^{IF}}$ denotes the number of interface nodes. The interface cut face indices are used to select a refiner type.

- Child element definition $\mathbf{LNI_l}[\mathbf{N_{ChildNodes}}]$, $\mathbf{l}=\mathbf{0},\cdots,\mathbf{N_{ChildElememts}}$. Here $\mathbf{N_{ChildElements}}$ denotes the number of child elements and $\mathbf{N_{ChildNodes}}$ denotes the number of local node indices defining child element l. The child elements are defined in terms of the local node indices of the parent element augmented with local node indices of the cut nodes.

The class diagram is given in Figure 4.10.

4.4 *hp*GEM

The process of implementing a DG method is very time consuming, due to the relatively high complexity of these methods. However, implementations will often have a common basis of basic and potentially re-usable DG functionality. This observation motivated the development of *hp*GEM [65], an object-oriented software package for DG methods.

The codes implementing the two-fluid method used for the simulations discussed in Chapter 3 were all created with help of the *hp*GEM package. The *hp*GEM package is implemented using the standardized and commonly used OOP language C++ and therfore shares many of the benefits

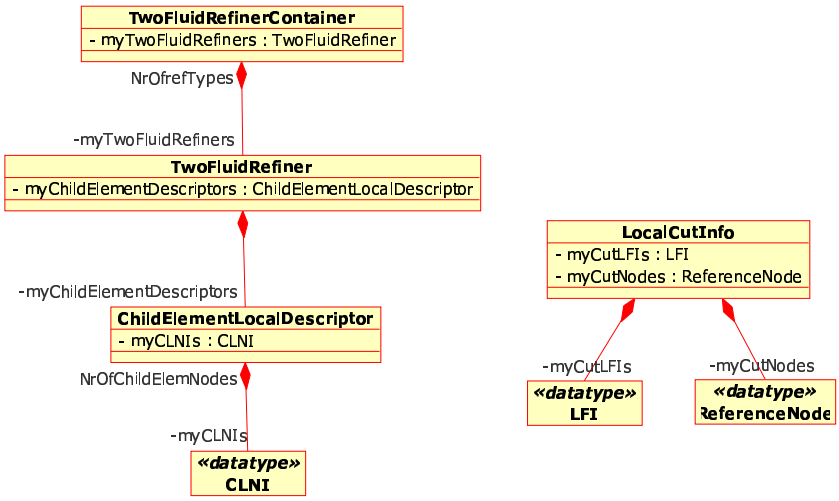

Figure 4.10: UML class diagram for the two-fluid refiner classes. The TwoFluidRefiner data is static and defines for each refinement type the child elements. The LocalCutInfo class is used during runtime for selecting a refiner and temporary storing the local cut nodal data. It stores the cut node positions and the local face indices.

and drawbacks of this programming language. Polymorphism is featured through the use of templates, which allow functions and classes to operate with generic types. C++ templates add flexibility and because these are instantiated at compile-time and after instantiation the resulting code is equivalent to code written specifically for the passed arguments. Encapsulation is implemented by allowing all members of a class to be declared as either public, private, or protected, which allows for safer handling of data. Namespaces are used to prohibit naming collisions and help make the code structure more transparent. Also, C++ incorporates the Standard Template Library (STL) which provides a number of container classes for often used storage structures and access in an uniform manner by means of iterators. The use of the *hp*GEM package allows for a reduction of development time and provides quality control. In addition, *hp*GEM also provides a

coding standard benefitting code sharing and maintenance.

In order to implement the method using the *hp*GEM package, a separation was made between parts that could be handled by *hp*GEM and parts that needed to be added. The *hp*GEM package can handle basic DG functionality available from the following subpackages:

- Geometry: Mesh generation, element and face factories, element, face and node containers, physical and reference geometries and mappings.
- Base: Data storage on elements and faces, expansions, basis functions.
- Integration: Element and face integration including numerous integration rules.
- GlobalAssembly: Global assembly.
- Output: Output in Tecplot format.

The parts which needed to be added were mainly problem specific, consisting of a main program, classes for the flow and level set variables, initial and boundary conditions, element and face integrands, numerical fluxes and the Runge-Kutta solver. In addition all classes related to the two-fluid refinement had to be added. The problem specific parts were encoded using the DG classes available from the *hp*GEM package as much as possible.

4.5 Discussion

In this chapter the object oriented programming (OOP) design and implementation of the two-fluid method in *hp*GEM, an OOP package for DG methods, were discussed. The choice for the OOP language C++ was motivated by the general advantages of OOP such as reusability, reliability, robustness, extensibility and maintainability. In addition the use of OOP allowed for a strong connection between the numerical method and its implementation. The use of *hp*GEM allowed for a reduction of the development time and provided quality control and a coding standard benefitting sharing and maintenance of the codes.

Chapter 5

Conclusions and Further Research

5.1 Conclusions

A novel numerical method for two-fluid flows has been presented. Here the results of Chapters 2, 3 and 4 will be discussed in the contex of the research objectives posed in the introduction.

