Modelling, Analysis and Simulation

A Runge-Kutta discontinuous-Galerkin level-set method for unsteady compressible two-fluid flow

J. Naber

REPORT MAS-E0601 JANUARY 2006

ISSN 1386-3703

A Runge-Kutta discontinuous-Galerkin level-set method for unsteady compressible two-fluid flow

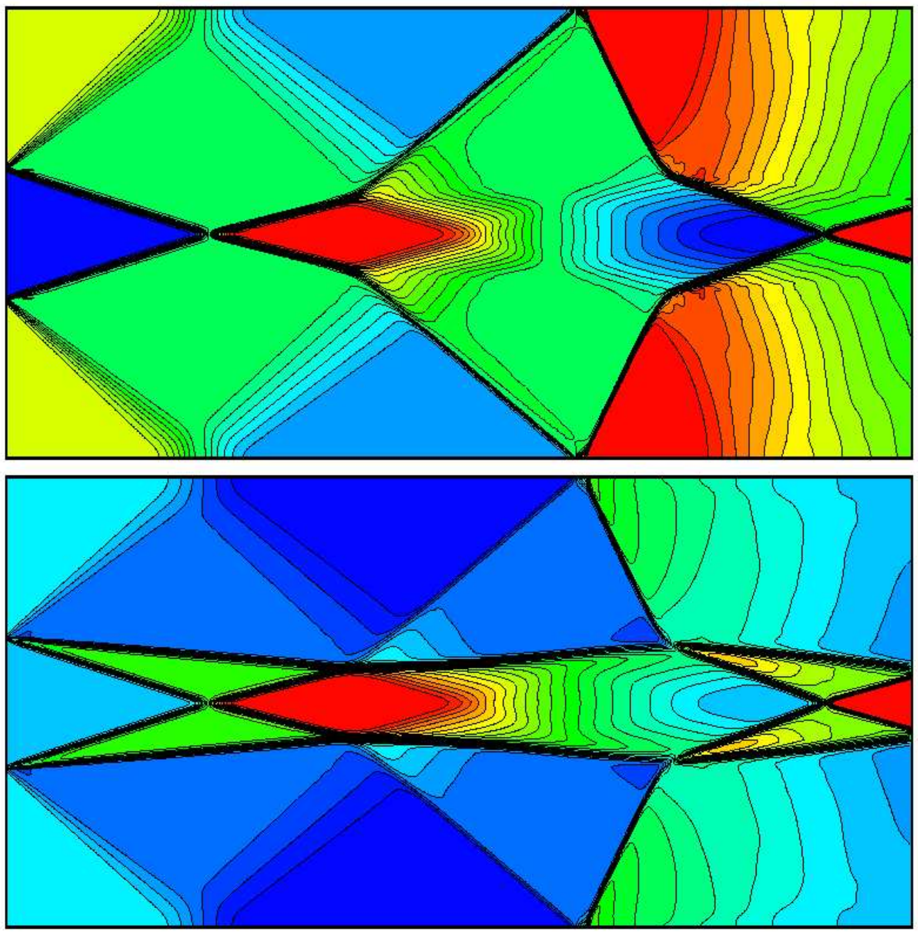
ABSTRACT
A typically interesting type of flow problem is that of flows involving multiple fluids. Especially two-fluid flows, where two non-mixing fluids are separated by a sharp fluid interface, find many applications in both engineering and physics. Although experimental and analytical results have provided us a solid foundation for two-fluid dynamics, it is the research area involved with numerically solving the governing flow equations, better known as *computational fluid dynamics* (CFD), that is presently leading the way in two-fluid research. Unfortunately, there is reason to believe that, in contrast to the simulation of single-fluid flows, there is still no sufficiently accurate and efficient simulation method available for general two-fluid flows. This is mainly due to difficulties that arise when treating the interface between the two fluids and the lack of accuracy of the numerical methods that can cope with these interface problems. It is therefore that this thesis governs the development of a highly accurate numerical solver for the simulation of compressible, unsteady and inviscid two-fluid flows described by the two-dimensional Euler equations of gas dynamics. The two-fluid flow solver that is developed in this thesis is based on the *level-set* (LS) method. This so-called *interface-tracking* method describes the evolution of the two-fluid interface in the flow and as such is able to distinguish between both fluids. The LS method has been applied to two-fluid flow simulation regularly and corresponding results have effectively shown its competence. The novelty of the solver developed here is the application of a highly accurate *Runge-Kutta discontinuous Galerkin* (RKDG) method for the temporal and spatial discretization of the governing equations. The possibility of obtaining very high orders of accuracy and the relatively easy implementation of mesh and order refinement (*hp*-refinement) techniques makes the RKDG method an attractive method for solving fluid flow problems. Applying a RKDG discretization to a two-fluid flow solver based on the LS method, combines the accuracy of the former with the efficiency and easy implementation of the latter, and as such results in an attractive numerical solver for two-fluid flows. Since the combination of a RKDG method for the Euler equations and a level-set method is a novelty, several aspects of both methods required further investigation. It is shown that the level-set equation has to be used in its advective form since this approach, as opposed to the frequently used conservative form, does not generate an erroneous off-set in the interface location. A simple fix is applied to prevent the solution from becoming oscillatory near the two-fluid interface. Application of this fix requires the development of a special two-fluid slope limiter for the RKDG method. Numerical results of several one- and two-dimensional problems show the competence of the developed method.

2000 Mathematics Subject Classification: 65M60,76N15,76T10
Keywords and Phrases: gas dynamics, compressible two-gas flows, level-set method, Runge-Kutta discontinuous Galerkin method, slope limiters
Note: Part of this research has been funded by the Dutch BSIK/BRICKS project.

A Runge-Kutta Discontinuous-Galerkin Level-Set Method for Unsteady Compressible Two-Fluid Flow

December 2005

Jorick Naber



Delft University of Technology

Centrum voor Wiskunde en Informatica

A Runge-Kutta Discontinuous-Galerkin Level-Set Method for Unsteady Compressible Two-Fluid Flow

Master of Science Thesis

Chair of Aerodynamics

Department of Aerospace Engineering

Delft University of Technology.

December 2005

by

Jorick Naber

born in Zeist, The Netherlands

This dissertation work is approved by the supervisor:

Prof. dr. ir. B. Koren

Composition of the exam committee:

Dr. ir. E.H. van Brummelen,	Delft University of Technology, Delft
Dr. ir. M.I. Gerritsma,	Delft University of Technology, Delft
Prof. dr. ir. B. Koren,	CWI, Amsterdam / Delft University of Technology, Delft
Dr. ir. C. Vuik,	Delft University of Technology, Delft
Prof. dr. B. van Leer	University of Michigan, Ann Arbor, USA (external reviewer)

Prof. dr. ir. B. Koren en Prof. dr. B. van Leer have contributed greatly to the realisation of this work.

Part of this research has been funded by the Dutch BSIK/BRICKS project.

Preface

The thesis presented here is the result of roughly one year of hard work at the Centre for Mathematics and Computer Science (CWI) in Amsterdam. This work is a continuation and extension of the research performed during my internship at the University of Michigan in Ann Arbor, which took place from August to December 2004. I am fully aware that this work would not have been the same without the support and advice of several people, to which I owe much gratitude.

First of all I would like to thank my supervisor Barry Koren for giving me the opportunity to perform my research under his supervision at CWI. His everlasting enthusiasm and profound knowledge have not only contributed greatly to the present work, but have also shaped me as a researcher and consequently paved the way for my future career. During this period Barry Koren provided a pleasant distraction by enabling me to participate as a teaching assistant in the course Computational Fluid and Solid Mechanics at the Delft University of Technology.

Further acknowledgment goes out to professor van Leer who enabled me to perform my internship in his research group at the University of Michigan. It was a great pleasure to work with a person that has contributed so greatly to our research area and possibly even more important, has so much joy sharing this with his students.

A few of the people that made my stay at CWI an interesting and pleasant one are: Reinout vander Meulen, Anton Kuut, David Echeverria, Domenico Lahaye, Jeroen Wackers, Marc van Raalte, Margreet Nool and many others. I am looking forward to working with you in the future.

Last but not least my gratitude goes out to those that had to cope with me when I was not working, my parents, brother, sister, friends and most of all Marloes. Thank you for making this year as worthwile as it was.

Amsterdam, December 2005,

Jorick Naber

Abstract

A typically interesting type of flow problem is that of flows involving multiple fluids. Especially two-fluid flows, where two non-mixing fluids are separated by a sharp fluid interface, find many applications in both engineering and physics. Although experimental and analytical results have provided us a solid foundation for two-fluid dynamics, it is the research area involved with numerically solving the governing flow equations, better known as *computational fluid dynamics* (CFD), that is presently leading the way in two-fluid research. Unfortunately, there is reason to believe that, in contrast to the simulation of single-fluid flows, there is still no sufficiently accurate and efficient simulation method available for general two-fluid flows. This is mainly due to difficulties that arise when treating the interface between the two fluids and the lack of accuracy of the numerical methods that can cope with these interface problems. It is therefore that this thesis governs the development of a highly accurate numerical solver for the simulation of compressible, unsteady and inviscid two-fluid flows described by the two-dimensional Euler equations of gas dynamics.

The two-fluid flow solver that is developed in this thesis is based on the *level-set* (LS) method. This so-called *interface-tracking* method describes the evolution of the two-fluid interface in the flow and as such is able to distinguish between both fluids. The LS method has been applied to two-fluid flow simulation regularly and corresponding results have effectively shown its competence. The novelty of the solver developed here is the application of a highly accurate *Runge-Kutta discontinuous Galerkin* (RKDG) method for the temporal and spatial discretization of the governing equations. The possibility of obtaining very high orders of accuracy and the relatively easy implementation of mesh and order refinement (hp-refinement) techniques makes the RKDG method an attractive method for solving fluid flow problems. Applying a RKDG discretization to a two-fluid flow solver based on the LS method, combines the accuracy of the former with the efficiency and easy implementation of the latter, and as such results in an attractive numerical solver for two-fluid flows.

Since the combination of a RKDG method for the Euler equations and a level-set method is a novelty, several aspects of both methods required further investigation. It is shown that the level-set equation has to be used in its advective form since this approach, as opposed to the frequently used conservative form, does not generate an erroneous off-set in the interface location. A simple fix is applied to prevent the solution from becoming oscillatory near the two-fluid interface. Application of this fix requires the development of a special two-fluid slope limiter for the RKDG method. Numerical results of several one- and two-dimensional problems show the competence of the developed method.

Contents

Chapter 1

Introduction

A typically interesting type of flow problem is that of flows involving multiple fluids. Especially two-fluid flows, where two non-mixing fluids are separated by a sharp fluid interface, find many applications in both engineering and physics. Jets exiting a nozzle into a different fluid, bubbles immersed in another fluid, air-water flows around a ship and the creation of complex two-fluid vortex patterns due to flow instabilities. These are only a few of the numerous examples of flows of multiple fluids as we encounter them in our everyday lives. It is because of man's desire to understand the dynamical processes that are involved in these types of fluid flows and the ability to use this knowledge for practical purposes, that made two-fluid flows the subject of elaborate research in recent years.

Although experimental and analytical results have provided us a solid foundation for two-fluid dynamics, it is the research area involved with numerically solving the governing flow equations, better known as *computational fluid dynamics* (CFD), that is presently leading the way in two-fluid research. The still increasing computational power of modern computers combined with the extensive knowledge of numerical methods gained in the past decades, makes the use of simulation techniques a realistic option for solving even the most complex flow problems.

Unfortunately, there is reason to believe that, in contrast to the simulation of single-fluid flows, there is still no sufficiently accurate and efficient simulation method available for general two-fluid flows. This is mainly due to difficulties that arise when treating the interface between the two fluids and the lack of accuracy of the numerical methods that can cope with these interface problems. It is therefore that this thesis governs the development of a highly accurate numerical solver for the simulation of compressible, unsteady and inviscid two-fluid flows described by the two-dimensional Euler equations of gas dynamics.

The two-fluid flow solver that is developed in this thesis is based on the *level-set* (LS) method. This so-called *interface-tracking* method describes the evolution of the two-fluid interface in the flow and as such is able to distinguish between both fluids. The LS method has been applied to two-fluid flow simulation regularly and corresponding results have effectively shown its competence. The novelty of the solver developed here is the application of a highly accurate *Runge-Kutta discontinuous Galerkin* (RKDG) method for the temporal and spatial discretization of the governing equations. The possibility of obtaining extremely high orders of accuracy and the relatively easy implementation of mesh and order refinement (hp-refinement) techniques makes the RKDG method an attractive method for solving fluid flow problems. Applying a RKDG discretization to a two-fluid flow solver based on the LS method, combines the accuracy of the former with the efficiency and easy implementation of the latter, and as such results in an attractive numerical solver for two-fluid flows.

1.1 The level-set method

Although many claim to be the first to have developed the level-set (LS) method, it is generally assumed that in 1964 Markstein [36] was the first to use a signed distance function, only later to be called the level-set function, to track the propagation of a flame front. A solid mathematical foundation for the LS method was developed by Osher and Sethian [41]. Together with Mulder they were also responsible for the application of this LS method to the Euler equations of gas dynamics, as described in [37]. Since then the LS method has found many applications ranging from two-fluid flow simulation, to structure optimization and even the enhancement of video images. For an excellent overview of the LS method see the books of Sethian [50], and Fedkiw and Osher [40].

The LS method is one of many interface methods. Interface methods for two-fluid flow simulation can generally be divided into three main classes; *interface-fitting*, *interface-capturing* and *interface-tracking methods*, where the division is based on how the interface is handled. A more general classification of interface methods divides them into *Lagrangian* and *Eulerian* methods. Interface-fitting belongs to the former class and interface-capturing and tracking to the latter. In the case of Lagrangian methods near the interface the flow equations are reduced to equations for the interface only, while in the case of Eulerian methods the flow itself is solved and the interface location is either handled as a separate unknown or follows implicitly from the flow solution. Since Lagrangian methods are less useful in the case of large deformations, Eulerian methods are used more often for two-fluid flow simulation. For a complete overview of these methods see [7].

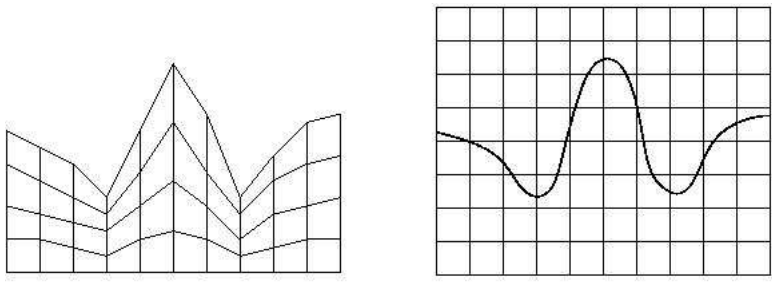


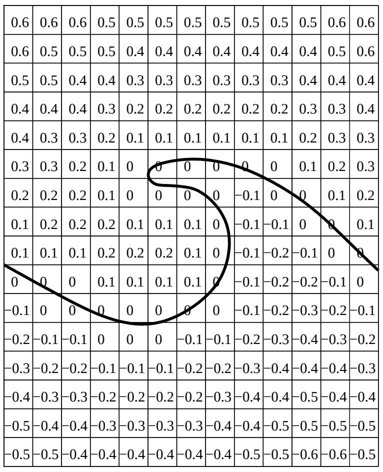
Figure 1.1: *Left:* Interface-fitting methods use an interface-aligned grid. *Right:* Interface-capturing and tracking methods use a domain-dependent grid.

The first class of interface methods, the interface-fitting methods, makes use of an interface-aligned grid that evolves together with the interface. This implies that in the case of large deformations of the interface, as is the case for many two-fluid flow problems such as for example breaking waves, the grid becomes extremely complicated, which makes this class of methods unsuitable.

The class of interface-capturing methods obtains the interface location from the flow solution itself. For this often an extra unknown, such as the mass-fraction or ratio of specific heats, is added to the system of flow quantities to describe the multi-fluid nature of the flow. See [11, 24, 60] for several of these methods. As opposed to interface-fitting methods a 'simple' grid can be used since the interface location is embedded in the flow solution itself. A disadvantage of this approach is the rather complicated system of equations that is required to describe the complete two-fluid flow.

Together with the *marker and cell* (MAC) and *volume of fluid* (VOF) methods (see [26]) the LS method is a member of the class of *interface-tracking* methods. All these methods use information from the fluid flow to track the location of the two-fluid interface as it is transported by the flow. For this purpose a variable or function is introduced that labels each point in the flow domain (depending on the numerical method these points are either grid points, cells, etc.) as either one fluid or the other. In the case of the LS method this labelling is done using a so-called *signed distance function* that labels every point with a sign to indicate the fluid (a $+$ for one fluid and a $-$ for the other) and a number representing the shortest distance to the interface. See figure 1.2 for an example of such a level-set distribution. To track the interface the LS function is simply advected with the local flow velocity, such that the well-known advection equation holds. Characteristic for the LS method, and other interface tracking methods, is that the equation for tracking the interface is almost completely decoupled from the flow equations. The only couplings that exist are those between the fluid properties (am I looking at fluid 1 or 2?) used in the Euler equations and the location of the interface, and the coupling between the velocity field and the evolution of the interface. This allows for the development of a separate solver for the Euler equations and for the LS equation.

Although both the interface-capturing and interface-tracking methods allow for large deformations,



0.6	0.6	0.6	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.6	0.6
0.6	0.5	0.5	0.5	0.4	0.4	0.4	0.4	0.4	0.4	0.4	0.5	0.6
0.5	0.5	0.4	0.4	0.3	0.3	0.3	0.3	0.3	0.3	0.4	0.4	0.4
0.4	0.4	0.4	0.3	0.2	0.2	0.2	0.2	0.2	0.2	0.3	0.3	0.4
0.4	0.3	0.3	0.2	0.1	0.1	0.1	0.1	0.1	0.1	0.2	0.3	0.3
0.3	0.3	0.2	0.1	0	0	0	0	0	0	0.1	0.2	0.3
0.2	0.2	0.2	0.1	0	0	0	0	−0.1	0	0	0.1	0.2
0.1	0.2	0.2	0.2	0.1	0.1	0.1	0	−0.1	−0.1	0	0	0.1
0.1	0.1	0.1	0.2	0.2	0.2	0.1	0	−0.1	−0.2	−0.1	0	0
0	0	0	0.1	0.1	0.1	0.1	0	−0.1	−0.2	−0.2	−0.1	0
−0.1	0	0	0	0	0	0	0	−0.1	−0.2	−0.3	−0.2	−0.1
−0.2	−0.1	−0.1	0	0	0	−0.1	−0.1	−0.2	−0.3	−0.4	−0.3	−0.2
−0.3	−0.2	−0.2	−0.1	−0.1	−0.1	−0.2	−0.2	−0.3	−0.4	−0.4	−0.4	−0.3
−0.4	−0.3	−0.3	−0.2	−0.2	−0.2	−0.2	−0.3	−0.4	−0.4	−0.5	−0.4	−0.4
−0.5	−0.4	−0.4	−0.3	−0.3	−0.3	−0.3	−0.4	−0.4	−0.5	−0.5	−0.5	−0.5
−0.5	−0.5	−0.4	−0.4	−0.4	−0.4	−0.4	−0.4	−0.5	−0.5	−0.6	−0.6	−0.5

Figure 1.2: The level-set method labels every grid cell of a computational domain with a signed number. The zero contour represents the interface. Negative numbers indicate one fluid and positive numbers the other.

and are therefore especially useful for the treatment of general two-fluid flow problems, it has been shown in [2] that when they make use of an extra equation for a two-fluid or interface parameter (the LS function, mass-fraction, VOF fraction, ratio of specific heats, etc.), they generally suffer from spurious pressure errors (often referred to as *pressure oscillations*). And since it is especially the interface (location and thickness) that is important for two-fluid flow simulation, it is extremely important that these oscillations are prevented. Several fixes for this problem have been proposed, see for example [1, 22, 28, 29, 32, 47], but these all use a locally non-conservative formulation of the flow equations near the interface, which can lead to large errors for problems with strong shocks. Fortunately it can be shown that for the problems we are interested in these non-conservative fixes suffice. Especially a fix proposed by Abgrall and Karni in [2], from now on called the *simple fix* because of its extremely easy implementation, appears to effectively prevent the spurious oscillations while side-effects are reduced to a minimum. In [39] several test problems have been solved using a finite-volume discretization of the LS method with this simple fix. The numerical results showed no spurious oscillations from which it became clear that the simple fix works.

However, the same report also mentions a numerical error that results in an off-set of the interface location. Although present in several test results in existing literature, this numerical error has not been mentioned before. In the present thesis this numerical error will be subject of further research, resulting in an analytical expression for it and the corresponding conclusion that the error only occurs when the LS equation (or a similar equation) is used in its so-called *conservative form*. Writing the LS equation as an *advection equation* prevents these errors, indicating that this is clearly the approach that should be taken. Together with the simple fix, this so-called *advective form* of the LS equation provides an excellent oscillation-free method for two-fluid flow simulation of the Euler equations.

1.2 The Runge-Kutta discontinuous Galerkin method

Numerically solving an equation requires it to be discretized. Often the choice of discretization method depends on the equations that need to be solved and the desired accuracy (how well does the numerical solution represent the exact solution?), robustness (how sensitive is my method?) and efficiency (how large are the computational costs?) of the discretization method. Given the large amount of numerical methods available, choosing the most optimal one (note that here explicitly the term *most optimal* instead of *best* is used, since in computational fluid dynamics there is no such thing as a best method!) is not an easy task at all.

An advantage of the level-set method is the ability to solve the flow equations and the level-set equation separately. Since the level-set equation is a 'simple' advection equation for which an extensive numerical theory exists, our focus will be on the Euler equations. The unsteady Euler equations form a system of nonlinear hyperbolic conservation laws. Characteristic for these and similar purely advective equations is the possibility that discontinuities are developed in the solution. These discontinuous phenomena often display a very rich and complex structure, which puts a lot of requirements - and often restraints - on the numerical method to be used. In order to capture these discontinuities a sufficiently accurate method is required, which delivers physically relevant discontinuities (entropy satisfying discontinuities) without introducing spurious oscillations (Godunov [23] showed that higher-than-first-order methods always introduce spurious oscillations near discontinuities when no further measures are taken).

The numerical methods available for the spatial discretization can generally be divided into *finite difference* (FD), *finite volume* (FV) and *finite element* (FE) methods. Especially the second has been used most frequently for solving nonlinear hyperbolic systems. Extended with numerical techniques as *numerical fluxes* and *slope limiters* these higher-order FV methods fulfilled all the conditions required for a correct approximation of discontinuous phenomena. It was only since the development of the so-called *discontinuous Galerkin* (DG) method that also FE methods became suitable for these types of problems.

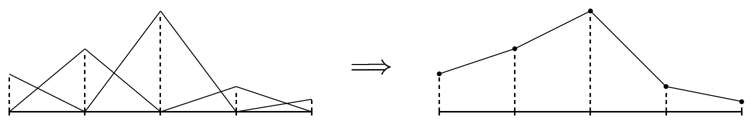


Figure 1.3: Continuous basis functions (hat functions) are used to give a linear representation of the solution on a discrete domain.

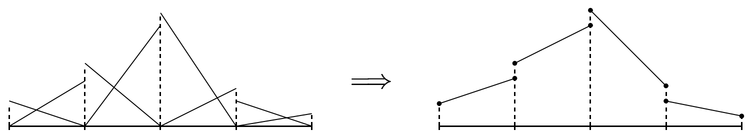


Figure 1.4: Discontinuous basis functions are used to give a linear representation of the solution on a discrete domain.

The DG method distinguishes itself from the more standard continuous Galerkin (CG) method by the way the *approximate solution* is represented on the numerical domain. In the CG case the approximate solution is assumed to be continuous over the cell faces that separate neighboring grid cells, while in the DG case this continuity restriction is absent. This difference is clarified in figures 1.3 and 1.4. Here we see an approximate solution on a grid consisting of several cells. In both the CG and the DG case, linear *basis functions* are used to represent the approximate solution (note that in the CG case these linear basis functions are often called *hat functions*), but while the continuous solution is restricted to one single solution value at each cell face, the discontinuous solution can have two. Although this means that for each cell more unknowns have to be solved, it is obvious that the discontinuous approach is better able to represent discontinuous phenomena. Another advantage of the DG method is the fact that it is *fully local*; all the necessary information for calculating the approximate solution in a cell is contained in that cell itself, such that no other information of neighboring cells is necessary [1]. This means that the resulting mass matrix, with a smart choice of basis functions, will always be block diagonal, while the CG method will result in a band-shaped mass matrix. It is well-known that a block diagonal mass matrix allows the use of explicit time-stepping methods, which are in general far less computationally expensive than implicit

[1]This is indeed the case for the DG method itself. However, numerically solving most equations requires also the calculation of so-called *numerical fluxes* over the cell faces. Especially when the equations are highly nonlinear these fluxes are calculated using information from both the cell under consideration and (several of) its neighboring cells.

methods. An additional advantage of the locality property of the DG method is that it allows for an easy implementation of refinement techniques. Since there is no restriction on the solution near cell faces, the order of the approximating polynomials (*p*-refinement) and the grid size (*h*-refinement) can differ per grid cell. Due to its local nature the DG method is also suitable for parallelization, which becomes an ever more interesting topic in the computational sciences.

The DG method was first introduced by Reed and Hill [44] in 1973 for solving the neutron transport equation. The first mathematical analysis of this method was performed in 1974 by LeSaint and Raviart [35], followed by many others [27, 42, 45]. This combined work resulted in an efficient and robust method for the treatment of linear equations. The application of DG to nonlinear equations, however, required a different approach. Finding a suitable time discretization for the existing DG method, such that the resulting method was besides efficient and formally higher-order accurate also stable, proved a difficult task. Stable methods could be obtained using implicit time discretizations but this always resulted in extremely computationally inefficient schemes. The first explicit DG method was developed by Chavent and Salzano [14]. Unfortunately this method proved unconditionally unstable, making it clear that different techniques were necessary. The first step towards such an efficient, robust and accurate explicit DG method was taken by Chavent and Cockburn [13]. They modified the method of Chavent and Salzano using a, in FV methods well-known and established, technique known as *slope limiting* [2]. This slope limiting, introduced by van Leer [33, 34] in 1974, locally introduces a nonlinear dissipative mechanism into the scheme to avoid spurious oscillations. Following this work Cockburn and Shu [20] developed the *Runge-Kutta discontinuous Galerkin* (RKDG) method based on the higher-order Runge-Kutta time discretization. This formally second-order RKDG method was extended to higher orders in [20]. Further work resulted in RKDG schemes for one-dimensional systems of nonlinear equations [18] and later for multi-dimensional systems such as the Euler equations of gas dynamics [4, 16]. More recent developments are the extension of the DG method to elliptic problems (diffusion) and consequently the application of those methods to convection-diffusion equations such as the Navier-Stokes equations [3]. Other research interests are *hp*-refinement (order and grid refinement) techniques and related to that the development of error estimation techniques [5, 6, 8, 9, 10]. An excellent overview of the development of DG methods can be found in [17].

In this thesis we will follow the work of Cockburn and Shu *et al.* and use a RKDG method together with the LS method for the simulation of two-fluid flows described by the Euler equations. Since the RKDG has never been used in this context several practical problems have to be tackled for a correct implementation of this method. Special attention is required for the spurious pressure oscillations mentioned earlier. Although an efficient simple fix is present for a FV solver, no such thing exists for a DG method. Based on previous work by Naber [39] this simple fix will be modified to work for the RKDG method. This also requires the development of a novel two-fluid slope limiter.

1.3 Relevant flow problems

The Euler equations of gas dynamics describe fluid flows that are unsteady, inviscid and compressible. Although viscosity plays an important role in most fluids, for many problems (especially those with high Reynolds numbers) satisfying results can be obtained with an Euler solver. Our interest lies in the application of these Euler equations to two-fluid flows with a sharp fluid interface. Several of these flows have been mentioned above. The numerical solver developed in this thesis will be tested using several one-dimensional so-called *shock tube* problems taken from [38, 57], before being applied to two-dimensional problems. The resulting two-dimensional solver is validated with well-known tests previously used in [39]. Besides acting as benchmarks these two-dimensional test problems also have a distinct physical relevance. Understanding the physical processes that take place is of prime importance. The two-dimensional problems that are treated in this thesis are:

[2]The idea of applying a for FV methods developed technique to DG methods is not as strange as it seems. In fact the DG method can be seen as a generalization of the FV method since it also assume piecewise discontinuous approximate solutions. While FV methods solely make use of a piecewise constant representation of the numerical solution, DG methods can use basis polynomials of any order (piecewise constant, linear, quadratic, cubic, etc.). A numerical technique such as slope limiting can therefore be implemented very naturally into the DG formulation.

Shock-bubble interaction

A shock in air interacts with a bubble of a different fluid resulting in the deformation of this bubble and the creation of a complex and highly structured wave pattern consisting of shock waves and expansion fans. For the fluids Helium and Refrigerant 22 is used. The former is a lighter-than-air fluid, while the latter is heavier. This density difference results in two completely different wave patterns. Especially the process of break-up of the bubble is interesting and requires a sufficiently accurate numerical solver. Experiments with this problem have been performed by Haas and Sturtevant [25], while numerical results can be found in [43].

Kelvin-Helmholtz instability

The Kelvin-Helmholtz instability is probably the best-known and most frequently occurring flow instability in nature. This unstable flow situation occurs when two layers of fluid move in different directions over each other, creating a so-called *shear layer*. A small disturbance in the interface will roll up to create a characteristic vortex pattern. The test case used in this thesis has been proposed by Mulder, Osher and Sethian in [37].

Supersonic free jets

The supersonic free jet is an excellent problem to investigate using numerical methods since analytical theories come short for a complete understanding of the gas dynamical processes taking place. A supersonic free jet occurs when a high velocity flow exits a nozzle into a gas, often air, producing a jet with characteristics depending on the properties of the jet and the ambient gas. Two specifically interesting situations occur when the jet exits in a gas with either a lower or a higher pressure. In the former case the jet is called *underexpanded* referring to the jet not being expanded to a low enough pressure (the ambient gas pressure). The latter jet is called *overexpanded* because the jet has a pressure lower than the gas it exits in. In both cases the jet will deform due to the appearance of shocks and expansion fans that try to adapt the jet pressure to the ambient pressure. For an analytical discussion of the supersonic free jet one can refer to any textbook on supersonic gasdynamics such as [21]. Several numerical results of these jets can be found in [39].

1.4 Major accomplishments

In this thesis a novel numerical solver for two-dimensional, unsteady, inviscid, compressible two-fluid flows is developed. The full method consists of a LS method for the interface treatment and a RKDG method (using piecewise quadratic basis functions) for the discretization of the Euler equations. Although the method is based on existing techniques the final method as such is new. Major accomplishments are:

- The derivation of an analytical expression for the numerical error that occurs when the LS equation is used in its *conservation* form. It is shown that the *advection* form does not generate these errors and is therefore the method to use.

- The *simple fix* that prevents spurious pressure oscillations to occur near the two-fluid interface is modified such that it works for the DG method. This requires the development of a novel *two-fluid slope limiter for DG*.

- The LS advection equation is considered in full detail. Especially the combination of the LS equation and the RKDG method for the Euler equations requires special attention. A suitable discretization method for the LS equation is chosen from several options.

- Several relevant two-fluid flow problems are treated in detail, showing the capabilities of the solver. Special attention is paid to the *supersonic jet*.

1.5 Outline of this thesis

This thesis is generally divided into three main parts. The first part is governed with the derivation and explanation of the flow model. In chapter 2 the Euler equations for compressible, unsteady and inviscid flow are given. The following chapter 3 is devoted to the level-set method. The different possible forms of the level-set equation are derived and a choice is made between these forms. Chapter 4 treats the spurious pressure oscillations that are present in the level-set method. A simple fix that prevents these oscillations from occurring is given and evaluated.

The second part of this thesis is governed with the flow solver. It treats the RKDG discretization of the Euler equations in chapter 5. Part of this chapter is the development of a novel two-fluid slope limiter that extends the simple fix of chapter 4 such that it also works for DG. The following chapter 6 discusses the discretization of the level-set equation. Also treated is the discretization and implementation of the redistancing procedure for the level-set function. Finally the complete flow solver is considered in more detail in chapter 7. Technical aspects that need further explanation can be found here.

The third and final part of this thesis treats the flow problems. In chapter 8 several one-dimensional problems are considered, which mainly act as a validation of the numerical solver. In the following chapter 9 the more complicated two-dimensional problems are treated. The thesis is concluded with research conclusions and several recommendations for further research.

Part I

Flow Model

Chapter 2

The Euler equations

The unsteady flow of a compressible fluid, where the effects of viscosity, heat conduction, external heating and external forces are neglected, can be described by the so-called *Euler equations* of gas dynamics. These equations describe the conservation of mass, momentum and energy in a fluid flow. To be more exact; when one uses the term *Euler* (or for example *Navier-Stokes*) equations, one officially only refers to the momentum equations. The equation of conservation of mass, also known as the *continuity equation*, is a separate equation. The same holds for the equation describing the conservation of energy. However, it is generally accepted that with the Euler equations one means the full set of flow equations.

The derivation of the Euler equations is quite straightforward and shall therefore not be elaborated here (for a detailed derivation one can use any textbook on fluid dynamics). The full set of laws for the conservation of mass, momentum and energy, reads in differential form:

$$\frac{\partial \rho}{\partial t} + \nabla \cdot \rho \boldsymbol{V} = 0, \tag{2.1}$$

$$\frac{\partial \rho \boldsymbol{V}}{\partial t} + \nabla \cdot \rho \boldsymbol{V}\boldsymbol{V} + \nabla p = 0, \tag{2.2}$$

$$\frac{\partial \rho E}{\partial t} + \nabla \cdot \rho H \boldsymbol{V} = 0. \tag{2.3}$$

Here ρ is the density of the fluid, $\boldsymbol{V} = (u, v, w)^T$ the velocity vector, p the pressure, E the total energy and H the total enthalpy. Of these we call the density, the velocity components and the pressure the *primitive variables*. In this thesis only ideal gases are considered, which allows for an explicit expression relating the primitive thermodynamic variables:

$$p = \rho e \left(\gamma - 1\right), \tag{2.4}$$

where e is the internal energy of the flow and γ the ratio of specific heats, being equal to 1.4 for air. Since we make use of ideal gasses, this ratio of specific heats is what distinguishes one fluid from the other. Hence, when treating a two-fluid flow, both fluids are characterized by their own specific value for γ. The total energy can be written as a combination of the internal and the kinetic energy:

$$E = e + \frac{1}{2}|\boldsymbol{V}|^2, \tag{2.5}$$

where e follows from (2.4). The total enthalpy H is defined as:

$$H = E + \frac{p}{\rho}. \tag{2.6}$$

2.1 General formulation

The Euler equations form a system of nonlinear hyperbolic conservation laws. These systems of conservation laws are often written as one single vector equation, where the vector components each represent one of the conservation laws. This general formulation reads:

$$\frac{\partial \boldsymbol{q}}{\partial t} + \nabla \cdot \boldsymbol{F}(\boldsymbol{q}) = \mathbf{0}. \tag{2.7}$$

Here $\boldsymbol{q}$ is the conservative state vector and $\boldsymbol{F}$ the general flux vector. Depending on the number of dimensions, $\boldsymbol{F}$ contains several directional flux vectors. For both the one and two-dimensional case this general formulation shall be worked out in more detail.

2.1.1 The one-dimensional equations

The general conservative form for the one-dimensional equations can be written as:

$$\frac{\partial \boldsymbol{q}}{\partial t} + \nabla \cdot \boldsymbol{F}(\boldsymbol{q}) = \frac{\partial \boldsymbol{q}}{\partial t} + \frac{\partial \boldsymbol{f}(\boldsymbol{q})}{\partial x} = \mathbf{0}. \tag{2.8}$$

In the one-dimensional case both the state and flux vectors have three components, representing the equations of conservation of mass, momentum and energy, respectively:

$$\boldsymbol{q} = \begin{pmatrix} \rho \\ \rho u \\ \rho E \end{pmatrix} \quad \text{and} \quad \boldsymbol{f} = \begin{pmatrix} \rho u \\ p + \rho u^2 \\ \rho u H \end{pmatrix}. \tag{2.9}$$

From the above it is obvious that all components of the conservative state vector and the flux vectors can be written as functions of the primitive variables. For convenience's sake therefore often the so-called primitive state vector $\boldsymbol{w} = (\rho, u, p)^T$ is introduced. Transformation between the conservative and primitive state vector is straightforward and shall therefore not be elaborated here.

2.1.2 The two-dimensional equations

In the two-dimensional case the general formulation of the Euler equations can be written as:

$$\frac{\partial \boldsymbol{q}}{\partial t} + \nabla \cdot \boldsymbol{F}(\boldsymbol{q}) = \frac{\partial \boldsymbol{q}}{\partial t} + \frac{\partial \boldsymbol{f}(\boldsymbol{q})}{\partial x} + \frac{\partial \boldsymbol{g}(\boldsymbol{q})}{\partial y} = \mathbf{0}. \tag{2.10}$$

The flux vector $\boldsymbol{F}$ now contains a flux vector for the x direction $\boldsymbol{f}$, and one for the y direction $\boldsymbol{g}$. For the two-dimensional version of the Euler equations these vectors each have four components, representing the equations for conservation of mass, momentum in x and y directions and energy, respectively:

$$\boldsymbol{q} = \begin{pmatrix} \rho \\ \rho u \\ \rho v \\ \rho E \end{pmatrix}, \quad \boldsymbol{f} = \begin{pmatrix} \rho u \\ p + \rho u^2 \\ \rho u v \\ \rho u H \end{pmatrix} \quad \text{and} \quad \boldsymbol{g} = \begin{pmatrix} \rho v \\ \rho u v \\ p + \rho v^2 \\ \rho v H \end{pmatrix}. \tag{2.11}$$

For the two-dimensional case the primitive state vector reads $\boldsymbol{w} = (\rho, u, v, p)^T$.

2.2 Boundary conditions

For the Euler equations five different boundary conditions can be distinguished, being: supersonic outflow, supersonic inflow, subsonic outflow, subsonic inflow and the solid wall boundary condition. Depending on the type of boundary condition a specific number of state variables, (ρ, u, v, p), should be prescribed at the boundary under consideration:

- Supersonic outflow: 0 state variables should be prescribed,
- Supersonic inflow: 4 state variables should be prescribed,

- Subsonic outflow: 1 state variable should be prescribed,
- Subsonic inflow: 3 state variables should be prescribed,
- Solid wall: 1 state variable should be prescribed (the solid wall boundary is assumed to be a limit case of the subsonic outflow boundary).

Whenever less than four variables need to be prescribed, the remaining variables can be left ’untouched’, i.e. a Neumann condition can be used. For a derivation and a more detailed description of these boundary conditions one is referred to [54].

Chapter 3

The level-set method

In this chapter the level-set method is treated in more detail. The level-set function is introduced and the governing equation is given. Both the advective and the conservative form of the level-set equation are derived. It is shown that the conservative form is undesirable since it introduces a numerical error that results in an off-set in the interface location. The advective form does not generate these errors and is therefore the form to use. To keep the level-set function at all time a true distance function it has to be redistanced, which is also subject of discussion. Finally the treatment of the two-fluid interface is discussed in more detail.

3.1 Distance function

The level-set (LS) method is an implicit method for describing the evolution of an interface between two fluids. It makes use of a *distance function* φ, referred to as the *level-set function*, which labels every point in a domain with a value representing the shortest distance to the interface. As such this function is equal to zero at the interface itself and non-zero in the fluids. To distinguish between the two fluids, one often uses a *signed* function, being negative in one fluid and positive in the other. In figure 3.1 a breaking wave is shown with several LS contours. It is clear that each contour represents those points that are a certain distance away from the interface. In the same figure the discretized flow domain is shown. Each grid cell contains a value of the level-set function.

The reason for using a linear representation for the LS function φ is that in this way the numerical solution of the LS function is not affected by numerical diffusion. A numerical scheme often introduces diffusive second-order derivatives into the governing equations. If the LS function is kept a true linear distance function at all times, the second and higher order derivatives of this function will always be equal to zero. This is desirable since higher order effects often result in undesirable errors in the numerical solution. Especially in the case of a sharp fluid interface, of which the location and thickness is of prime importance, these errors should be prevented.

In the context of two-fluid simulation the LS function is used as a switch to distinguish between the two fluids present in the computational domain. Whenever the LS distribution is known at a certain moment in time, the zero contour of the level-set function indicates the location of the interface. Those grid cells that have a LS function with a positive number contain one fluid, and those with a negative number contain the other. As such the LS function indicates which fluid properties should be used in which grid cell. This information is then used to solve the Euler equations with the correct fluid parameters. This procedure will be explained in more detail in one of the following sections.

3.2 Level-set equation

The LS function φ is a scalar parameter that is advected by the flow with the local flow velocity without influencing the flow itself; a passive scalar. Note that the LS function does influence the flow, it determines

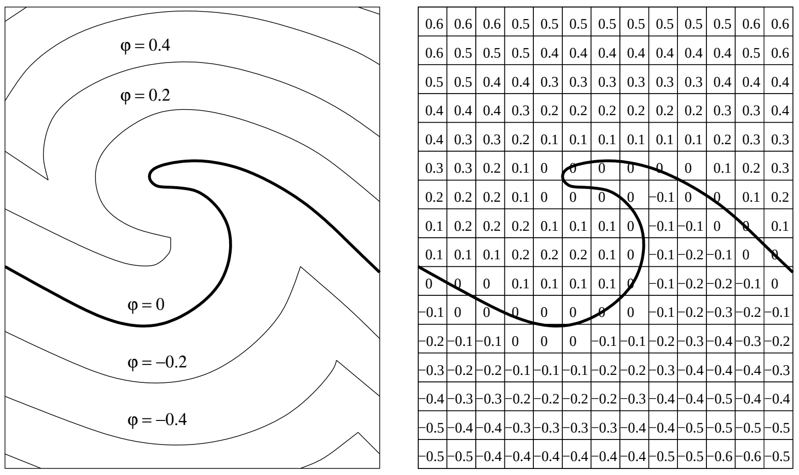


Figure 3.1: Two fluids separated by an interface. The LS set function φ is a signed distance function, being zero at the interface, positive in one fluid and negative in the other. *Left:* Several LS contours. *Right:* Each grid cell contains a LS value.

the location of the interface and thus the fluid properties to be used when a certain region is under consideration, but this can be seen as an indirect influence. The procedure of determining the interface can be performed after the flow has been solved. During the process of solving the flow itself the LS function is just simply advected with the flow [1]. The well-known advection or transport equation can therefore be used to describe the motion of φ, and thus the evolution of the interface, in time.