1. The first research objective was to investigate and develop a discontinuous Galerkin method which could improve upon existing methods for fluid flows with interfaces such as front tracking and front capturing methods. This objective was addressed in Chapter 2 where a novel numerical method for two-fluid flow computations was presented, which combines the space-time discontinuous Galerkin (STDG) finite element discretization with the level set method and cut-cell based front 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. For the interface handling a front tracking approach was chosen because front tracking methods are capable of high accuracy. It was chosen to track the interface using

cut-cell mesh refinement. Because this type of refinement is very local in nature it combines well with the STGD. To compute the interface dynamics the level set method (LSM) was chosen, because of its ability to deal with merging and breakup, since it was expected that the LSM combines well with the cut-cell mesh refinement and also because the LSM is easy to extend to higher dimensions. The small cell problem caused by the cut-cell refinement was solved by using a merging procedure involving bounding box elements, which improved stability and performance of the method. The interface conditions could be 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. All possible cuts the 0-level set could make with square and cube shaped background elements were identified and for each cut an element refinement was defined explicitly. To ensure connectivity of the refined mesh, the *dim*-dimensional face refinements were defined equal to the $dim - 1$-dimensional element refinements. It is expected that the scheme can accurately solve smaller scale problems where the interface shape is of importance and where complex interface physics are involved.

2. The second research objective was to investigate the numerical properties and performance of this method for model and real life problems. This objective was addressed in Chapter 3 where the method was applied to a number of one and two dimensional single-fluid and two-fluid test problems. The order of accuracy of the flow STDG discretization was checked for a number of 1D linear advection tests, and found to be approximately 2 for continuous and 0.36 for discontinuous initial conditions. In order to test the level set accuracy the Zalesak disc test problem was examined and it was found that that the linear approximation of the level set gave a decrease in accuracy near sharp corners. The accuracy of the method was examined for a single-fluid ideal gas shocktube problem and a magma-air shocktube problem featuring high density and pressure ratio's. To remove flow oscillations near the interface a novel interface flux was developed,

based on the HLLC flux for a contact discontinuity, which could compensate for small errors in the interface position by allowing for a small mass loss. The resulting interface flux could handle interface discontinuities well. For both tests orders of accuracy of about 0.5 for discontinuous flow solutions were found. Slope limiting was found to reduce the spikes in the solution well but at the cost of a decrease in accuracy. Next, the performance of the method was checked using cut-cell and boundary wrapped meshes, by computing the subsonic flow around a cylinder at a subsonic Mach number of 0.38. It was found that the results were comparable. It was observed that for the cut-cell mesh h-refinement should be applied to the background mesh to increase performance. Lastly, the method was applied to calculate the interaction between a helium cylinder and a shock wave. It was found that the level set deformation restricted the simulation lengths. The addition of a level set reinitialization procedure seems to be the most effective way to fix this problem. To improve the efficiency and stability of the method the incorporation of hp-refinement and a multigrid algorithm seems promising.

3. The third research objective was to investigate the design and implementation aspects for this method. This objective was addressed in Chapter 4 where the Object Oriented Programming (OOP) design and implementation of the two-fluid method were discussed. The choice for the OOP language C++ was motivated by the general advantages of OOP such as reusability, reliability, robustness, extensibility and maintainability. In addition the use of OOP allowed for a strong connection between the numerical method and its implementation. In addition, *hp*GEM, an OOP package for DG methods was presented. The use of *hp*GEM allowed for a reduction of the development time and provided quality control and a coding standard benefitting sharing and maintenance of the codes.

5.2 Further research

The following recommendations are made for further research:

- The strategy to define two-fluid elements can be generalized such that it extends to four dimensions. Given the degree of generalization in the current algorithm this appears to be possible although it will be nontrivial.
- More advanced level set techniques can be implemented to reduce level set deformation over time and improve the overall accuracy of the method. This includes the narrow band approach, in which the level set is solved only locally around the interface, level set reinitialization and also the advection of the level set with the interface velocity, which can be achieved by means of an extention velocity. For further information, see [80].
- The incorporation of h-Refinement and/or multigrid algorithms to improve the performance of the method.

Some interesting applications involve:

- The shallow water equations to simulate flooding and drying [5, 6, 15, 70, 86] two-phase flows [72] and other applications [3, 87, 101]. Because of the methods' flexibility in defining flow domains with interfaces it is expected that valuable contributions are possible in these fields.
- The well known Rayleigh–Taylor and Kelvin-Helmholtz instability tests, which are interesting because they feature extreme interface deformation.
- Interface related phenomena with the method including interface curvature, tension and contamination, chemical reactions and membranes, which play a role in many real life two-fluid problems.

Bibliography

[1] Adalsteinsson, D. & Sethian, J.A. 1995 A fast level set method for propagating interfaces, *J. Comp. Phys.* 118, **269–277**.

[2] Ahn, H.T., Shashkov, M. 2009 Adaptive moment-of-fluid method, *J. Comp. Phys.* 228, **2792–2821**.

[3] Akers, B., Bokhove, O. 2008 Hydraulic Flow through a Channel Contraction: Multiple Steady States, *Phys. Fluids*, 20, 056601.

[4] Almgren, A.S., Bell, J.B., Collela, P. & Marthaler, T. 1997 A Cartesian mesh projection method for the incompressible Euler equations in complex geometries, *SIAM J. Sci. Comp.* 18(5), **1289–1309**.

[5] Ambati, V.R. 2006 Flooding and drying in discontinuous Galerkin discretizations of shallow water equations, ECCOMAS Egmond aan zee, European Conference on Computational Fluid Dynamics. http://proceedings.fyper.com/eccomascfd2006/.

[6] Ambati, V.R. and Bokhove, O. 2007 Space-time discontinuous Galerkin discretization of rotating shallow water equations, *J. Comp. Phys.*, 225(2), **1233-1261**.

[7] Ambati, V.R. and Bokhove, O. 2007 Space-time discontinuous Galerkin finite element method for shallow water flows, *J. Comp. Appl. Math.*, 204(2), **452-462**.

[8] Atkins, H.L. & Shu, C.-W. 1998 Quadrature-free implementation of discontinuous Galerkin methods for hyperbolic equations, *AIAA J.* 36, **775–782**.

[9] Bassi, F. & Rebay, S. 1997 High-order accurate discontinuous finite element solution of the 2D Euler equations, *J. Comp. Phys*, 138, **251–285**.

[10] Bassi, F. & Rebay, S. 1997 A high-order accurate discontinuous finite element method for the numerical solution of the compressible Navier-Stokes equations, *J. Comp. Phys.* 131, **267–279**.

[11] Biswas, R., Devine, K.D. & Flaherty, J. 1994 Parallel, adaptive finite element methods for conservation laws, *Appl. Numer. Math.* 14, **255-283**.

[12] Bokhove, O. 2001 Numerical modeling of magma-repository interactions, University of Twente, 97 pp, http://eprints.eemcs.utwente.nl/.

[13] Bokhove, O. 2002 Decompressie van magma in opslagtunnels, *Nederlands Tijdschrift voor Natuurkunde* 68, **232–235**, English version: http://eprints.eemcs.utwente.nl/.

[14] Bokhove, O., Woods, A.W., & de Boer, A. 2005 Magma Flow through Elastic-Walled Dikes, *Theor. Comp. Fluid Dyn.* 19, **261–286**.

[15] Bokhove, O. 2005 Flooding and drying in finite-element Galerkin discretizations of shallow-water equations. Part I: One dimension, *J. Sci. Comp.* 22, **47-82**.

[16] Causon, D.M., Ingram, D.M., Mingham, C.G., Yang, G. & Pearson, R.V. 2000 Calculation of shallow water flows using a Cartesian cut element approach, *Adv. Water Resour.* 23, **545–562**.

[17] Cockburn, B. & Shu, C.W. 1989 TVB Runge-Kutta local projection discontinuous Galerkin finite element method for scalar conservation laws II: General framework, *Math. Comp.* 52, **411-435**.

[18] Cockburn, B., Lin, S.Y. & Shu, C.W. 1989 TVB Runge-Kutta local projection discontinuous Galerkin finite element method for scalar conservation laws III: One dimensional systems, *J. Comp. Phys.* 84, **90–113**.

[19] Cockburn, B., Hou, S. & Shu, C.W. 1990 TVB Runge-Kutta local projection discontinuous Galerkin finite element method for scalar conservation laws IV: The multidimensional case, *Math. Comp.* 54, **545–581**.

[20] Cockburn, B. & Shu, C.W. 1998 The Runge-Kutta discontinuous Galerkin finite element method for conservation laws V: Multidimensional systems, *J. Comp. Phys.* 141, **199–224**.

[21] Cockburn, B. & Shu, C.W. 1998 The local discontinuous Galerkin method for time-dependent convection-diffusion systems, *SIAM J. Numer. Anal.* 35(6), **2440–2463**.

[22] Cockburn, B., Karniadakis, G.E. & Shu, C.W. 2000 Discontinuous Galerkin methods theory, computation and applications, Lecture Notes in Computational Science and Engineering. Vol.11, Springer, Berlin.

[23] Cockburn, B. et al. 2002 Enhanced accuracy by post-processing for finite element methods for hyperbolic equations, *Math. Comp.* 72, **577–606**.

[24] Coirier, W.J. & Powell, K.G. 1995 An accuracy assessment of Cartesian-mesh approaches for Euler equations, *J. Comp. Phys.* 117, 121–131.

[25] Cuenot, B., Magnaudet, J. & Spennato, B. 1997 The effects of slightly soluble surfactants on the flow around a spherical bubble, *J. Fluid Mech.* 339, **25–53**.

[26] Daly, B.J. 1969 Numerical study of the effect of surface tension on interface instablility, *Phys. Fluids* 12, **1340**.

[27] Dandy, D.S. & Leal, L.G. 1989 Buoyancy-driven motion of a deformable drop through a quiescent liquid at intermediate Reynolds numbers, *J. Fluid Mech.* 339, **161–192**.

[28] De Zeeuw, D., Powell, K.G. 1993 An adaptively refined Cartesian mesh solver for the Euler equations, *J. Comp. Phys.* 104(1), **56–68**.