3.2.1 Advective form

For the two-dimensional case the advection equation for the LS function can be written as:

$$\frac{\partial \varphi}{\partial t} + \boldsymbol{V} \cdot \nabla \varphi = \frac{\partial \varphi}{\partial t} + u\frac{\partial \varphi}{\partial x} + v\frac{\partial \varphi}{\partial y} = 0. \tag{3.1}$$

Here u and v are the velocities in x and y directions, respectively. Given a velocity field $\boldsymbol{V}$ and an initial distribution of the LS function φ this equation describes how the LS distribution changes in time. The advantage of this approach is the fact that the numerical solver for the LS equation can be completely decoupled from the Euler equations. An existing Euler solver can be used, which can simply be extended with a solver for the LS equation. The major disadvantage is that this does require two separate numerical schemes to be used.

3.2.2 Conservative form

Besides the advective form of the level-set equation there exists a second frequently used form; the conservative form. When the level-set equation (3.1) is used together with a hyperbolic conservation law (or a system of hyperbolic conservation laws, such as the Euler equations) it is often written as a conservation law itself. Multiplying equation (3.1) with the density ρ and adding it to the equation for conservation of mass (2.2) multiplied with the level-set function φ, gives:

$$\frac{\partial \rho\varphi}{\partial t} + \nabla \cdot \rho \boldsymbol{V} \varphi = \frac{\partial \rho\varphi}{\partial t} + \frac{\partial \rho u \varphi}{\partial x} + \frac{\partial \rho v \varphi}{\partial y} = 0, \tag{3.2}$$

[1]For a proper understanding of how the LS function is affected by the fbw, compare the LS function with a concentration of a certain substance in the fluid. The LS function indicates how much of this substance is present at a certain place. When a fbw is present, the concentration just moves with the local fbw velocity without influencing the fbw.

which can be interpreted as an equation for the conservation of the distance function times the mass. This conservation law for the level-set function can be added to the system of conservation laws (the Euler equations) as an additional equation:

$$\boldsymbol{q} = \begin{pmatrix} \rho \\ \rho u \\ \rho v \\ \rho E \\ \rho\varphi \end{pmatrix}, \quad \boldsymbol{f} = \begin{pmatrix} \rho u \\ p + \rho u^2 \\ \rho uv \\ \rho uH \\ \rho u\varphi \end{pmatrix} \quad \text{and} \quad \boldsymbol{g} = \begin{pmatrix} \rho v \\ \rho uv \\ p + \rho v^2 \\ \rho vH \\ \rho v\varphi \end{pmatrix}. \tag{3.3}$$

As such the LS equation can be solved with exactly the same numerical techniques as the other Euler equations. Although this conservation approach seems preferable above a separate treatment of the LS equation - mainly for aesthetic reasons since numerically it is not cheaper and practically it is not as easy as it seems [2] - there is a convincing reason for not doing so.

3.3 Why not conservative?

Solving for the level-set function in its conservative form is not desirable since it introduces numerical errors which lead to an unphysical off-set of the level-set function and thus a wrong location of the interface. The existence of these errors has already been shown in a report by Naber [39]. Although the presence of a numerical off-set was noticed in that report, the origin was still unknown. Here we shall give for a specific numerical scheme the corresponding expression for the numerical error. Based on this a proposal is done to prevent the error.

3.3.1 Translating interface problem

In [39] a two-fluid solver was developed, which was based on a finite volume discretization of the Euler equations and the conservation form of the LS equation. Using this solver several one-dimensional test cases were solved. The numerical results of these tests all showed an undesirable off-set in the numerical interface location, which was worse when a larger density jump was present in the test problem. A specific test where the off-set was clearly visible was the so-called *translating interface problem*. This one-dimensional problem considers an initial interface separating two fluids with similar pressures but different densities. The interface and both fluids travel with a constant velocity to the right. In time, the interface is required to move with this constant velocity, while the pressure and the density remain unchanged.

Figure 3.2 shows the numerical and exact distributions of the ratio of specific heats for this translating interface with a strong initial density jump. The corresponding initial conditions are:

$$\begin{aligned} (\rho, u, p, \gamma)_L &= (1000.0, 1.0, 1.0, 1.4)\,, \\ (\rho, u, p, \gamma)_R &= (1.0, 1.0, 1.0, 1.6)\,. \end{aligned}$$

Clearly visible in the plot of the distribution of the ratio of specific heats γ is the difference between the location of the numerical and the exact interface. The location of the numerical interface has a clear off-set to the right compared to the exact interface location. This off-set leads to the use of the wrong fluid properties near the interface and thus in an incorrect calculation of the flow solution (density, velocity and pressure).

With this in mind several numerical results in other papers have been investigated more closely, among which the well-known overview paper by Abgrall and Karni [2], and also in these the interface off-set was present. Although less visible due to a smaller initial density jump, the LS function showed indeed a numerical error when a conservative approach was used.

[2]Standard approximate Riemann solvers have to be adapted for the treatment of the level-set function φ. Although satisfying results have been obtained in the past this approach is not as straightforward as often is believed. In [37] such an adapted Riemann solver is discussed in more detail.

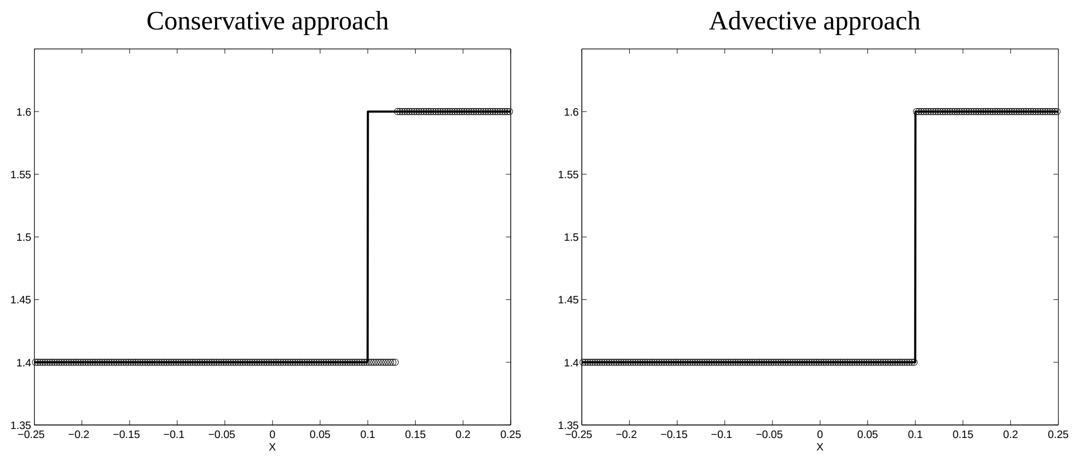


Figure 3.2: 1D translating interface problem. Ratio of specific heats γ at $t = 0.1$. Grid has 200 cells and $\Delta t/\Delta x = 0.5$. Solid lines are exact solutions, dotted lines are numerical solutions. *Left:* Conservative approach. *Right:* Advective approach.

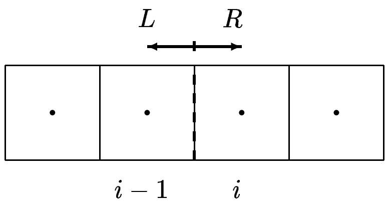


Figure 3.3: Grid cells containing a traveling interface separating two fluids with $\gamma_L \neq \gamma_R$. The density jumps over the interface, $\rho_L \neq \rho_R$, while the pressure and velocity are both constant, $p_L = p_R = p$ and $u_L = u_R = u$.

3.3.2 Origin of the error

The origin of the errors shown above can be determined with an analytical treatment of the translating interface problem. This analysis is done for the finite volume solver developed in [39] but the conclusions hold generally for every numerical scheme. We assume that initially the interface is located exactly on the cell face separating cells $i-1$ and i (see figure 3.3). The interface moves with a velocity u to the right. The pressure in both fluids is equal, $p_L = p_R = p$, while the density jumps over the interface, $\rho_L \neq \rho_R$. The flow of the two fluids and their interface can be described by the one-dimensional Euler equations for the conservation of mass, momentum and energy together with the one-dimensional LS equation in conservation form (3.2):

$$\frac{\partial \rho}{\partial t} + \frac{\partial \rho u}{\partial x} = 0, \tag{3.4}$$

$$\frac{\partial \rho u}{\partial t} + \frac{\partial \rho u^2 + p}{\partial x} = 0, \tag{3.5}$$

$$\frac{\partial \rho E}{\partial t} + \frac{\partial \rho u H}{\partial x} = 0, \tag{3.6}$$

$$\frac{\partial \rho \varphi}{\partial t} + \frac{\partial \rho u \varphi}{\partial x} = 0. \tag{3.7}$$

Here $E = \frac{1}{\gamma-1}\frac{p}{\rho} + \frac{1}{2}u^2$ is the total energy and $H = E + \frac{p}{\rho}$ the total enthalpy. Applying the first-order upwind scheme for the discretization of the spatial derivative and the forward Euler scheme for the time derivative, together with the notations $\Delta x = x_i - x_{i-1}$ and $\Delta t = t^{n+1} - t^n$, to equations (3.4) and (3.7) gives:

$$\frac{\rho_i^{n+1} - \rho_i^n}{\Delta t} = -\frac{1}{\Delta x}\left[(\rho u)_i^n - (\rho u)_{i-1}^n\right], \tag{3.8}$$

$$\frac{(\rho\varphi)_i^{n+1} - (\rho\varphi)_i^n}{\Delta t} = -\frac{1}{\Delta x}\left[(\rho u\varphi)_i^n - (\rho u\varphi)_{i-1}^n\right]. \tag{3.9}$$

Because the velocity is constant over the interface, $u_{i-1} = u_i = u$, these expressions can be rewritten as:

$$\rho_i^{n+1} = \sigma\rho_{i-1}^n + (1-\sigma)\,\rho_i^n, \tag{3.10}$$

$$(\rho\varphi)_i^{n+1} = \sigma\,(\rho\varphi)_{i-1}^n + (1-\sigma)\,(\rho\varphi)_i^n, \tag{3.11}$$

where $\sigma = \frac{u\Delta t}{\Delta x}$. At $t = t^n$ the level-set function is a linear distribution with $\varphi = 0$ at the interface, $\varphi > 0$ to the left of this interface and $\varphi < 0$ to the right of this interface. This means that $\varphi_{i-1}^n = \frac{\Delta x}{2}$ and $\varphi_i^{n-1} = -\frac{\Delta x}{2}$. Inserting these expressions into (3.11) and replacing ρ_{i-1}^n and ρ_i^n with ρ_L and ρ_R, respectively, gives:

$$\rho_i^{n+1} = \sigma\rho_L + (1-\sigma)\,\rho_R, \tag{3.12}$$

$$(\rho\varphi)_i^{n+1} = (\sigma\rho_L - (1-\sigma)\,\rho_R)\,\frac{\Delta x}{2}. \tag{3.13}$$

Such that the new value of the level-set function in cell i becomes:

$$\varphi_i^{n+1} = \frac{\sigma\rho_L - (1-\sigma)\,\rho_R}{\sigma\rho_L + (1-\sigma)\,\rho_R}\frac{\Delta x}{2}. \tag{3.14}$$

The fact that this expression is not the desired result becomes obvious when we compare it to the exact expression for the level-set function in cell i at time level n. We know that the interface moves with the velocity u to the right such that after a time step Δt it has moved a distance $\sigma\Delta x$. The value of the level-set function in cell i at $t = t^n$ thus becomes:

$$\left(\varphi_i^{n+1}\right)_{exact} = -\frac{\Delta x}{2} + \sigma\Delta x. \tag{3.15}$$

The difference between (3.14) and (3.15) is the error in the LS distribution, and thus in the interface location:

$$\begin{aligned}\Delta\varphi_i^{n+1} &= \varphi_i^{n+1} - \left(\varphi_i^{n+1}\right)_{exact} \\ &= \frac{\sigma\rho_L - (1-\sigma)\,\rho_R}{\sigma\rho_L + (1-\sigma)\,\rho_R}\frac{\Delta x}{2} - \left(-\frac{\Delta x}{2} + \sigma\Delta x\right) \\ &= \frac{\sigma\,(1-\sigma)\,(\rho_L - \rho_R)}{\sigma\rho_L + (1-\sigma)\,\rho_R}\Delta x. \end{aligned} \tag{3.16}$$

Only when the density is constant over the interface, $\rho_L = \rho_R$, or when $\sigma = 0$ or $\sigma = 1$, the interface is in its correct location. Note that $\sigma = 0$ corresponds to an interface that did not move, and $\sigma = 1$ to an interface that in one time step exactly moved one cell length, which are both unrealistic options. For any other case the location of the interface has an undesired off-set from the exact location, which depends on the magnitude of the density jump over the interface. The larger the density jump, the larger the error. This is clearly visible when we compare the weaker translating interface treated in [2] with the stronger one treated above and shown in figure 3.2. The latter shows a much larger off-set in the interface location than the former. From (3.16) it further becomes clear that this off-set is of the order $\mathcal{O}\,(\Delta x)$ for the first-order upwind scheme. It can he shown however that even the higher-than-first-order numerical schemes result in an error that depends linearly on the mesh size.

3.3.3 How to prevent the error?

It is obvious that treating the LS equation in its *advective form*, equation (3.1), does not produce the numerical problems described above since one does not need to transform φ to and from a conservative form $\rho\varphi$. This becomes clear when one analyzes the translating interface problem with the same numerical solver as used before, but now with the LS equation in its advective form. Performing a similar analysis as was done above, i.e. forward euler for time and first-order upwind for space, one can write for the update equation for the LS function:

$$\frac{(\varphi)_i^{n+1} - (\varphi)_i^n}{\Delta t} = -\frac{u}{\Delta x}\left[(\varphi)_i^n - (\varphi)_{i-1}^n\right]. \tag{3.17}$$

Writing this out using the notations defined above, one ends up with the following expression for the LS function at the new time level:

$$\varphi_i^{n+1} = \sigma\varphi_{i-1}^n + (1-\sigma)\,\varphi_i^n. \tag{3.18}$$

Introducing the expressions for the LS function at the old time level, $\varphi_{i-1}^n = \frac{\Delta x}{2}$ and $\varphi_i^{n-1} = -\frac{\Delta x}{2}$, finally gives:

$$\varphi_i^{n+1} = \sigma\frac{\Delta x}{2} - (1-\sigma)\,\frac{\Delta x}{2} = -\frac{\Delta x}{2} + \sigma\Delta x, \tag{3.19}$$

which is identical to the expression for the exact LS function that was found in (3.15). The corresponding numerical result is given in figure 3.2 too. It is clear that the interface is now in the right location.

Although it seems advantageous to add the level-set equation to the system of Euler equations, it is definitely not. There is no decrease in numerical costs, the implementation is not easier and above all, it produces first-order numerical errors in the interface location. Simply solving the advection equation for the LS function prevents these errors and does not add any difficulties or computational costs. This is why the advective form is the one that shall be used in this thesis.

3.4 Redistancing

The advection equation for the level-set function solely transports the values of φ with the local velocity. In the general case of a non-uniform velocity field this results in the advection of the different level-set values such that the signed distance function is not preserved. As an example consider a simple one-dimensional case with a signed distance function distribution (linear) and a velocity distribution such that the velocity has its maximum at the interface ($\varphi = 0$). See figure 3.4. Due to this non-uniform velocity field the value $\varphi = 0$ moves faster than the other values for φ. This means that the distance between the interface and the point with $\varphi = 1$ for example, is initially equal to 1, but will be different from 1 after one time step already. It is obvious that the distribution of the level-set function is distorted in time, i.e. it is no longer a distance function. Only the interface, having $\varphi = 0$, remains correct because it is not measured relative to another point.

Because we are only interested in this zero-level, i.e. the interface, this seems to lead to no further difficulties. Unfortunately, in a numerical approximation using a finite grid, the LS function will, besides maybe at time zero, never be exactly equal to zero in a cell. The points closely around the interface therefore need a correct value of the LS function too. Another reason why the LS function should be a true linear distance function is that earlier mentioned fact that a nonlinear function will introduce numerical diffusion into the numerical scheme. To be able to give an accurate representation of the interface and prevent undesirable numerical errors it is therefore necessary that near the interface the LS function is the real distance function and not the distorted version. This requires for a *redistancing* procedure of the LS function.

A commonly used method for the redistancing of the level-set function, i.e. giving φ its correct value at each point of the treated domain, is the so-called *PDE approach* presented by Sussman *et al.* in [56]. The idea is to solve the following differential equation [3] until its steady state is reached:

$$\frac{\partial \varphi}{\partial \tau} = S\left(\varphi_{\tau=0}\right)\left(1 - |\nabla \varphi|\right), \tag{3.20}$$

where τ is an artificial time scale only used within the redistancing procedure and $S\left(\varphi_{\tau=0}\right)$ the sign function of φ_0. A steady state, i.e. $\frac{\partial \varphi}{\partial \tau} = 0$, is reached when the gradient of φ is equal to one. When this is the case, the level-set function has been transformed to a real (signed) distance function again. The numerical treatment of this equation shall be discussed in the chapter 6.

3.5 Interface treatment

It has been mentioned that the signed distance function works as a switch to distinguish between the different fluids; a positive φ means one fluid and a negative value the other. Because here only ideal gasses are treated, which follow the same thermodynamical laws, the only difference between the two fluids is the value of their ratio of specific heats γ. The value of φ can therefore be easily used to determine the local value of γ to be used in the flow equations:

$$\gamma = \begin{cases} \gamma_1 & \text{if} \quad \varphi > 0, \\ ? & \text{if} \quad \varphi = 0, \\ \gamma_2 & \text{if} \quad \varphi < 0. \end{cases} \tag{3.21}$$

Here γ_1 is the ratio of specific heats of one fluid and γ_2 of the other. Unfortunately this rather simplistic approach leads to some difficulties. The first of these is the fact that due to the discontinuity over the interface, γ is not explicitly defined exactly at the interface. It is therefore unclear what value of γ to use here.

The second problem has to do with the fact that the interface will run through cells instead of neatly coinciding with their boundaries in the case of a discrete approximation of the domain using grid cells. When the interface moves with the local flow velocity it will move away from the cell faces into the cells. This results in cells containing portions of both fluids. Because the numerical solver only uses the flow variables of one fluid, the one that is present in the cell center, the interface is in fact *pushed* to the cell faces, resulting in the interface to follow the sharp corners of the grid cells (see figure 3.5). This effect is called *staircasing* of the interface. Due to this staircasing the location of the interface, and thus the local values of the ratio of specific heats, are used incorrectly when solving fluxes over the cell faces. This can lead to numerical errors, which in turn can result in spurious oscillations in the interface.

[3]This equation is also referred to as the *eikonal equation* for propagating wave fronts. This is a clear way of looking at the redistancing equation since we want to propagate our interface outwards from itself in the direction of its own normal vector. Every point in the domain is in this way turned into a real distance function where each point has a value representing the distance to the closest interface point.

To avoid this, the interface can be smeared out to some extent (in the discrete approximation over some cells). Thus instead of using a 'standard' Heaviside function $H(\varphi)$ for the determination of the local value of γ, a smooth function is used (see figure 3.6). This smooth step function $H_\epsilon \in [0,1]$ is defined as:

$$H_\epsilon(\varphi) = \begin{cases} 0 & \text{if} \quad \varphi < -\epsilon, \\ \frac{1}{2}\left[1 + \frac{\varphi}{\epsilon} + \frac{1}{\pi}\sin\left(\frac{\pi\varphi}{\epsilon}\right)\right] & \text{if} \quad -\epsilon \leq \varphi \leq \epsilon, \\ 1 & \text{if} \quad \varphi > \epsilon, \end{cases} \tag{3.22}$$

where ϵ is a small distance often taken equal to one and a half cell size ($\epsilon = \frac{3\Delta x}{2}$) [4]. This specific value of ϵ is used since it is the smallest distance possible in the two-dimensional case such that smoothing is always symmetrically applied (to be more precise; the minimum value is $\epsilon = \sqrt{2\Delta x}$, but for convenience's sake we shall use the value mentioned above). Choosing a lower value can lead to an asymmetrical distribution because of the discrete nature of the state-variable distribution. A higher value leads to more smearing of the interface and is not desired. Using this Heaviside function we can write the following equation for the determination of γ:

$$\gamma = H_\epsilon \gamma_1 + (1 - H_\epsilon)\gamma_2, \tag{3.23}$$

where γ_1 and γ_2 are again the ratios of specific heats of both fluids. Because the interface is no longer sharp but smeared out to some extent, the sign function $S(\varphi_0)$ used in the redistancing equation (3.20) is no longer correct. Therefore a new sign function S_ϵ based on H_ϵ is defined:

$$S_\epsilon(\varphi) = 2H_\epsilon(\varphi) - 1. \tag{3.24}$$

[4]Note that for this smoothing function also an *error function* could be used. Another approach could be to use a VOF-type-of-approach to determine an intermediate value for γ.

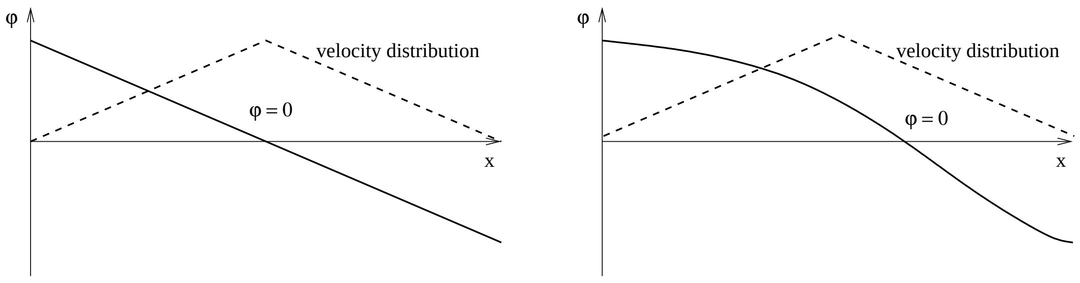


Figure 3.4: Under influence of a non-uniform velocity field (in this case a hat distribution) the LS function does not remain a true distance function. Only the interface ($\varphi = 0$ location remains correct. *Left:* Initial LS distribution. *Right:* LS distribution after a few time steps.

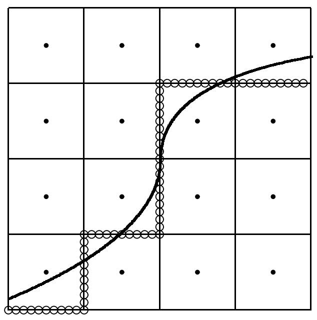


Figure 3.5: Grid cells containing an interface separating two fluids (solid line). The interface is *pushed* to the cell faces (dotted line) leading to *staircasing* of the interface. This staircasing leads to numerical errors which can induce spurious oscillations in the state variable distributions.

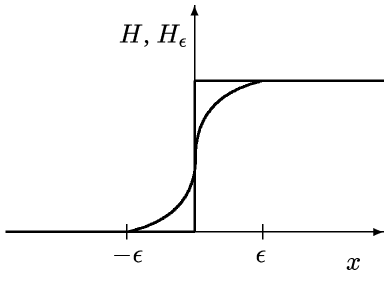


Figure 3.6: The Heaviside function $H(\varphi)$ is smoothed to the transformed $H_\epsilon(\varphi)$ to avoid spurious oscillations due to staircasing of the interface.

Chapter 4

Pressure oscillations in the LS method

One of the major problems of conservative methods for compressible two-fluid flows is the occurrence of *spurious pressure oscillations* [1]. Due to the inconsistency of conservative schemes near the interface large errors occur that do not disappear with decreasing mesh-size. These oscillations have been the subject of extensive research for the past ten years, leading to several adequate fixes, of which the *ghost-fluid method* by Fedkiw *et al.* [22] and the fixes by Karni [28, 29] are best known. In this chapter a locally non-conservative method, first developed by Abgrall and Karni in [2], is discussed. The method is comparable to the ghost-fluid method but is more simple to implement and requires less computational power.

4.1 Origin of the oscillations

The existence of these so-called *spurious pressure oscillations* can easily be shown by performing a simple numerical experiment. For this again the translating interface problem from chapter 3 is used. The initial properties of both fluids are taken to be equal to $(\rho, u, p, \gamma)_L = (1.0, 1.0, 1.0, 1.4)$ and $(\rho, u, p, \gamma)_R = (0.1, 1.0, 1.0, 1.6)$. Results at time $t = 0.02$ are plotted in figure 4.1. One immediately sees that the velocity and pressure distributions show significant errors, i.e. the spurious pressure oscillations. Further experiments show that the severity of these pressure errors does not decrease with decreasing mesh-size, indicating a zero-th order behavior. It should further be mentioned that these errors do not occur in the case of single-fluid flows, i.e a constant value of γ.

So where do these pressure oscillations come from? As the name already suggests they originate from numerical errors in the pressure distribution. It has been shown by Abgrall and Karni [2] that using a conservative scheme for two-fluid simulation always results in an inconsistency in the pressure update (the energy equation) near the interface, where the distribution of γ is *not* constant (in [11] it has been shown that for barotropic two-fluid flows it is indeed possible to obtain oscillation-free solutions with a conservative approach). This is mainly due to the fact that the values of γ used for the calculation of the interface fluxes are different, resulting in a loss of continuity in the pressure distribution.

To show this we use the earlier seen translating interface problem. Assume the interface to be initially located between the left cell face of cell i and its cell center (see figure 4.2). This implies that, when using a non-smooth step-function for γ, the left interface has γ_L and the right cell face γ_R which are different in the case of two different fluids. The flow is described by the one-dimensional Euler equations (2.8), which are discretized in space by the simple first-order upwind scheme and in time using the forward Euler scheme [2]. This results in the following discretized vector equation:

$$\boldsymbol{q}_i^{n+1} - \boldsymbol{q}_i^n = -\frac{\Delta t}{\Delta x}\left[\boldsymbol{f}_i^n - \boldsymbol{f}_{i-1}^n\right], \tag{4.1}$$

[1]Incompressible flows do not suffer from these spurious pressure oscillations since the energy equation can be omitted in case of constant density.

[2]Note that for our analysis we use a FV instead of a DG approach. This is done for simplicity's sake. The conclusions that will be based on these results are however also valid for the DG approach used in this report. A small extension has to be made to the simple fix developed here in order to work for our DG solver but this will be topic of discussion in chapter 5

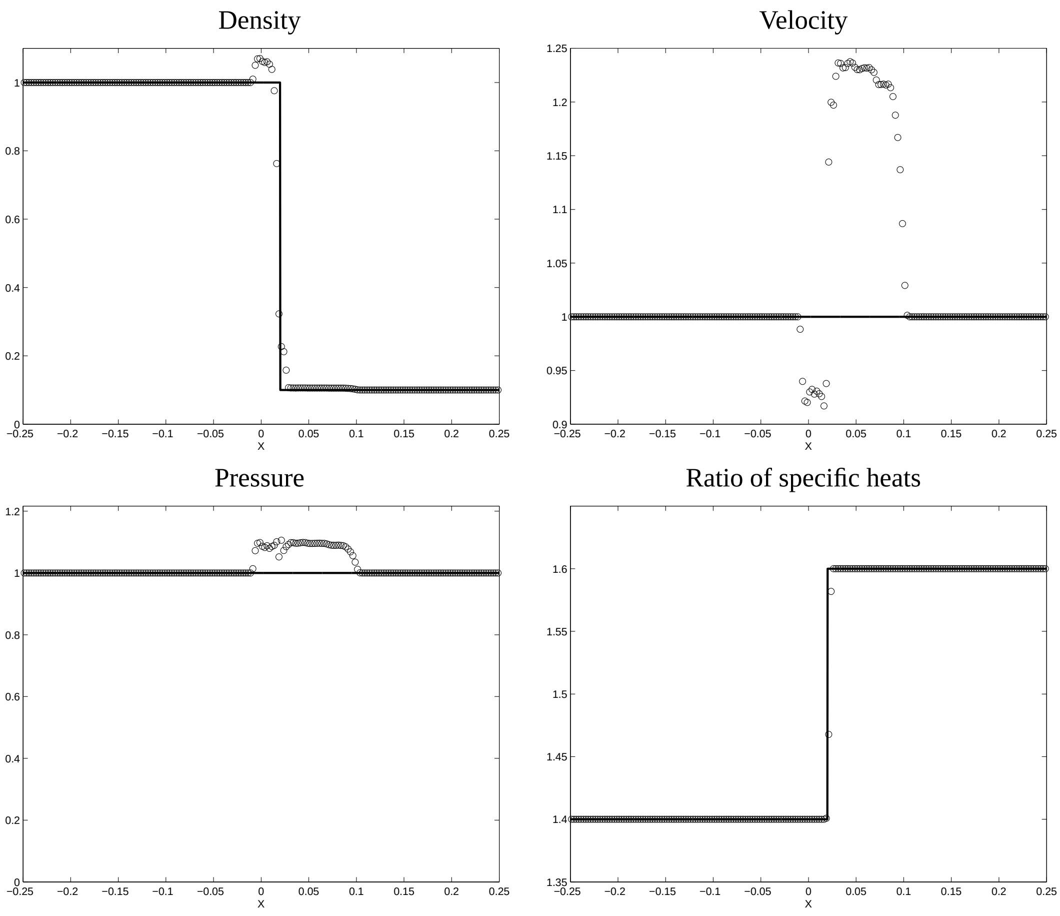


Figure 4.1: 1D translating interface problem with pressure oscillations. (ρ, u, p, γ) at $t = 0.02$. Grid has 200 cells and $\Delta t/\Delta x = 0.5$. Solid lines are exact solutions, dotted lines are numerical solutions.

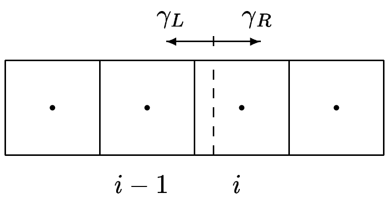


Figure 4.2: Grid cells containing an interface separating two fluids with $\gamma_L \neq \gamma_R$.

or written as separate equations for the conservation of mass, momentum and energy:

$$\rho_i^{n+1} - \rho_i^n = -\frac{\Delta t}{\Delta x}\left[(\rho u)_i^n - (\rho u)_{i-1}^n\right], \tag{4.2}$$

$$(\rho u)_i^{n+1} - (\rho u)_i^n = -\frac{\Delta t}{\Delta x}\left[\left(\rho u^2 + p\right)_i^n - \left(\rho u^2 + p\right)_{i-1}^n\right], \tag{4.3}$$

$$(\rho E)_i^{n+1} - (\rho E)_i^n = -\frac{\Delta t}{\Delta x}\left[(\rho u H)_i^n - (\rho u H)_{i-1}^n\right]. \tag{4.4}$$

Assuming that both the velocity u and the pressure p are continuous over the interface (at time level n), such that $u_i^n = u_{i-1}^n = u$ and $p_i^n = p_{i-1}^n = p$ (since an interface is a contact discontinuity this is indeed the case). From (4.2) and (4.3) it follows that $u_i^{n+1} = u_i^n = u$, i.e. the velocity remains uniform. Unfortunately this is not the case for the pressure. To see this we rewrite the energy equation (4.4) as follows:

$$\left(\frac{p}{\gamma - 1}\right)_i^{n+1} - \left(\frac{p}{\gamma_R - 1}\right) = -\frac{u\Delta t}{\Delta x}\left[\left(\frac{p}{\gamma_R - 1}\right) - \left(\frac{p}{\gamma_L - 1}\right)\right], \tag{4.5}$$

where we already took into account that the velocity and the pressure are continuous over the interface. It is obvious that the required situation $p_i^{n+1} = p_i^n = p$ is only the case when the following holds:

1. $\gamma_L = \gamma_R$, which corresponds to not having an interface,
2. $\gamma_i^{n+1} = \gamma_R = \gamma_i^n$, which implies that the interface has not moved.

Since in the chosen approach both situations are in general not the case, pressure oscillations are the undesired result. From expression (4.5) it thus follows that the presence of the factor $\frac{1}{\gamma-1}$, which jumps over an interface, is the origin of our spurious pressure oscillations. Every time there is a difference between these factors numerical errors appear in the distribution of the pressure which, due to the coupling of the flow equations, immediately result in oscillations in the other state variable distributions too. Going to a higher order does not change this because there will always be a certain change in γ. A way to the reduce the magnitude of these errors is the earlier mentioned *interface smoothing*. This interface smoothing will smear the change of γ over more than one cell and as such reduce the local jumps in the γ distribution, which will result in a decrease of the severity of the errors. Note however that interface smoothing does *not* remove the oscillations completely.

4.2 Simple fix

To remove the pressure oscillations from the solution, without using a completely different method (for example a fully non-conservative method), a fix has to be applied. In the analysis of the numerical discretization treated in the previous section it was shown that the solution would remain oscillation free only in the special case that:

$$\gamma_L = \gamma_R \qquad \text{and} \qquad \gamma_i^{n+1} = \gamma_i^n .$$

It is not hard to see that this is therefore the situation we are looking for. Unfortunately the implementation is not as straightforward as one may believe based on the above. In the following this simple fix is explained in more detail. It consists of two parts, which are based on the two requirements derived above. The first works during the calculation of the interface fluxes and the second during the pressure update.

4.2.1 Part 1 - Flux calculation

It has been shown that part of the errors resulting in the spurious pressure oscillations occur due to the jump in the ratio of specific heats over an interface. It therefore is a logical choice to look for a fix that removes this jump. Unfortunately, due to the discrete nature of a numerical method, this jump will always be present. Having no jump would mean having only one fluid. Fortunately this is not completely true, which allows for a way out. The oscillations only occur because, when updating the state variables in the cell centers, the fluxes are calculated using different values of the ratio of specific heats γ (in the case of a smoothed interface the value of γ at the cell center is also different). Using the same ratio of specific heats for these flux calculations and the update of the cell centered state vector removes this inconsistency. Thus instead of interpolating the ratio of specific heats to the cell faces and calculating the interface fluxes using these values of γ, the value for γ in the cell center is also used for the fluxes. In other words; a cell is updated using its own single value of γ. As such we indeed have locally $\gamma_L = \gamma_R$.

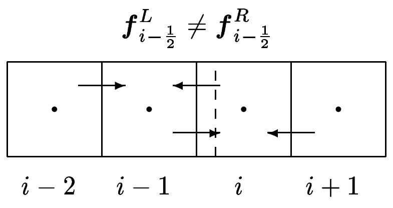

$\boldsymbol{f}^L_{i-\frac{1}{2}} \neq \boldsymbol{f}^R_{i-\frac{1}{2}}$
$i-2$
$i-1$
i
$i+1$

Figure 4.3: The simple fix applied to grid cell i containing an interface (dotted line) separating two fluids with $\gamma_L \neq \gamma_R$. The simple fix locally abandons conservation by using two different numerical fluxes $\boldsymbol{f}$ at the cell interface separating both fluids.

To clarify the chosen approach the situation in figure 4.3 is considered. When updating the state variables in cell i two numerical fluxes are required; the flux at $i-\frac{1}{2}$ and the flux at $i+\frac{1}{2}$ (in the two-dimensional case four fluxes are required, a north, east, south and west flux, indicated with N, E, S, W). When no fix is applied, the flux at $i-\frac{1}{2}$ updating cell i, here indicated with $\boldsymbol{f}^R_{i-\frac{1}{2}}$ because it updates the cell to the right of the cell face, is equal to the flux updating cell $i-1$ left of the cell face $\boldsymbol{f}^L_{i-\frac{1}{2}}$, which complies with the requirement that the method should be conservative [3]. Unfortunately this approach will lead to the spurious oscillations mentioned because the value of γ used for the flux calculation is different from the γ used for the update of the state variables. In the fix presented here each cell is updated with fluxes using the cell centered values of the ratio of specific heats. So the flux updating cell i will use γ_R and the flux updating cell $i-1$ will use γ_L, which can be expressed as follows:

$$\boldsymbol{f}^R_{i-\frac{1}{2}} = \boldsymbol{f}\left(\gamma_R\right), \tag{4.6}$$

$$\boldsymbol{f}^L_{i-\frac{1}{2}} = \boldsymbol{f}\left(\gamma_L\right). \tag{4.7}$$

This means that the flux updating the left cell is no longer equal to the flux in the right cell, i.e. the method is no longer conservative when this fix is applied. In the next section this disadvantage will be considered in more detail. An advantage of this fix is that it only works when necessary. A cell face that separates two cells that both contain the same fluid does not require two fluxes to be calculated. The fix automatically takes this into account. Consider for example cells i and $i-1$. Because the fluids in cells i and $i-1$ are the same, the right and left *pointing* [4] fluxes at interface $i-\frac{1}{2}$ are exactly the same, so the fix does not work in this case and only one flux is calculated. Because the interface will be smeared-out over some grid cells due to the smoothing procedure, the fix will in reality be applied to several interfaces instead to only one as figure 4.3 maybe suggests.

Another advantage of this fix is its *single-fluid nature*. Since the method only uses one value of γ at all time, the numerical solver can be built from single-fluid building blocks such as a single-fluid Riemann solver. This especially eases the development of the full flow solver to a large extent.

4.2.2 Part 2 - Pressure update

The second cause of the numerical errors that lead to spurious pressure oscillations is the difference between the ratio of specific heats at the *old* γ_i^n and at the *new* time level γ_i^{n+1}. Since the interface moves in time,

[3]The normal vectors with which both fluxes are multiplied are pointing in opposite directions, so the contribution of these fluxes to the total flux will differ in sign for both cells.

[4]Note that with *pointing* is meant which cell (left or right) the flux updates not in which direction it is physically pointing (the direction of the local flow velocity).

the distribution of the ratio of specific heats will not stay the same at all time. The resulting difference in old and new γ will cause a numerical error to occur during the calculation of the pressure p_i^{n+1} from the value of the total energy ρE_i^{n+1} during the pressure update.

To prevent this error one has to *freeze* the distribution of the ratio of specific heats. This freezing implies that for the calculation of the pressure from the total energy, the old value of γ should be used, i.e. $\gamma_i^{n+1} = \gamma_i^n$. After the calculation of this correct pressure the new ratio of specific heats can be used again. This rather simple fix, combined with the first part mentioned above, prevents the solution from becoming oscillatory.

4.3 Errors due to fix

Although the simple fix discussed here is able to remove the undesired pressure oscillations, it has several side effects. The errors induced by the simple fix will be discussed in more detail in this section. Analytical expressions will be derived for the errors of both parts of the fix. Note again that these errors hold for the FV method taken from [39] and used here for the derivation of the simple fix. However, the conclusions drawn here will give a good qualitative indication of the errors to be expected when using the DG method. Due to the complexity of the DG method it is not yet possible to derive similar expressions for the errors due to the simple fix.

4.3.1 Part 1 - Abandoning conservation

The first type of error introduced by the fix presented above is that it locally *abandons conservation* of energy. Only when a cell uses one single value of γ, the spurious oscillations are prevented. Unfortunately this implies that in certain situations at one cell face two different numerical fluxes are used. Clearly this means that conservation is abandoned. The corresponding error can be written as the difference between the flux that updates the cell to the left of the cell face under consideration and the cell to the right of this cell face. Assuming a simple first order upwind scheme we can write for this difference at cell face $i - \frac{1}{2}$:

$$\Delta \boldsymbol{f} = \boldsymbol{f}^R_{i-\frac{1}{2}} - \boldsymbol{f}^L_{i-\frac{1}{2}} = \boldsymbol{f}\left(\gamma_R\right)_{i-\frac{1}{2}} - \boldsymbol{f}\left(\gamma_L\right)_{i-\frac{1}{2}} \,. \tag{4.8}$$

For the calculation of the numerical flux Roe's approximate Riemann solver is used. For this specific Riemann solver the numerical fluxes read as follows:

$$\boldsymbol{f}^L_{i-\frac{1}{2}} = \boldsymbol{f}^L_{i-1} + \sum_{\lambda^{L-}} \Delta v^L |\lambda^L| \boldsymbol{r}^L, \tag{4.9}$$

$$\boldsymbol{f}^R_{i-\frac{1}{2}} = \boldsymbol{f}^R_{i} - \sum_{\lambda^{R+}} \Delta v^R |\lambda^R| \boldsymbol{r}^R. \tag{4.10}$$

Here $\lambda^{\pm}$ are the positive and negative eigenvalues respectively. Further we can write:

$$\begin{array}{lll} \Delta v_1 = \frac{\Delta p - \rho c \Delta u}{2c^2}, & \lambda_1 = u - c, & \boldsymbol{r}_1 = \left(u, u-c, H-uc\right)^T, \\ \Delta v_2 = \frac{c^2\Delta\rho - \Delta p}{c^2}, & \lambda_2 = u, & \boldsymbol{r}_2 = \left(1, u, \frac{1}{2}u^2\right)^T, \\ \Delta v_3 = \frac{\Delta p + \rho c \Delta u}{2c^2}, & \lambda_3 = u + c, & \boldsymbol{r}_3 = \left(u, u+c, H+uc\right)^T. \end{array} \tag{4.11}$$

Substitution of these expressions into (4.8), together with $\Delta u = \Delta p = 0$, $u_{i-1} = u_i = u$ and $p_{i-1} = p_i = p$, finally results in the following expression for the difference between the numerical fluxes at cell face $i - \frac{1}{2}$:

$$\Delta \boldsymbol{f} = \begin{pmatrix} 0 \\ 0 \\ up\Delta\left(\frac{1}{\gamma-1}\right) \end{pmatrix}. \tag{4.12}$$

It is clear that the error due to abandoning conservation near the two-fluid interface results in a numerical error that is dependent on the magnitude of the velocity, pressure and the factor $\frac{1}{\gamma-1}$. For the problems considered in this report this error remains small. Only near shocks this abandoning of conservation can introduce difficulties. Fortunately, since a shock and a contact discontinuity move with different wave speeds, a two-fluid interface will generally not coincide with a shock long enough to produce large errors.