[29] Edwards, D.A., Brenner, H. & Wasan, D.T. 1991 *Interfacial processes and rheology,* Butterworth-Heineman, Stoneham, Reed publishing.

[30] Fedkiw, R.P. et al. 1999 A Non-Oscillatory Eulerian Apporach to Interface in Multimaterial Flows (The Ghost Fluid Method), *J. Comp. Phys.* 152, **457–492**.

[31] Fidkowski, K.J. & Darmofal, D.L. 2007 A triangular cut-cell adaptive method for high-order discretizations of the compressible Navier-Stokes equations, *J. Comp. Phys.* 225, **1653–1672**.

[32] Fritts, M.J., Cowley, W., Trease, H.E, eds. 1985 *The Free Lagrange Method,* Lect. Notes Phys. Vol. 238, Springer-Verlag, New York.

[33] Fukai, J., Shiiba, Y., Yamamoto, T., Miyatake, O. Poulikakos. D. et al. 1993 Wetting effects on the spreading of a liquid droplets colliding with a flat surface: experiment and modeling, *Phys. Fluids A.* 5, **2588–2599**.

[34] Fyfe, D.E., Oran, E.S. & Fritts, M.J. 1988 Surface tension and viscosity with Lagrangian hydrodynamics on a triangular mesh, *J. Comp. Phys.* 76, **349–384**.

[35] Glimm, J., Isaacson, E., Marchesin, D. & McBryan, O. 1981 Front tracking for hyperbolic systems, *Adv. Appl. Math.* 2, **91-119**.

[36] Glimm, J. & McBryan, O. 1985 A computational model for interfaces, *Adv. Appl. Math.* 6, **422–435**.

[37] Glimm, J., Grove, J.W., Li, X.L., Shyue, K.-M., Zhang, Q. & Zeng, Y. 1998 Three-dimensional front tracking, *SIAM J. Sci. Comp.* 19, **201–225**

[38] Glimm, J. et al. 1999 Simple Front Tracking, *Contemp. Math.* 238, **133–149**.

[39] Glimm, J. et al. 2003 Conservative front tracking with improved accuracy, *SIAM J. Num. Anal.* 41-5, **1926–1947**.

[40] Greaves, M.D. 2005 Simulation of viscous water column collapse using adapting hierarchial grids, *Int.J.Num.Meth.Fluids* 50, **693–711**.

[41] Greaves, D.M. 2006 Viscous wave interaction with structures using adapting quadtree grids and cartesian cut cells, ECCOMAS CFD 2006, Delft.

[42] Haas, J.-F. & Sturtevant, B. 1987 Interaction of weak shock waves with cylindrical and spherical gas inhomogeneities, *J.Fluid Mech.* 181, **41–76**.

[43] Harlow, F.H. & Welch, J.E. 1965 Numerical calculation of time-dependent viscous incompressible flow, *Phys. Fluids* 8, **2182**.

[44] Hirt, C.V. & Nichols, B.D. 1981 Volume of fluid (VOF) methods for the dynamics of free boundaries, *J. Comp. Phys* 39, **201–255**.

[45] Hu, C. & Shu, C.W.- 1998 A Discontinuous Galerkin Finite Element Method for Hamilton-Jacobi Equations, NASA/CR-1998-206903, Hampton.

[46] Hyman, J.M. 1984 Numerical methods for tracking interfaces, *Phys. D.* 12, **396–407**.

[47] Jaffre, J., Johnson, C. & Szepessy, A. 1995 Convergence of the discontinuous Galerkin finite element method for hyperbolic conservation laws, *Math.Models Meth.Appl.Sci.* 5, **367**.

[48] Johansen, H. & Colella, P. 1998 A Cartesian mesh embedded boundary method for Poisson's equation on irregular domains, *J. Comp. Phys.* 147(1), **60–85**.

[49] Klaij, C.M., van der Vegt, J.J.W. & van der Ven, H. 2006 Space-time discontinuous Galerkin method for the compressible Navier-Stokes equations, *J. Comp. Phys.* 217, **589–611**.

[50] Klaij, C.M., van der Vegt, J.J.W. & van der Ven, H. 2006 Pseudo-time stepping methods for space-time discontinuous Galerkin discretizations of the compressible Navier-Stokes equations, *J. Comp. Phys.* 219, **622–643**.

[51] Krivodonova, L., Xin, J., Remacle, J.-F., Chevaugeon, N. & Flaherty, J.E. 2004 Shock detection and limiting with discontinuous Galerkin methods for hyperbolic conservation laws, *Appl. Num. Math.* 48, **323–338**.

[52] Krivodonova, L. & Berger, M. 2006 High-order Accurate Implementation of Solid Wall Boundary Conditions in Curved Geometries, *J. of Comp. Physics*, Vol. 211, **492-512**.

[53] Kröner 1996 *Numerical schemes for conservation laws.*, Wiley und Teubner.

[54] Kucharik, M., Limpouch, J. & Liska, R. 2006 Laser plasma simulations by Arbitrary Lagrangian Eulerian method, *J. de Phys.*, IV, Vol. 133, **167–169**.

[55] LeVeque, R.J. & Shyue, K.-M. 1996 2-dimensional front tracking based on high resolution wave propagation methods, *J. Comp. Phys.* 123, **354–368**.

[56] Luo, H., Baum, J.D. & Lohner, R. 2007 A Hermite WENO-based limiter for discontinuous Galerkin method on unstructured grids, *J. Comp. Phys.* 225, **686–713**.

[57] Magnaudet, J., Rivero, M. & Fabre, J. 1995 Accelerated flows around a rigid sphere or a spherical bubble. Part I: Steady straining flow, *J. Fluid Mech.* 284, **97–136**.

[58] Moretti, G., Grossman, B. & Marconi, F. 1972 *A complete numerical technique for the calculation of three dimensional inviscid supersonic flow*, Rep. No. 72-192, American Institute for Aerodynamics and Astronautics, New York.

[59] Oden, J.T., Babuska, I. & Baumann, C.E. 1998 A discontinuous hp finite element method for diffusion problems, *J. Comp. Phys.* 146, **495–519**.

[60] Osher, S.J. & Sethian, J.A. 1988 Fronts propagating with curvature dependent speed: algorithms based on Hamilton-Jacobi formulations, *J. Comp. Phys.* 79, **12–49**.

[61] Osher, S. & Shu, C.W. 1991 High-order essentially nonoscillatory schemes for Hamilton-Jacobi equations, *SIAM J.Num.Anal.* 28, No. 4, **907–922**.

[62] Palaniappan, J., Haber, R.B. & Jerrard, R.L. 2004 A spacetime discontinuous Galerkin method for scalar conservation laws, *Comp. Meth. Appl. Mech. Eng.* 193, **3607–3631**.

[63] Pember, R.B., Bell, J.B., Collela, P., Crutchfield, W.Y. & Welcome, M.L. 1994 An adaptive Cartesian mesh method for unsteady compressible flow in irregular regions, *J. Comp. Phys.* 120, **278-304**.

[64] Peng, D. et al. 1999 A PDE-Based Fast Local Level Set Method, *J. Comp. Phys.* 155, **410–438**.

[65] Pesch, L., Bell, A., Sollie, W.E.H., Ambati, V.R., Bokhove, O. & van der Vegt, J.J.W. 2007 hpGEM- A software framework for discontinuous Galerkin finite element methods, *ACM Transactions on Mathematical Software*. 33(4).

[66] Puckett, E.G. & Saltzman, J.S. 1992 A 3D adaptive mesh refinement algorithm for interfacial gas dynamics, *Physica D.* 60, **84-93**.

[67] Qiu, J., Liu, T. & Khoo, B.C. 2008 Simulations of Compressible Two-Medium Flow by Runge-Kutta Discontinuous Galerkin Methods with the Ghost Fluid Method, *Commun. Comp. Phys.* 3-2, **479–504**.

[68] Quirk, J. 1994 An alternative to unstructured meshes for computing gas dynamic flows around arbitrarily complex two-dimensional bodies, *Comp. Fluids* 23, **125–142**.

[69] Reed, W.H. & Hill, T.R. 1973 *Triangular mesh methods for the neutron transport equation*, Los Alamos Scientific Labaratory report LA-UR-73-479, Los Alamos, NM.

[70] Remacle, J.-F. et al. 2006 An adaptive discretization of shallow-water equations based on discontinuous Galerkin methods, *Int. J. Numer. Meth. Fluids* 52, 8, **902–923**.

[71] Rhebergen, S., Bokhove, O. & Van der Vegt, J.J.W. 2008 Discontinuous Galerkin finite element methods for hyperbolic nonconservative partial differential equations, *J. Comp. Phys.* 227, **1887–1922**.

[72] Rhebergen, S., Bokhove, O. and van der Vegt, J.J.W. 2009 Discontinuous Galerkin finite element method for shallow two-phase flows *Comp. Methods Appl. Mech. Engrg.*, Vol.198, **819-830**.

[73] Richtmyer, R.D. & Morton, K.W. 1967 *Difference Methods for Initial-Value Problems,* Inter-science, New york.

[74] Rudman, J.M. 1997 Volume-tracking methods for interfacial flow calculations, *Int. J. Numer. Heat Fluid Flow* 24, **671–691**.

[75] Ryskin, G. & Leal, L.G. 1984 Numerical solution of free-boundary problems in fluid mechanics, Part 2: Buoyancy-driven motion of a gas bubble through a quiescent liquid, em J. Fluid Mech. 148, **19–36**.

[76] Saurell, R. & Abgrall, R. 1998 A simple method for compressible multifluid flows, *SIAM J. Sci. Comp.* 21, no 3,**1115-1145**.

[77] Scardovelli, R. & Zaleski, S. 1999 Direct numerical simulation of free-surface and interfacial flow, *Annu. Rev. Fluid Mech.* 31, **567–603**.

[78] Scriven, L.E. 1960 Dynamics of a fluid interface, *Chem. Eng. Sci.* 12, **98–108**.

[79] Sethian, J.A. 1996 *Level Set Methods,* Cambridge Univ.Press.