4.3.2 Part 2 - Freezing of γ

The second source of errors is due to the freezing of the thermodynamic properties. Using the old distribution of γ for obtaining the pressure from the total energy can introduce errors when during the time step the interface moves to another cell. In this case the pressure is obtained using a *wrong* value of γ. Though the way this method is constructed requires this specific approach. The total energy change due to this freezing can be written as:

$$\left(E|_{\gamma=\gamma_L} - E|_{\gamma=\gamma_R}\right) = -\Delta\left(\frac{1}{\gamma-1}\right)p. \tag{4.13}$$

The numerical error due to the temporarily frozen ratio of specific heats also depends on the factor $\frac{1}{\gamma-1}$ and the pressure. As was the case for the error due to abandoning conservation, this error remains relatively small for the problems considered here and can therefore be neglected. It has further been shown by Abgrall and Karni [2] that the contributions of (4.12) and (4.13) have opposite effects on the solution.

Part II

Flow Solver

Chapter 5

RKDG discretization of the Euler equations

The previous chapters were devoted to our flow model for two-fluid flows. In the following chapters the corresponding governing equations are discretized. Here the Runge-Kutta discontinuous Galerkin (RKDG) method is considered. The approach of Cockburn *et al.* [15, 16, 18, 20] is followed. The spatial grid consists of uniform rectangular cells on which the numerical solution is represented using the discontinuous Galerkin (DG) method, based on orthogonal quadratic (or lower order) basis functions. The cell-face fluxes are computed using Roe's approximate Riemann solver. Integrals are approximated using standard Gaussian quadrature rules. Time discretization is done by a Runge-Kutta method, with the same order of approximation as the spatial discretization. A slope limiter is introduced to prevent the solution from becoming oscillatory near flow discontinuities. Since a single-fluid slope limiter introduces the same spurious pressure oscillations as discussed in the previous chapter, a novel two-fluid slope limiter for DG is developed that delivers oscillation-free results.

5.1 Discontinuous Galerkin method

Here the general framework of the discontinuous Galerkin method is treated. The method is set up for a general choice of basis functions but will be worked out for orthogonal linear and quadratic basis functions for two-dimensional rectangular elements.

5.1.1 Weak formulation

For the Euler equations we obtain the weak formulation as follows: multiply the Euler equations (2.7) with an arbitrary continuous test function $\boldsymbol{r}$ and integrate over the domain Ω such that [1]:

$$\int_\Omega \boldsymbol{r}\frac{\partial \boldsymbol{q}}{\partial t}d\Omega + \int_\Omega \boldsymbol{r}\nabla\cdot\boldsymbol{F}\left(\boldsymbol{q}\right)d\Omega = \boldsymbol{0}, \qquad \forall \boldsymbol{r}. \tag{5.1}$$

Using integration by parts we can rewrite (5.1) as follows:

$$\int_\Omega \boldsymbol{r}\frac{\partial \boldsymbol{q}}{\partial t}d\Omega + \int_\Gamma \boldsymbol{r}\boldsymbol{F}\left(\boldsymbol{q}\right)\cdot\boldsymbol{n}d\Gamma - \int_\Omega \nabla\boldsymbol{r}\cdot\boldsymbol{F}\left(\boldsymbol{q}\right)d\Omega = \boldsymbol{0}, \qquad \forall \boldsymbol{r}, \tag{5.2}$$

where Γ is the boundary of the domain Ω and $\boldsymbol{n}$ the outward-pointing normal vector on this boundary. This weak formulation can be discretized in space by subdividing the domain Ω into a collection of nonoverlapping elements $E_{i,j}$ with boundaries $\partial E_{i,j}$, where the indices i, j are the counters corresponding to the two spatial dimensions of these elements. Together these elements form the triangulation or partition Ω_h. We

[1] Note the use of $\boldsymbol{r}$ for the test functions instead of the more familiar v. To avoid confusion with the velocity in y-direction we have chosen to use an $\boldsymbol{r}$ here.

now define the approximate solution $\boldsymbol{q}_h$ and test function $\boldsymbol{r}_h$ such that they are in the finite dimensional function space:

$$R_h = \{\boldsymbol{r}_h \in L^2(\Omega) : \boldsymbol{r}_h|_{E_{i,j}} \in \mathcal{P}^p(E_{i,j}), \forall E_{i,j} \in \Omega_h\}, \tag{5.3}$$

where $\mathcal{P}^p$ is the collection of polynomials of degree p. A common way of representing the approximate solution $\boldsymbol{q}_h$ and the corresponding approximate test function $\boldsymbol{r}_h$ is as a combination of coefficients ($\boldsymbol{q}_{i,j}^{(l)}$ and $r_{i,j}^{(l)}$) and basis functions $\phi_{i,j}^{(l)}$, with $l = 0, \ldots, m$ where m is the total amount of unknowns. Within one element this can be written as:

$$\begin{aligned} \boldsymbol{q}_h(x,y,t) &= \textstyle\sum_{l=0}^{m} \boldsymbol{q}_{i,j}^{(l)}(t)\,\phi_{i,j}^{(l)}(x,y), \\ \boldsymbol{r}_h(x,y) &= \textstyle\sum_{l=0}^{m} r_{i,j}^{(l)}\phi_{i,j}^{(l)}(x,y), \end{aligned} \qquad \forall x,y \in E_{i,j}. \tag{5.4}$$

The approximate solution on the entire domain Ω_h is simply obtained by summing the expression above over all the elements $E_{i,j}$. Note that different from the one-dimensional case where $m = p$, with p the order of the approximating polynomial ($p = 1$ for linear basis functions, $p = 2$ for quadratic basis functions, etc.), m is not equal to p for two-dimensional elements but depends on the specific choice of elements and basis functions. This will be elaborated in the section treating the basis functions. The semidiscrete (discrete in space not in time) system of equations now becomes:

$$\int_{E_{i,j}} \boldsymbol{r}_h \frac{\partial \boldsymbol{q}_h}{\partial t} d\Omega + \sum_{e \in \partial E_{i,j}} \int_e \boldsymbol{r}_h \boldsymbol{F}(\boldsymbol{q}_h) \cdot \boldsymbol{n}_e d\Gamma - \int_{E_{i,j}} \nabla \boldsymbol{r}_h \cdot \boldsymbol{F}(\boldsymbol{q}_h)\, d\Omega = \boldsymbol{0}, \tag{5.5}$$
$$\forall E_{i,j} \in \Omega_h,$$

where the boundary $\partial E_{i,j}$ is split up into its edges e. Consequently on each of these edges we define an outward-pointing normal vector $\boldsymbol{n}_e$. Since the Galerkin method chooses $\boldsymbol{r}_h$ in each cell $E_{i,j}$ such that it is a linear combination of $m+1$ basis functions $\phi_{i,j}^{(l)}$, equation (5.5) is equivalent to a system of $m+1$ equations:

$$\int_{E_{i,j}} \phi_{i,j}^{(l)} \frac{\partial \boldsymbol{q}_h}{\partial t} d\Omega + \sum_{e \in \partial E_{i,j}} \int_e \phi_{i,j}^{(l)} \boldsymbol{\Psi}_e(\boldsymbol{q}_h)\, d\Gamma - \int_{E_{i,j}} \nabla \phi_{i,j}^{(l)} \cdot \boldsymbol{F}(\boldsymbol{q}_h)\, d\Omega = \boldsymbol{0}, \tag{5.6}$$
$$\forall E_{i,j} \in \Omega_h, \qquad l = 0, \ldots, m,$$

where we write replace the expression for the flux in normal direction at cell face e, $\boldsymbol{F} \cdot \boldsymbol{n}_e$, with $\boldsymbol{\Psi}_e$, which is generally defined as follows:

$$\boldsymbol{\Psi}_e = \boldsymbol{F} \cdot \boldsymbol{n}_e = \boldsymbol{f} n_{x,e} + \boldsymbol{g} n_{y,e} = \begin{pmatrix} \rho v_n \\ \rho v_n u + p n_x \\ \rho v_n v + p n_y \\ \rho v_n H \end{pmatrix}_e, \tag{5.7}$$

where v_n is the normal velocity defined as $v_n = u n_x + v n_y$. Note that the above equation (5.6) requires an evaluation of the flux at the cell-boundaries. However, the approximate solution $\boldsymbol{q}_h$ is discontinuous, and thus not uniquely defined on these boundaries. Therefore we have to replace this flux with one that is defined at the cell faces. We will come back to this issue later.

5.1.2 Elements

For the elements $E_{i,j}$ we will use uniform rectangular cells. Although other choices, such as triangles, are common practice too, rectangles will suffice for the problems treated. An element is defined as $E_{i,j} = \left(x_{i-\frac{1}{2}}, x_{i+\frac{1}{2}}\right) \times \left(y_{j-\frac{1}{2}}, y_{j+\frac{1}{2}}\right)$ for $i, j = 1, \ldots, N_{i,j}$. Note that since we will only work with uniform rectangular cells we can drop the dependency of x on the index j and similarly the dependency of y on the index i. Each element has four edges, which make up the boundary $\partial E_{i,j} = \sum_{N,E,S,W} e$. These edges e have length Δs_e, being either $\Delta x = x_{i+\frac{1}{2}} - x_{i-\frac{1}{2}}$ or $\Delta y = y_{j+\frac{1}{2}} - y_{j-\frac{1}{2}}$, which are both constants for the entire domain since uniform rectangular grids are used. In the center of each edge we define an outward-pointing unit normal vector $\boldsymbol{n}_e = (n_x, n_y)_e^T$. See figure 5.1 for an overview of such a rectangular element.

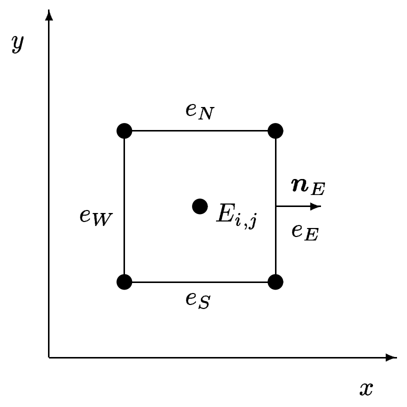


Figure 5.1: A rectangular finite element $E_{i,j}$ with cell interfaces e_N, e_E, e_S and e_W. At each interface mid-point the outward unit normal vector $\boldsymbol{n}_e$ is defined.

5.1.3 Basis functions

Solving the system (5.6) for the unknown coefficients $\boldsymbol{q}_{i,j}^{(l)}$ requires a choice of basis functions. A smart choice of basis functions can to a large extent reduce the amount of computations and therefore requires some attention. It has been mentioned that for the basis functions polynomials of the order p are used. The higher the order of the polynomials the higher the order of the spatial discretization. In [16, 20] it was shown that for systems of hyperbolic conservation laws the DG method is formally $\mathcal{O}(\Delta x^{p+1})$ accurate. Hence, linear basis functions ($p = 1$) lead to a scheme with second-order accuracy and quadratic basis functions ($p = 2$) to a scheme with third-order accuracy. Going to an even higher order is formally possible too.

Further note that by taking the basis functions in each cell equal to one (order zero), i.e. constant distributions, the well-known finite volume method is obtained. Now the solution is approximated by a piecewise-constant distribution of the state variables. In this case one can obtain a higher-order scheme by using a broader stencil of grid cells. Instead of using more equations one uses more cells for each equation.

To avoid a, for the discontinuous Galerkin method characteristic, block-diagonal mass matrix to appear in system (5.6) we choose the basis functions $\phi_{i,j}^{(l)}$ such that they form a local orthogonal basis over $E_{i,j}$. Orthogonal basis functions will result in a single-diagonal mass matrix such that we can completely decouple the system of equations. A straightforward way of obtaining orthogonal basis functions for two-dimensional rectangular elements is by taking the tensor product of orthogonal basis functions in one dimension. Because both sets of basis functions form an orthogonal basis in their own dimension, the tensor product does too. This means that the basis functions $\phi_{i,j}^{(l)}(x, y)$ are split into two separate parts, which only depend on one dimension. The approximate solution can now be written as:

$$\boldsymbol{q}_h(x, y, t) = \sum_{k,l=0}^{p} \boldsymbol{q}_{i,j}^{(k,l)}(t)\, \phi_i^{(k)}(x)\, \psi_j^{(l)}(y), \qquad \forall x, y \in E_{i,j}, \tag{5.8}$$

where $\phi_i^{(k)}$ are the basis functions in x direction and $\psi_j^{(l)}$ in y direction and $\boldsymbol{q}_{i,j}^{(k,l)}$ the *degrees of freedom* or *unknowns* of the approximate solution. For these basis functions we use one-dimensional *Legendre polynomials* [2]. This results in the following basis functions in x direction:

$$\phi_i^{(0)} = 1, \quad \phi_i^{(1)} = \frac{x - x_i}{\Delta x}, \quad \phi_i^{(2)} = \frac{(x - x_i)^2}{\Delta x^2} - \frac{1}{12}, \quad \ldots, \tag{5.9}$$

[2] Here we will make use of Legendre polynomials defined on the domain $E_i = \left(x_{i-\frac{1}{2}}, x_{i+\frac{1}{2}}\right)$ with length Δx, such that $E_i = \left(x_i - \frac{\Delta x}{2}, x_i + \frac{\Delta x}{2}\right)$, while the original Legendre polynomials are defined on $(-1, 1)$. This results in a somewhat different notation than usual. Note however that the polynomials are in fact exactly the same.

and for the basis functions in y direction:

$$\psi_j^{(0)} = 1, \quad \psi_j^{(1)} = \frac{y - y_j}{\Delta y}, \quad \psi_j^{(2)} = \frac{(y - y_j)^2}{\Delta y^2} - \frac{1}{12}, \quad \ldots . \tag{5.10}$$

These $(p+1) \times (p+1)$ basis functions span the complete two-dimensional solution in a rectangular cell. The orthogonality property of these basis functions becomes clear when we take a closer look at the left most integral in (5.6). Working out this integral, after moving the time derivative out of the integral (which is exact for time-independent grids), leads to the following expression:

$$\int_{E_{i,j}} \phi_i^{(k)} \psi_j^{(l)} \boldsymbol{q}_h d\Omega = \boldsymbol{q}_{i,j}^{(k,l)}(t) \left[\int_{x_{i-\frac{1}{2}}}^{x_{i+\frac{1}{2}}} \left(\phi_i^{(k)} \right)^2 dx \right] \left[\int_{y_{j-\frac{1}{2}}}^{y_{j+\frac{1}{2}}} \left(\psi_j^{(l)} \right)^2 dy \right], \quad k, l = 0, \ldots, p. \tag{5.11}$$

The left most integrals of each equation of (5.6) can thus be written in terms of the degrees of freedom of the numerical solution multiplied with the factors $\int_{x_{i-\frac{1}{2}}}^{x_{i+\frac{1}{2}}} \left(\phi_i^{(k)} \right)^2 dx$ and $\int_{y_{j-\frac{1}{2}}}^{y_{j+\frac{1}{2}}} \left(\psi_j^{(l)} \right)^2 dy$, which can easily be computed given the basis functions (5.9) and (5.10) (in appendix A these factors are given together with plots of the various basis functions). Each equation of (5.6) thus becomes an explicit equation for one unknown $\boldsymbol{q}_{i,j}^{(k,l)}$ instead of an implicit equation containing all unknowns.

Linear basis functions

Given the choice of approximation defined in (5.8) with basis functions (5.9) and (5.10) we can write for the numerical solution based on linear polynomials ($p = 1$) in one grid cell:

$$\boldsymbol{q}_h(x, y, t) = \boldsymbol{q}_{i,j}^{(0,0)} + \phi_i^{(1)} \boldsymbol{q}_{i,j}^{(1,0)} + \psi_j^{(1)} \boldsymbol{q}_{i,j}^{(0,1)} + \phi_i^{(1)} \psi_j^{(1)} \boldsymbol{q}_{i,j}^{(1,1)}. \tag{5.12}$$

Note that we have omitted the time dependency of the unknowns $\boldsymbol{q}_{i,j}^{(k,l)}(t)$ and the space dependency of the basis functions. After solving the system of equations (5.6) for the unknowns, the complete solution can be reconstructed using this expression. For convenience's sake we will replace the notation for the unknowns $\boldsymbol{q}_{i,j}^{(k,l)}$ with the following:

$$\boldsymbol{q}_{i,j}^{(0,0)} = \overline{\boldsymbol{q}}_{i,j}, \quad \boldsymbol{q}_{i,j}^{(1,0)} = \overline{\Delta_x \boldsymbol{q}}_{i,j}, \quad \boldsymbol{q}_{i,j}^{(0,1)} = \overline{\Delta_y \boldsymbol{q}}_{i,j}, \quad \boldsymbol{q}_{i,j}^{(1,1)} = \overline{\Delta_{xy} \boldsymbol{q}}_{i,j}, \tag{5.13}$$

which is based on the Taylor series expansion around the point (x_i, y_j). As such the approximate solution $\boldsymbol{q}_h$ can be interpreted to consist of a mean $\overline{\boldsymbol{q}}_{i,j}$ and several derivatives. The more derivatives are used to approximate the solution, the more accurate it becomes.

Quadratic basis functions

In the same way as we did for the linear case we can write the approximate solution for the quadratic case ($p = 2$) as follows:

$$\begin{aligned} \boldsymbol{q}_h(x, y, t) &= \boldsymbol{q}_{i,j}^{(0,0)} + \phi_i^{(1)} \boldsymbol{q}_{i,j}^{(1,0)} + \psi_j^{(1)} \boldsymbol{q}_{i,j}^{(0,1)} \\ &+ \phi_i^{(2)} \boldsymbol{q}_{i,j}^{(2,0)} + \phi_i^{(1)} \psi_j^{(1)} \boldsymbol{q}_{i,j}^{(1,1)} + \psi_j^{(2)} \boldsymbol{q}_{i,j}^{(0,2)} \\ &+ \phi_i^{(2)} \psi_j^{(1)} \boldsymbol{q}_{i,j}^{(2,1)} + \phi_i^{(1)} \psi_j^{(2)} \boldsymbol{q}_{i,j}^{(1,2)} + \phi_i^{(2)} \psi_j^{(2)} \boldsymbol{q}_{i,j}^{(2,2)}. \end{aligned} \tag{5.14}$$

The corresponding unknowns are the same as those for the linear case but now extended with:

$$\begin{aligned} \boldsymbol{q}_{i,j}^{(2,0)} &= \overline{\Delta_{xy} \boldsymbol{q}}_{i,j}, \quad \boldsymbol{q}_{i,j}^{(0,2)} = \overline{\Delta_{xx} \boldsymbol{q}}_{i,j}, \quad \boldsymbol{q}_{i,j}^{(2,1)} = \overline{\Delta_{xxy} \boldsymbol{q}}_{i,j}, \\ \boldsymbol{q}_{i,j}^{(1,2)} &= \overline{\Delta_{xyy} \boldsymbol{q}}_{i,j}, \quad \boldsymbol{q}_{i,j}^{(2,2)} = \overline{\Delta_{xxyy} \boldsymbol{q}}_{i,j}. \end{aligned} \tag{5.15}$$

Note that when one looks at the orders of magnitude of the different terms in the approximate solution it becomes clear that especially the terms which contain third and fourth-order derivatives are of a lower order of magnitude than the other terms. The same holds for the second-order derivative in the linear approximation. It is because of this that sometimes these terms are neglected in the computations. As such the linear approximation would contain 3 unknowns instead of 4 and the quadratic approximation would have 6 instead of 9. Especially when going to higher orders this 'neglect' of the higher-order terms pays off. Since here only $p \leq 2$ is used, this shall not be done.

5.1.4 Cell face fluxes

It has been mentioned that the cell face fluxes are not uniquely defined since the approximate solution is discontinuous at a cell face. At each cell face two distinct solution values are present. Therefore we replace the undefined real flux $\boldsymbol{\Psi}_e(\boldsymbol{q_h})$ with a numerical flux $\tilde{\boldsymbol{\Psi}}_e(\boldsymbol{q}_h^-, \boldsymbol{q}_h^+)$, which depends upon the two states at the cell face. This numerical flux has to satisfy the following conditions in order to guarantee the formal accuracy of the scheme:

$$\tilde{\boldsymbol{\Psi}}_e(\boldsymbol{q}, \boldsymbol{q}) = \boldsymbol{\Psi}_e(\boldsymbol{q}) \tag{5.16}$$
$$\tilde{\boldsymbol{\Psi}}_e(\boldsymbol{q}, \boldsymbol{r}) = -\tilde{\boldsymbol{\Psi}}_e(\boldsymbol{r}, \boldsymbol{q}) \tag{5.17}$$

For the calculations of these interface fluxes we borrow a well-established approach from the finite volume methods; the Riemann solver. For a discussion of the Riemann problem and corresponding solvers see [38]. The idea is that given a left and right state (which are not equal in most cases) a Riemann solver is able to calculate the cell face value by taking into account all physical information. As such it is actually a method to calculate the correct upwind values for the non-linear Euler equations. Here we will make use of the approximate Riemann solver by Roe [46], which is discussed in more detail in the next section. The two input states for this solver are easily computed from the expression for $\boldsymbol{q}_h$, either (5.12) or (5.14), by substituting the location of the interface (x, y).

The concept of a Riemann solver for the cell flux calculations we borrow from finite volume methods and apply it to a finite element approach. Unfortunately these two methods are not the same; the finite volume method makes use of piecewise constant distributions in a cell while the finite element method often uses higher order polynomials. This difference leads to the question whether the standard Riemann problem as we use it here suffices, since this Riemann approach uses the concept of piecewise constant initial distributions. Using for example piece-wise linear distributions requires a different approach to calculate the correct upwind values at the cell-face. Solving for these situations is done using the so-called *generalized Riemann problem*.

Based on the above it seems that if we use, together with a higher-order finite element method, an *old fashioned* Riemann solver instead of a more advanced *generalized* Riemann solver, we neglect information and thus reduce the order of our numerical scheme. However, it can be shown that using a time marching method with several stages, such as the Runge-Kutta methods, together with a standard Riemann solver is equivalent to the generalized Riemann approach with a single-stage scheme. Both deliver the required order of accuracy. It is due to the multi-stage nature of the Runga-Kutta method that a generalized Riemann approach is in fact not necessary. The linear, or higher order, distribution inside a cell is automatically taken into account when the solution is advanced to a new time level with several smaller time steps, as is the case for the Runge-Kutta methods. For a more detailed derivation of the equivalence property of the Runge-Kutta methods see appendix C.

5.1.5 Roe's approximate Riemann solver

Roe's method [46] is a *flux differencing method* for the calculation of the cell face fluxes based on averaged flow characteristics described by the eigensystem (the eigenvalues and eigenvectors) and the jump relations. It calculates the numerical fluxes $\tilde{\boldsymbol{\Psi}}_e$ using the following approximation:

$$\tilde{\boldsymbol{\Psi}}_e = \frac{1}{2}\left(\tilde{\boldsymbol{\Psi}}_e^- + \tilde{\boldsymbol{\Psi}}_e^+\right) - \frac{1}{2}\sum_{k=1}^{4}\left|\hat{\lambda}_k\right| \Delta v_k \hat{\boldsymbol{r}}_k, \tag{5.18}$$

where $\boldsymbol{\Psi}_e^-$ and $\boldsymbol{\Psi}_e^+$ are the fluxes calculated using the initial conditions $\boldsymbol{q}_{h,e}^-$ and $\boldsymbol{q}_{h,e}^+$ respectively. The eigenvalues λ_k and eigenvectors $\boldsymbol{r}_k$ are used here in their modified form. These modified forms follow from the expressions for the eigenvalues and eigenvectors by replacing the regular state quantities with the so-called *Roe averaged state quantities*:

$$\begin{aligned}\hat{\rho} &= \omega\rho^-, &(5.19)\\ \hat{u} &= \frac{u^- + \omega u^+}{1+\omega}, &(5.20)\\ \hat{v} &= \frac{v^- + \omega v^+}{1+\omega}, &(5.21)\\ \hat{H} &= \frac{H^- + \omega H^+}{1+\omega}, &(5.22)\end{aligned}$$

where the ratio ω is used, which is defined as:

$$\omega = \sqrt{\frac{\rho^+}{\rho^-}}. \tag{5.23}$$

Using (5.20) and (5.21) the Roe averaged normal and tangential velocities are defined as: $\hat{v}_n = \hat{u}n_x + \hat{v}n_y$ and $\hat{v}_t = -\hat{u}n_y + \hat{v}n_x$. The Roe averaged speed of sound is given by:

$$\hat{c} = \sqrt{(\gamma - 1)\left(\hat{H} - \frac{1}{2}\left(\hat{u}^2 + \hat{v}^2\right)\right)}. \tag{5.24}$$

Note that this expression requires a value for the ratio of specific heats γ. Here one immediately experiences the enormous advantage of the fact that the two-fluid method developed here is built from single-fluid building blocks. The simple fix that prevents the numerical solution from becoming oscillatory makes the numerical method only dependent upon one single fluid and its corresponding fluid properties. No further measures have to be taken and therefore this rather simple single-fluid approximate Riemann solver can be used (for a two-fluid Roe solver see [37]).

The jump relations Δv_k in (5.18) are obtained from the discrete version of the differential relation $d\boldsymbol{v} = \hat{\boldsymbol{R}}^{-1}d\boldsymbol{q}$ where $\hat{\boldsymbol{R}}$ is the matrix containing the Roe averaged right eigenvectors of the Euler equations (2.10) where the state variables are replaced by the Roe averaged variables. From standard matrix algebra it follows that the inverse of the matrix $\boldsymbol{R}$ is equal to the matrix $\boldsymbol{L}$ containing the *left eigenvectors*, such that the jump conditions can finally be written as: $\Delta\boldsymbol{v} = \hat{\boldsymbol{L}}\Delta\boldsymbol{q}$, where $\Delta[\cdots] = [\cdots]^+ - [\cdots]^-$. See [37] for expressions for the left eigenvalues.

With the above all necessary information for calculating the interface fluxes is known. The procedure is as follows; given the left and right initial interface conditions it is possible to calculate the fluxes left and right of the interface $\boldsymbol{\Psi}_l^-$ and $\boldsymbol{\Psi}_l^+$. The initial conditions can further be used to calculate Roe's averaged state quantities. This allows for the calculation of the modified eigenvectors $\hat{\boldsymbol{r}}_k$ and the modified absolute eigenvalues $\left|\hat{\lambda}_k\right|$. Also the jump relations $\Delta\boldsymbol{v}$ can be obtained from these averaged quantities together with the jumps in density, normal velocity, tangential velocity, pressure and level-set function, which allows for the calculation of the interface fluxes using (5.18)

A well-known problem that occurs when using a Riemann solver is the possibility of having unphysical *expansion shocks* in the solution. Instead of the required fan, a discontinuous shock is chosen by the solver to represent the expansion. To avoid this situation the absolute eigenvalues $\left|\hat{\lambda}_k\right|$ are modified using an *entropy fix*, which removes this unphysical situation from the possible solutions. Therefore the following parameter is introduced:

$$\begin{aligned}\tfrac{\delta\lambda_k}{2} &= 0 && \text{for} \quad k = 2,3,\\ \tfrac{\delta\lambda_k}{2} &= \min\left[\hat{c}, \max\left(0, 2\left(\lambda_{k+} - \lambda_{k-}\right)\right)\right] && \text{for} \quad k = 1,4.\end{aligned} \tag{5.25}$$

Such that the modified absolute eigenvalues $\left|\hat{\lambda}_k\right|^*$ can be expressed as:

$$\begin{aligned}\left|\hat{\lambda}_k\right|^* &= \left|\hat{\lambda}_k\right| &&\text{for } \left|\hat{\lambda}_k\right| \geq \tfrac{\delta\lambda_k}{2},\\ \left|\hat{\lambda}_k\right|^* &= \tfrac{(\hat{\lambda_k})^2}{\delta\lambda_k} + \tfrac{\delta\lambda_k}{4} &&\text{for } \left|\hat{\lambda}_k\right| < \tfrac{\delta\lambda_k}{2}.\end{aligned} \tag{5.26}$$

5.1.6 Quadrature rules

In order to numerically solve the system of equations (5.6) the edge and volume integrals have to be approximated. For this purpose we use *quadrature rules*. In [16] it was shown that the quadrature rules that are used should be exact for polynomials of degree $2p+1$ in case of the edges and of degree $2p$ in case of the interior, for the full method to keep its formal order of accuracy. Here p is the earlier mentioned order of the approximating polynomials.

The set of quadrature rules that is most accurate with the least amount of support points is the well-known set of Gaussian quadrature rules. The amount of support points or nodes that is required depends upon the accuracy that needs to be obtained. For the Gaussian rules one requires n nodes to exactly integrate a polynomial of degree $2n-1$. This means that for integrating a polynomial of degree $2p+1$, which is the requirement for the edge integrals, we need $p+1$ nodes. Such that for $p=1$ it suffices to use the following two-point Gaussian rule:

$$\int_{E_i} \boldsymbol{g}(x)dx = \frac{\Delta x}{2}\left[\boldsymbol{g}\left(x_{i-\frac{1}{6}\sqrt{3}}\right) + \boldsymbol{g}\left(x_{i+\frac{1}{6}\sqrt{3}}\right)\right], \tag{5.27}$$

and for the $p=2$ case we can use the following three-point version:

$$\int_{E_i} \boldsymbol{g}(x)dx = \frac{\Delta x}{18}\left[5\boldsymbol{g}\left(x_{i-\frac{1}{10}\sqrt{15}}\right) + 8\boldsymbol{g}(x_i) + 5\boldsymbol{g}\left(x_{i+\frac{1}{10}\sqrt{15}}\right)\right], \tag{5.28}$$

Similar to the notation for the locations of the cell faces here the locations are expressed relative to the cell center, thus $x_{i-\frac{1}{6}\sqrt{3}} = x_i - \frac{\sqrt{3}}{6}\Delta x$. The integrals over the interior of the elements are approximated using tensor-products of the approximations over the edges (5.27) and (5.28). This leads to a four-point rule for the $p=1$ case and a nine-point rule for the $p=2$ case. Note that we thus use quadrature rules that are exact for polynomials of degree $2p+1$ for the interior integrals, which is higher than the required $2p$.

5.2 Runge-Kutta method

For the time discretization we use the so-called explicit Runge-Kutta time-marching method which was first introduced for discontinuous Galerkin (the RKDG method) by Cockburn and Shu [16, 18, 20]. The system to be solved can be written as follows:

$$\frac{d}{dt}\boldsymbol{q}_h = \boldsymbol{L}_h\left(\boldsymbol{q}_h\right), \tag{5.29}$$

where the term $\boldsymbol{L}_h\left(\boldsymbol{q}_h\right)$ represents the right-hand-side of the system of equations to be solved. The Runge-Kutta method now discretizes the time derivative present in this system.

5.2.1 Time discretization

We assume our temporal domain $[0,T]$ to be partitioned into discrete time levels $\{t^n\}_{n=0}^N$, with time steps $\Delta t^n = t^{n+1} - t^n$, $n = 0,\ldots,N-1$. Further we assume that the initial state $\boldsymbol{q}_h(x,y,t=0) = \boldsymbol{q}_h^0$ is given. Our time-marching algorithm then looks as follows:

$$\begin{aligned}\boldsymbol{q}_h^{(0)} &= \boldsymbol{q}_h^n,\\ \boldsymbol{q}_h^{(k)} &= \sum_{m=0}^{k-1}\left\{\alpha_{km}\boldsymbol{q}_h^{(m)} + \beta_{km}\Delta t^{(m)}\boldsymbol{L}_h\left(\boldsymbol{q}_h^{(m)}\right)\right\}, \quad k=1,\ldots,p+1,\\ \boldsymbol{q}_h^{n+1} &= \boldsymbol{q}_h^{(p+1)},\end{aligned} \tag{5.30}$$

where the coefficients α_{km} and β_{km} are given in table 5.1. Do not confuse the superscripts (k) with the superscripts used for the different unknowns in $\boldsymbol{q}_h$. It can further be noted that these type of Runge-Kutta methods, which use the intermediate solution values $\boldsymbol{q}_h^{(m)}$ and are often called *Shu-Osher* Runge-Kutta schemes, can be derived from the more standard or *general* Runge-Kutta methods quite easily. If we denote the matrix composed of all α_{km} with $\boldsymbol{A}$ and the matrix of all β_{km} with $\boldsymbol{B}$, then we can introduce the matrix $\boldsymbol{C}$ with components γ_{km}, such that $\boldsymbol{B} = \boldsymbol{C} - \boldsymbol{AC}$. The *general* explicit Runge-Kutta methods can now be written as:

$$\begin{aligned}
\boldsymbol{q}_h^{(0)} &= \boldsymbol{q}_h^n, \\
\boldsymbol{q}_h^{(k)} &= \boldsymbol{q}_h^{(0)} + \sum_{m=0}^{k-1} \left\{ \gamma_{km} \Delta t^{(m)} \boldsymbol{L}_h \left(\boldsymbol{q}_h^{(m)} \right) \right\}, \quad k = 1, \ldots, p+1, \\
\boldsymbol{q}_h^{n+1} &= \boldsymbol{q}_h^{(p+1)}.
\end{aligned} \tag{5.31}$$

order	α_{km}			β_{km}		
2	1	-		1	-	
$(p=1)$	1/2	1/2		0	1/2	
3	1	-	-	1	-	-
$(p=2)$	3/4	1/4	-	0	1/4	-
	1/3	0	2/3	0	0	2/3

Table 5.1: Coefficients for the Runge-Kutta scheme

In order to keep the temporal discretization of the same order as the spatial discretization we use here the index p which we introduced before as the order of the approximating polynomial ($p = 1$ for linear basis functions and $p = 2$ for quadratic basis functions). This implies that when the linear basis functions are used we obtain the second order Runge-Kutta scheme, i.e., both the temporal and spatial discretization are second-order accurate. And in the case of quadratic basis functions we obtain the third-order Runge-Kutta scheme such that both the temporal and spatial schemes are third-order accurate.

5.2.2 Stability

In [15] it has been shown that for the scheme described above to be numerically stable the following condition has to be satisfied:

$$c\frac{\Delta t}{\Delta x} \leq \frac{1}{2p+1}, \tag{5.32}$$

where c is maximum wave speed present in the problem treated. In practice this wave speed is often estimated based on the initial flow conditions of the problem under consideration. This roughly estimated wave speed will then be used during the whole series of calculations. Another, and more reliable, approach would be to calculate after each time step the maximum wave speed and use this one for the next calculations. Since we require our solver to be as efficient as possible we shall not go that far.

5.3 Slope limiting

It is a known fact that any second or high-order accurate numerical scheme leads to oscillations near flow discontinuities (see [23]). To avoid these errors we need to introduce a nonlinear dissipative mechanism into our numerical scheme. Following the work of Van Leer [33, 34], Cockburn and others have applied the so-called *local projection limiting* or *slope limiting* technique to the discontinuous Galerkin method. This approach works fine in the case of single-fluid flows. However, when we introduce this limiter to a two-fluid solver (in our case the LS method), spurious oscillations occur. To avoid these errors the *simple fix* developed in chapter 4 has to be extended to the slope limiter, resulting in a novel two-fluid slope limiter for DG.

5.3.1 TVDM slope limiting for DG

In order to develop the required two-fluid slope limiter we first need a thorough understanding of the original single-fluid slope limiter for DG. Following the work of Cockburn *et al.* we will apply TVDM (Total Variation Diminishing in the Means) slope limiting to the DG approximation. This TVDM slope limiting ensures that the accuracy of the numerical method does not degrade to first-order accuracy, which is what a slope limiter formally tends to do locally. The limiter used here is based upon the one-dimensional limiter developed in [15]. In the case of linear basis functions ($p = 1$) the approximate solution can be written as:

$$\boldsymbol{q}_h(x,y,t) = \overline{\boldsymbol{q}}_{i,j} + (x - x_i)\,\frac{\overline{\Delta_x \boldsymbol{q}}_{i,j}}{\Delta x} + (y - y_j)\,\frac{\overline{\Delta_y \boldsymbol{q}}_{i,j}}{\Delta y} + \cdots. \qquad (5.33)$$

In order to implement slope limiting this approximation is replaced by its limited version indicated by the $*$:

$$\boldsymbol{q}^*_h(x,y,t) = \overline{\boldsymbol{q}}_{i,j} + (x - x_i)\,\frac{\overline{\Delta_x \boldsymbol{q}}^*_{i,j}}{\Delta x} + (y - y_j)\,\frac{\overline{\Delta_y \boldsymbol{q}}^*_{i,j}}{\Delta y} + \cdots. \qquad (5.34)$$

Here we introduced the limited slopes $\overline{\Delta_x \boldsymbol{q}}^*_{i,j}$ and $\overline{\Delta_y \boldsymbol{q}}^*_{i,j}$, which are defined as:

$$\overline{\Delta_x \boldsymbol{q}}^*_{i,j} = \mathrm{m}\left(\overline{\Delta_x \boldsymbol{q}}_{i,j}, \frac{\overline{\boldsymbol{q}}_{i+1,j} - \overline{\boldsymbol{q}}_{i,j}}{\lambda}, \frac{\overline{\boldsymbol{q}}_{i,j} - \overline{\boldsymbol{q}}_{i-1,j}}{\lambda}\right), \qquad (5.35)$$

$$\overline{\Delta_y \boldsymbol{q}}^*_{i,j} = \mathrm{m}\left(\overline{\Delta_y \boldsymbol{q}}_{i,j}, \frac{\overline{\boldsymbol{q}}_{i,j+1} - \overline{\boldsymbol{q}}_{i,j}}{\lambda}, \frac{\overline{\boldsymbol{q}}_{i,j} - \overline{\boldsymbol{q}}_{i,j-1}}{\lambda}\right), \qquad (5.36)$$

where $\mathrm{m}(a_1, \ldots, a_\nu)$ is the well known *minmod* function defined as follows:

$$\mathrm{m}(a_1, \ldots, a_\nu) = \begin{cases} s \cdot \min_{1\le n\le \nu} |a_n| & \text{if} \quad s = \mathrm{sign}(a_1) = \cdots = \mathrm{sign}(a_\nu), \\ 0 & \text{otherwise.} \end{cases} \qquad (5.37)$$

When we take $\lambda = 1$ this is the slope limiter used in the MUSCL schemes of Van Leer [33, 34]. A less restrictive limiter is obtained when we use $\lambda = \frac{1}{2}$. Although other limiters could be used too, one could for example think of the limiter by Koren [30], here only the 'simple' limiter stated above shall be applied. It is interesting however to investigate whether the accuracy properties of the RKDG method could be improved with other limiters. Note that the cross derivative in (5.12) is not affected by the limiting, which complies with the orthogonality property.

The limiter described above only holds for the linear case since all higher-order derivatives are neglected. A general limiter that also works for higher-order approximations can be derived from the above quite straightforwardly. This generalized limiter makes use of the so-called *limited cell face values* of the approximate solution. For the x direction these cell face values read:

$$\boldsymbol{q}^*_{h,W} = \overline{\boldsymbol{q}}_{i,j} - \frac{1}{2}\mathrm{m}\left(2\left(\overline{\boldsymbol{q}}_{i,j} - \boldsymbol{q}_{h,W}\right), \frac{\overline{\boldsymbol{q}}_{i+1,j} - \overline{\boldsymbol{q}}_{i,j}}{\lambda}, \frac{\overline{\boldsymbol{q}}_{i,j} - \overline{\boldsymbol{q}}_{i-1,j}}{\lambda}\right), \qquad (5.38)$$

$$\boldsymbol{q}^*_{h,E} = \overline{\boldsymbol{q}}_{i,j} + \frac{1}{2}\mathrm{m}\left(2\left(\boldsymbol{q}_{h,E} - \overline{\boldsymbol{q}}_{i,j}\right), \frac{\overline{\boldsymbol{q}}_{i+1,j} - \overline{\boldsymbol{q}}_{i,j}}{\lambda}, \frac{\overline{\boldsymbol{q}}_{i,j} - \overline{\boldsymbol{q}}_{i-1,j}}{\lambda}\right). \qquad (5.39)$$

Here $\boldsymbol{q}_{h,E}$ and $\boldsymbol{q}_{h,W}$ are the original values of the approximate solution at cell face e_E and e_W, i.e. the *right* and *left* cell faces respectively, which are obtained by substitution of the locations of the centers of the respective cell faces. Similar expressions can be found for the y direction, with the only difference that now the cell face values are located at the *bottom* $\boldsymbol{q}_{h,S}$ and *top* $\boldsymbol{q}_{h,N}$ of the cell. See figure 5.2 for an overview of these cell face values. Note that expressions (5.38) and (5.39) indeed use information of the higher-order derivatives, since they are used to obtain the original cell face values $\boldsymbol{q}_{h,e}$.

Now the limited cell face values are defined we can introduce the generalized higher-order slope limiter. The limiting algorithm for this limiter in x direction reads as follows:

1. Compute the limited cell-face values $\boldsymbol{q}^*_{h,W}$ and $\boldsymbol{q}^*_{h,E}$ using (5.38) and (5.39),

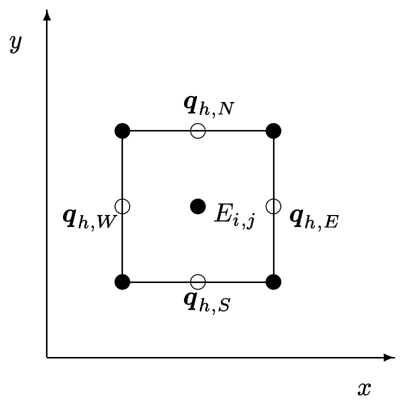


Figure 5.2: A rectangular finite element $E_{i,j}$ with left $\boldsymbol{q}_{h,W}$, right $\boldsymbol{q}_{h,E}$, bottom $\boldsymbol{q}_{h,S}$ and top $\boldsymbol{q}_{h,W}$ cell face values. These values are obtained by substitution of the locations of the cell face centers into the expression for $\boldsymbol{q}_h(x, y, t)$.

2. If this leads to the original quadratic approximation, i.e. $\boldsymbol{q}^*_{h,W} = \boldsymbol{q}_{h,W}$ and $\boldsymbol{q}^*_{h,E} = \boldsymbol{q}_{h,E}$, no limiting is applied,
3. If not, take the limited linear approximation, i.e. calculate $\overline{\Delta_x \boldsymbol{q}}^*_{i,j}$ and $\overline{\Delta_y \boldsymbol{q}}^*_{i,j}$ from (5.35) and (5.36) and set all higher-order derivatives equal to zero .

Running some experiments with simple one-dimensional test cases has shown that the best results are obtained when λ is taken equal to 1 in step 1. and $\lambda = \frac{1}{2}$ in step 3. In words; the restrictive limiter is used to determine whether one should limit or not (do either 2. or 3.), but when it is decided that limiting is applied (do 3.), the less restrictive limiter is used.