[80] Sethian, J.A. & Smereka, P. 2003 Level set methods for fluid interfaces, *Annu. Rev. Fluid Mech.* 35, **341–372**.

[81] Sollie, W.E.H., Bokhove, O. & Van der Vegt, J.J.W. 2009 Two Fluid Space-Time Discontinuous Galerkin Finite Element Method, Part I: Theory, *Submitted.*

[82] Sollie, W.E.H. & Van der Vegt, J.J.W. 2009 Two Fluid Space-Time Discontinuous Galerkin Finite Element Method, Part II: Applications, *Submitted.*

[83] Sudirham, J.J., van der Vegt, J.J.W. & van Damme, R.M.J. 2006 Space-time discontinuous Galerkin method for advection-diffusion problems on time-dependent domains, *Appl. Num. Math.* 56, **1491–1518**

[84] Sussman, M., Smereka, P. & Osher, S. 1994 A level set approach for computing solutions to incompressible two-phase flow, *J. Comp. Phys.* 114, **146–159**.

[85] Sussman, M. & Puckett, E.G. 2000 A coupled level set and volume-of-fluid method for computing 3D and axisymmetric incompressible two-phase flows, J. Comp. Phys. 162, **301–337**.

[86] Tassi P.A., Bokhove O. and Vionnet C. A. 2007 Space discontinuous Galerkin method for shallow water flows-kinetic and HLLC flux, and potential vorticity generation *Advances in Water Resources*, 30(4).

[87] Tassi, P., Rhebergen, S., Vionnet, C. and Bokhove,O. 2008 A discontinuous Galerkin finite element model for bed evolution under shallow flows *Comp. Methods Appl. Mech. Eng.* 197, **2930-2947**.

[88] Toro, E. F. 1997 *Riemann solvers and numerical methods for fluid dynamics.* Springer Verlag, Berlin.

[89] Tucker, P.G. & Pan, Z. 2000 A Cartesian cut element method for incompressible viscous flow, *Appl. Math. Modell.* 24, **591–606**.

[90] Tryggvason, G. & Unverdi, S.O. 1990 Computations of three-dimensional Rayleigh-Taylor instability, *Phys. Fluids A.* 5, **656–659**.

[91] Udaykumar, H.S., Kan, H.C., Shyy, W., Tran Son Tay, R. 1997 Multiphase dynamics in arbitrary geometries on fixed Cartesian meshes, *J. Comp. Phys.* 137(2), **366–405**.

[92] Udaykumar, H.S., Mittal, R., Rampunggoon, P. & Khanna, A. 2001 A sharp interface Cartesian mesh method for simulating flows with complex moving boundaries, *J. Comp. Phys.* 174(1), **345–380**.

[93] Udaykumar, H.S., Mittal, R. & Rampunggoon, P. 2002 Interface tracking finite volume method for complex solid-fluid interactions on fixed meshes, *Comm. Nummer. Meth. Eng.* 18(2), **89–97**.

[94] Ungor, A. & Sheffer, A. 2002 Pitching tents in space-time: mesh generation for discontinuous Galerkin method, *Int. J. Found. Comp. Sci.* 13, **201–221**.

[95] Unverdi, S.O. & Tryggvason, G. 1992 Computations of multi-fluid flows, *Phys. Fluids D* 60, **70–83**.

[96] Unverdi, S.O. & Tryggvason, G. 1992 A front-tracking method for viscous, incompressible multi-fluid flows, *J. Comp. Phys.* 100, **25–37**.

[97] van der Vegt, J.J.W. & van der Ven, H. 1998 Discontinuous Galerkin finite element method with anisotropic local grid refinement for inviscid compressible flows, *J. Comp. Phys.* 141, **46–77**.

[98] van der Vegt, J.J.W. & van der Ven, H. 2002 Space-time discontinuous Galerkin finite element method with dynamic mesh motion for Inviscid Compressible Flows, *J. Comp. Phys.* 182, **546–585**.

[99] van der Vegt, J.J.W. & van der Ven, H. 2002 Slip Flow Boundary Conditions in Discontinuous Galerkin Discretizations of the Euler Equations of Gas Dynamics, *WCCM V.*, Fifth World Congress on Computational Mechanics, Vienna.

[100] Van der Vegt, J.J.W. & Xu, Y. 2007 Space-time discontinuous Galerkin method for nonlinear water waves, *J. Comp. Phys.* 224, **17–39**.

[101] Vreman, A.W. et.al. 2007 Supercritical shallow granular flow through a contraction: experiment, theory and simulation. *J. Fluid Mech.* 578, **233-269**.

[102] Wackers, J. & Koren, B. 2005 A fully conservative model for compressible two-fluid flow, *Int. J. Numer. Meth. Fluids* 47, **1337–1343**.

[103] Woods, A. W., Sparks, S., Bokhove, O., LeJeune, A.-M., Connor, C. B. and Hill, B. E. 2002 Modeling magma-drift interaction at the proposed high-level radioactive waste repository at Yucca Mountain, Nevada, USA. *Geophys. Res. Lett.*, 29(13), **1641**.

[104] Woods, A.W., Bokhove, O., Boer de, A., Hill, B.E. 2006 Compressible magma flow in a two-dimensional elastic-walled dike. *Earth Planet. Sci. Lett.* 246, **241-250**.

[105] Yang, G., Causon, D.M., Ingram, D.M., Saunders, R. & Batten, P. 1997 A Cartesian cut element method for compressible flows. Part A: Static body problems, *Aeronaut. J.* 2, **47–56**.

[106] Yang, G., Causon, D.M., Ingram, D.M., Saunders, R. & Batten, P. 1997 A Cartesian cut element method for compressible flows. Part B: Moving body problems, *Aeronaut. J.* 2, **57–65**.

[107] Yang, G., Causon, D.M. & Ingram 2000 Calculation of compressible flows about complex moving geometries using a three-dimensional Cartesian cut element method, *Int. J. Numer. Meth. Fluids* 33, **1121–1151**.

[108] Ye, T., Mittal, R., Udaykumar, H.S. & Shyy, W. 1999 An accurate Cartesian mesh method for viscous incompressible flows with complex immersed boundaries, *J. Comp. Phys.* 156(2), **209–240**.

[109] Young, S.S., White, J.J., Clark, E.S. & Oyanagi, Y. 1980 A basic experimental study of sandwich injection moulding with sequential injection, *Pol. Eng. Sci.* 20 **798–804**.

[110] Zalesak, S.T. 1979 Fully multidimensional flux-corrected transport algorithms for fluids, *J. Comp. Phys.* 31, **335-362**.

Summary

Multifluid and multiphase flows involve combinations of fluids and interfaces which separate these. These flows are of importance in many natural and industrial processes including fluidized beds and bubble columns. Often the interface is not static but moves with the fluid flow velocity. Also, interface topological changes due to breakup and coalescence processes may occur. Solutions typically have a discontinuous character at the interface between different fluids because of curvature and surface tension 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.

The aim of this research project was to develop a discontinuous Galerkin method for two-fluid flows, which is accurate, versatile and can alleviate some of the problems commonly encountered with existing methods.

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. A front tracking approach is chosen because these methods ensure a sharp interface between the fluids are capable of high accuracy. The front tracking is incorporated by means of cut-cell mesh refinement, because this type of refinement is very local in nature and hence combines well with the STGD. To compute the interface dynamics the level set method (LSM) is chosen,

because of its ability to deal with merging and breakup, since it was expected that the LSM combines well with the cut-cell mesh refinement and also because the LSM is easy to extend to higher dimensions. The small cell problem caused by the cut-cell refinement is solved by using a merging procedure involving bounding box elements, which improves stability and performance of the method. 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. All possible cuts the 0-level set can make with square and cube shaped background elements are identified and for each cut an element refinement is defined explicitly. To ensure connectivity of the refined mesh, the dim-dimensional face refinements are defined equal to the $dim-1$-dimensional element refinements. It is expected that this scheme can accurately solve smaller scale problems where the interface shape is of importance and where complex interface physics are involved.

To investigate the numerical properties and performance of the numerical algorithm it is applied to a number of one and two dimensional single and two-fluid test problems, including a magma - ideal gas shocktube and a helium cylinder - shock wave interaction problem. To remove oscillations in the flow field near the interface a novel interface flux is presented, which is based on the HLLC flux for a contact discontinuity and can compensate for small errors in the interface position by allowing for a small mass loss. Slope limiting was found to reduce spikes in the solution at the cost of a decrease in accuracy. It was found that the level set deformation restricted the simulation lengths. This problem can be solved by adding a level set reinitialization procedure. To improve the efficiency and stability of the two-fluid numerical algorithm it is advised to incorporate *hp*-refinement and a multigrid algorithm.

Next, the Object Oriented Programming (OOP) design and implementation of the two-fluid method were discussed. The choice for the OOP language C++ was motivated by the general advantages of OOP such as reusability, reliability, robustness, extensibility and maintainability. In addition the use of OOP allowed for a strong connection between the numerical method and its implementation. In addition, *hp*GEM, an OOP package

for DG methods was presented. The use of *hp*GEM allowed for a reduction of the development time and provided quality control and a coding standard which benefitted the sharing and maintenance of the codes.

Samenvatting

Meer vloeistof en meer fase stromingen worden gekenmerkt door de aanwezigheid van meerdere vloeistoffen en/of gassen en interfaces welke deze van elkaar scheiden. Dit type stromingen is belangrijk in veel natuurlijke en industriële processen zoals bijvoorbeeld gefluïdiseerde bedden en bellenkolommen. De interface is meestal niet statisch maar beweegt met de vloeistof. Ook kan de interface topologie veranderen door breking of samenvoeging van interfaces. Grootheden zoals de dichtheid, tangentiële snelheid en druk zijn vaak discontinu over de interface vanwege oppervlakte krommings en -spannings effecten. Deze discontinuïteiten kunnen bovendien erg sterk zijn, zoals in het geval van vloeistof-gas stromingen. Additionele discontinuïteiten zoals schokgolven of contact discontinuïteiten kunnen ook een rol spelen.