5.3.2 TVBM slope limiting for DG

It is important to notice that the generalized TVDM limiter for third and higher-order approximations introduced above degrades the accuracy of a higher-order method to that of a second-order method when limiting is applied [3]. When the limiter decides to take the limited linear approximation we do nothing less than ignoring the higher-order derivatives. This means that for problems which require much limiting it is almost impossible to obtain a higher than second-order accuracy. This property of our TVDM limiter will also be shown in the numerical results. To avoid this Shu [52] has introduced the TVBM (Total Variation Bounded in the Means) slope limiter.

This TVBM generalized slope limiter makes use of a TVB corrected *minmod* function which is defined as follows:

$$\overline{m}(a_1, \ldots, a_m) = \begin{cases} a_1 & \text{if} \quad |a_1| \le M(\Delta x)^2, \\ m(a_1, \ldots, a_m) & \text{otherwise,} \end{cases} \tag{5.40}$$

where M is a constant equal to the upper bound of the absolute value of the second-order derivative of the solution at local extrema. For piecewise C^2 initial data this constant can be expressed as follows:

$$M = \sup\{|\boldsymbol{q}^0_{xx}(y)|, y : \boldsymbol{q}^0_x(y) = 0\}. \tag{5.41}$$

It is clear that the constant M is problem dependent. And even within one problem several local extrema can exist with different values for the local second-order derivatives. It is especially this that makes this approach not very suitable for the implementation in a robust and general solver.

[3] A limiter actually makes a method locally fi rst-order accurate. By stating that the higher-order methods are degraded to second-order accuracy we mean that when limiting is applied the higher-order solution becomes equal to the second-order limited solution.

Another disadvantage of this approach is that it only works well when, for systems of equations, limiting is applied using the characteristic variables. These characteristic variables are the normal system components transformed to the characteristic space spanned by the eigenvectors of the Jacobian of the system. Unfortunately these transformation procedures require a large amount of computational power. Especially in the DG framework, where many unknowns per grid cell have to be calculated, this procedure would require too much computations and it shall therefore not be applied here. Unfortunately this means that the higher-order solution will be limited using the TVDM limiter which degrades the order of accuracy to that of the second-order solution (or lower). For problems which require a lot of limiting it is therefore not very useful to use a higher-than-second-order method. But, when a problem contains highly structured smooth regions, the higher order does pay-off.

5.3.3 Pressure oscillations due to limiter

It has been shown in chapter 4 that a numerical two-fluid solver based on the LS method suffers from spurious pressure oscillations due to an inconsistency in the pressure update. Because the ratio of specific heats jumps over an interface, undesired errors in the pressure distribution were introduced. To prevent these errors we introduced a simple fix that makes the numerical method *blind* for other fluids. During an update of the pressure the method only uses the properties of one fluid instead of two. Results obtained with a finite volume based solver have shown the competence of this simple fix.

Unfortunately, test results obtained using the DG method developed in this chapter are not oscillation free [4]. In figure 5.3 numerical results of the by now well-known *translating interface* test problem are given. As was done before, the initial properties of both fluids are taken to be equal to $(\rho, u, p, \gamma)_L = (1.0, 1.0, 1.0, 1.4)$ and $(\rho, u, p, \gamma)_R = (0.125, 1.0, 1.0, 1.6)$. Although small, one can clearly see the oscillations present in the state variable distributions. These oscillations are comparable to the spurious pressure oscillations described above. They are not present in the single-fluid case which indicates that they are induced by jumps in the distribution of the ratio of specific heats.

When we use this information and take a closer look at the limiter we introduced above it is not difficult to notice the origin of these oscillations. The limiter as we use it *looks out of the cell*; it uses information from the neighboring grid cells, $\overline{\boldsymbol{q}}_{i\pm1,j}$, to determine which gradient to use. This means that when one of the cells under consideration contains the interface, i.e. the ratio of specific heats jumps ($\gamma_{i,j} \neq \gamma_{i-1,j}$ or $\gamma_{i,j} \neq \gamma_{i+1,j}$), the total energies in these cells are calculated from the pressure ($\rho E = \frac{p}{\gamma-1} + \frac{1}{2}\rho(u^2+v^2)$) with different values for γ. The limited solution thus becomes dependent upon the fluid properties of two fluids instead of one.

5.3.4 Two-fluid slope limiting for DG

To prevent these spurious oscillations we borrow from the simple fix we have discussed before. Also in the slope limiter we apply a fix that prevents the limiter from *seeing* two different values for γ. Since the oscillations result from the equation for the total energy this fix is only applied to the approximation of the total energy ρE. The original slope limiter for the energy equation reads as follows:

$$\rho E_h^* = \overline{\rho E}_{i,j} + (x - x_i)\,\frac{\overline{\Delta_x \rho E}^*_{i,j}}{\Delta x} + (y - y_j)\,\frac{\overline{\Delta_y \rho E}^*_{i,j}}{\Delta y} + \cdots, \tag{5.42}$$

with the limited slopes:

$$\overline{\Delta_x \rho E}^*_{i,j} \;=\; \mathrm{m}\left(\overline{\Delta_x \rho E}_{i,j}, \frac{\overline{\rho E}_{i+1,j} - \overline{\rho E}_{i,j}}{\lambda}, \frac{\overline{\rho E}_{i,j} - \overline{\rho E}_{i-1,j}}{\lambda}\right), \tag{5.43}$$

$$\overline{\Delta_y \rho E}^*_{i,j} \;=\; \mathrm{m}\left(\overline{\Delta_y \rho E}_{i,j}, \frac{\overline{\rho E}_{i,j+1} - \overline{\rho E}_{i,j}}{\lambda}, \frac{\overline{\rho E}_{i,j} - \overline{\rho E}_{i,j-1}}{\lambda}\right), \tag{5.44}$$

[4]Since at this point we have not yet treated the discretization of the LS equation and the implementation of the LS method into the two-fluid solver, it suffices to mention that the FV approach used in [39] is followed. So the Euler equations are discretized using the DG method discussed here, while the LS equation is treated in a FV manner.

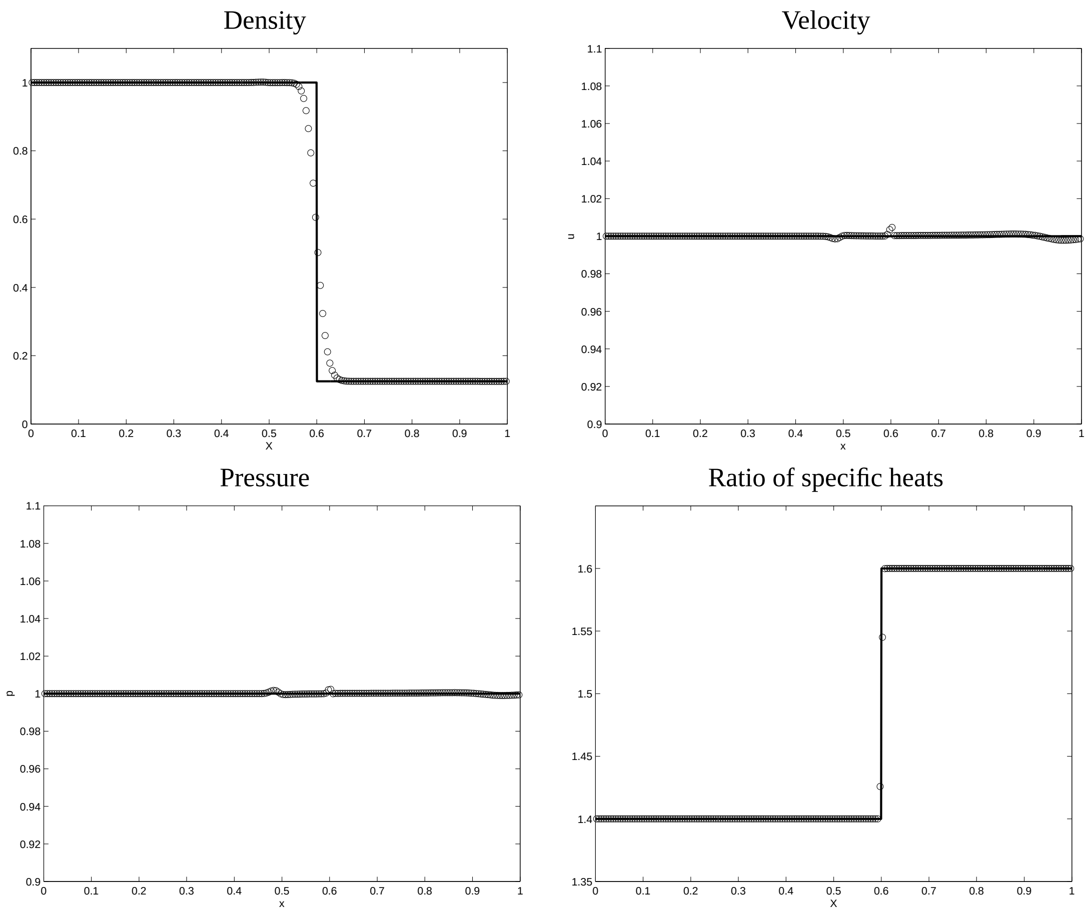


Figure 5.3: 1D translating interface problem solved with DG method ($p = 1$). (ρ, u, p, γ) at $t = 0.1$. Grid has 200 cells and $\Delta t/\Delta x = 0.1667$. Solid lines are exact solutions, dotted lines are numerical solutions.

Spurious oscillations occur when this limiter uses two different values for γ during the calculations. It is clear that only the terms $\overline{\rho E}_{i-1,j}$, $\overline{\rho E}_{i+1,j}$, $\overline{\rho E}_{i,j-1}$ and $\overline{\rho E}_{i,j+1}$ can be responsible for such a situation. Whenever an interface is present in the cells under consideration, such that for example $\gamma_{i,j} \neq \gamma_{i-1,j}$, these expressions will introduce different values of γ into the limiter, resulting in the spurious oscillations visible in 5.3.

To prevent the solution from becoming oscillatory we require the limiter to use only $\gamma_{i,j}$ instead of $\gamma_{i\pm1,j}$. Therefore we replace the original expressions for the total energy in the neighboring cells with *fixed* expressions: $\overline{\rho E}^*_{i\pm1,j} = \overline{\rho E}_{i\pm1,j}(\gamma_{i,j})$, and the same in the y direction. Given the expression for the total energy $\rho E = \frac{p}{\gamma-1} + \frac{1}{2}\rho(u^2+v^2)$ these expressions for the x direction can be written as follows:

$$\overline{\rho E}^*_{i-1,j} = \overline{\rho E}_{i-1,j} + \overline{p}_{i-1,j}\left[\left(\frac{1}{\gamma-1}\right)_{i,j} - \left(\frac{1}{\gamma-1}\right)_{i-1,j}\right], \tag{5.45}$$

$$\overline{\rho E}^*_{i+1,j} = \overline{\rho E}_{i+1,j} + \overline{p}_{i+1,j}\left[\left(\frac{1}{\gamma-1}\right)_{i,j} - \left(\frac{1}{\gamma-1}\right)_{i+1,j}\right], \tag{5.46}$$

where $\overline{p}_{i,j}$ is the mean pressure that can easily be calculated from the mean total energy. Numerical results in chapter 8 show that the simple fix of chapter 4 extended with this novel two-fluid slope limiter provides oscillation free results.

It is clear that, similar to the simple fix applied in the flux calculation and the pressure update, the *fixed* slope limiter developed above also introduces numerical errors that are not present in the original single-fluid slope limiter. From (5.45) and (5.46) it immediately follows that the difference between using the *correct* value of γ and the *fixed* value of γ is equal to:

$$\overline{\rho E}^*_{i\pm1,j} - \overline{\rho E}_{i\pm1,j} = \overline{p}_{i\pm1,j}\Delta\left(\frac{1}{\gamma-1}\right), \tag{5.47}$$

which is similar to the expression we found in chapter 4. Given the severity of the errors when this fix would not be present, the magnitude of the errors introduced by the fix can be neglected.

5.3.5 Why not in FV?

It is interesting to compare the two-fluid flow solver based on the DG method we have developed so far with the more familiar solver based on the FV method (see [39]). In the results of the latter the secondary pressure oscillations discussed above were not present while in the former they obviously are, requiring a novel two-fluid slope limiter. This can be explained by the way both methods obtain and use the flow information present in each cell. The FV method only uses cell-centered values. Each state variable is assumed to have a piecewise constant distribution. Slope limiting comes in when one wants to obtain cell face values of these state variables, which in turn are used to calculate the cell face fluxes using a Riemann solver. The FV method thus uses slope limiting as an interpolation method.

The DG method already has these cell face values available. In each cell a certain number of unknowns is defined. The more unknowns the better the approximation to the exact solution. When the unknowns are known, the complete distribution in each cell is known and thus also the cell face values. These cell face values could be used directly to calculated the cell face fluxes. But to avoid oscillations we first apply slope limiting. This limiting does *not* produce interpolated values but completely changes the state variable distribution. The effect of the limiter is thus also sensed in the update of the state variables and not only in the flux calculation. It is this full dependency that makes the DG method sensitive to the pressure oscillations.

5.4 Boundary conditions

The boundary conditions for the Euler equations have been treated in chapter 2. Depending on the type of boundary conditions several flow variables have to be prescribed at the boundary under consideration.

Implementation of these boundary conditions is done by adding an extra row or column of grid cells, better known as *ghost cells*, to the edges of the computational domain. Within these ghost cells the required flow variables will be prescribed.

Due to the local character of the DG method at each boundary only one additional column or row of ghost cells is necessary. As opposed to the FV or FD methods, which often use a broader stencil, the DG method only requires information from one neighboring cell for the calculation of a higher-order numerical approximation.

Note that besides the state variables also the ratio of specific heats should be prescribed in the ghost cells. Both the approximate Riemann solver and the slope limiter require the ratio of specific heats of the neighboring cells.

Chapter 6

Discretization of the LS method

This chapter treats the discretization and implementation of the level-set method. This requires the discretization of the time-dependent advective LS equation for a given velocity field (obtained from solving the Euler equations). Further the redistancing procedure has to be implemented, which implies solving for the steady state of the redistancing equation. Special attention is paid to the discretization scheme for the LS equation. Since the application of a LS method to an Euler flow solver based on a DG discretization is a novelty, several choices have to be made concerning the type of discretization that is used. Decisive factors for the method of choice are the accuracy of the discretization and the ease of implementation into the numerical solver. The governing LS equation is used in its advective form (3.1), which reads:

$$\frac{\partial \varphi}{\partial t} + \boldsymbol{V} \cdot \nabla \varphi = \frac{\partial \varphi}{\partial t} + u\frac{\partial \varphi}{\partial x} + v\frac{\partial \varphi}{\partial y} = 0. \tag{6.1}$$

This scalar advection equation has been subject of extensive research, which resulted in a large amount of suitable discretization methods. Because of its ease of implementation we will separate the spatial and temporal discretization, allowing for the use of an explicit time-marching method. We shall first consider the discretization of the spatial part, followed by the choice of a suitable time-marching method for the temporal part. This chapter finishes with the discretization and implementation of the redistancing procedure.

6.1 Spatial discretization of the LS equation

A suitable discretization method for the spatial part of (6.1) will be chosen. Since the LS method is *effectively* decoupled from the RKDG method that solves the Euler equations, i.e. given the velocity field the LS equation can be solved independently, it is not necessary to use the same numerical approach for both methods. However, to ease the implementation of the LS method into the Euler solver we shall apply a discretization for the LS equation that uses the same grid as the RKDG method. This grid consists of uniform rectangular grid cells, allowing for the use of either a FV (piecewise constant approximation) or DG (piecewise higher-order approximation) approach. As was explained earlier, the former approach obtains its higher-order accuracy by using broader stencils for the calculation of the interface fluxes, while the latter makes use of higher-order basis functions to reach the same accuracy. To select between these two classes of methods the following additional requirements are used:

- At least second-order accurate advection of $\varphi = 0$ level-set,
- Use of given piecewise quadratic velocity-field,
- Easy implementation of redistancing procedure.

Probably the most important aspect of two-fluid flow simulation is the location of the interface. Since this interface location follows from the location of the zero LS contour ($\varphi = 0$) it is necessary that the advection of the LS function is as accurate as possible. Here we require the numerical method to be at least second-order accurate.

The second requirement concerns the velocity field that is obtained by solving the Euler equations. The RKDG method delivers a velocity distribution which is a piecewise higher-order approximation (linear or higher-order polynomials). Different from the FV-type LS method used in [39] we can thus make use of a velocity field which contains more information; besides a mean value the solution in a cell also contains derivatives. This highly accurate velocity field seems advantageous for obtaining a higher-order accurate numerical scheme for the LS equation. One can think of a DG discretization which uses the same polynomial basis for the LS function and the velocity field. As such the derivatives of the velocity field are used to their full extent.

Unfortunately, the resulting LS distribution is redistanced after each time step. As was mentioned before, this redistancing changes the LS distribution such that its gradient $\nabla\varphi$ is equal to 1 everywhere. The higher-order derivatives are clearly set to zero in this procedure. Using a DG method that uses other than piecewise constant or linear approximations is therefore not necessary [1]. Hence, our class of possible methods is reduced to the higher-than-second-order FV methods and the DG methods using piecewise linear basis functions.

The final decision is made after considering the implementation of the redistancing procedure. For convenience the discretization of the redistancing procedure is taken from [39]. The discretization scheme used here is based on a second-order FV approximation. For an easy insertion of this discretization into the LS method it is advantageous to use a FV approach for the discretization of the LS equation too. This does imply that only the cell-mean values of the velocity distribution are used. Choosing a DG method for the LS equation would require the development of a DG method for the redistancing equation too. Although this provides an interesting research topic, this shall not be addressed here.

6.1.1 FV discretization

For the discretization of the LS equation we have chosen the well-established FV approach. The LS function is assumed to be piecewise constant. Also the velocity field is reduced to a piecewise constant distribution by neglecting all the higher-order derivatives. The corresponding discretized LS equation can be derived from (6.1) by integration over an element $E_{i,j} = \left(x_{i-\frac{1}{2}}, x_{i+\frac{1}{2}}\right) \times \left(y_{j-\frac{1}{2}}, y_{j+\frac{1}{2}}\right)$:

$$\int_{E_{i,j}} \frac{\partial\varphi}{\partial t} d\Omega + \int_{E_{i,j}} u\frac{\partial\varphi}{\partial x} + v\frac{\partial\varphi}{\partial y} d\Omega = 0. \tag{6.2}$$

Applying Gauss' divergence theorem to the integral term containing the spatial derivatives leads to:

$$\frac{\partial}{\partial t}\int_{E_{i,j}} \varphi d\Omega + u_{i,j}\Delta y \int_{x_{i-\frac{1}{2}}}^{x_{i+\frac{1}{2}}} \varphi dx + v_{i,j}\Delta x \int_{y_{j-\frac{1}{2}}}^{y_{j+\frac{1}{2}}} \varphi dy = 0, \tag{6.3}$$

where $u_{i,j}$ and $v_{i,j}$ are the cell-mean values of the velocities in x and y directions respectively. Taking the time derivative out of the integral, which is allowed for a time-independent grid and introducing the cell-mean value $\varphi_{i,j} = \frac{1}{\Delta x \Delta y}\int_{E_{i,j}} \varphi d\Omega$ finally leads to the FV discretization of the LS equation:

$$\frac{\partial\varphi_{i,j}}{\partial t} = -\frac{u_{i,j}}{\Delta x}\left[\varphi_{i+\frac{1}{2},j} - \varphi_{i-\frac{1}{2},j}\right] - \frac{v_{i,j}}{\Delta y}\left[\varphi_{i,j+\frac{1}{2}} - \varphi_{i,j-\frac{1}{2}}\right] = 0. \tag{6.4}$$

It is clear that the x and y directions can be split into separate parts. From now on we will only consider the x direction. Since φ is approximated with a piecewise constant distribution the cell-face values $\varphi_{i\pm\frac{1}{2},j}$ are not uniquely defined. Similar to what was done for the Euler equations we therefore apply *upwinding*. Depending on the flow direction we choose the physically correct value of φ that should be used. Simply taking the upwind values of φ results in a first-order accurate discretization. Since we require our method to be at least second-order accurate this does not suffice. Therefore a broader interpolation stencil is used

[1]An interesting topic of further research would be to see whether it is possible to create an advection scheme that is able to use the highly detailed velocity fi eld obtained from a DG method.

that is higher-order accurate. It has been mentioned that a second-or-higher-order method for hyperbolic problems always results in undesired oscillations requiring the use of a non-linear dissipative mechanism, i.e. slope limiting.

6.1.2 Two slope limiters

In order to obtain a stable higher-order accurate numerical method for the spatial part of the LS equation a slope limiter has to be introduced. This slope limiter calculates the physically correct cell-face values of the LS function $\varphi_{i\pm\frac{1}{2},j}$ and $\varphi_{i,j\pm\frac{1}{2}}$ by taking into account the flow direction, without producing undesired oscillations near discontinuities. Here two different slope limiters shall be considered. In the following section both limiters will be compared using the *Molenkamp* test problem, sometimes also referred to as the *solid body rotation* problem or the *Crowley* test, taken from [59].

Minmod limiter

The well-known *minmod limiter* reads as follows (only the x direction is shown):

$$\text{if} \quad u_{i,j} \geq 0 \quad \text{then:} \quad \begin{cases} \varphi_{i+\frac{1}{2},j} = \varphi_{i,j} + \frac{\Delta\varphi_{i,j}}{2}, \\ \varphi_{i-\frac{1}{2},j} = \varphi_{i-1,j} + \frac{\Delta\varphi_{i-1,j}}{2}, \end{cases} \tag{6.5}$$

$$\text{if} \quad u_{i,j} < 0 \quad \text{then:} \quad \begin{cases} \varphi_{i+\frac{1}{2},j} = \varphi_{i+1,j} - \frac{\Delta\varphi_{i+1,j}}{2}, \\ \varphi_{i-\frac{1}{2},j} = \varphi_{i,j} - \frac{\Delta\varphi_{i,j}}{2}. \end{cases} \tag{6.6}$$

Here $\Delta\varphi_{i,j}$ is defined as:

$$\Delta\varphi_{i,j} = \mathrm{m}\left(\varphi_{i+1,j} - \varphi_{i,j}, \varphi_{i,j} - \varphi_{i-1,j}\right), \tag{6.7}$$

with $\mathrm{m}\left(a_1, \ldots, a_\nu\right)$ the *minmod* function defined in (5.37). This limiter results in a second-order accurate scheme.

Koren's limiter

A formally third-order accurate scheme is obtained by using the limiter developed by Koren [30]. This limiter is derived from the $\kappa = \frac{1}{3}$ scheme. This limiter is defined as:

$$\text{if} \quad u_{i,j} \geq 0 \quad \text{then:} \quad \begin{cases} \varphi_{i+\frac{1}{2},j} = \varphi_{i,j} + \frac{1}{2}\Phi\left(r_{i+\frac{1}{2},j}\right)\left[\varphi_{i,j} - \varphi_{i-1,j}\right], \\ \varphi_{i-\frac{1}{2},j} = \varphi_{i-1,j} + \frac{1}{2}\Phi\left(r_{i-\frac{1}{2},j}\right)\left[\varphi_{i-1,j} - \varphi_{i-2,j}\right], \end{cases} \tag{6.8}$$

$$\text{if} \quad u_{i,j} < 0 \quad \text{then:} \quad \begin{cases} \varphi_{i+\frac{1}{2},j} = \varphi_{i+1,j} - \frac{1}{2}\Phi\left(r_{i+\frac{1}{2},j}\right)\left[\varphi_{i+1,j} - \varphi_{i+2,j}\right], \\ \varphi_{i-\frac{1}{2},j} = \varphi_{i,j} - \frac{1}{2}\Phi\left(r_{i-\frac{1}{2},j}\right)\left[\varphi_{i,j} - \varphi_{i+1,j}\right], \end{cases} \tag{6.9}$$

where:

$$r_{i+\frac{1}{2},j} = \frac{\varphi_{i+1,j} - \varphi_{i,j}}{\varphi_{i,j} - \varphi_{i-1,j}}, \tag{6.10}$$

and:

$$\Phi\left(r\right) = \max\left(0, \min\left(2r, \min\left(\frac{1}{3} + \frac{2}{3}r, r\right)\right)\right). \tag{6.11}$$

6.1.3 Molenkamp test

In order to compare the two limiters and their corresponding numerical schemes the advection equation is applied to the *Molenkamp* problem. This test case considers an initial scalar distribution (in our case the distribution of the LS function), which is advected in a circular velocity field with $u = -wy$, $v = wx$ and $w = 2\pi$. On the inflow boundaries the initial LS distribution is prescribed. See figure 6.1 for a more detailed description of this problem. After one full rotation the resulting LS distribution can be compared to the initial distribution, since the exact solution is identical to the initial solution. Note that no redistancing is used. Time-marching is obtained using the explicit Runge-Kutta 4 scheme; see the next section for a discussion of this time-marching method. Performing the calculations for both numerical schemes on several grids results in the plots shown in figures 6.2 and 6.3. Note that the exact solution is the so-called *exact-discrete* solution, i.e. the exact solution plotted on the discrete grid.

It becomes immediately clear that the limiter by Koren performs, as expected, significantly better than the minmod limiter. Especially on the coarser grids the minmod limiter shows a significant amount of numerical smearing. The interface, or zero LS contour, has almost completely disappeared and the other contours show a significant phase shift. Koren's limiter on the contrary produces even on the coarsest grid a solution that resembles the exact solution. The interface has not been smoothed out and the phase shift is negligible. It is clear that the limiter by Koren is the limiter of choice.

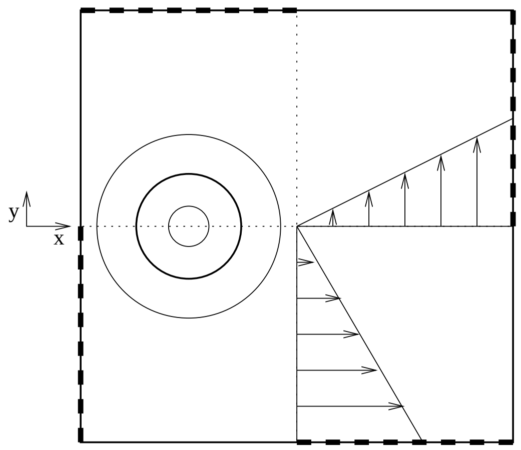


Figure 6.1: Molenkamp's test problem for advection schemes. An initial level-set distribution is advected by a circular velocity field with $u = -wy$, $v = wx$ and $w = 2\pi$. The dashed parts of the boundary are inflow boundaries and the non-dashed parts outflow boundaries.

6.2 Temporal discretization of the LS equation

The temporal part of the LS equation is discretized using a sufficiently accurate explicit time-marching scheme. For this we use the *standard* Runge-Kutta 4 scheme [2]. We assume our temporal domain $[0, T]$ to be partitioned into discrete time levels $\{t^n\}_{n=0}^{N}$, with time steps $\Delta t^n = t^{n+1} - t^n$, $n = 0, \ldots, N-1$. If we rewrite (6.4) as:

$$\frac{d}{dt}\varphi_{i,j} = L(\varphi_{i,j}), \tag{6.12}$$

where $L(\varphi_{i,j})$ represents the right-hand-side of (6.4), our RK algorithm looks as follows:

$$\begin{aligned} \varphi_{i,j}^{(0)} &= \varphi_{i,j}^{n}, \\ \varphi_{i,j}^{(k)} &= \varphi_{i,j}^{(0)} + \sum_{m=0}^{k-1} \left\{ \gamma_{km} \Delta t^{(m)} L_{i,j}\left(\varphi_{i,j}^{(m)}\right) \right\}, \quad k = 1, \ldots, 4, \\ \varphi_{i,j}^{n+1} &= \varphi_{i,j}^{(4)}, \end{aligned} \tag{6.13}$$

where the coefficients γ_{km} are given in table 6.1.

[2]There are several RK4 schemes that are fourth-order accurate. The difference between these schemes is the choice of coeffi cients. When we refer to the standard RK4 scheme, we mean the scheme with the coeffi cients given in table 6.1.

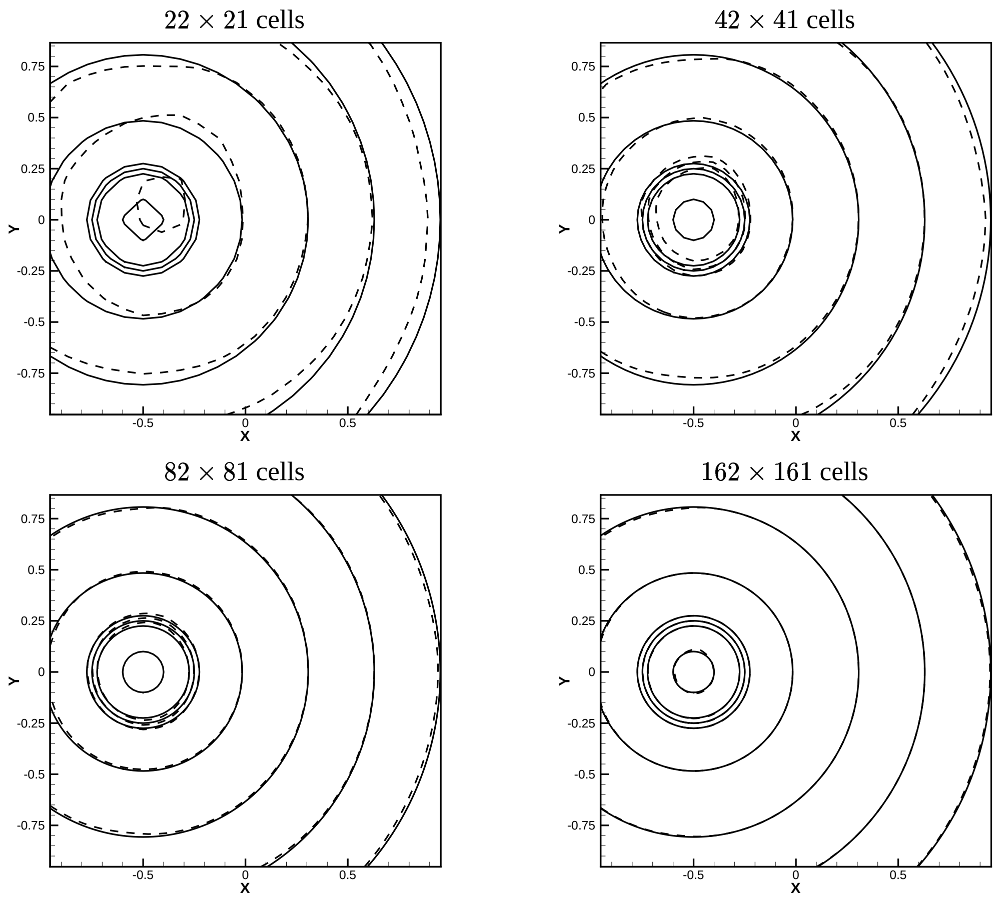


Figure 6.2: Results of the Molenkamp problem with minmod limiter. Plotted are several level-set contours. The three dense contours are respectively $\varphi = -0.025, 0, 0.025$. Grid has 100×200 cells and $\Delta t = 1 \times 10^{-3}$. $-$ line is 'exact' solution, $\cdots$ line is 'numerical' solution.

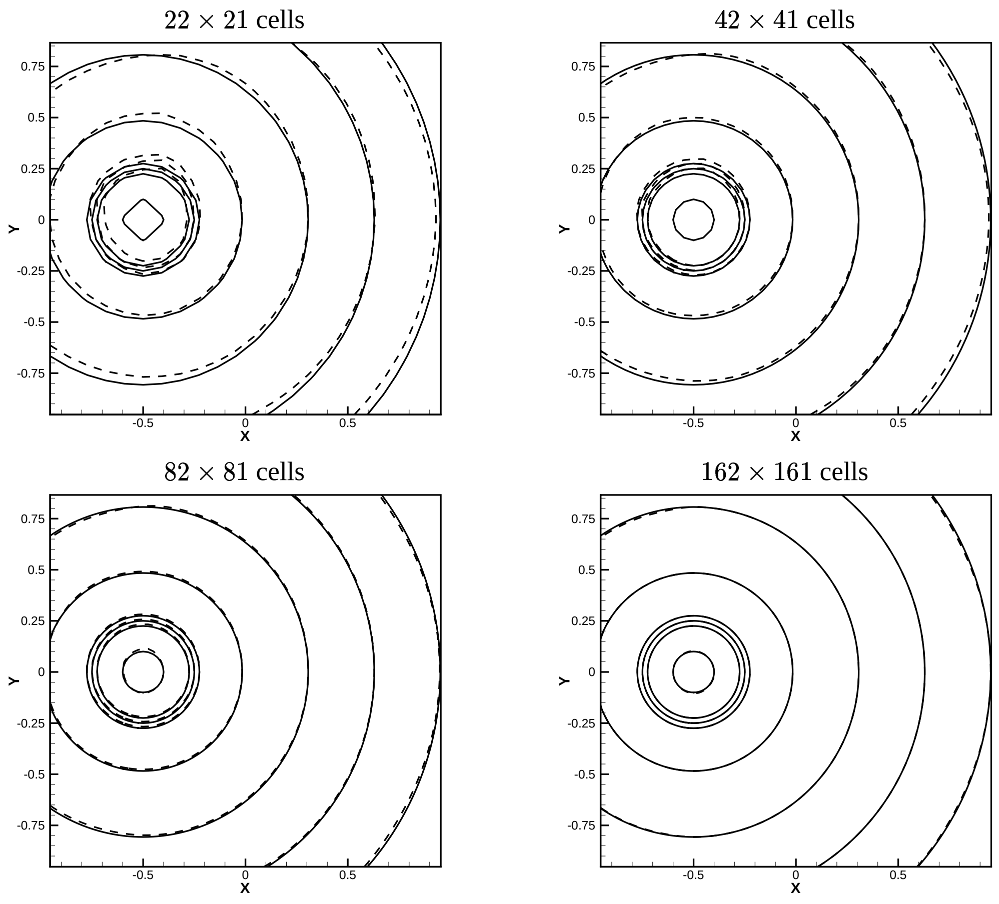


Figure 6.3: Results of the Molenkamp problem with Koren's limiter. Plotted are several level-set contours. The three dense contours are respectively $\varphi = -0.025, 0, 0.025$. Grid has 100×200 cells and $\Delta t = 1 \times 10^{-3}$. $-$ line is 'exact-discrete' solution, $\cdots$ line is numerical solution.

k \ m	0	1	2	3
1	1/2	-	-	-
2	0	1/2	-	-
3	0	0	1	-
4	1/6	1/3	1/3	1/6

Table 6.1: Coefficients for the *standard* Runge-Kutta 4 scheme

6.3 Discretization of the redistancing procedure

The redistancing procedure that changes the distorted level-set function back into a true signed distance function is governed by the PDE equation (6.14) as explained in chapter 3. Solving for this equation means looking for that distribution of φ such that its absolute gradient equal one everywhere; $|\nabla\varphi| = 1$. As was mentioned this means solving for the steady state of equation (3.20). The corresponding procedure can be written as:

$$\begin{aligned} d\left(\boldsymbol{x}, \tau=0\right) &= \varphi\left(\boldsymbol{x}, t\right), \\ \frac{\partial d}{\partial \tau} &= S\left(\varphi_0\right)\left(1-|\nabla d|\right), \\ \varphi\left(\boldsymbol{x}, t\right) &= d\left(\boldsymbol{x}, \tau=\infty\right), \end{aligned} \tag{6.14}$$

where the pseudo-time τ is used instead of the real time t to avoid confusion. The sign function is defined as $S_\epsilon\left(\varphi\right) = 2H_\epsilon\left(\varphi\right) - 1$. For an accurate representation of φ (and thus the interface) it does not suffice to use a simple first-order scheme for the numerical approximation of (6.14). Therefore a scheme is chosen that is second-order in both space and time. The update in time is chosen to be a RK2 scheme (see (6.13)). The value of d at a certain discrete pseudo-time level n_τ is updated to the new time level $n_\tau + 1$ using two intermediate stages:

$$\begin{aligned} d_{i,j}^{(0)} &= d_{i,j}^{n_\tau}, \\ d_{i,j}^{(1)} &= d_{i,j}^{(0)} + \frac{\Delta\tau}{2}\left[S\left(\varphi_0\right)\left(1-\left|\nabla d^{(0)}\right|\right)\right]_{i,j}, \\ d_{i,j}^{(2)} &= d_{i,j}^{(0)} + \Delta\tau\left[S\left(\varphi_0\right)\left(1-\left|\nabla d^{(1)}\right|\right)\right]_{i,j}, \\ d_{i,j}^{n_\tau+1} &= d_{i,j}^{(2)}. \end{aligned} \tag{6.15}$$

In the case of a uniform Cartesian grid, the spatial discretization of the gradient $\left|\nabla d^{(m_\tau)}\right|_{i,j}$ with $m_\tau = 1, 2$ can be written as:

$$|\nabla d|_{i,j} = \sqrt{\left(\frac{\Delta_x d}{\Delta x}\right)_{i,j}^2 + \left(\frac{\Delta_y d}{\Delta y}\right)_{i,j}^2}, \tag{6.16}$$

where (m_τ) is suppressed. A second-order approximation for the term $\Delta_x d$ is given by the following scheme:

$$\left(\Delta_x^L d\right)_{i,j} = d_{i,j} - d_{i-1,j} + \frac{1}{2}\,|\mathrm{min}|\left(d_{i+1,j} - 2d_{i,j} + d_{i-1,j}, d_{i,j} - 2d_{i-1,j} + d_{i-2,j}\right), \tag{6.17}$$

$$\left(\Delta_x^R d\right)_{i,j} = d_{i+1,j} - d_{i,j} - \frac{1}{2}\,|\mathrm{min}|\left(d_{i+1,j} - 2d_{i,j} + d_{i-1,j}, d_{i+2,j} - 2d_{i+1,j} + d_{i,j}\right), \tag{6.18}$$

$$w_x^L = S_\epsilon\left(\varphi_{i,j}\right)\left(\Delta_x^L d\right)_{i,j}, \tag{6.19}$$

$$w_x^R = S_\epsilon\left(\varphi_{i,j}\right)\left(\Delta_x^R d\right)_{i,j}, \tag{6.20}$$

$$\left(\Delta_x d\right)_{i,j} = \begin{cases} \left(\Delta_x^L d\right)_{i,j} & \text{if} \quad w_x^L > 0 \quad \text{and} \quad w_x^L + w_x^R > 0, \\ \left(\Delta_x^R d\right)_{i,j} & \text{if} \quad w_x^R < 0 \quad \text{and} \quad w_x^L + w_x^R < 0, \\ 0 & \text{if} \quad w_x^L < 0 \quad \text{and} \quad w_x^R > 0. \end{cases} \tag{6.21}$$

A similar scheme can be derived for $\Delta_y d$, which shall not be done here.

The second-order scheme used here requires information of five cells to determine the required differences. Near the boundaries this is a problem since there is only one virtual cell and two are needed.

Therefore in these cells the first-order approximation is used, which follows from (6.17) and (6.18) by removing the most right terms ($|\text{min}|\,(\ldots)$). What remains is a stability condition relating $\Delta\tau$, Δx and Δy. A safe choice will be: $\Delta t = \frac{1}{2}\text{min}\,(\Delta x, \Delta y)$, where we already took into account that the wavespeed of this problem is equal to one.

An advantage of the method used here is that the level-set function is redistanced in an outward direction from the interface. Because only the cells near the interface require an exact distance function, it is possible to take only a small amount of time steps in the redistancing procedure. It has been shown that six time steps will do.

6.4 Boundary conditions

The boundary conditions for the LS equation and for the redistancing equation are implemented in the same way as was done for the Euler equations. The computational domain is extended with the same set of ghost cells in which the boundary conditions are prescribed. For both the LS equation and for the redistancing equation the same boundary conditions are used. Only in the case that the LS function is explicitly known a value is prescribed. In all other situations a soft boundary is used. Although the LS function represents a distance function it is not necessary to extend this into the boundary cells. The LS value is simply copied to the ghost cell instead of extrapolated using information from neighboring cells. This rather simplistic approach seems to generate no additional inaccuracies. This is most probably due to the redistancing procedure, which redistributes the level-set distribution after each time step.

Chapter 7

Implementation aspects

In the previous sections a full RKDG LS method for the solution of two-dimensional two-fluid flows is developed. This chapter is devoted to several implementation aspects regarding the different algorithms involved. The RKDG algorithm for the Euler equations is discussed first, followed by the treatment of the LS algorithm. Special attention is paid to the implementation of the simple fix. Also the transformation procedure for transforming between primitive and conservative flow variables is discussed. The chapter is concluded with an overview of the full algorithm.

7.1 The RKDG algorithm

The RKDG method solves for the state variables of the two-dimensional Euler equations. This is done for the conservative state variables $\boldsymbol{q} = (\rho, \rho u, \rho v, \rho E)^T$, while data processing and plotting is done for the primitive state variables $\boldsymbol{w} = (\rho, u, v, p)^T$. The full method consists of a time-marching method that 'steps' from the approximate solution at the old time level $\boldsymbol{q}_h^n$ in several stages, where the amount of stages depends on the required order of accuracy of the full RKDG method, to the solution at the new time level $\boldsymbol{q}_h^{n+1}$. See (5.30) for this RK algorithm. At each intermediate Runge-Kutta stage a right-hand-side (RHS) $\boldsymbol{L}\left(\boldsymbol{q}_h^{(k)}\right)$ has to be calculated. Here the superscript (k) indicates the kth intermediate RK stage of general time level n. This RHS follows from the discontinuous Galerkin discretization of the spatial part of the Euler equations.

The calculation of this RHS governs several steps, of which the first is the limiting procedure. Here the approximate solution at each intermediate RK stage $\boldsymbol{q}_h^{(k)}$, which we shall for convenience denote here with $\boldsymbol{q}_h$, is transformed to the limited approximate solution $\boldsymbol{q}_h^*$. The limiter already contains part of the simple fix that prevents solution errors around the two-fluid interface to appear. The remaining part of this fix is implemented in the calculation of the RHS and the solution at the new time level, and will be explained in more detail below.

After being limited the approximate solution is used to determine the RHS. This RHS generally consists of two parts; a set of edge integrals and a volume integral. Both the edge and volume integral evaluations require the use of quadrature rules. Evaluation of the interior terms in each of the volume quadrature points is quite straightforward since it only requires substitution of the approximate solution into the expression for the fluxes, i.e. $\boldsymbol{f}(\boldsymbol{q}_h)$ and $\boldsymbol{g}(\boldsymbol{q}_h)$. Evaluation of the cell-face fluxes in the edge quadrature points however, requires the use of Roe's approximate Riemann solver. At each quadrature point two cell-face values, $\boldsymbol{q}_h^-$ and $\boldsymbol{q}_h^+$, are used for the calculation of the numerical flux $\tilde{\boldsymbol{\Psi}}\left(\boldsymbol{q}_h^-, \boldsymbol{q}_h^+\right)$.

Using the resulting RHS the approximate solution at the new time level $n+1$ can be calculated. This approximate solution can in turn be marched to a new time level by performing the same procedure again. For plotting and data processing purposes the approximate solution consisting of conservative variables $\boldsymbol{q}_h$ has to be transformed to the primitive state vector $\boldsymbol{w}_h$. This transformation procedure will be discussed in more detail below.

7.2 The LS algorithm

For the LS method, which solves for the distribution of the LS function φ, the same time loop is used as was done for the RKDG method. After one full solution step of the RKDG method a new velocity field is known, which in turn can be used to march the old LS distribution, and thus the old interface location ($\varphi = 0$), to a new time level. For this time marching a four stage RK scheme is used (see (6.13)). At each intermediate RK stage the RHS of the discretized LS equation, $L\left(\varphi_{i,j}\right)$, has to be calculated.

Solving for the RHS of the LS equation requires the calculation of the cell-face values of the LS function. These cell-face values are obtained using the limiter by Koren in which an upwind procedure is implemented. This upwinding mechanism uses the flow velocity in a cell to determine the correct interpolation scheme that should be used to determination the correct cell-face values of φ. For this purposes only the cell-mean values of the velocity are used, i.e. $\overline{u}_{i,j}$ and $\overline{v}_{i,j}$.

After one complete cycle of this update procedure the new LS distribution is known. Using the new interface location, which is represented by the zero LS contour, the new distribution of the ratio of specific heats γ can be determined. With this new distribution of the ratio of specific heats a new solution step of the RKDG method can be performed. But before such a new step is taken first the LS function has to be redistanced again. Since only the LS contour with $\varphi = 0$ is transported in the correct way, the LS function has to be transformed to a true signed distance function after each update of the LS function.

7.3 Implementation of the simple fix

In the solver described above a simple fix has to be implemented. This fix prevents the numerical solution to the Euler equations from becoming erroneous near the two-fluid interface. The fix consists of two parts; one that is governed with the calculation of the numerical fluxes and one that is governed with the update of the pressure distribution.

The first part of the simple fix is governed with the RKDG method only. This part of the fix requires each cell to only *see* one single fluid. In other words; each cell is only allowed to cope with one single fluid. Whenever the RKDG method has to perform certain calculations within a cell and therefore needs information from neighboring cells, every flow variable except the ratio of specific heats from these neighbors can be used. This 'looking out of the cell' only occurs in the approximate Riemann solver and in the slope limiter. In both cases information from several cells is required for the calculations. Implementation of the simple fix requires in both of these situations that only one ratio of specific heat is used.

The second part of the simple fix has to do with the update of the pressure distribution. Given the distribution of the total energy and the ratio of specific heats at the new time level, the pressure at the new time level can be calculated. However, the simple fix requires the update of the pressure to be performed using a *frozen* ratio of specific heats. This means that the pressure at the new time level is obtained from the total energy at the new time level using the old value for the ratio of specific heats. When this new pressure distribution is transformed back to a total energy distribution in order to perform a new RKDG solution step, the new ratio of specific heat distribution should be used.

7.4 Transforming the state variables

Transformation of the state variables from conservative ($\boldsymbol{q}_h$) to primitive ($\boldsymbol{w}_h$) and vice versa is not as straightforward as it may seem. This is due to the fact that the approximate solution $\boldsymbol{q}_h$ within a cell is constructed as the combination of degrees of freedom and polynomial basis functions (5.8):

$$\boldsymbol{q}_h\left(x, y, t\right) = \sum_{k,l=0}^{p} \boldsymbol{q}_{i,j}^{(k,l)}\left(t\right) \phi_i^{(k)}\left(x\right) \psi_j^{(l)}\left(y\right).$$

The approximate solution is thus a nonlinear combination of unknowns and the locations x and y within a cell. Therefore it is not possible to simply divide certain components of $\boldsymbol{q}_h$ to obtain another component. To clarify this we treat the transformation from conservative to primitive variables of the second component of

$\boldsymbol{q}_h$, i.e. $(\rho u)_h$ to u_h. Since both $(\rho u)_h$ and ρ_h are nonlinear functions of x and y it is clear that $\frac{(\rho u)_h}{\rho_h} \neq u_h$. Division of the separate degrees of freedom does not give the correct answer either.

A correct transformation procedure is obtained when first the different components of the approximate solution are evaluated in $(p+1)^2$ locations in a cell, before they are transformed from conservative to primitive or vice versa. So instead of transforming the $(p+1)^2$ degrees of freedom $\boldsymbol{q}_{i,j}^{(k,l)}$, one transforms the approximate solution in $(p+1)^2$ basis points. Evaluation of the approximate solution in each of these points is simply done by substitution of the corresponding locations (x, y). The points to use are free to be chosen, but a smart choice can significantly reduce the required computations. In appendix B the corresponding transformation procedures are explained in more detail.

7.5 The full algorithm

The complete algorithm for the numerical solution of unsteady compressible two-fluid flows described by the Euler equations can be summarized as follows:

- Initialize approximate solution $\boldsymbol{q}_h$ at t^0
- For $n = 0, \ldots, N-1$ march from *old* time level t^n to *new* time level t^{n+1}:
  - Update state vector $\boldsymbol{q}_h$ using $(p+1)$ stage RK scheme:
    - Apply slope limiter to obtain limited approximate solution $\boldsymbol{q}_h^*$,
    - Calculate RHS of discretized Euler equations $\boldsymbol{L}(\boldsymbol{q}_h^*)$,
    - March to new intermediate RK stage,
  - Transform from conservative $\boldsymbol{q}_h$ to primitive $\boldsymbol{w}_h$ variables using *frozen* γ^n,
  - Update LS function φ using four-stage RK scheme:
    - Calculate RHS of discretized LS equation $L(\varphi_{i,j})$,
    - March to new intermediate RK stage,
  - Redistance LS function φ to a true distance function using several steps with a two-stage RK scheme:
    - Calculate RHS of discretized redistancing equation,
    - March to new intermediate RK stage,
  - Given the *new* interface location, calculate distribution of ratio of specific heats γ,
  - Transform from primitive $\boldsymbol{w}_h$ to conservative $\boldsymbol{q}_h$ variables using *new* γ^{n+1},

Part III

Flow Problems

Chapter 8

One-dimensional problems

Given the discontinuous Galerkin solver as described in the previous chapters it is possible to run several test cases to validate its performance. The one-dimensional test problems that will be used are referred to as *shock tube problems*. A shock-tube problem is defined as a one-dimensional problem consisting of two initially separated states, L left of $x = 0$ and R right of $x = 0$. At time $t = 0$ the virtual membrane separating both states is removed, allowing the two fluid states to interact with each other. A set of waves (shock waves, expansion fans and contact discontinuities) will appear that run into both fluid states, changing the state variable distributions along their path. These shock tube problems are desirable for the validation of our solver since for each problem an exact solution can be found, i.e. the solution to the Riemann problem present initially, allowing for an easy comparison of the performance of the solver.

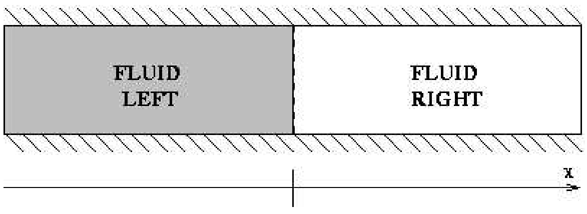


Figure 8.1: A shock tube with initially two fluids separated by an interface at $x = 0$. At time $t = 0$ the fluids are *released* resulting a characteristic wave pattern consisting of shock waves, expansion fans and/or contact discontinuities.

Unless stated otherwise, for all test cases a grid of 200 cells is used. The temporal step size Δt follows from the stability condition (5.32), where for each test a sufficiently large maximum wave speed is estimated. When possible, the less restrictive MUSCL-limiter ($\lambda = \frac{1}{2}$) is used in the slope limiting procedure. For each problem the piecewise quadratic approximations ($p = 2$), which are formally third order accurate, are used. Unfortunately this order of convergence will not be reached due to the slope limiter, which degrades the order of accuracy. The effect of the higher-order discretization only becomes visible in those regions containing smooth, highly structured phenomena.

8.1 Translating interface

This one-dimensional test problem was encountered several times already in this report. Two fluids separated by a sharp interface travel with a constant speed to the right. The density and the ratio of specific heats jump over the interface, while the velocity and the pressure are constant. The exact solution indicates that in time the conditions left and right of the travelling interface remain unchanged. Except for some numerical smearing over the interface this should hold for the numerical solution too.

We will consider two different situations. The first treats a translating interface with a weak jump of the density distribution over the interface. The second case treats the same problem with a 125 times stronger density jump. A sufficiently robust and accurate numerical solver should keep the density jump as sharp as possible without producing oscillations. The simple pressure fix should prevent spurious pressure oscillations from occurring near the two-fluid interface, at the price of locally abandoning conservation. These fix-induced errors, if present, should especially become apparent in the stronger test case.

8.1.1 Weak density jump

For this problem the initial left and right flow states are given by:

$$\begin{aligned}(\rho, u, p, \gamma)_L &= (1.0, 1.0, 1.0, 1.4)\,,\\ (\rho, u, p, \gamma)_R &= (0.125, 1.0, 1.0, 1.6)\,.\end{aligned}$$

The maximum wave speed is estimated to be equal to $c = 5$ such that for the quadratic approximation $\Delta t = \frac{1}{25}\Delta x$ should be used.

The results are plotted in figure 8.2. It is clear that the numerical solution approximates the exact solution quite well. The numerical interface is, although smeared out due to numerical diffusion, in the right location. It is obvious that the DG discretization of the Euler equations results in a relatively sharp jump in the density distribution. Especially when one compares the results obtained here with those mentioned in [39], which were obtained using a third-order FV method, it becomes clear that the method developed here results in a desirable increase of resolution near the interface. It can further be seen that the velocity and pressure distributions are oscillation free, indicating that the simple pressure fix performs as desired.

8.1.2 Strong density jump

In order to push the numerical solver a bit further the same translating interface problem shall be treated again, but now with a stronger density jump. This could possibly result in larger numerical errors due to the presence of our simple fix, which makes the method abandon conservation near the interface. Also the limiter is tested more severely this way. For this problem the following initial conditions are used:

$$\begin{aligned}(\rho, u, p, \gamma)_L &= (1000.0, 1.0, 1.0, 1.4)\,,\\ (\rho, u, p, \gamma)_R &= (1.0, 1.0, 1.0, 1.6)\,.\end{aligned}$$

The maximum wave speed is roughly estimated to be equal to $c = 40$, requiring the following time step: $\Delta t = \frac{1}{200}\Delta x$.

The corresponding results are given in figure 8.3. Again the results are satisfactory. No significant difference between the weak and the strong problem is visible, indicating that the fix-induced errors are negligible. The interface is still in its correct location and the density jump shows no extra smearing.

8.2 Sod's problem

Sod's problem is probably the best-known shock tube problem. The exact solution to this problem consists of a left-running expansion fan and both a right-running contact discontinuity (here also the two-fluid interface) and shock wave. Here the problem is solved using piecewise constant, linear and quadratic approximations in order to investigate the limiter implemented in the numerical method. This TVDM limiter, discussed in chapter 5, should effectively prevent the solution from becoming oscillatory near discontinuous phenomena, without a loss of accuracy. However, the TVDM limiter applied here degrades the accuracy obtained using piecewise quadratic approximations to that of a solution using piecewise linear approximations. We shall show this effect by solving Sod's problem. To indicate that this effect is solely the result of the limiter, also the numerical results obtained without a limiter (thus showing numerical oscillations!) are given.

The initial data for this problem is:

$$\begin{aligned}(\rho, u, p, \gamma)_L &= (1.0, 0.0, 1.0, 1.4)\,,\\ (\rho, u, p, \gamma)_R &= (0.125, 0.0, 0.1, 1.6)\,.\end{aligned}$$

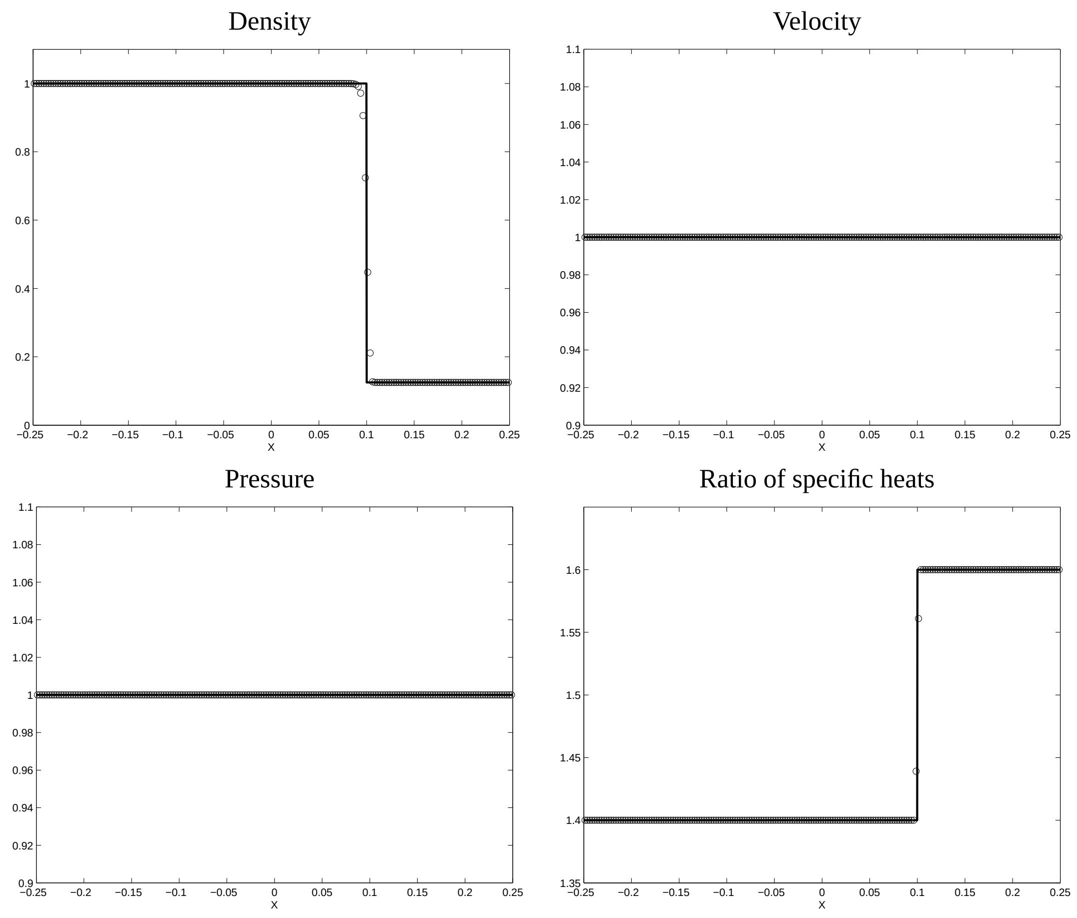


Figure 8.2: Results of the translating wave problem, with weak density jump. Primary variables against the x location in the tube at $t = 0.1$. Grid consists of 200 cells. Plotted are values in cell centers. $-$ line is exact solution, $\cdots$ line is numerical solution.

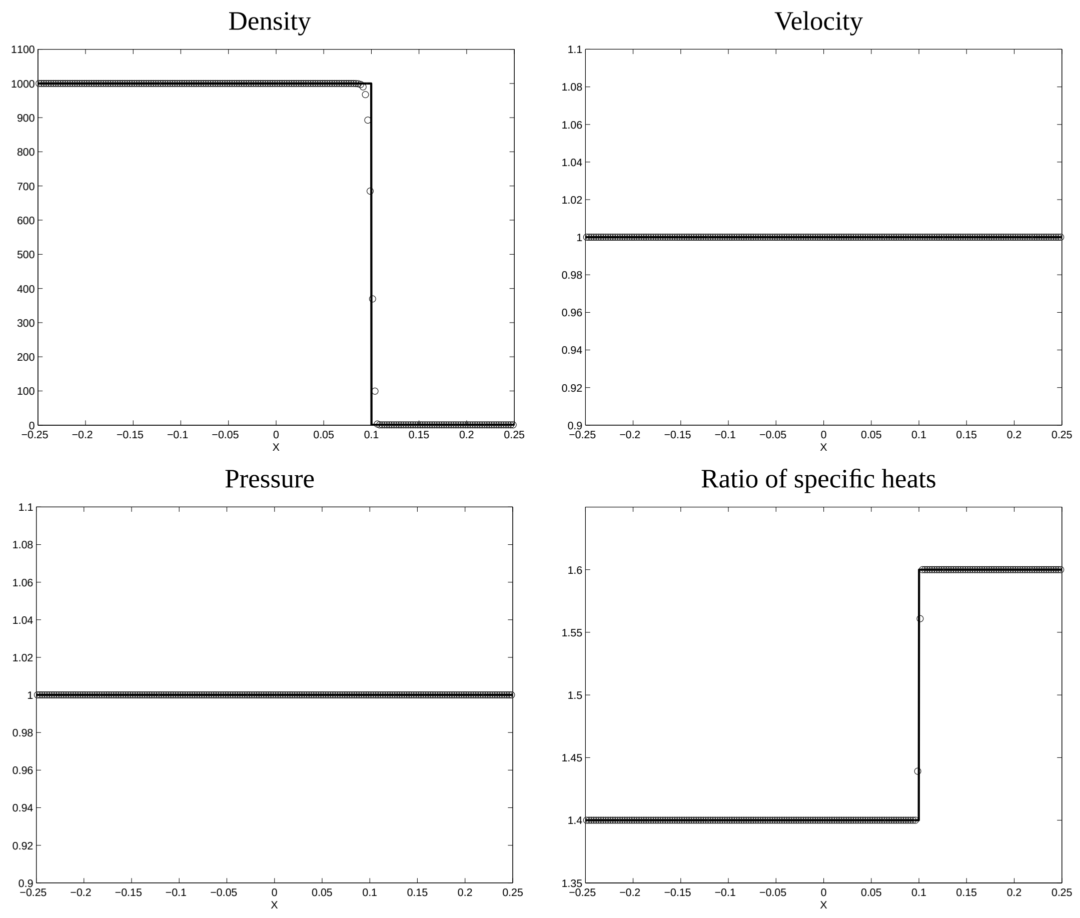


Figure 8.3: Results of the translating wave problem, with strong density jump. Primary variables against the x location in the tube at $t = 0.1$. Grid consists of 200 cells. Plotted are values in cell centers. $-$ line is exact solution, $\cdots$ line is numerical solution.

Based on these conditions the maximum wave speed is estimated to be equal to $c = 2.5$ such that for the constant approximation one finds a time step equal to $\Delta t = \frac{2}{5}\Delta x$. For the linear approximation a time step equal to $\Delta t = \frac{2}{15}\Delta x$ is used and for the quadratic approximation $\Delta t = \frac{2}{25}\Delta x$. The results are plotted in figure 8.4. The numerical results comply well with the exact solution. The discontinuous phenomena, i.e. the contact discontinuity and the shock wave, are in their correct locations. The first-order solution ($p = 0$) shows a large amount of smearing near the discontinuous phenomena. Also the sharp corners of the expansion fan show signs of smearing. The higher-order solutions are clearly much more accurate. The shock and contact discontinuity are sharp and show no signs of undesired oscillations, indicating that the limiter performs well. Although a limiter basically prevents a solution from becoming oscillatory by using first-order techniques, our TVDM limiter clearly prevents the method from losing accuracy.

As expected, the third-order solution is almost identical to the second-order solution. This lack of accuracy can completely be ascribed to the slope limiter. To prove this also the results of the same calculations but now without the use of a slope limiter is shown (for most problems this will lead to too large oscillations such that no results can be obtained). In figure 8.5 again the first, second and third order results of the Sod problem are given. Since a first-order method does not require limiting, the solution remains unchanged compared to the results shown in figure 8.4. In the case of the second and third-order method one does see a significant difference. Although the solution now has undesired oscillations near the discontinuous phenomena and near the front and back of the expansion wave, stipulating the need for a limiter, one does notice an increase in accuracy of the third-order solution especially near the contact discontinuity. This proves that TVDM limiting indeed degrades the accuracy of higher-order methods to that of the limited second-order method, which is in fact locally first-order accurate. From this it can be concluded that for problems containing strong discontinuous phenomena it does not pay-off to go to higher-than-second-order using the present method. Only in smooth regions the higher accuracy will become visible. An interesting topic for further research will be to develop a two-fluid slope limiter for DG that does not degrade the order of accuracy of the DG method as severe as is the case now. In chapter 5 the TVBM slope limiter, which possesses such requirements, was already mentioned. Similar work has been done by Burbeau *et al.* in [12], resulting in a problem-independent higher-order limiter. Another option could be the use of so-called *stability operators*. This more recent development is the finite-element specialist's answer to the problems involved with slope limiting. See [58] for a more elaborate discussion of these methods.

8.3 No-reflection problem

As a third test we take the *no-reflection problem* presented in [60]. This rather 'hard' test problem treats an initial two-fluid interface that is hit by a strong shock wave, which conditions are chosen such that after the interaction the interface travels to the right accompanied by a right-running shock wave. Typical for this problem is that no reflection (left-running) wave occurs. Due to the severity of the interaction between the strong shock and the interface especially the limiter is required to perform well without introducing to much numerical smearing. The initial data for this problem are:

$$\begin{aligned}(\rho,u,p,\gamma)_L &= (3.1748, 9.4350, 100.0, 1.667)\,,\\ (\rho,u,p,\gamma)_R &= (1.0, 0.0, 1.0, 1.2)\,.\end{aligned}$$

The time step is taken equal to $\Delta t = \frac{1}{125}\Delta x$. At time $t = 0.016$ this results, for a grid consisting of 400 cells, in the numerical solution plotted in figure 8.6. Although this problem is called the no-reflection problem the numerical results show a small *bump* that travels with the local wave velocity to the left. This small wave is the result of errors occurring during the interaction of the shock with the interface during the start-up process of the problem. A possible source for these errors is the lack of conservation due to the simple pressure fix. However, results for the same problem obtained in [39] show that applying no fix produces even worse numerical errors. It can further be noted that results from [60] show the same little left-running bump in the solution. Apparently the *bump* and responsible errors are typical for the no-reflection problem and do not depend on the type of numerical solver used.

Taking a closer look at the numerical solution in figure 8.6 and the solution from [39] and [60], brings us to the conclusion that our numerical solver performs substantially better. The shock wave and the interface

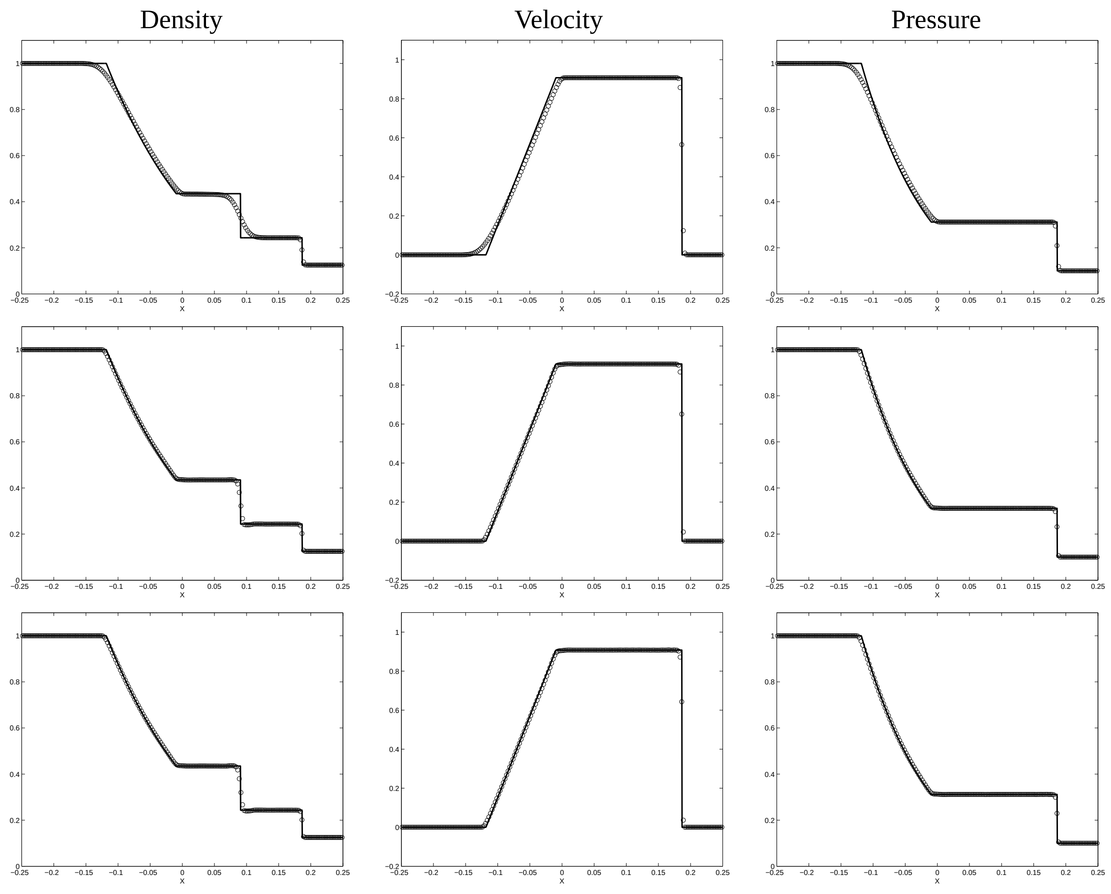


Figure 8.4: Results of Sod's problem. Primary variables against the x location in the tube at $t = 0.25$. Grid consists of 200 cells. Plotted are values in cell centers. $-$ line is exact solution, $\cdots$ line is numerical solution. *Top:* First order. *Middle:* Second order. *Bottom:* Third order.

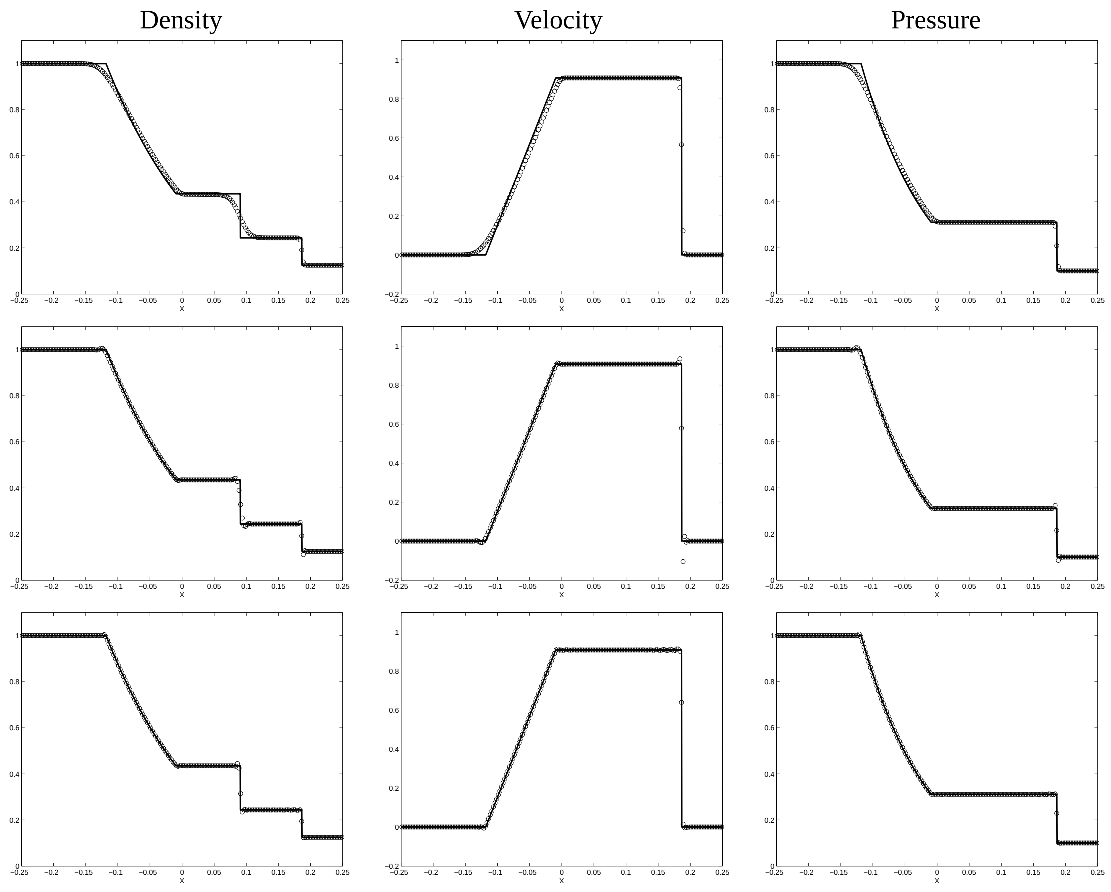


Figure 8.5: Results of Sod's problem without limiting. Primary variables against the x location in the tube at $t = 0.25$. Grid consists of 200 cells. Plotted are values in cell centers. $-$ line is exact solution, $\cdots$ line is numerical solution. *Top:* First order. *Middle:* Second order. *Bottom:* Third order.

are both sharper in the case of the DG discretization. Also the transition between the interface and the shock wave is captured more accurately. Less smearing is present in this region resulting in the correct values for the flow variables.

To further compare the solver developed here with other comparable methods a grid refinement study is performed. On several grids, ranging from 50 to 800 cells, the errors in the state variable distributions are calculated. For this purpose the L_1-error norm is used, such that the total error of the state variable distribution can be written as $\epsilon_{L_1} = \frac{\sum_1^N |\epsilon|}{N}$. These errors for the different primary variables are given in tables 8.1 and 8.2, for the linear and quadratic case respectively. The order of accuracy n, i.e. $\mathcal{O}\left(\Delta x^n\right)$), is obtained from these errors and their corresponding grid size using a least-squares method.

J	ϵ_ρ	ϵ_u	ϵ_p
50	0.2732876	0.1854001	1.7579671
100	0.1438852	0.0862774	0.9211628
200	0.0892548	0.0579925	0.5719657
400	0.0450280	0.0320273	0.3121799
800	0.0209017	0.0098080	0.1142802
n	0.9093476	0.9910768	0.9447601

Table 8.1: Results of grid refinement study for 1D no-reflection problem. Linear basis functions, $p = 1$.

J	ϵ_ρ	ϵ_u	ϵ_p
50	0.2680861	0.1787391	1.7165797
100	0.1379616	0.0841464	0.9188234
200	0.0851093	0.0547675	0.5618585
400	0.0435939	0.0306400	0.3058662
800	0.0204000	0.0094221	0.1129150
n	0.9094177	0.9948811	0.9439343

Table 8.2: Results of grid refinement study for 1D no-reflection problem. Quadratic basis functions, $p = 2$.

Comparing the results from this grid refinement study with those given in [39] (formally third-order FV method, i.e. effectively a second-order accurate method with improved resolution) and [60] (formally second-order FV method), it becomes clear that the RKDG LS method performs significantly better. Already for the case with linear basis functions the magnitudes of the errors are approximately 2 times lower. Also the order of accuracy, which approaches one, is higher than was the case for both FV methods. The fact that one does not find an order equal to 2 is solely due to the discontinuities in the problem. For smooth continuous problems the formal order of accuracy should be obtained. The third-order results, i.e. quadratic basis functions, do not show any improvement compared to the second-order solution, which can again be subscribed to the slope limiter.

From the above it can safely be concluded that although its two-fluid flow limiter is not yet performing optimally, the numerical solver developed here is more accurate than similar solvers. The second-order RKDG method already shows a significant reduction in the magnitude of the numerical errors, while the order of accuracy is somewhat higher. It appears that the RKDG method is able to more accurately capture discontinuous phenomena.

8.4 Shu-Osher problem

The fourth test is the Shu-Osher problem (example 8 from [53]). This problem consists of a Mach 3 shock which interacts with sine-shaped waves in the density distribution. This problem is particularly interesting due to the complex structure in the smooth regions. In the previous section it was shown that in the case of

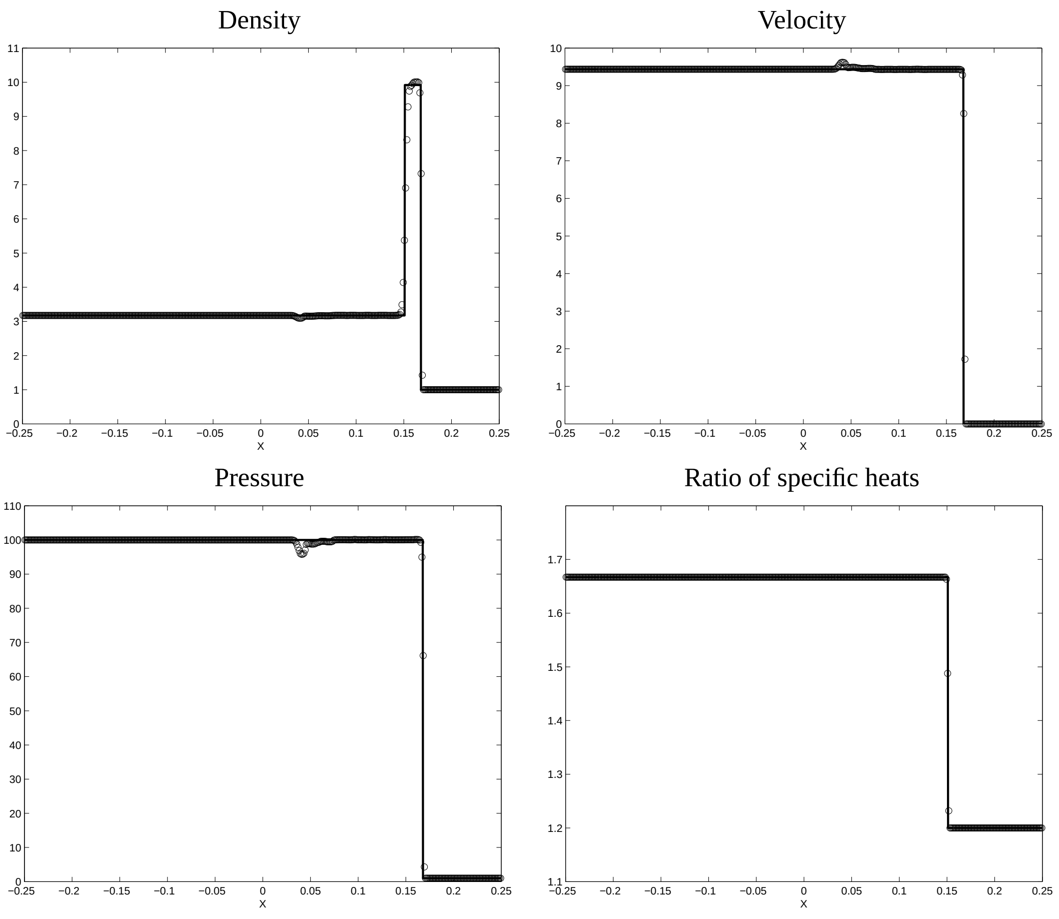


Figure 8.6: Results of the no-reflection problem. Primary variables against the x location in the tube at $t = 0.016$. Grid consists of 400 cells. Plotted are values in cell centers. $-$ line is exact solution, $\cdots$ line is numerical solution.

discontinuous phenomena the numerical solution obtained using quadratic basis functions does not reach its formal third-order accuracy, which is due to the TVDM limiter. However, near smooth continuous phenomena an increase in accuracy should be seen. The initial data for this problem is:

$$\begin{aligned}(\rho, u, p)_L &= (3.857143, 2.629369, 10.33333)\,,\\ (\rho, u, p)_R &= (1+\epsilon \mathrm{sin} 5x, 0.0, 1.0)\,,\end{aligned}$$

where the left and right regions are at $x = 1$ with x ranging from 0 to 10. Note that the Shu-Osher problem is a single-fluid problem, where we take γ equal to 1.4; the value for air. We choose $\epsilon = 0.2$. The 'exact' solution is obtained by running the third-order calculations on a grid consisting of 1600 cells. Our 'numerical' solutions follows from the calculations on a 200 cell grid.

The results for $p = 1$ are plotted in figure 8.7 and for $p = 2$ in figure 8.8. The highly structured region is almost completely ignored on the 400 cells grid. The 800 grid cell solution is significantly better, although the strong oscillatory part just in front of the shock is still captured poorly. Further it becomes immediately clear that the third-order solution is not better than the second-order solution. Although this behavior was seen already in the previously treated problems, the Shu-Osher problem was our best chance of making full use of the increased resolution of the third-order method. The highly structured smooth region formally does not require any limiting. However, it is obvious that near the crests and troughs of the oscillatory part the limiter is indeed activated since no significant difference between the second and third-order solution can be distinguished. Obviously this problem is still not smooth enough for the third-order method to really show its competence.

In [18] results of the same tests are given but now TVBM limiting using characteristic variables has been applied. Corresponding results are promising but is should be noted that application of the TVBM limiter requires the estimation of the problem dependent parameter M. This especially makes the TVBM approach less suited for the application in a general and robust solver.

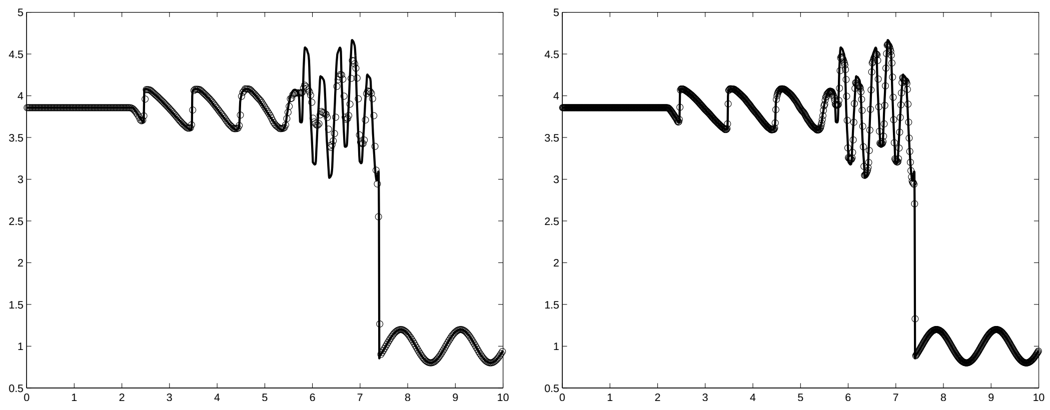


Figure 8.7: Results of Shu-Osher problem. Second-order solution. Density against the x location in the tube at $t = 1.8$. Plotted are values in cell centers. – line is 'exact' solution, $\cdots$ line is 'numerical' solution. *Left:* 400 grid cells. *Right:* 800 grid cells.

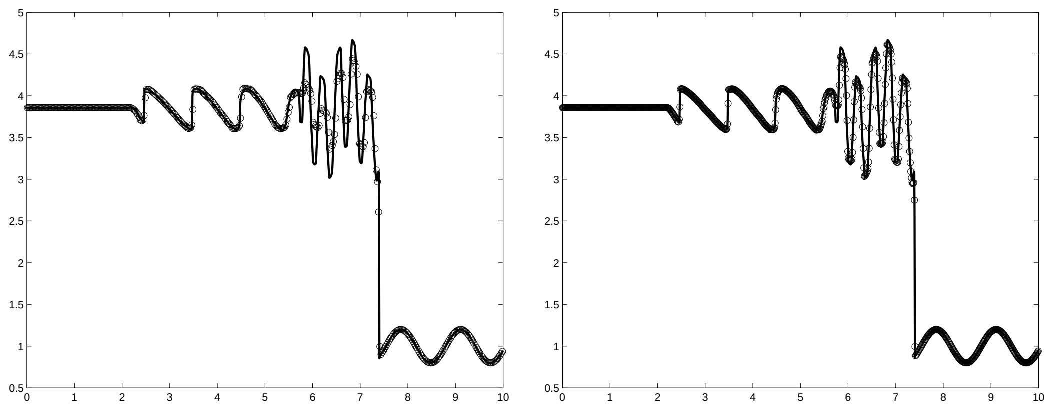


Figure 8.8: Results of Shu-Osher problem. Third-order solution. Density against the x location in the tube at $t = 1.8$. Plotted are values in cell centers. – line is 'exact' solution, $\cdots$ line is 'numerical' solution. *Left:* 400 grid cells. *Right:* 800 grid cells.

Chapter 9

Two-dimensional problems

In the previous chapter the numerical solver was tested using several one-dimensional benchmark problems. This chapter is devoted to the application of the solver to two-dimensional flow problems with significant physical relevance. The first of these problems is the well-known shock-bubble interaction taken from the experiments by Haas and Sturtevant [25]. This specific problem has been solved numerically many times before, see [39, 43, 60], such that plenty of data is available for comparison. Secondly the Kelvin-Helmholtz instability is considered. This problem is taken from [37] and has recently been repeated in [39]. Although this problem contains several interesting aspects, as a benchmark it is less suited due to its intrinsically unstable character. Finally the solver will be applied to several supersonic free jet problems. This novel test problem was proposed in [39] and will be studied more extensively here.

9.1 Shock-bubble interaction

The performance of the numerical solver for two-dimensional problems is tested on a problem treating the interaction of a shock moving in air and a bubble containing a different gas which is initially at rest (see figure 9.1). This problem is taken from the experiments by Haas and Sturtevant [25]. Numerical treatment of this problem has among others been done by Quirk and Karni [43] and more recently by Wackers and Koren [60] and by Naber [39]. The experimental setup is as follows: a gas bubble at rest is located in the center of a wind tunnel containing air at rest. The gas in the bubble is separated from the air by a micro-film. At the right of the bubble a shock moves towards the bubble. When the bubble is hit by the shock the micro-film tears apart and the shock and the bubble start to interact, resulting in a specific wave pattern depending on the type of gas in the bubble. Due to this interaction the bubble will start to deform and finally break-up into two separate halves. Because the interaction between the bubble and the shock is extremely fast (the shock passes the bubble in approximately 10^{-4} seconds in the case of an air speed of sound of 343 m/s) it is allowed to assume that the two fluids do not mix, such that the level-set method is well-suited for its simulation.