Het doel van dit promotie onderzoek was het ontwikkelen van een discontinue Galerkin eindige elementen methode voor twee vloeistof en/of gas stromingen, welke nauwkeurig en breed toepasbaar is en welke een alternatief kan bieden voor gangbare methoden.

Een nieuwe methode voor twee vloeistof en/of gas stromingen wordt gepresenteerd, welke een ruimte-tijd discontinue Galerkin eindige elementen discretisatie combineert met de level set methode en cut-cell interface tracking. De ruimte-tijd discontinue Galerkin (RTDG) eindige elementen methode biedt een hoge nauwkeurigheid, kan goed omgaan met discontinuïteiten en heeft een erg lokaal rekenrooster, waardoor de methode relatief makkelijk te combineren is met lokale *hp*-verfijning. Er is gekozen voor een front tracking methode vanwege de scherpe interface voorstelling en de hoge

nauwkeurigheid van dit type methoden. De front tracking maakt gebruik van cut-cell verfijning, welke vanwege haar lokale natuur goed te combineren is met de RTDG. De interface dynamiek wordt berekend met behulp van de level set methode (LSM). De LSM kan omgaan met interface samenvoeging en breking en is makkelijk uit te breiden naar hogere dimensies. Een merging procedure wordt gebruikt om verfijnde elementen welke te klein zijn samen te voegen, wat de nauwkeurigheid en stabiliteit van de methode ten goede komt. De interface condities worden verwerkt in de numerieke flux op de interface. De RTDG discretisatie zorgt ervoor dat het schema conservatief is zolang als de numerieke fluxen conservatief zijn. Alle mogelijk doorsnijdingen van de 0-level set met vierkante en kubusvormige element zijn geïdentificeerd en voor elke type doorsnijding is een expliciete element verfijning bepaald. Om de connectiviteit van het verfijnde mesh te kunnen garanderen zijn de dim-dimensionale face verfijningen gelijk aan de dim-1-dimensionale element verfijningen. Verwacht wordt dat deze methode nauwkeurig kleinschalige problemen kan oplossen waar de interface vorm en complexe interface fysica belangrijk zijn.

Om haar numerieke eigenschappen en efficiëntie te kunnen bepalen is de numerieke methode gebruikt om een aantal meer vloeistof test problemen op te lossen in een en twee ruimte dimensies. Onder andere een magma-lucht schokbuis probleem en een helium cilinder - schok golf interactie probleem zijn opgelost. Om oscillaties in de stromings velden bij de interface te verwijderen wordt een alternatieve interface flux gepresenteerd welke gebaseerd is op de HLLC flux voor een contact discontinuïteit, en die kan compenseren voor kleine fouten in de interface positie door een klein massa verlies toe te staan. Om pieken in de oplossing te kunnen verwijderen is gebruik gemaakt van slope limiting, welke echter een verlies van nauwkeurigheid tot gevolg had. Er is gevonden dat de level set deformatie de maximale simulatie lengte begrensde, wat opgelost kan worden door het toevoegen van een level set re-initialisatie procedure. Om de efficiëntie van de methode te verbeteren wordt geadviseerd hp-verfijning en een multigrid algoritme toe te voegen.

Het objectgeoriënteerde programma (OOP) ontwerp en de implementatie van de methode zijn beschreven. De keuze voor de OOP taal C++

was gemotiveerd door de voordelen van OOP zoals herbruikbaarheid, betrouwbaarheid, robuustheid, uitbreidbaarheid en onderhoudbaarheid. Gebruik van de OOP resulteerde in een sterke connectie tussen de numerieke methode en de implementatie. Ook is *hp*GEM, een OOP pakket voor DG methoden gepresenteerd. Het gebruik van *hp*GEM bied een reductie van de implementatie tijd, kwaliteits controle en een programmerings standaard wat delen en onderhoud van de implementaties ten goed komt.

Acknowledgements

Without the help, support and encouragement of my friends, colleagues and family this thesis could not have been completed. Therefore I would like to express my sincerest gratitude to all of them here.

First and foremost, I would like to thank my supervisors prof. dr. ir. Jaap van der Vegt and dr. ir. Onno Bokhove. Thank you for giving me the opportunity to work in the Numerical Analysis and Computational Mechanics (NACM) group. I am very grateful for your involvement, interest, patience, motivational support and proofreading.

I would like to thank my colleagues Vijaya and Sander for their advice and contributions to this thesis.

I would like to thank the members and users of the hpGEM project, Lars, Alex, Vijaya, Sander, Jaap, Onno, Tito, Masoumeh, Shavarsh and Anthony.

I would like to thank my colleagues from the NACM group, Jaap, Onno, Ruud, Brigit, Bernhard, Mike, Chris, Federik, Ference, Olena, Wenny, Tito, Masoumeh, Shavarsh, Anthony, Julia, Lilya and David, in particular to my room mates, Sander, Bob, Ivan, Tim and Lie, the secretaries Marielle, Diana, Linda and Carin and all the other nice people I forgot about.

I would like to thank my team members and friends from Harambee for all the fun time spend playing volleyball.

I would like to thank Albert van Eijk for his support and help.

And last but not least, I would like to thank my parents, brother and

sister for their constant support and encouragement.

Henk Sollie, 2010.