In these tests the circular bubble in air contains either helium (lighter than air) or the refrigerant gas R22 (heavier than air). Due to the different densities of both gasses the characteristic wave patterns resulting from the interaction between the shock and the bubble differ significantly. However, for both situations several characteristic waves can be distinguished. The first of these is the *incoming shock* travelling towards the bubble and continuing above the bubble after the interaction. After hitting the bubble the incoming shock continues in the bubble as a *refracted shock*. In the helium case the speed of sound in the bubble is higher than the speed of sound in the air resulting in a refracted shock that travels faster than the shock outside of the bubble. In the R22 case the speed of sound in the bubble is lower than in the air, which makes the refracted shock to lag behind the shock in the air. After a while the wave interaction becomes more complex producing characteristic wave patterns for both cases.

For the numerical computations the speed of sound in air is taken equal to one instead of 343 m/s. For completeness all velocities should therefore be scaled with 343. Given this speed of sound the incoming shock moves with a Mach number equal to 1.22. The initial conditions for both the helium and the air case

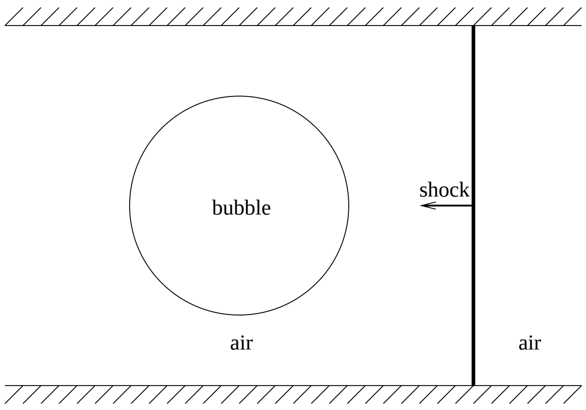


Figure 9.1: A shock hitting a bubble immersed in air. Depending on the fluid it contains the bubble deforms due to the shock-bubble interaction.

are given in table 9.1. Note that the ratio of specific heats and the density for helium differ from text-book values. This is to take into account the contamination of the helium with air which was the case in the experiment.

In order to reduce the amount of computations the numerical flow domain is halved by making use of symmetry properties. This results in a domain with lengths 0.08 and 0.045 in x and y directions respectively. The center of the bubble is initially located at a distance of 0.035 from the left boundary of the domain. The grid is taken equal to 400×200 cells in accordance to the simulation in [60]. The time step is chosen such that for $p = 0$ it is equal to $\Delta t = 1.25 \times 10^{-5}$ for the helium bubble and $\Delta t = 2.5 \times 10^{-5}$ for the R22 bubble. When higher-order basis functions are applied these time steps are reduced according to the stability condition (5.32). To allow the shock to develop before it hits the bubble, its initial position is taken 5 cells right of the bubble interface. This means that it takes the shock 2.24×10^{-3} seconds to reach the bubble. From now on only the time from the start of the calculations is used.

In order to prevent unphysical oscillations to occur the density distribution near the bubble interface is smoothed in exactly the same way as is done for the ratio of specific heats. Using the smoothed-step function H_ϵ the bubble interface is smeared-out over a few cells. This prevents the density distribution from *staircasing*. See [39] for an elaborate discussion of this initial density smoothing.

	γ	ρ	u	v	p
air$_{stagnant}$	1.4	1.40000	0.00000	0.00000	1.00000
air$_{moving}$	1.4	1.92691	0.33361	0.00000	1.56980
helium	1.648	0.25463	0.00000	0.00000	1.00000
R22	1.249	4.41540	0.00000	0.00000	1.00000

Table 9.1: Initial conditions for the 'shock hitting bubble' test case.

9.1.1 Helium bubble

In this first test the helium case is considered. The helium used here has a sound speed that is significantly higher than the speed of sound in air ($c_{\text{helium}} = 2.44$ and $c_{\text{air}} = 1.0$). This makes that the *refracted shock*, i.e. the incoming shock continuing in the bubble, moves faster than the incoming shock in the air. Due to this difference in velocity the refracted shock and the incoming shock do not intersect the interface at the same location, which is depicted in figure 9.2. At the point where the refracted shock meets the interface therefore another wave occurs, which is called the *transmitted shock*. This transmitted shock is curved backwards such that it is directed towards the incoming shock. After crossing the incoming shock this transmitted shock interacts with the expansion that originates from the interaction between the incoming shock and the interface. Due to this interaction the transmitted shock is bended inwards (towards the symmetry line). When the refracted shock passes again through the bubble interface, now at the left, it continues outside the bubble as the earlier mentioned transmitted shock. This shock now has the shape of a bow-shock. The bubble itself is now moving with a constant velocity to the left. During the following period the right side of the bubble will continue to steepen, finally resulting in the rolling-up of the bubble in a vortex shape. Unfortunately this break-up has not been reached numerically due to computational limitations. In [43] this break-up is discussed in more detail.

In the plots of the density (figure 9.3) and the pressure (figure 9.4) one recognizes the for the helium bubble characteristic wave pattern. The refracted shock, travelling faster than the incoming shock, is clearly visible. After leaving the bubble again this refracted shock becomes the transmitted shock, which has the shape of a bow-shock. Even after passing through the bubble twice this shock is still surprisingly sharp and in the correct location, indicating that there is no apparent loss of accuracy due to the method locally being non-conservative. Only near the back of the bubble the density distribution is smeared-out to some extent. However, this smearing is significantly less than was the case in the results in [39] and [60]. Similar to the observation made in the previous chapter one can conclude that the RKDG method introduces less numerical diffusion near discontinuous phenomena resulting in sharper shocks and contact discontinuities. Within the bubble the contours of the pressure show some oscillatory behavior, which can be subscribed to plotting.

After being hit by the incoming shock the bubble starts deforming. This deformation process is clearly visible in figure 9.5, which shows plots of the contours of the ratio of specific heats. The part of the bubble right of the point where the incoming shock and the interface intersect starts moving to the left while the part left of the intersection point remains unchanged. This results in a steepening of the right side of the bubble. In time the right part of the bubble moves further towards its left part, hereby decreasing the width of the bubble. Since the LS method only smears the interface over a maximum of 3 cells the interface remains sharp at all time.

To further qualify the results obtained with the present method the velocities of several waves are compared with known data. In table 9.2 the velocities of the incoming shock c_s, the refraction shock c_r and the interface c_i are given. The former is measured at the top of the computational domain while the latter two are obtained at the center or symmetry line. Velocities are obtained by interpolation of several locations and the corresponding times. Since all velocities are scaled with the speed of sound of air, i.e. 343 m/s, all results are multiplied with this number. From the table it becomes clear that the present method performs reasonably well. The incoming shock speed complies very well with previous results. The refracted shock speed and the interface speed lie somewhat higher than is the case for the other results, but are still within an acceptable region.

	$c_s[m/s]$	$c_r[m/s]$	$c_i[m/s]$
Present method	419	955	181
Quirk & Karni [43]	422	943	178
Wackers & Koren [60]	419	950	173
Haas & Sturtevant [25]	410	900	170

Table 9.2: Wave speeds in helium bubble test. Incoming shock speed c_s measured at the top of the domain, refraction shock c_r and interface c_i speeds at symmetry line. The velocities are scaled back to $[m/s]$ by taking the speed of sound equal to $343m/s$.

9.1.2 R22 bubble

In the second test case the bubble is filled with refrigerant gas R22, which is a HCFC used in for example air conditioners and heat pumps. R22 is a heavy gas, having a density which is three times larger than the density of air, and a ratio of specific heats which is somewhat lower. This makes that the sound speed in R22 is lower than the sound speed in air ($c_{\text{R22}} = 0.532$ against $c_{\text{air}} = 1.0$). This lower speed of sound

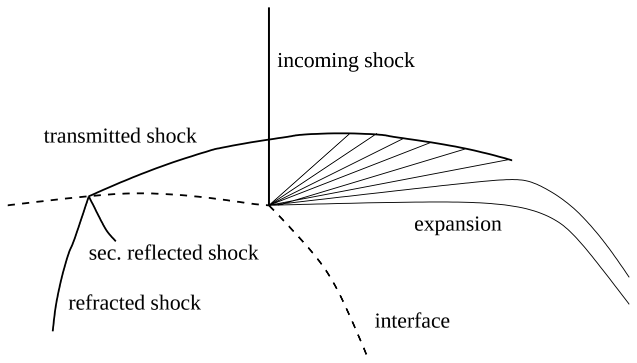


Figure 9.2: Wave pattern resulting from the interaction of the shock and the helium bubble (taken from [60] with permission).

results in a refracted shock that moves slower than the incoming shock. Different from the case for the helium bubble, the refracted shock and the incoming shock keep intersection the interface at the same point at all time. This causes the top of the refracted wave to bend forward when time passes. In time also the incoming shock starts to curve inwards near the bubble, resulting a complicated wave pattern in front of the bubble. Due to the initial interaction of the shock and the bubble a reflected wave appears that travels backwards and is finally reflected by the solid wall on the top. For a complete description of the wave pattern see again [60].

In figures 9.6, 9.7 and 9.8 the contours of the density, pressure and ratio of specific heats are plotted, respectively. Again the density and pressure plots comply very well with the expectations. All waves are in their correct locations and show significantly less smearing than was the case in [39] and [60]. The incoming shock and the refracted shock intersect the interface at the same location at all time requiring both shocks to significantly curve. Due to this curvature both shocks show somewhat more smearing than the same shocks in the helium case. The deformation of the bubble, clearly visible in the plots of the ratio of specific heats, figure 9.8, again complies with comparable results. The center line locations of both the left and the right bubble interface are correct. It is obvious that the deformation of the R22 bubble is different from the deformation of the helium bubble. Instead of breaking-up at its right side, resulting in two backward curved vortexes, the bubble breaks-up at its left. This process is clearly explained in [43]. Due to computational limitations this break-up has not been computed here.

Again the velocities of several characteristic waves are computed. These values can be found in table 9.3. Again the different shock and interface speeds show good comparison with similar data. Results even show a better similarity than was the case for the helium bubble. This can probably be subscribed to the R22 shock-bubble interaction being less severe due to the lower speeds of sound.

	$c_s[m/s]$	$c_r[m/s]$	$c_i[m/s]$
Present method	419	230	73
Quirk & Karni [43]	420	254	70
Wackers & Koren [60]	419	241	75
Haas & Sturtevant [25]	415	240	73

Table 9.3: Wave speeds in R22 bubble test. Incoming shock speed c_s measured at the top of the domain, refraction shock c_r and interface c_i speeds at symmetry line. The velocities are scaled back to $[m/s]$ by taking the speed of sound equal to $343m/s$.

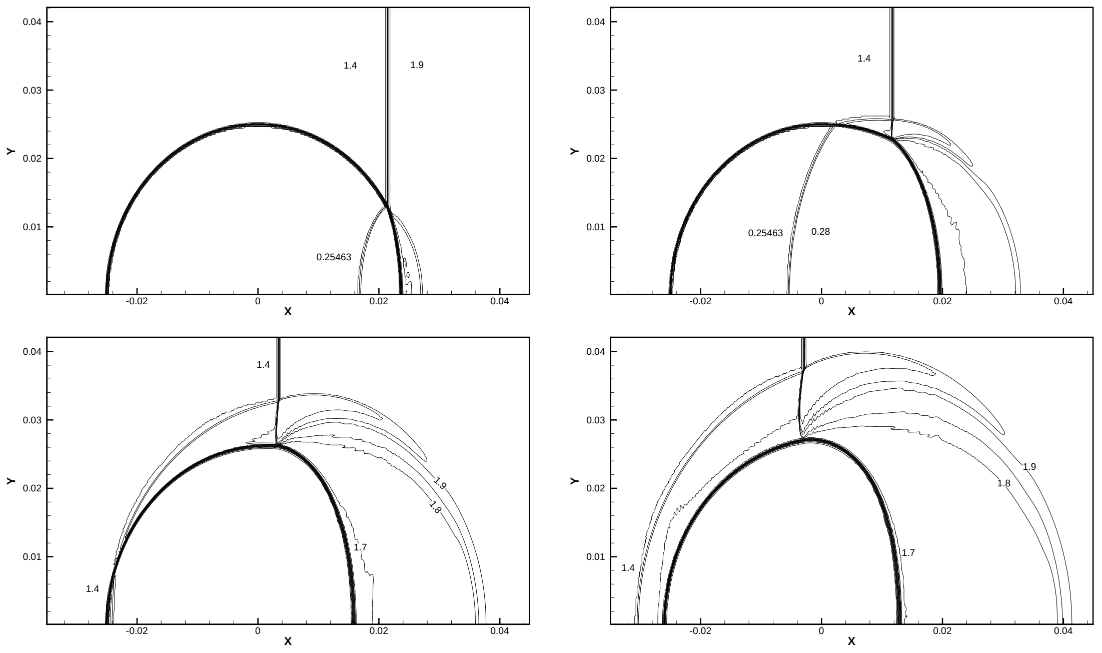


Figure 9.3: Shock hitting helium bubble. Density at $t = 5 \times 10^{-3}$, $t = 13 \times 10^{-3}$, $t = 19.8 \times 10^{-3}$ and $t = 25 \times 10^{-3}$. Grid has 400×200 cells and $\Delta t = 0.25 \times 10^{-5}$.

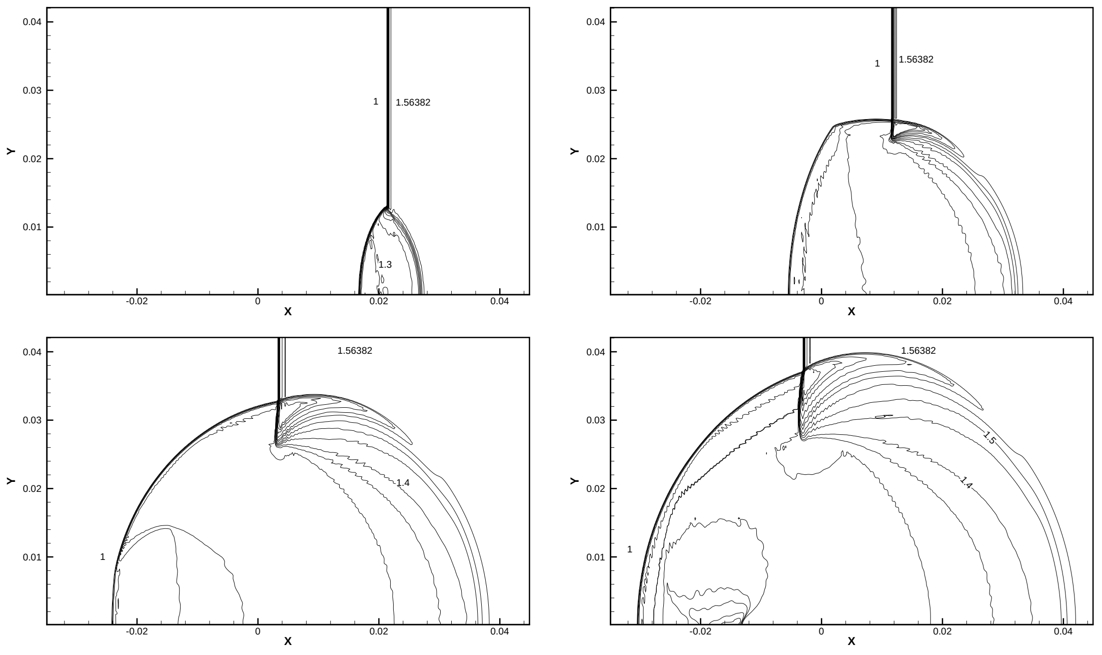


Figure 9.4: Shock hitting helium bubble. Pressure at $t = 5 \times 10^{-3}$, $t = 13 \times 10^{-3}$, $t = 19.8 \times 10^{-3}$ and $t = 25 \times 10^{-3}$. Grid has 300×150 cells and $\Delta t = 0.25 \times 10^{-5}$.

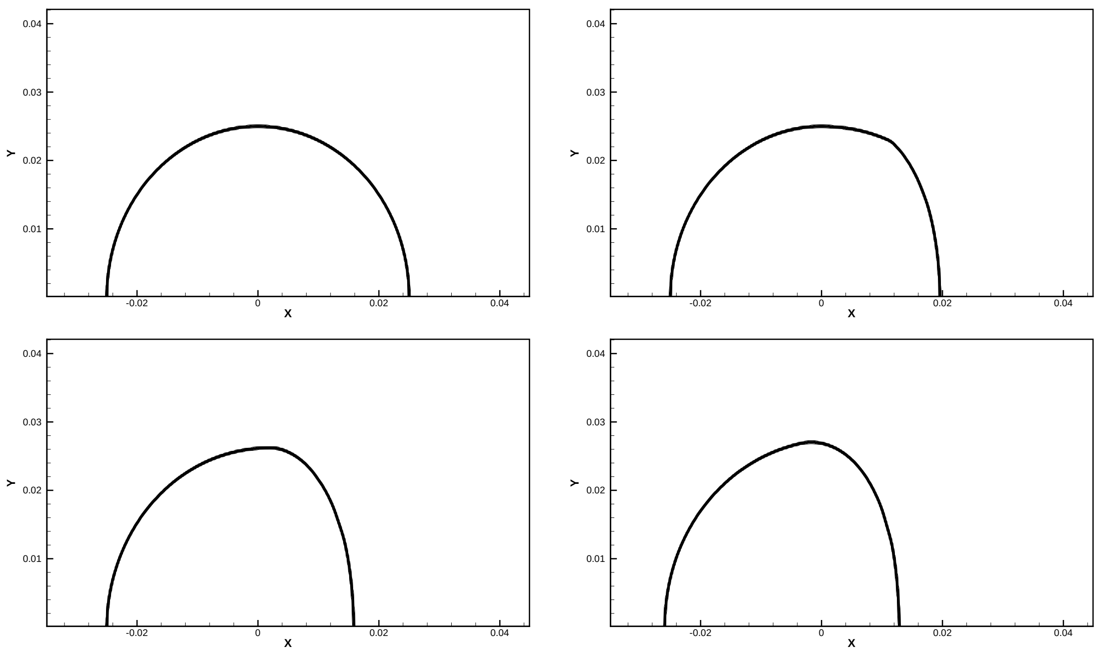


Figure 9.5: Shock hitting helium bubble. Ratio of specific heats at $t = 5 \times 10^{-3}$, $t = 13 \times 10^{-3}$, $t = 19.8 \times 10^{-3}$ and $t = 25 \times 10^{-3}$. Grid has 300×150 cells and $\Delta t = 0.25 \times 10^{-5}$.

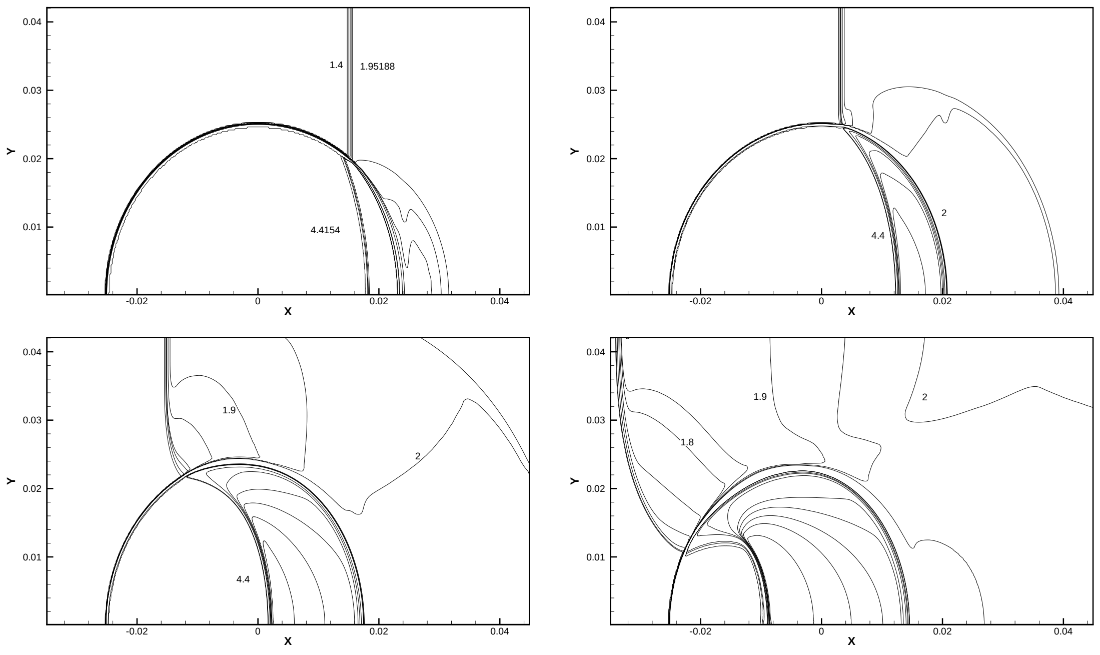


Figure 9.6: Shock hitting R22 bubble. Density at $t = 10 \times 10^{-3}$, $t = 20 \times 10^{-3}$, $t = 35 \times 10^{-3}$ and $t = 50 \times 10^{-3}$. Grid has 300×150 cells and $\Delta t = 0.125 \times 10^{-5}$.

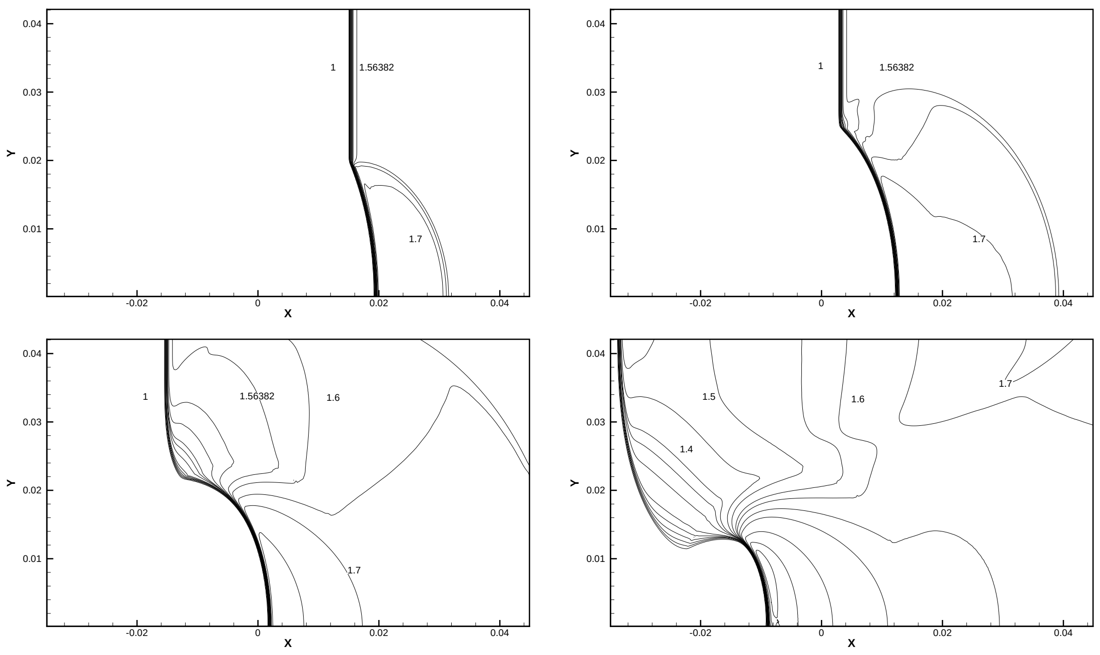


Figure 9.7: Shock hitting R22 bubble. Pressure at $t = 10 \times 10^{-3}$, $t = 20 \times 10^{-3}$, $t = 35 \times 10^{-3}$ and $t = 50 \times 10^{-3}$. Grid has 300×150 cells and $\Delta t = 0.125 \times 10^{-5}$.

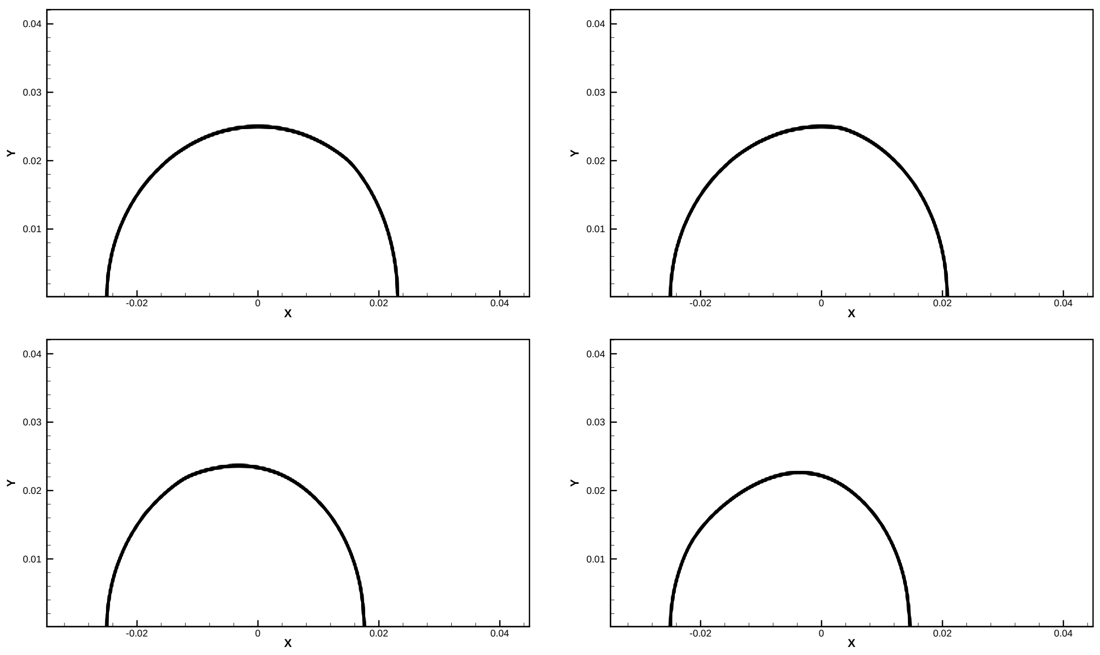


Figure 9.8: Shock hitting R22 bubble. Ratio of specific heats at $t = 10 \times 10^{-3}$, $t = 20 \times 10^{-3}$, $t = 35 \times 10^{-3}$ and $t = 50 \times 10^{-3}$. Grid has 300×150 cells and $\Delta t = 0.125 \times 10^{-5}$.

9.2 Kelvin-Helmholtz instability

To further investigate the possibilities of the LS method the Kelvin-Helmholtz instability is treated. This unstable flow situation occurs when two layers of fluid move with different velocities relative to each other. A small initial disturbance in the interface separating these two *shearing* fluids will roll up in a characteristic vortex pattern, of which the exact shape depends on the amplitude of the initial disturbance and the flow conditions. Examples of such Kelvin-Helmholtz vortex patterns can be found in atmospheric clouds in between two layers of shearing air, in tubes containing shearing two-fluid flows and many other similar problems. In figure 9.9 the Kelvin-Helmholtz instability of an initially sine-shaped interface is shown.

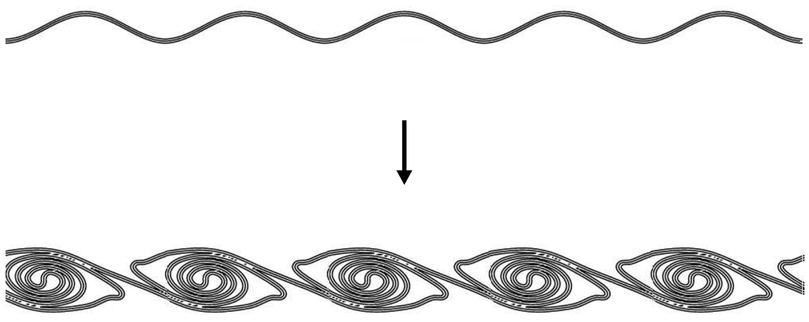


Figure 9.9: The Kelvin-Helmholtz instability occurs when two layers of fluid move with a relative velocity over each other. An initial disturbance in the interface (*top*) will roll-up into the characteristic vortex pattern (*bottom*).

Due to the Kelvin-Helmholtz (KH) instability, the two-fluid interface is severely distorted, possibly leading to topology changes, which requires a sufficiently accurate numerical interface method for its simulation. As such the KH instability acts as a good test case for the LS method developed here. Besides acting as a benchmark problem for our method, it also provides a way of clearly showing the competence of the redistancing procedure. However, due to the intrinsically unstable nature of the KH instability it is difficult to compare the results with previously obtained data. Far-stretching qualifications considering the present method should not be made.

The test case performed here is taken from [37]. Two layers of fluids (top and bottom) are separated by an initially perturbed interface. This perturbation is take equal to a sine-function. Although we speak here about an interface separating two fluids, the first test shall be done using air both above and below the interface. The second test will treat a real two-fluid interface separating air and helium, where the less dense helium is the upper fluid.

9.2.1 Air-air

The first case treats two layers of air moving in opposite directions. The air above the interface moves with velocity $u = 0.25$ to the left and the air below the interface with the same velocity to the right. The density is taken equal to $\rho = 1.4$, the pressure to $p = 1$ and the ratio of specific heats to $\gamma = 1.4$ (this makes the speed of sound equal to $c = 1$). The computational domain has dimensions 1×2 in respectively x and y directions. The grid is chosen to have 100×200 cells. The initial interface perturbation has amplitude 0.1.

The numerical results at several time steps are depicted in figure 9.10. Clearly visible is the initial sine-shaped disturbance in the interface, which grows in time under influence of the shearing flows. The part left of the top of the sine steepens while the right part flattens. The interface further starts to show several disturbances. It is interesting to notice that these disturbances are more severe than was the case in the results in [39]. After a few time steps the steepened part of the interface starts to roll over. In time this roll-up process continues finally leading to the characteristic KH pattern. The results are however significantly different from those in [39]. The cause of these differences is still unknown but can most probably subscribed to the intrinsic nature of the KH instability. Small differences in the velocity field can lead to large differences in the final results.

This sensitive dependency upon the velocity becomes clear when the same problem is done but now with a velocity field that is initially smeared over the interface. Using the smoothed step function H_ϵ the x velocity u is smeared over 3 cells around the interface. This prevents the solution from staircasing. In figure 9.11 the results of this smeared KH problem are given. The results show a significant difference with those obtained with the non-smeared velocity field, indicating that a small change in the velocity field can change the final result completely. This may be the explanation for the difference between the results obtained here and those obtained in [37] and [39].

The final figure shows a topological change in the LS contours. Some contours start to merge together while others split. Clearly the LS method has no difficulties capturing these phenomena. Note that although this merging of the interface implies some kind of mixing of the two fluids, this is not the case. The development of our method is based upon the assumption that both fluids do not mix. If the resolution would be high enough one would observe that the interface remains clearly intact. The *artificial* mixing that is noticed in our results is therefore non-physical. However, an interesting topic for further investigation would be to see whether this artificial mixing could be used as a coarse approximation to physical mixing.

Especially interesting is the behavior of the LS contours away from the interface. Although the LS function distorts due to the non-homogeneous flow field, the LS contours that are shown remain real distances from the interface. This is obviously the result of the redistancing procedure. By redistancing after each time step, the distorted LS function is turned back into a true signed distance function.

9.2.2 Air-helium

The second test treats the KH instability of an interface separating helium and air. The fluid above the interface is helium and the fluid below is air, which density is larger than that of helium. The properties of both air and helium are taken equal to the values used in the shock-bubble interaction problem (table 9.1). The helium used is thus contaminated with a small amount of air. The velocities above and below the interface are pointed in respectively left and right directions but have now a magnitude equal to $u = 0.5$. The grid is the same as for the air-air case.

In figure 9.12 the contours of the LS function are depicted. The interface clearly behaves differently from the interface in the air-air situation. The initial instability now grows to a very sharp wave that *breaks* instead of *rolls*. The lighter helium on top of the air makes that the instability is no longer symmetric. After only a few time steps the interface deforms to such an extent that it is difficult to indicate which fluid is which. This specific breaking-up of the wave resembles the break-up pattern of waves in the ocean, with the difference that no gravity is present in the current model. Again the redistancing procedure has not difficulties with the strongly deformed interface. All other LS contours are neatly placed at their correct distance from the interface.

9.3 Supersonic free jets

An important application from aerospace engineering these days is the interaction between the exhaust plume of a rocket engine and the air flowing around this rocket. Although this problem has been the subject of extensive research already it is still not yet fully understood, which can be partly subscribed to the complexity of the problem. In this problem one is not dealing with a supersonic exhaust plume only, also the interaction of this plume, which often has completely different thermodynamic properties than air, and the surrounding air flow is of major importance. Several experimental and computational studies have already been performed, see for example [48] and [49], resulting in a profound understanding of the problem. However, these computational studies were all based on single-fluid dynamics, which severely reduces the applicability of the results. A first step towards a full two-fluid solution of this problem is made here using the treatment of a supersonic free two-fluid jet.

A free jet occurs when a high-velocity flow exits a nozzle into a gas, often air, producing a jet with characteristics depending on the properties of the jet and the ambient gas. Since we do not want to treat all possible situations, we shall restrict ourselves to one typically interesting case; a supersonic jet exiting

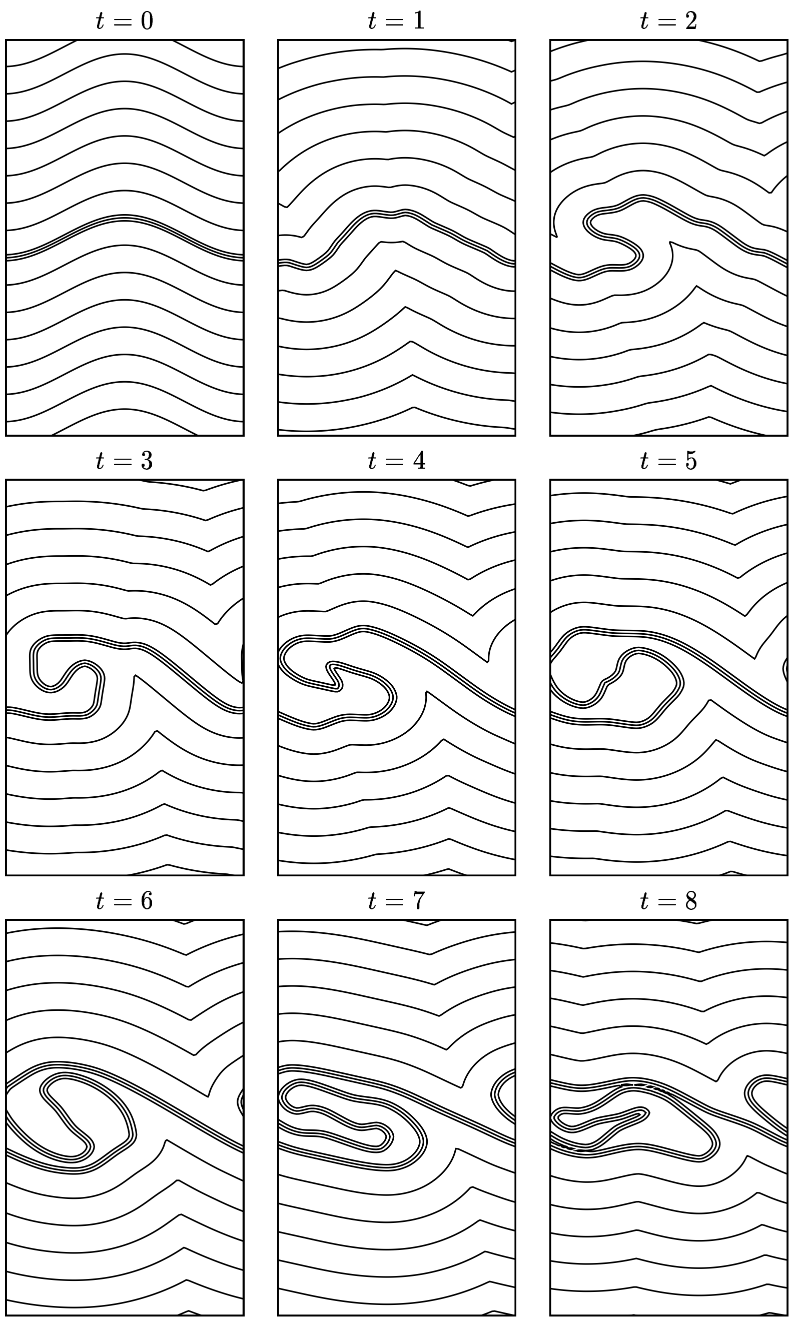


Figure 9.10: Kelvin-Helmholtz instability for air-air interface. Level set function at different time levels. The three dense LS contours are respectively $\varphi = -\frac{3\Delta y}{2}, 0, \frac{3\Delta y}{2}$. Grid has 100×200 cells and $\Delta t = 0.2 \times 10^{-3}$.

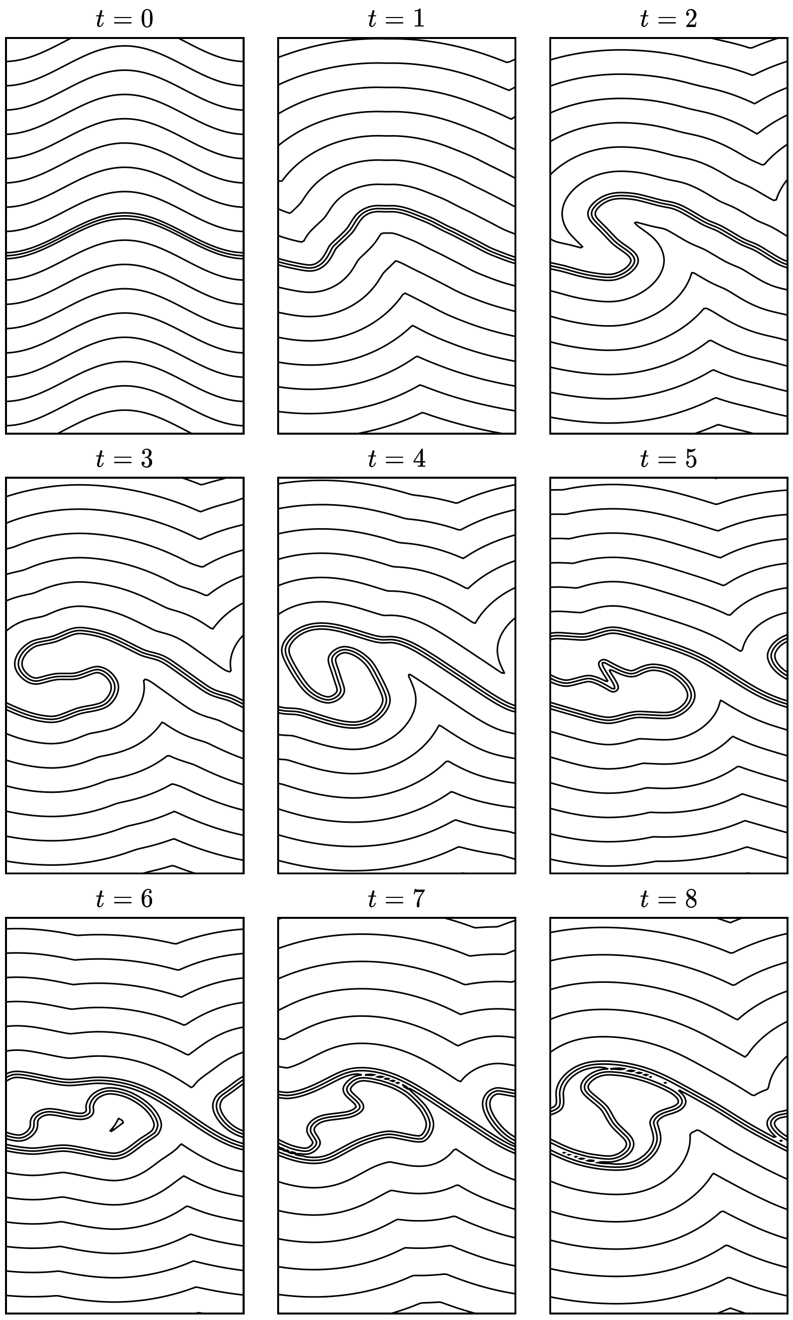


Figure 9.11: Kelvin-Helmholtz instability for air-air interface with smoothed initial velocity field. Level set function at different time levels. The three dense LS contours are respectively $\varphi = -\frac{3\Delta y}{2}, 0, \frac{3\Delta y}{2}$. Grid has 100×200 cells and $\Delta t = 0.2 \times 10^{-3}$.

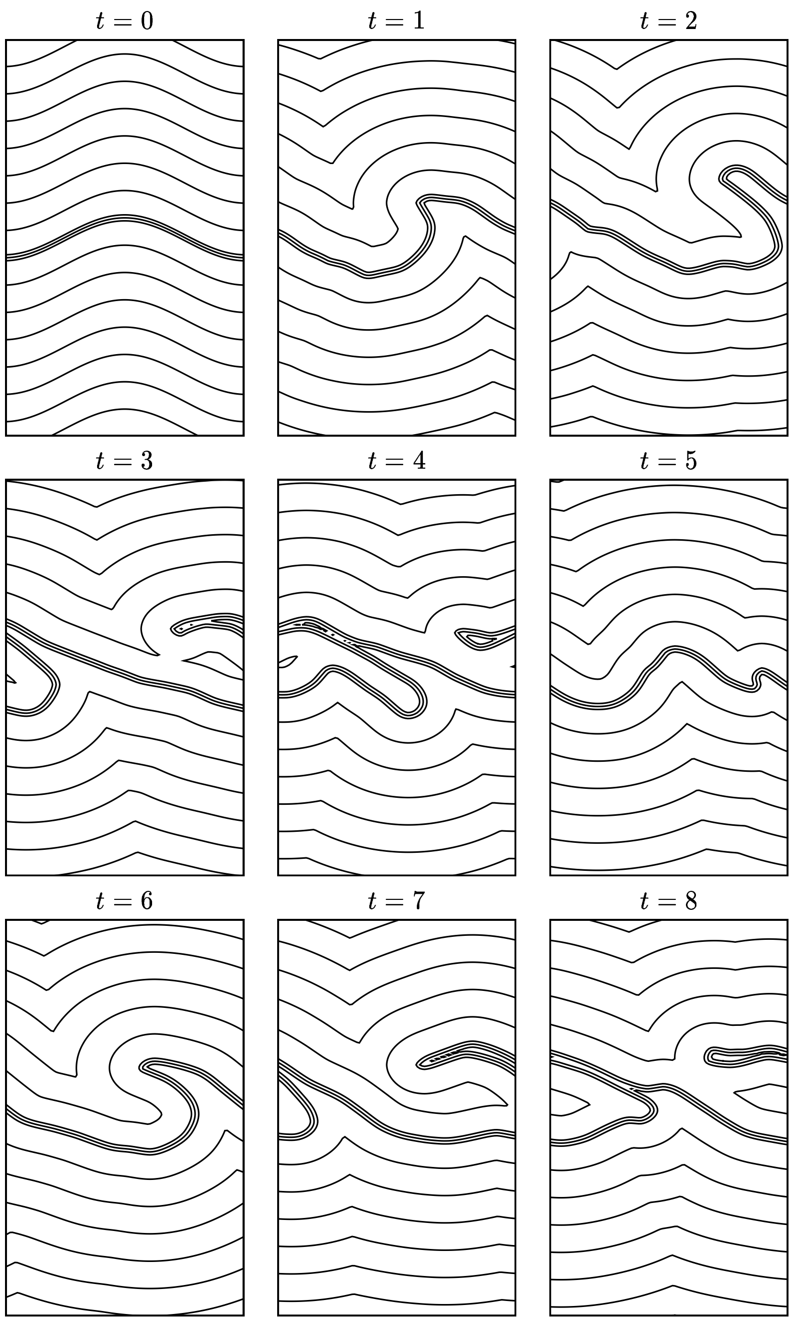


Figure 9.12: Kelvin-Helmholtz instability for air-helium interface. Level set function at different time levels. The three dense LS contours are respectively $\varphi = -\frac{3\Delta y}{2}, 0, \frac{3\Delta y}{2}$. Grid has 100×200 cells and $\Delta t = 1 \times 10^{-3}$.

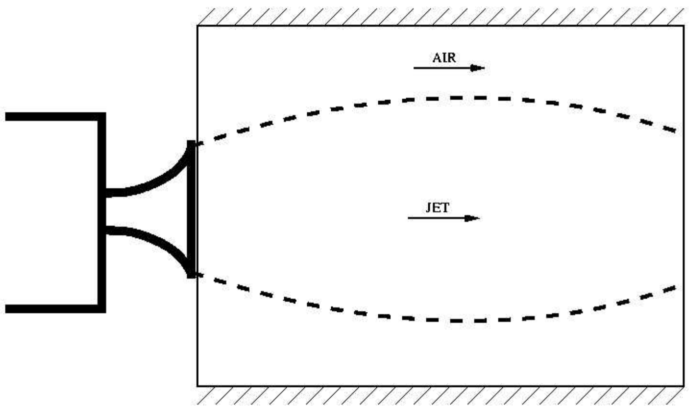


Figure 9.13: A supersonic free jet exits into supersonic ambient air. The rectangular domain represents the computational domain.

in a supersonic co-flowing ambient gas. This specific case is especially interesting since it allows for the waves present in the jet to continue in the air, creating a relatively complex wave pattern. Note that this is different from what was done in [39], where the air outside the jet was assumed to be stagnant. Since only inviscid flows are covered by the solver developed here it is not yet possible to also simulate the flow around the complex geometry of the jet nozzle. Therefore the computational domain is taken as depicted in figure 9.13.

Two specifically interesting situations occur when the jet exits in a gas with either a higher or a lower pressure. Since a jet often has a fixed nozzle geometry, the exhaust conditions of the jet will always be the same, while the ambient pressure will decrease with increasing height. When the jet is below its so-called *design height*, i.e. the height for which the jet and the air have exactly the same pressure, the jet is called *overexpanded*; the exhaust pressure is lower than the ambient pressure. In case the jet is above its design height it is called *underexpanded*, referring to the jet not being expanded to a low enough pressure (that of the ambient air). In both cases the jet will deform due to the appearance of shocks and expansion fans, which adapt the jet pressure to the ambient pressure. For an analytical discussion of the supersonic free jet one can refer to any textbook on supersonic gasdynamics such as [21].

Different from the supersonic free jet treated in [39] here also the start-up process of both over- and underexpanded jets is treated. Initially the jet is assumed to cover a very small circular shaped region in the computational domain. In time this small initial jet will grow under the influence of the continuous exhaust that flows into the domain. In front of the most right part of the jet a region of high pressure is present, which restricts the jet from growing longer and consequently makes it grow wider. After a while the front of the start-up jet will form a characteristic *mushroom shape* known from for example the explosion of an atomic bomb. The detailed shape of this start-up jet depends on the initial conditions and the type of jet considered.

The exhaust gas is taken to be *carbon dioxide*, CO_2, which is the primary product of a burning process, and for the ambient gas air is used. The initial thermodynamic state variables (scaled) for both the over- and underexpanded case are given in table 9.4. The ratio of specific heats for CO_2 is assumed to be 1.3 (based on a tri-atomic gas), while that of air is 1.4. For the air atmospheric conditions at sea level are used, which are scaled such that the speed of sound is equal to 1. The air is further assumed to move with a supersonic velocity to the right, i.e. $M > 1$. The conditions for the CO_2 are chosen such that also the jet has a speed of sound equal to 1. Note that these conditions correspond to a jet temperature of $T_{CO_2} = 1.65T_{\text{air}}$ (based on the specific gas constants $R_{\text{air}} = 287.15J/KgK$ and $R_{CO_2} = 188.96J/KgK$). Such that for atmospheric conditions with $T_{\text{air}} = 288.2K$ the jet has an exit temperature equal to $T_{CO_2} = 475K$, which is a reasonable assumption for rocket engines.

For the computations the lengths of the numerical domain are taken equal to 3×1.5 in x and y direction respectively. This domain is discretized using 400×200 grid cells. Supersonic inflow boundary conditions are prescribed on the left boundary of the domain (see figure 9.13). The outflow at the right boundary of

	γ	ρ	u	v	p
Air (overexp.)	1.4	1.4	2.0	0.0	2.0
CO_2 (overexp.)	1.3	1.3	4.0	0.0	1.0
Air (underexp.)	1.4	1.4	1.25	0.0	0.5
CO_2 (underexp.)	1.3	1.3	2.5	0.0	1.0

Table 9.4: Initial conditions for the supersonic free jet problem.

the domain is assumed to be supersonic such that no conditions have to be prescribed here. The upper and lower boundaries are assumed to be symmetry walls, resembling a wind-tunnel experiment for example. Although this implies that waves reaching the domain boundaries are reflected back towards the jet, this specific choice eases the set-up of the numerical problem considerably. Unless stated otherwise linear basis functions are used in the computations.

9.3.1 Overexpanded jet

The overexpanded jet is characterized by an exhaust jet pressure that is lower than the ambient pressure. The resulting wave pattern is sketched in figure 9.14. Characteristic for the overexpanded jet are the two shocks that originate from the nozzle exit and compress the jet pressure to the ambient pressure. When these shocks meet they deflect each other (sometimes forming compression fans). The jet boundary is initially *sucked* inside but deflects toward the outside when the shock hits it. The shocks reflect as expansion fans and lower the pressure of the jet again. This process continues itself. Due to its specific pattern this jet is sometimes referred to as the *diamond jet*.

In figure 9.14 only the wave pattern inside the jet is sketched. However, this is not the full story. Because the ambient flow is supersonic, the shocks and expansions hitting the interface may also be transmitted partly into the outer flow. In that case, the part of the wave that is reflected back into the jet is clearly weaker than the original wave hitting the interface. Therefore the regions of higher and lower pressure created by the reflected waves may not be as distinct as expected for the subsonic situation in which no transmission occurs. What further complicates the flow problem is the existence of two outward-pointing expansion fans emanating from the nozzle exit. Since the incoming air flow is taken supersonic, two expansion fans appear that curve the outer flow inwards such that it follows the overexpanded jet.

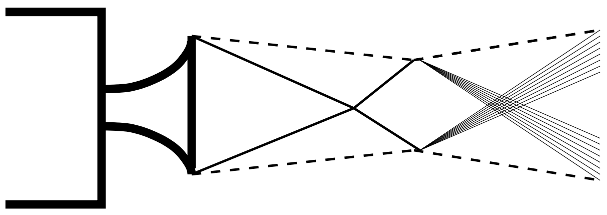


Figure 9.14: A supersonic jet exits into a stagnant gas with higher pressure; the overexpanded jet. Two shock waves increase the exhaust pressure towards the ambient pressure. As a result the interface, which is initially *pushed* inwards, curves outwards again.

The numerical results for the overexpanded jet, using the first-order approximation, are given in figures 9.15 (ratio of specific heats γ), 9.16 (density ρ) and 9.17 (pressure p). In the plots of the ratio of specific heats γ, figure 9.15, the start-up of the jet is clearly visible. Due to the start-up bow shock, a negative pressure gradient in transverse direction is built up around the head of the jet. Hence, the head also grows wider. At later stages the outer parts of this jet head curve backwards and form the earlier mentioned characteristic *mushroom* shape. Together with the start-up shock also the jet head moves out of the computational domain.

When all start-up processes have moved out of the domain, the resulting overexpanded jet pattern becomes visible. At the nozzle exit two straight shocks appear that increase the jet pressure. When these shocks hit eachother they are curved slightly inwards. At a later stage these shocks hit the contact discontinuity at the interface, which results in a complex wave pattern. Part of the shocks continue outside the jet, be it less strong, while the remainder of the shock reflects as an expansion fan. These expansion fans meet each other in the center of the jet, creating a region of lower pressure. These regions combined form the characteristic *diamond* pattern; a region sharply bounded by shocks followed by a region smoothly bounded by expansion fans. This periodic pattern is clearly visible in the pressure plot at time $t = 2.5$.

Outside of the jet one can distinguish two expansion fans emanating from the nozzle exit that curve the flow inwards, leaving a region of lower pressure and density. The air in this region is compressed again by the shock that leaves the jet. When these waves hit the symmetry walls at the top and bottom of the computational domain they are reflected back towards the jet. At a later stage these waves interact with the jet, and as such increase the effect of the expansion fans in the jet.

Besides these outward pointing expansion fans emanating from the nozzle exit, an interesting shock pattern is created outside of the jet. This pattern consists of the shock that emanates from the nozzle exit, propagates into the jet and is partially transmitted outside the jet. When hitting the boundary of the domain this shock is reflected back towards the jet, where it creates a *diamond* of its own, and as such magnifies the original wave pattern in the jet. Also visible outside of the jet is a shock downstream of the reflection of the initial expansion fans. This slightly backwards tilted shock moves to the right. When this tilted shock hits the original shock leaving the jet it creates a shock pattern that resembles a *lambda* shock. At a later stage this moving shock leaves the computational domain leaving a *clean* diamond pattern.

It is interesting to note that these waves outside of the jet were not present in the results in [39]. This is due to the fact that in the present results the outer air flow is taken supersonic instead of stagnant. Since rocket-engine jets often emanate into supersonic ambient flows, the present situation is definitely of interest.

9.3.2 Underexpanded jet

In figure 9.18 the characteristic jet pattern occurring when an underexpanded jet enters a stagnant gas is sketched. Because the pressure in the jet is higher than the ambient pressure two expansion fans originate from the nozzle exit into the jet, lowering the jet pressure. Meanwhile the interface is pushed outwards to some extent. Because the streamlines cross the expansion fans twice the pressure is expanded too much (overexpanded) resulting in a region of low pressure. The same expansion fans are reflected from the jet boundaries, curving the boundaries inwards, creating two reflected compression fans. The pressure in the jet is increased again to a value higher than the ambient pressure, resulting in a region comparable to the inflow region (nozzle exit). When the fans of one compression fan meet they form a shock. If this shock hits the jet boundary it is curved outward while the shock reflects as an expansion fan, which starts the process all over again.

Similar to the preceding overexpanded jet with supersonic ambient air flow a wave pattern exists outside of the jet. Since the jet expands after exiting the nozzle, two outward pointing shocks occur at the nozzle exit that deflect the outer air flow in the direction of the interface. The air is compressed by these shocks resulting in a region of higher pressure and density. Further downstream the air in this region is expanded again by the expansion fans that crossed the interface.

Plots of the distributions of the ratio of specific heats γ, the density ρ and the pressure p are given in figures 9.19, 9.16 and 9.17, respectively. Results are obtained using the first-order accurate solver. A higher-order oscillation-free solution could not yet be obtained. Again one can clearly distinguish the start-up process of the jet. The head starts expanding due to the earlier mentioned pressure gradient. Note that this process differs significantly from that in the overexpanded case. The head of the jet clearly grows wider.

In the final figure the steady state of the underexpanded jet is given. Visible is the initial region of high pressure. Two expansion fans expand the jet to a lower pressure and density. When hitting the interface these fans are transmitted mostly, resulting in two expansion fans in the outer flow. The expansions that are reflected by the interface form compression fans that compress the gas in the jet again. These reflected expansions curve the boundary inward again, resulting in the characteristic underexpanded shape of the

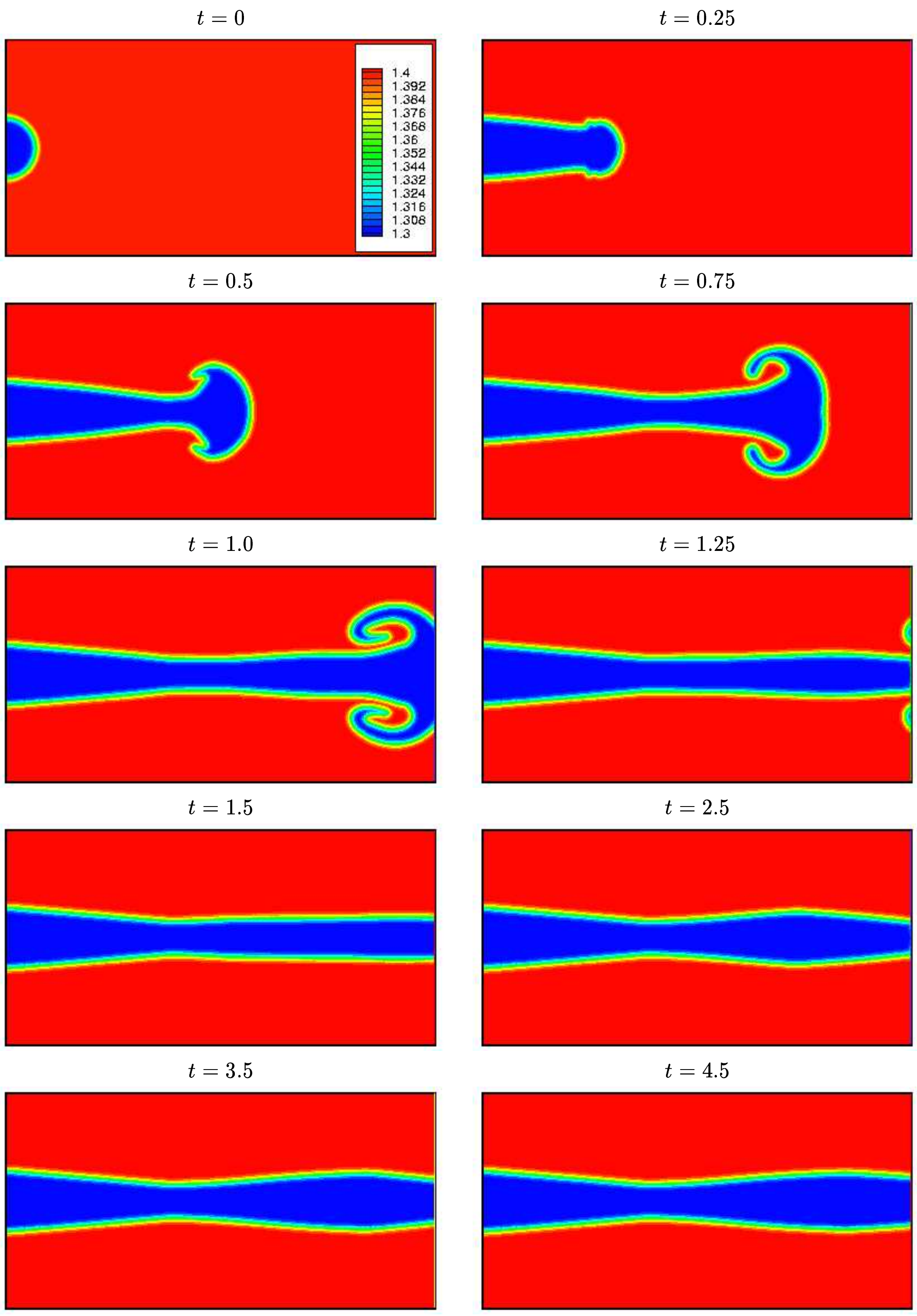


Figure 9.15: Supersonic overexpanded jet. Ratio of specific heats γ at different time levels. Pressure of ambient air is 2 times higher than the exhaust pressure. Grid has 400×200 cells and $\Delta t = 1.25 \times 10^{-4} \Delta x$.

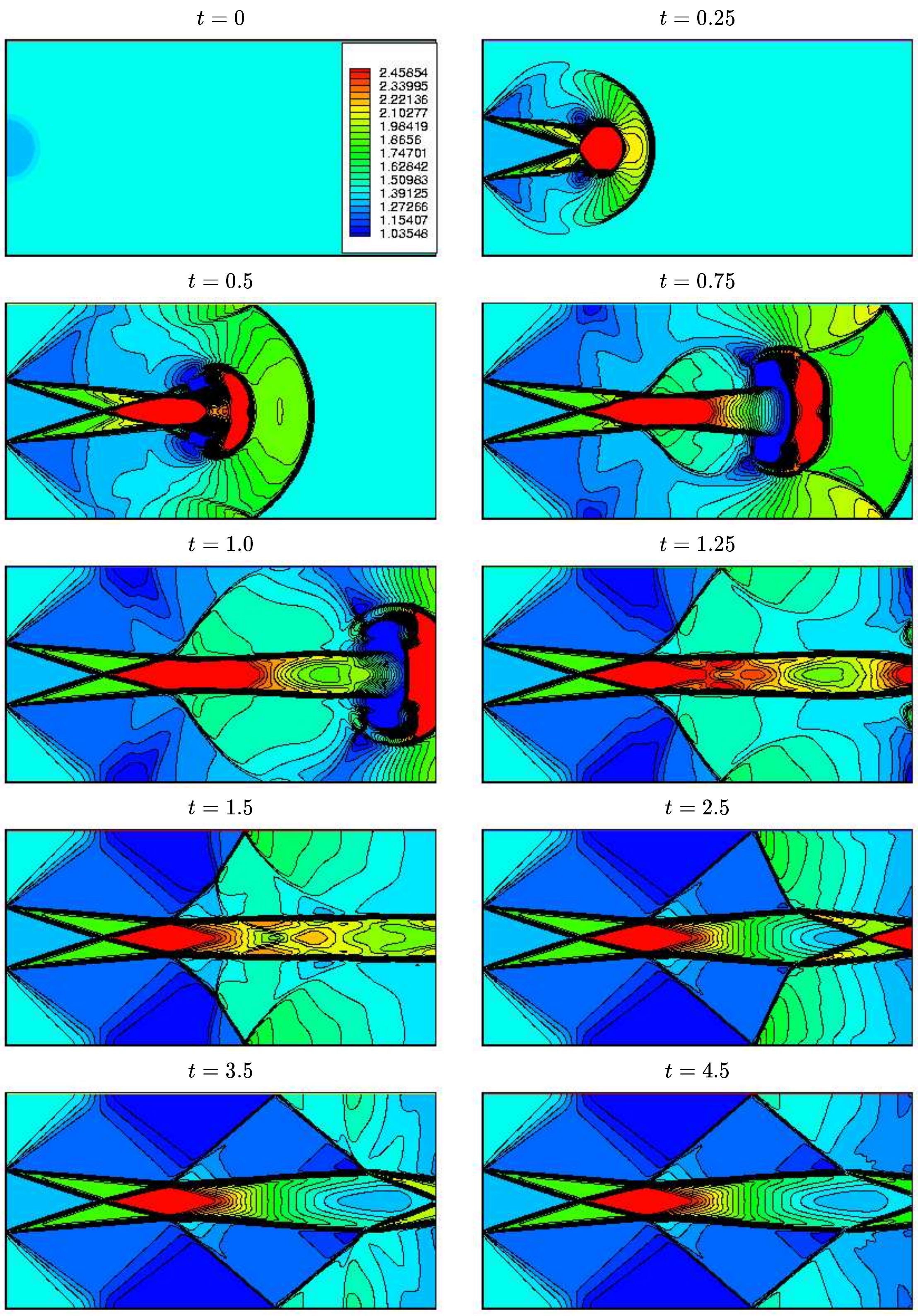


Figure 9.16: Supersonic overexpanded jet. Density ρ at different time levels. Pressure of ambient air is 2 times higher than the exhaust pressure. Grid has 400×200 cells and $\Delta t = 1.25 \times 10^{-4} \Delta x$.

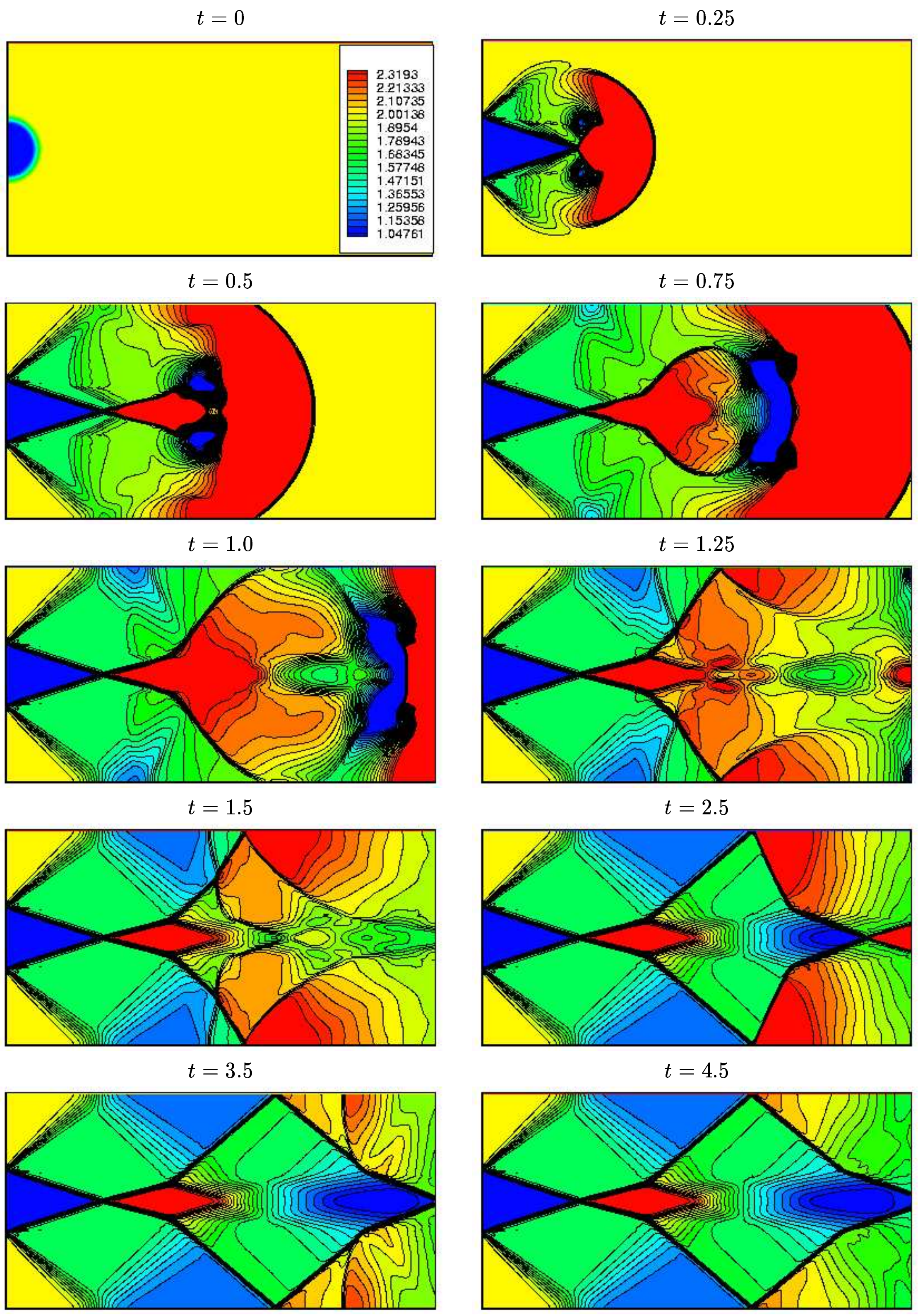


Figure 9.17: Supersonic overexpanded jet. Pressure p at different time levels. Pressure of ambient air is 2 times higher than the exhaust pressure. Grid has 400×200 cells and $\Delta t = 1.25 \times 10^{-4} \Delta x$.

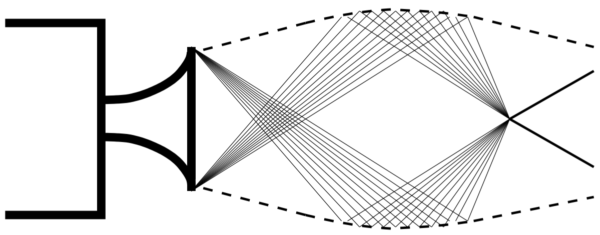


Figure 9.18: A supersonic jet exits into a stagnant gas with lower pressure; the underexpanded jet. Two expansion fans lower the exhaust pressure towards the ambient pressure. As a result the interface, which is initially *pushed* outwards, curves inwards again.

jet. Note that the regions of higher and lower pressure and density are less distinct than is the case for the overexpanded jet. This is clearly due to the first-order accuracy of the present method.

Interesting to note is that different from the overexpanded case, the underexpanded jet requires more time to fully converge to a ’steady’ state. Different from the overexpanded jet, which showed a nearly steady jet when all waves left the outflow boundary, the underexpanded jet clearly shows several slowly moving regions. The most right region of lower pressure and density moves instead of out of the domain, back into the visible region. This behavior can most probably be subscribed to the lower velocities of the underexpanded problem. In order to show several periods of this jet the flow velocities are chosen lower than was the case for the overexpanded case. However, the final figure at $t = 8.4$ seems to be steady. No further movement is seen, indicating a converged jet.

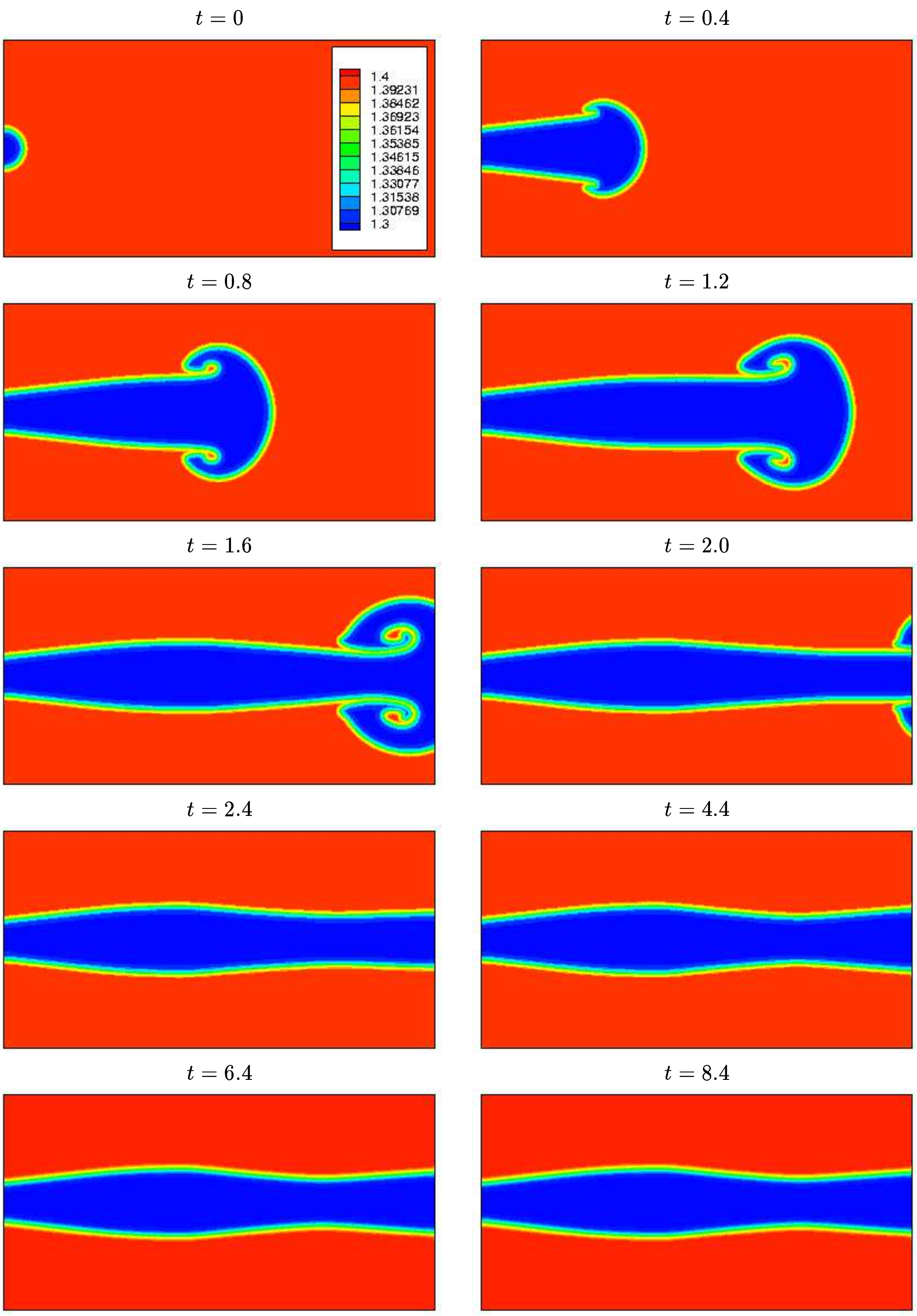


Figure 9.19: Supersonic underexpanded jet. Ratio of specific heats γ at different time levels. Pressure of ambient air is 2 times lower than the exhaust pressure. Grid has 400×200 cells and $\Delta t = 3.75 \times 10^{-4} \Delta x$. Note that results are first-order accurate.

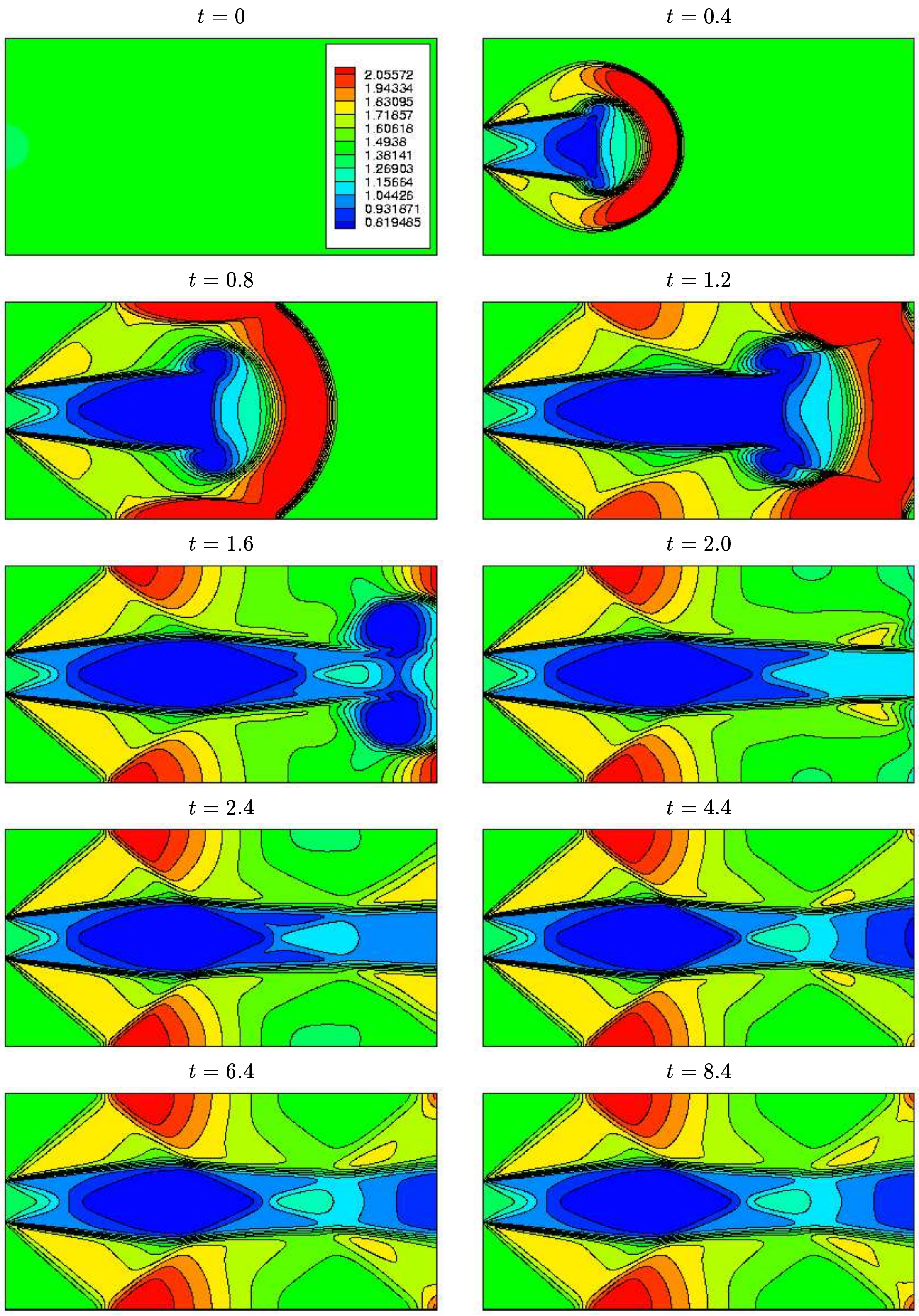


Figure 9.20: Supersonic underexpanded jet. Density ρ at different time levels. Pressure of ambient air is 2 times lower than the exhaust pressure. Grid has 400×200 cells and $\Delta t = 3.75 \times 10^{-4} \Delta x$. Note that results are first-order accurate.

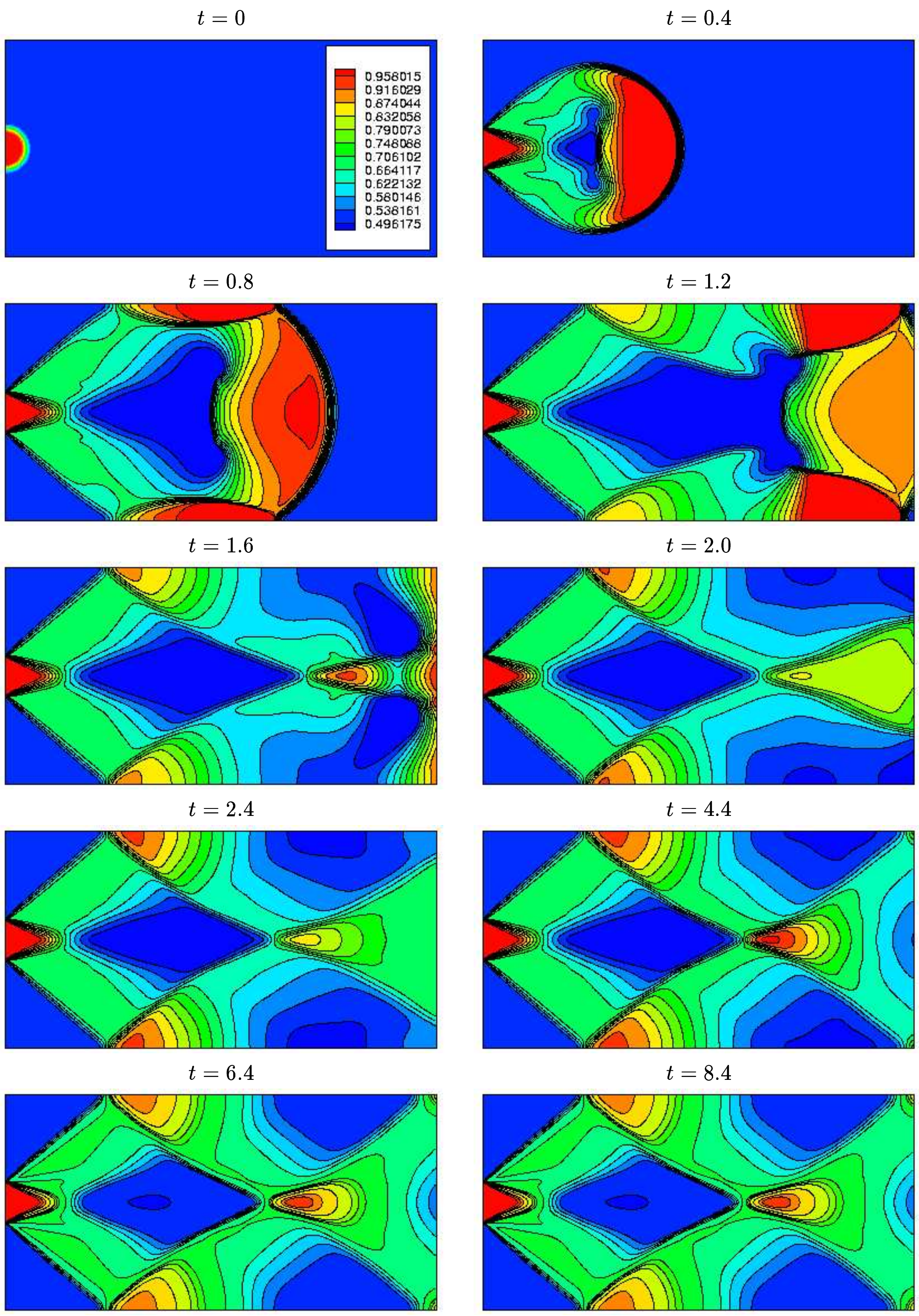


Figure 9.21: Supersonic underexpanded jet. Pressure p at different time levels. Pressure of ambient air is 2 times lower than the exhaust pressure. Grid has 400×200 cells and $\Delta t = 3.75 \times 10^{-4} \Delta x$. Note that results are first-order accurate.

Chapter 10

Conclusions

10.1 Current work

In this work a numerical method for the solution of the two-dimensional Euler equations describing unsteady compressible two-fluid flows is presented. The method is based on the combination of a Runge-Kutta discontinuous Galerkin discretization method for the Euler equations and a level-set method for the treatment of the two-fluid interface.

To describe the evolution of the two-fluid interface a level-set (LS) equation is added to the system formed by the Euler equations. This LS equation is a transport equation for the LS function which is a signed distance function that labels every point in the computational domain with a value representing the shortest distance to the interface and a sign corresponding to either one of the fluids. For the LS equation two possible forms are available; the advective form which is simply the advection equation for the LS function, and the conservative form which is obtained when the LS equation is combined with the equation for the conservation of mass. In chapter 3 it is shown that the frequently used conservative formulation of the LS equation generates an erroneous off-set in the interface location. An analytical expression for this error is derived which clearly shows that the corresponding error depends upon the density jump over the interface. When the LS equation is used in its advective form this error does not occur indicating that this is clearly the approach to be used. To prevent the LS function from becoming distorted a redistancing procedure is applied. This redistancing is done using a PDE-method which transforms the distorted LS function back to a real signed distance function. For a consistent and error-free interface treatment the two-fluid interface is smeared out over several cells using a smoothed-step function.

Typical for conservative two-fluid methods is that they generate spurious oscillations near the two-fluid interface. In order to prevent these oscillations a simple fix is applied. The first part of this fix is based on the requirement that a numerical cell can only use information from one fluid. The second part of the fix requires that during an update of the state variables only the old value of ratio of specific heats can be used. An analytical treatment proves the competence of this fix.

A Runge-Kutta discontinuous Galerkin (RKDG) method is used for the discretization of the Euler equations. For the piecewise discontinuous basis functions orthogonal Legendre polynomials are used. Depending on the required order of accuracy $p+1$ these polynomials are of the order p, i.e. linear basis functions for a second-order scheme and quadratic basis functions for a third-order scheme. For the calculation of the cell-face fluxes Roe's approximate Riemann solver is used. Gaussian quadrature rules are used for the approximation of the volume integrals. Time marching is done using a $p+1$ stage Runge-Kutta scheme, which gives the complete RKDG method a $p+1$ order of accuracy in both space and time. To prevent the solution from becoming oscillatory a slope limiter is implemented in the RKDG method. This special DG slope limiter changes the gradients of the approximate solution within a cell when spurious oscillations are generated. It is shown that the original single-fluid slope limiter does not suffice to prevent the earlier mentioned interface oscillations from occurring. Therefore a novel two-fluid slope limiter is developed which is based upon the simple pressure fix.

Several numerical tests are performed to test the performance of the present method. Results from one-dimensional shock tube problems show the competence of the numerical method. Discontinuous phenomena show less numerical diffusion than results obtained with similar methods. However, it is also shown that the third-order method ($p = 2$) does not show much improvement as compared to the second-order method ($p = 1$). This behavior can be fully subscribed to the slope limiter. For an effective use of the present method a more accurate slope limiter has to be developed. Several two-dimensional problems, being the shock-bubble interaction, the Kelvin-Helmholtz instability and the supersonic free jet, show again the improved accuracy of the RKDG LS method compared to other methods.

10.2 Future work

Advanced two-fluid slope limiter for DG

In order to fully use the possibilities of the RKDG method a more advanced two-fluid slope limiter is required. The limiter used in the present method degrades a third-order method completely to a second-order method when strong discontinuities are present in the problem under consideration. A possibility is the TVBM limiter by Cockburn, Hou and Shu [16]. This limiter is still based upon the rather simplistic minmod limiter. Improvement could be obtained using a more advanced limiter, such as the limiter developed by Koren [30]. More recent developments by Bruneau *et al.* in [12] in the direction of problem-independent limiters for the RKDG method show great potential too.

Stability operators for DG

A more recent development for DG methods is the application of stability operators instead of slope limiters. In [58] these stability operators are treated in more detail.

Adaptive mesh refinement.

To reduce the effect of the level-set method being locally non-conservative an adaptive mesh refinement method could be used. Increasing the number of cells near the interface will resolve this problem partially. Because the level-set method provides the exact location of the interface, mesh refinement can be easily implemented.

New interface approach.

The fix for the spurious pressure oscillations makes the level-set method locally non-conservative. Although this seems to have no large influence for the problems treated, it will when very strong shocks occur. It is therefore necessary to continue the search for a less severe fix. A different interface approach is necessary which prevents the spurious pressure oscillations but does not affect the conservation properties.

Extension to full 3D Navier-Stokes

In order to be able to fully tackle the exhaust plume problem, taking into account the flow of the ambient air over the nozzle, a full 3D Navier-Stokes solver is required. Extension of the present method to the 3D Navier-Stokes equations is therefore an interesting topic for further research.

Bibliography

[1] R. Abgrall, How to prevent pressure oscillations in multicomponent flow calculations: A quasi conservative approach, *J. Comp. Phys.* **125**, pages 150–160, 1996.

[2] R. Abgrall and S. Karni, Computations of compressible multifluids, *J. Comp. Phys.* **169**, pages 594–623, 2001.

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

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

[5] C.E. Baumann and J.T. Oden, *A discontinuous hp finite element method for the Navier-Stokes equations*, 10th. International Conference on Finite Elements in Fluids, 1998.

[6] C.E. Baumann and J.T. Oden, *A discontinuous hp finite element method for the Euler equation of gas dynamics*, 10th. International Conference on Finite Elements in Fluids, 1998.

[7] D.J. Benson, Computational methods for Lagrangian and Eulerian Hydroflow, *CMAME* **99**, pages 235–394, 1992.

[8] K.S. Bey and J.T. Oden, *A Runge-Kutta discontinuous Galerkin finite element method for high speed flows*, 10th. AIAA Computational Fluid Dynamics Conference, Honolulu, Hawaii, June 24–27, 1991.

[9] K.S. Bey and J.T. Oden, hp-version discontinuous Galerkin methods for hyperbolic conservation laws, *Comp. Meth. Appl. Mech. Eng.* **133**, pages 259–286, 1996.

[10] K.S. Bey, A. Patra and J.T. Oden, A parallel hp-adaptive discontinuous Galerkin method for hyperbolic conservation laws, *Appl. Num. Math.* **20**, pages 286-321, 1996.

[11] E.H. van Brummelen and B. Koren, A pressure-invariant conservative Godunov-type method for barotropic two-fluid flows, *J. Comp. Phys.* **185**, pages 289-308, 2003.

[12] A. Burbeau, P. Sagaut and Ch.-H. Bruneau, A problem-independent limiter for high-order Runge-Kutta discontinuous Galerkin methods, *J. Comp. Phys* **169**, pages 111–150, 2001.

[13] G. Chavent and B. Cockburn, The local projection P^0P^1-discontinuous Galerkin finite element method for scalar conservation laws, RAIRO modél, *Math, Anal. Numér.* **23**, pages 565–592, 1989.

[14] G. Chavent and G. Salzano, A finite element method for the 1D water flooding problem with gravity, *J. Comp. Phys.* **45**, pages 307–344, 1982.

[15] B. Cockburn, *An introduction to the discontinuous Galerkin method for convection-dominated problems*, Lecture notes, School of Mathematics, University of Minnesota.

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

[17] B. Cockburn, G.E. Karniadakis and C.W. Shu, *The development of discontinuous Galerkin methods*, Lecture Notes in Computational Science and Engineering, `http://www.msi.umn.edu/general/Reports/rptfiles/UMSI99-220/UMSI99-220.ps.Z`.

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

[19] B. Cockburn and C.W. Shu, TVB Runge-Kutta local projection discontinuous Galerkin finite element method for conservation laws II: general framework, *Math. Comp.* **52**, pages 411–435, 1989.

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

[21] R. Courant and K.O. Friedrichs, *Supersonic Flow and Shock Waves*, Springer Verlag, New York, 1976.

[22] R.P. Fedkiw, T. Aslam, B. Merriman and S. Osher, A non-oscillatory Eulerian approach to interfaces in multimaterial flows, *J. Comp. Phys.* **152**, pages 457–492, 1999.

[23] S.K. Godunov, A finite difference method for the computation of discontinuous solutions of the equations of fluid dynamics, *Mat. Sb.* **47**, pages 357–393, 1959.

[24] H. Guillard and A. Murrone, *A five-equation reduced model for compressible two phase flow problems*, INRIA Rapport de recherche N^0 4778, Institut National de Recherche en Informatique et en Automatique, Sophia Antipolis, 2003.

[25] J.F. Haas and B. Sturtevant, Interaction of weak shock waves with cylindrical and spherical gas inhomogeneities, *J. Fluid Mech.* **81**, pages 41–76, 1987.

[26] C.W. Hirt and B.D. Nichols, Volume of fluid (VOF) method for the dynamics of free boundaries, *J. Comp. Phys.* **39**, pages 201–225, 1981.

[27] C. Johnson and J. Pitkäranta, An analysis of the discontinuous Galerkin method for a scalar hyperbolic equation, *Math. Comp.* **46**, pages 1–26, 1986.

[28] S. Karni, Multicomponent flow calculations by a consistent primitive algorithm, *J. Comp. Phys.* **112**, pages 31–43, 1994.

[29] S. Karni, Hybrid multifluid algorithms, *SIAM J. Sci. Comp.* **17**, pages 1019–1039, 1996.

[30] B. Koren, A robust upwind discretization method for advection, diffusion and source terms, in: *Numerical Methods for Advection-Diffusion Problems* (C.B. Vreugdenhil and B. Koren, eds.), *Notes on Numerical Fluid Mechanics*, **45**, 117-138, Vieweg, Braunschweig (1993).

[31] B. Koren, M.R. Lewis, E.H. van Brummelen and B. van Leer, *Riemann-problem and level-set approaches for two-fluid flow computations, I: Linearized Godunov scheme*, Report MAS-R0112, CWI, `http://ftp.cwi.nl/CWIreports/MAS/MAS-R0112.pdf`, 2001.

[32] B. Koren, M.R. Lewis, E.H. van Brummelen and B. van Leer, *Riemann-problem and level-set approaches for two-fluid flow computations, II: Fixes for solution errors near interfaces,* Report MAS-R0113, CWI, `http://ftp.cwi.nl/CWIreports/MAS/MAS-R0113.pdf`, Amsterdam, 2001.

[33] B. van Leer, Towards the ultimate conservation difference scheme, II, *J. Comp. Phys.* **14**, pages 361–376, 1974.

[34] B. van Leer, Towards the ultimate conservation difference scheme, V, *J. Comp. Phys.* **32**, pages 1–136, 1979.

[35] P. LeSaint and P.A. Raviart, *On a finite element method for solving the neutron transport equation*, Mathematical aspects of finite elements in partial differential equations (C. de Boor, ed.), pages 89–145, Acadamic Press, 1975.

[36] G.H. Markstein, *Chapter B: Theory of Flame propagation*, in: Non-Steady Flame Propagation (G.H. Markstein, ed.), pages 5–14, Pergamon Press, New York, 1964.

[37] W.A. Mulder, S. Osher and J.A. Sethian, Computing interface motion in compressible gas dynamics, *J. Comp. Phys.* **100**, pages 209–228, 1992.

[38] J. Naber, *Building your own shock tube*, Report MAS-E0502, CWI, `http://ftp.cwi.nl/CWIreports/MAS/MAS-E0502.pdf`, Amsterdam, 2005.

[39] J. Naber, *A numerical solver for compressible two-fluid flow*, Report MAS-E0505, CWI, `http://ftp.cwi.nl/CWIreports/MAS/MAS-E0505.pdf`, Amsterdam, 2005.

[40] S. Osher and F.P. Fedkiw, *Level Set Methods and Dynamic Implicit Surfaces*, Applied Mathematical Sciences 152, Springer-Verlag, 2002.

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

[42] T. Peterson, A note on the convergence of the discontinuous Galerkin method for a scalar hyperbolic equation, *SIAM J. Numer. Anal.* **28**, pages 133–140, 1991.

[43] J.J. Quirk and S. Karni, *On the dynamics of a shock-bubble interaction*, ICASE Report 94-75, Institute for Computer Applications in Science and Engineering, NASA Langley Research Center, Hampton, VA, 1994.

[44] W.H. Reed and T.R. Hill, *Triangular mesh methods for the neutron equation*, Tech. Report LA-UR-73-479, Los Alamos Scientific Laboratory, 1973.

[45] G.R. Richter, An optimal-order error estimate for the discontinuous Galerkin method, *Math. Comp.* **50**, pages 75–88, 1988.

[46] P.L. Roe, Approximate Riemann solvers, parameter vectors, and difference schemes, *J. Comp. Phys.* **43**, pages 357-372, 1981.

[47] R. Saurel and R. Abgrall, A multiphase Godunov method for compressible multifluid and multiphase flows, *J. Comp. Phys.* **150**, pages 425–467, 1999.

[48] M.M.J. Schoones and W.J. Bannink, *Base Flow and Exhaust Plume Interaction, part 1: experimental study*, Series 01, Aerodynamics 15, Delft University Press, 1998.

[49] M.M.J. Schoones and W.J. Bannink, *Base Flow and Exhaust Plume Interaction, part 2: computational study*, Series 01, Aerodynamics 16, Delft University Press, 1998.

[50] J.A. Sethian, *Level Set Methods and Fast Marching Methods: Evolving Interfaces in Computational Geometry, Fluid Mechanics, Computer Vision, and Materials Science*, Cambridge University Press, 1999.

[51] J.A. Sethian, Curvature flow and entropy conditions applied to grid generation, *J. Comp. Phys.* **115**, pages 440–454, 1994.

[52] C.W. Shu, TVB uniformly high order schemes for conservation laws, *Math. Comp.* **49**, pages 105–121, 1987.

[53] C.W. Shu and S. Osher, Efficient implementation of Essentially Non-Oscillatory shock-capturing schemes, II, *J. Comp. Phys.* **83**, pages 32–78, 1989.

[54] S.P. Spekreijse, *Multigrid solution of the steady Euler equations*, PhD dissertation, Centre for Mathematics and Computer Science, Amsterdam 1987.

[55] M. Sussman and E. Fatemi, An efficient, interface-preserving level set redistancing algorithm and its application to interfacial incompressible fluid flow, *SIAM J. Scient. Comp.* **20**, pages 1165–1191, 1999.

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

[57] E.F. Toro, *Riemann Solvers and Numerical Methods for Fluid Dynamics, A Practical Introduction*, 2^{nd} edition, Springer-Verlag, Berlin, 1999.

[58] J.J.W. van der Vegt and H. van der Ven, Space–time discontinuous Galerkin finite element method with dynamic grid motion for inviscid compressible flows I: general formulation, *J. Comp. Phys.* **182**, pages 546–585, 2002.

[59] C.B. Vreugdehil, Introduction, in: *Numerical Methods for Advection-Diffusion Problems* (C.B. Vreugdenhil and B. Koren, eds.), *Notes on Numerical Fluid Mechanics,* **45**, 117-138, Vieweg, Braunschweig (1993).

[60] J. Wackers and B. Koren, *Five-equation model for compressible two-fluid flow*, Report MAS-E0414, CWI, `http://ftp.cwi.nl/CWIreports/MAS/MAS-E0414.pdf`, Amsterdam, 2004.

Appendix A

Orthogonal basis functions for Discontinuous Galerkin

The discontinuous Galerkin method used for the discretization of the Euler equations makes use of a set of basis functions that span the complete solution domain. Here we have chosen these basis as the collection of polynomials of order p, indicated with $\mathcal{P}^p$. In order to simplify the evaluation of the integral terms in the system of equations (5.6) these basis functions are chosen such that they are orthogonal within an element $E_{i,j} = (x_{i-\frac{1}{2}}, x_{i+\frac{1}{2}}) \times (y_{j-\frac{1}{2}}, y_{j+\frac{1}{2}})$. The Legendre polynomials applied here is one set of polynomials that satisfy this condition. Below we shall give these basis functions and work out the integral term containing the time derivative.

A.1 Basis functions for 1D

The approximate solution $\boldsymbol{q}_h$ can be written as a combination of degrees of freedom and basis functions, which gives in the one-dimensional case the following:

$$\boldsymbol{q}_h\left(x,t\right) = \sum_{l=0}^{p} \boldsymbol{q}_i^{(l)}\left(t\right)\phi_i^{(l)}\left(x\right), \qquad \forall x \in E_i, \tag{A.1}$$

For these basis functions Legendre polynomials are used, which are orthogonal on an element $E_i = (x_{i-\frac{1}{2}}, x_{i+\frac{1}{2}})$:

$$\phi_i^{(0)} = 1, \quad \phi_i^{(1)} = \frac{x - x_i}{\Delta x}, \quad \phi_i^{(2)} = \frac{\left(x - x_i\right)^2}{\Delta x^2} - \frac{1}{12}, \quad \ldots, \tag{A.2}$$

In figure A.1 these first three basis function are given for an element E_i with $x_i = 0$ and $\Delta x = 1$. Due to the orthogonality property of these basis functions the time-dependent integral term in (5.6) can be written as an explicit expression for one of the unknowns $\boldsymbol{q}_i^{(l)}$. This allows for the use of an explicit time-marching method, which simplifies the implementation of the DG method to a large extent. Working out the integral term for $l = 0, \ldots, 2$ gives:

$$\int_{E_i} \phi_i^{(l)} \boldsymbol{q}_h d\Omega \Rightarrow \begin{bmatrix} \boldsymbol{q}_i^{(0)} \\ \frac{1}{12}\boldsymbol{q}_i^{(1)} \\ \frac{1}{180}\boldsymbol{q}_i^{(2)} \end{bmatrix} \Delta x \tag{A.3}$$

Here we clearly see that each equation of our system of equations can be written as an explicit equation for one of the unknowns $\boldsymbol{q}_i^{(l)}$ when the corresponding factor is moved to the right. Note that we have moved the time derivative out of the integral, which is allowed in the case of constant size grid cells.

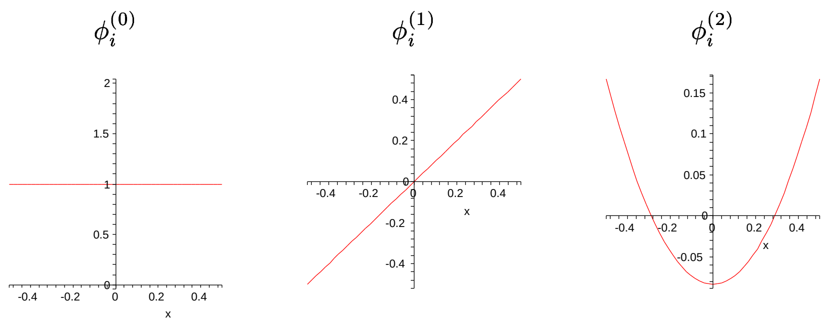


Figure A.1: 1D orthogonal basis functions based on Legendre polynomials scaled on the element $E_i = (x_{i-\frac{1}{2}}, x_{i+\frac{1}{2}})$.

A.2 Basis functions for 2D

The two-dimensional approximate solution $\boldsymbol{q}_h$ is written in a similar fashion as the one-dimensional approximation (5.8), only now we split our x and y directions by using separate one-dimensional basis functions. Using $\phi_i^{(k)}$ for the basis functions in x direction and $\psi_j^{(l)}$ for the y direction we can write:

$$\boldsymbol{q}_h(x,y,t) = \sum_{k,l=0}^{p} \boldsymbol{q}_{i,j}^{(k,l)}(t)\,\phi_i^{(k)}(x)\,\psi_j^{(l)}(y), \qquad \forall x,y \in E_{i,j}, \tag{A.4}$$

Here the basis functions are exactly the same as those used for the one-dimensional case. Thus:

$$\phi_i^{(0)} = 1, \quad \phi_i^{(1)} = \frac{x - x_i}{\Delta x}, \quad \phi_i^{(2)} = \frac{(x-x_i)^2}{\Delta x^2} - \frac{1}{12}, \quad \ldots, \tag{A.5}$$

for the x direction and:

$$\psi_j^{(0)} = 1, \quad \psi_j^{(1)} = \frac{y - y_j}{\Delta y}, \quad \psi_j^{(2)} = \frac{(y-y_j)^2}{\Delta y^2} - \frac{1}{12}, \quad \ldots, \tag{A.6}$$

for the y direction. In figure A.2 the first few basis function products $\phi_i^{(k)}\psi_j^{(l)}$ are plotted for an element $E_{i,j} = (x_{i-\frac{1}{2}}, x_{i+\frac{1}{2}}) \times (y_{j-\frac{1}{2}}, y_{j+\frac{1}{2}})$ with $(x_i, y_j) = (0,0)$ and $\Delta x = \Delta y = 1$. Since both $\phi_i^{(k)}$ and $\psi_j^{(l)}$ are orthogonal, their tensor product $\phi_i^{(k)}\psi_j^{(l)}$ is orthogonal too. Working out the time-dependent integral leads again to a set of explicit equations for the unknowns $\boldsymbol{q}_{i,j}^{(k,l)}$:

$$\begin{aligned}\int_{E_{i,j}} \phi_i^{(k)}\psi_j^{(l)}\boldsymbol{q}_h d\Omega \quad &= \quad \boldsymbol{q}_{i,j}^{(k,l)}(t)\left[\int_{x_{i-\frac{1}{2}}}^{x_{i+\frac{1}{2}}} \left(\phi_i^{(k)}\right)^2 dx\right]\left[\int_{y_{j-\frac{1}{2}}}^{y_{j+\frac{1}{2}}} \left(\psi_j^{(l)}\right)^2 dy\right] \\ &\Rightarrow \left[\begin{array}{ccc} \boldsymbol{q}_{i,j}^{(0,0)} & \frac{1}{12}\boldsymbol{q}_{i,j}^{(0,1)} & \frac{1}{180}\boldsymbol{q}_{i,j}^{(0,2)} \\ \frac{1}{12}\boldsymbol{q}_{i,j}^{(1,0)} & \frac{1}{144}\boldsymbol{q}_{i,j}^{(1,1)} & \frac{1}{2160}\boldsymbol{q}_{i,j}^{(1,2)} \\ \frac{1}{180}\boldsymbol{q}_{i,j}^{(2,0)} & \frac{1}{2160}\boldsymbol{q}_{i,j}^{(2,1)} & \frac{1}{32400}\boldsymbol{q}_{i,j}^{(2,2)} \end{array}\right] \Delta x \Delta y. \end{aligned} \tag{A.7}$$

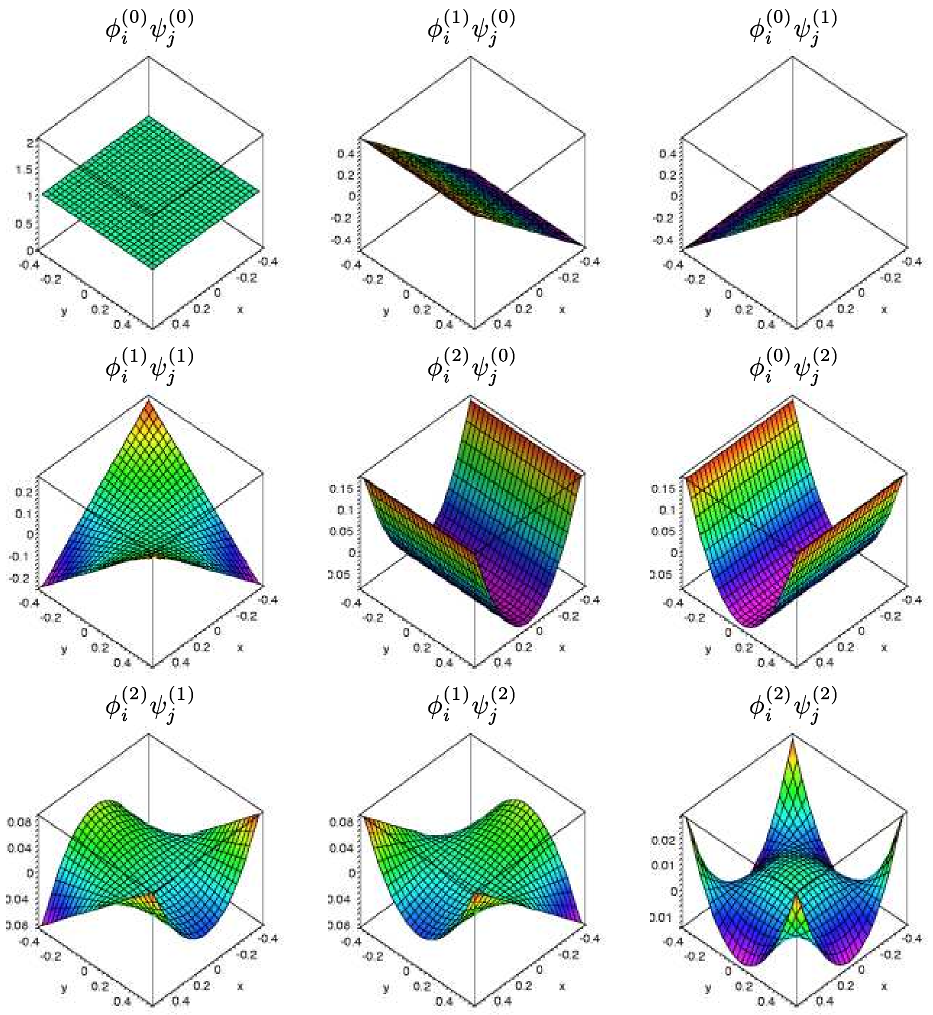


Figure A.2: 2D orthogonal basis functions based on Legendre polynomials scaled on the element $E_{i,j} = (x_{i-\frac{1}{2}}, x_{i+\frac{1}{2}}) \times (y_{j-\frac{1}{2}}, y_{j+\frac{1}{2}})$.

Appendix B

Transformation of degrees of freedom

In chapter 7 the transformation between primary $\boldsymbol{w}_h$ and conservative $\boldsymbol{q}_h$ state variables is discussed. Although the corresponding transformation is rather straightforward, due to the nonlinear dependency of the approximate solution $\boldsymbol{q}_h$ on the locations x and y it is not possible to simply transform the $(p+1)^2$ independent degrees of freedom $\boldsymbol{q}_{i,j}^{(k,l)}$. It is because of this that transforming between conservative and primary state variables should be done using the approximate solution $\boldsymbol{q}_h$ evaluated in $(p+1)^2$ points within a cell, i.e. $\boldsymbol{q}_h^{(m,n)} = \boldsymbol{q}_h\left(x^{(m,n)}, y^{(m,n)}, t\right)$ where m and n are the indices corresponding to the evaluation points. Given the expression for the approximate solution (5.8):

$$\boldsymbol{q}_h\left(x,y,t\right) = \sum_{k,l=0}^{p} \boldsymbol{q}_{i,j}^{(k,l)}\left(t\right)\phi_i^{(k)}\left(x\right)\psi_j^{(l)}\left(y\right),$$

these evaluated solutions can be written as:

$$\boldsymbol{q}_h^{(m,n)} = \boldsymbol{q}_h\left(x^{(m,n)}, y^{(m,n)}, t\right) = \sum_{k,l=0}^{p} \boldsymbol{A}^{(k,l,m,n)}\boldsymbol{q}_{i,j}^{(k,l)} \quad \text{for} \quad m,n = 0,\ldots,p. \tag{B.1}$$

where the basis functions evaluated in the corresponding evaluation point $\left(x^{(m,n)}, y^{(m,n)}\right)$ are gathered together into $\boldsymbol{A}^{(k,l,m,n)}$. If one now combines all degrees of freedom $\boldsymbol{q}_{i,j}^{(k,l)}$ into the vector $\boldsymbol{q}_M$ (*cell mean values*) and all evaluated solutions $\boldsymbol{q}_h^{(m,n)}$ into $\boldsymbol{q}_B$ (*cell basis values*) expression (B.1) can be written as:

$$\boldsymbol{q}_B = \boldsymbol{A}\boldsymbol{q}_M. \tag{B.2}$$

The matrix $\boldsymbol{A}$ thus transforms the *cell mean values* (degrees of freedom) to *cell basis values* (the evaluated solutions). Transformation from cell basis to cell mean values can now easily be expressed using the inverse of matrix $\boldsymbol{A}$:

$$\boldsymbol{q}_M = \left[\boldsymbol{A}\right]^{-1}\boldsymbol{q}_B. \tag{B.3}$$

When the elements of matrix $\boldsymbol{A}$ are known the transformation between degrees of freedom and evaluated values is straightforward. Evaluation of these elements requires a choice of evaluation points. A smart choice for these evaluation points can severely reduce the complexity of matrix $\boldsymbol{A}$. In figure B.1 the chosen points are given for a cell in the case of linear and quadratic basis functions. For both situations the transformation matrix $\boldsymbol{A}$ is given in tables B.1 and B.2 respectively.

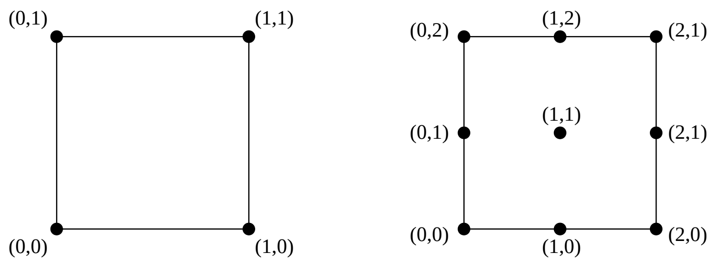


Figure B.1: Evaluation points within a grid cell for both linear (*Left*) and quadratic (*right*) basis functions. The number of evaluation points is equal to $(p+1)^2$.

$\boldsymbol{q}_B^{(m,n)} \backslash \boldsymbol{q}_M^{(k,l)}$	(0,0)	(1,0)	(0,1)	(1,1)
(0,0)	1	$-1/2$	$-1/2$	$1/4$
(1,0)	1	$1/2$	$-1/2$	$-1/4$
(0,1)	1	$-1/2$	$1/2$	$-1/4$
(1,1)	1	$1/2$	$1/2$	$1/4$

Table B.1: Transformation matrix for linear basis functions.

$\boldsymbol{q}_B^{(m,n)} \backslash \boldsymbol{q}_M^{(k,l)}$	(0,0)	(1,0)	(0,1)	(2,0)	(1,1)	(0,2)	(2,1)	(1,2)	(2,2)
(0,0)	1	$-1/2$	$-1/2$	$1/4$	$1/6$	$1/6$	$-1/12$	$-1/12$	$1/36$
(1,0)	1	0	$-1/2$	0	$-1/12$	$1/6$	$1/24$	0	$-1/72$
(0,1)	1	$-1/2$	0	0	$1/6$	$-1/12$	0	$1/24$	$-1/72$
(2,0)	1	$1/2$	$-1/2$	$-1/4$	$1/6$	$1/6$	$-1/12$	$1/12$	$1/36$
(1,1)	1	0	0	0	$-1/12$	$-1/12$	0	0	$1/144$
(0,2)	1	$-1/2$	$1/2$	$-1/4$	$1/6$	$1/6$	$1/12$	$-1/12$	$1/36$
(2,1)	1	$1/2$	0	0	$1/6$	$-1/12$	0	$-1/24$	$-1/72$
(1,2)	1	0	$1/2$	0	$-1/12$	$1/6$	$-1/24$	0	$-1/72$
(2,2)	1	$1/2$	$1/2$	$1/4$	$1/6$	$1/6$	$1/12$	$1/12$	$1/36$

Table B.2: Transformation matrix for quadratic basis functions.

Appendix C

The general Riemann problem

Many numerical methods require the calculation of fluxes over cell-faces for the calculation of the solution at a new time level. For the Euler equations of gasdynamics these cell-face fluxes can be calculated by finding the solution to the Riemann problem on each cell face. Given a left and right piecewise constant initial state the Riemann problem can be used to determine the correct upwind solution on the cell-face, and consequently the corresponding flux. However, this approach only works for a piecewise constant initial distribution in the cells. The more complex flow structure present in a cell in case of for example a piecewise linear distribution is neglected completely when the standard Riemann approach is used.

The problem sketched above can be fixed with the generalized Riemann problem approach (GR). In this method the cell-face fluxes are calculated with information about the complete solution in the neighboring cells. Higher-order spatial derivatives of the solution are taken into account which allows the numerical solution to obtain its optimal order of accuracy. However, a major draw back of this method is its increase in computational complexity. The GR approach requires the calculation of the derivative of the Jacobian of the Euler equations.

It can be shown that when a multi-stage method is used for the calculation of the numerical solution at a new time level, it is not necessary to use this GR approach. The multi-stage nature already takes into account the highly structured approximation within a cell. To prove this we shall derive the numerical order of accuracy of both a single-stage scheme with the GR approach and a multi-stage Runge-Kutta scheme with a standard Riemann approach. Both methods should be second-order accurate in space and time.

Before we compare the standard and the GR approach for the Euler equations, we first explain the GR approach in more detail by treating a simple model problem.

C.1 GR for a model problem

We shall introduce the generalized Riemann (GR) method for a piecewise linear discontinuous Galerkin approximation of a model problem such that we obtain a scheme that is both second-order accurate in space and time. The model problem that we will use is the following linear scalar advection equation:

$$\frac{\partial u}{\partial t} + a\frac{\partial u}{\partial x} = 0, \tag{C.1}$$

where u is the quantity we would like to solve for and a the advection coefficient, which can be interpreted as a velocity.

C.1.1 Discretization

For the spatial discretization we use the discontinuous Galerkin (DG) method with linear basis functions. We assume our solution in cell j to consist of a mean $\overline{u}_j$ and a gradient $\overline{\Delta u_j}$ such that we obtain a piecewise linear distribution (see figure C.1).

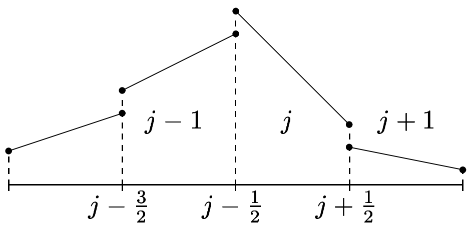


Figure C.1: A linear representation of the solution on a discrete domain.

Here it suffices to only treat the discrete equation for the mean. Writing out this semi-discrete equation (discrete in space but not in time) gives:

$$\Delta x \frac{d\overline{u}_j}{dt} + a\left[\overline{u}_{j+\frac{1}{2}} - \overline{u}_{j-\frac{1}{2}}\right] = 0, \tag{C.2}$$

where $\overline{u}_{j\pm\frac{1}{2}}$ are the cell-face values of $\overline{u}_j$. Since we work with a piecewise linear approximation which is discontinuous at the cell-faces these cell-face values are not defined. Therefore we have to decide which of the two values at the cell-face we should use. Based on the flow direction (the sign of a) we choose between the left or right cell-face values, $\overline{u}^L_{j\pm\frac{1}{2}}$ or $\overline{u}^R_{j\pm\frac{1}{2}}$:

$$\Delta x \frac{d\overline{u}_j}{dt} + \max(a,0)\left[\overline{u}^L_{j+\frac{1}{2}} - \overline{u}^L_{j-\frac{1}{2}}\right] + \min(a,0)\left[\overline{u}^R_{j+\frac{1}{2}} - \overline{u}^R_{j-\frac{1}{2}}\right] = 0. \tag{C.3}$$

For now we shall only treat the case of a flow to the right, $a > 0$, such that we can rewrite (C.3):

$$\Delta x \frac{d\overline{u}_j}{dt} + a\left[\overline{u}^L_{j+\frac{1}{2}} - \overline{u}^L_{j-\frac{1}{2}}\right] = 0. \tag{C.4}$$

We now introduce together with the time discretization, for which we take the simple first-order accurate *forward Euler* scheme, the generalized Riemann approach. The standard Riemann approach together with the forward Euler scheme calculates the solution at the new time level $n+1$ by using the solution and the fluxes at the previous time level n. The GR approach on the contrary, replaces these *old fluxes* with intermediate values at time level $n+\alpha$. This can be written as:

$$\Delta x \frac{\overline{u}^{n+1}_j - \overline{u}^n_j}{\Delta t} + a\left[\left(\overline{u}^L_{j+\frac{1}{2}}\right)^{n+\alpha} - \left(\overline{u}^L_{j-\frac{1}{2}}\right)^{n+\alpha}\right] = 0. \tag{C.5}$$

Given the DG approximation in each cell at time level n we can now write the upwind values in (C.5) as follows:

$$\begin{array}{rclcl}
\left(\overline{u}^L_{j+\frac{1}{2}}\right)^{n+\alpha} & = & \overline{u}^n_j + \frac{\Delta x}{2}\left(\frac{\partial \overline{u}_j}{\partial x}\right)^n & + & \alpha\Delta t \left(\frac{\partial \overline{u}^L_{j+\frac{1}{2}}}{\partial t}\right)^n, \\
\left(\overline{u}^L_{j-\frac{1}{2}}\right)^{n+\alpha} & = & \overline{u}^n_{j-1} + \frac{\Delta x}{2}\left(\frac{\partial \overline{u}_{j-1}}{\partial x}\right)^n & + & \alpha\Delta t \left(\frac{\partial \overline{u}^L_{j-\frac{1}{2}}}{\partial t}\right)^n, \\
 & & \Downarrow & & \Downarrow \\
 & & \text{DG part} & & \text{GR correction}
\end{array} \tag{C.6}$$

where the constant α is still to be determined. Note that taking $\alpha = 0$ returns us a scheme that is first-order in time and second-order in space. Introducing this intermediate time level at which we calculate the fluxes allows us to increase the order of our time discretization to second order. The following step could be to discretize the time derivatives in (C.6) but this would require the introduction of extra time levels. Here we will use the original equation (C.1) to obtain:

$$\left(\frac{\partial \overline{u}^L_{j+\frac{1}{2}}}{\partial t}\right)^n = -a\left(\frac{\partial \overline{u}^L_{j+\frac{1}{2}}}{\partial x}\right)^n = -a\left(\frac{\partial \overline{u}_j}{\partial x}\right)^n, \tag{C.7}$$

$$\left(\frac{\partial \overline{u}^L_{j-\frac{1}{2}}}{\partial t}\right)^n = -a\left(\frac{\partial \overline{u}^L_{j-\frac{1}{2}}}{\partial x}\right)^n = -a\left(\frac{\partial \overline{u}_{j-1}}{\partial x}\right)^n. \tag{C.8}$$

Substitution of these expressions into (C.6) and (C.5) finally leads to the following expression for the mean of the solution in cell j at time level $n+1$:

$$\overline{u}_j^{n+1} = \overline{u}_j^n - \frac{a\Delta t}{\Delta x}\left[\{\overline{u}_j - \overline{u}_{j-1}\} + \left(\frac{\Delta x}{2} - a\alpha\Delta t\right)\left\{\left(\frac{\partial \overline{u}_j}{\partial x}\right)^n - \left(\frac{\partial \overline{u}_{j-1}}{\partial x}\right)^n\right\}\right]. \tag{C.9}$$

C.1.2 Accuracy analysis

To determine the order of accuracy of the discretization applied to the advection equation we will derive the modified equation. Taylor series expansions around the point j, n are substituted into (C.9) such that we obtain after some rewriting the following expression:

$$\begin{aligned} u_t + au_x + \frac{1}{2}\Delta t u_{tt} - a^2\alpha\Delta t u_{xx} + \frac{1}{6}\Delta t^2 u_{ttt} - \frac{1}{12}\Delta x^2 a u_{xxx} + a^2\alpha\Delta t\frac{\Delta x}{2}u_{xxx} \\ = \mathcal{O}\left(\Delta x^3, \Delta x^2\Delta t, \Delta t^3\right). \end{aligned} \tag{C.10}$$

Taking the derivative with respect to time of this expression gives:

$$u_{tt} = -au_{xt} - \frac{1}{2}\Delta t u_{ttt} + a^2\alpha\Delta t u_{xxt} + \mathcal{O}\left(\Delta x^2, \Delta x\Delta t, \Delta t^2\right), \tag{C.11}$$

which can be substituted into (C.10) such that we finally obtain:

$$\begin{aligned} u_t + au_x - \frac{1}{2}a\Delta t\frac{\partial}{\partial x}\left(u_t + 2a\alpha u_x\right) - \frac{1}{12}\Delta t^2 u_{ttt} + \frac{1}{2}a^2\alpha\Delta t^2 u_{xxt} \\ + \left(\frac{1}{2}a^2\alpha\Delta t\Delta x - \frac{1}{12}\Delta x^2 a\right)u_{xxx} = \mathcal{O}\left(\Delta x^3, \Delta x^2\Delta t, \Delta x\Delta t^2, \Delta t^3\right). \end{aligned} \tag{C.12}$$

It is not difficult to see now that a smart choice for α would be $\alpha = \frac{1}{2}$. Working out (C.12) for this specific choice of α gives:

$$\begin{aligned} \left(1 - \frac{1}{2}a\Delta t\frac{\partial}{\partial x}\right)(u_t + au_x) - \frac{1}{12}\Delta t^2 u_{ttt} + \frac{1}{4}a^2\Delta t^2 u_{xxt} + \left(a\Delta t - \frac{1}{3}\Delta x\right)\frac{1}{4}a\Delta x u_{xxx} \\ = \mathcal{O}\left(\Delta x^3, \Delta x^2\Delta t, \Delta x\Delta t^2, \Delta t^3\right). \end{aligned} \tag{C.13}$$

Dividing the whole expression by the term in front of $(u_t + au_x)$ gives us the desired result:

$$u_t + au_x = \mathcal{O}\left(\Delta x^2, \Delta x\Delta t, \Delta t^2\right). \tag{C.14}$$

We thus see that a DG discretization with piecewise linear approximations together with a GR approach gives us a discretization which is second order accurate in both space and time. Note however that we have only derived the modified equation for the cell mean $\overline{u}_j$. A similar derivation can be performed for the derivative, but this shall not be elaborated here.

C.2 GR for the Euler equations

In the previous section we have seen that the GR method can be used to obtain a numerical scheme that is both second order accurate in space and time for the linear scalar advection equation. Here we shall follow a similar approach for the one-dimensional Euler equations of gasdynamics:

$$\frac{\partial \boldsymbol{q}}{\partial t} + \frac{\partial \boldsymbol{f}(\boldsymbol{q})}{\partial x} = \boldsymbol{0}. \tag{C.15}$$

Here $\boldsymbol{q}$ is the conservative state vector and $\boldsymbol{f}$ the flux vector. For the one-dimensional version of the Euler equations these vectors each have three components, representing the equations for conservation of mass, momentum and energy, respectively:

$$\boldsymbol{q} = \begin{pmatrix} \rho \\ \rho u \\ \rho E \end{pmatrix}, \qquad \boldsymbol{f} = \begin{pmatrix} \rho u \\ p + \rho u^2 \\ \rho u H \end{pmatrix}. \tag{C.16}$$

Their components include the three primary state variables being, ρ the density, u the velocity and p the pressure. Further they contain the total energy E and the total enthalpy H. In this discussion only ideal gasses are considered, which allows for an explicit expression relating the primary thermodynamic variables:

$$p = \rho e\left(\gamma - 1\right), \tag{C.17}$$

where e is the internal energy of the flow and γ the ratio of specific heats, equal to 1.4 for air. The total energy can then be written as:

$$E = e + \frac{1}{2}u^2, \tag{C.18}$$

where e follows from (C.17). The total enthalpy H is defined as:

$$H = E + \frac{p}{\rho}. \tag{C.19}$$

The Euler equations (3.5) form a system of purely hyperbolic conservation laws.

C.2.1 Discretization

Similar to the approach we followed for the linear scalar advection equation we discretize the Euler equations in space with a DG method based on piecewise linear approximations. This gives us:

$$\Delta x \frac{\partial \overline{\boldsymbol{q}}_j}{\partial t} + \left[\boldsymbol{f}\left(\overline{\boldsymbol{q}}_{j+\frac{1}{2}}\right) - \boldsymbol{f}\left(\overline{\boldsymbol{q}}_{j-\frac{1}{2}}\right)\right] = 0. \tag{C.20}$$

We define at each cell-face a numerical flux $\tilde{\boldsymbol{f}}_{j+\frac{1}{2}} = \tilde{\boldsymbol{f}}\left(\boldsymbol{q}^L_{j+\frac{1}{2}}, \boldsymbol{q}^R_{j+\frac{1}{2}}\right)$. This flux can for example be calculated using the exact Riemann solver by Godunov or by one of the approximate solvers. Substitution of these fluxes in (C.20) gives:

$$\Delta x \frac{\partial \overline{\boldsymbol{q}}_j}{\partial t} + \left[\tilde{\boldsymbol{f}}\left(\boldsymbol{q}^L_{j+\frac{1}{2}}, \boldsymbol{q}^R_{j+\frac{1}{2}}\right) - \tilde{\boldsymbol{f}}\left(\boldsymbol{q}^L_{j-\frac{1}{2}}, \boldsymbol{q}^R_{j-\frac{1}{2}}\right)\right] = 0. \tag{C.21}$$

For the discretization in time we again follow the GR approach. For the time derivative we use the forward Euler scheme but we calculate the numerical fluxes at an intermediate time level instead of at the old level. In the previous section we found that $\alpha = \frac{1}{2}$ yields second-order accuracy. Hence, here we directly take:

$$\Delta x \frac{\overline{\boldsymbol{q}}_j^{n+1} - \overline{\boldsymbol{q}}_j^n}{\Delta t} + \left[\tilde{\boldsymbol{f}}\left(\boldsymbol{q}^L_{j+\frac{1}{2}}, \boldsymbol{q}^R_{j+\frac{1}{2}}\right)^{n+\frac{1}{2}} - \tilde{\boldsymbol{f}}\left(\boldsymbol{q}^L_{j-\frac{1}{2}}, \boldsymbol{q}^R_{j-\frac{1}{2}}\right)^{n+\frac{1}{2}}\right] = 0. \tag{C.22}$$

This obviously requires the cell-face values at the intermediate time level:

$$\begin{array}{rcll}
\left(\overline{\boldsymbol{q}}^L_{j+\frac{1}{2}}\right)^{n+\frac{1}{2}} & = & \overline{\boldsymbol{q}}^n_j + \frac{\Delta x}{2}\left(\frac{\partial \overline{\boldsymbol{q}}_j}{\partial x}\right)^n & + \ \frac{1}{2}\Delta t\left(\frac{\partial \overline{\boldsymbol{q}}^L_{j+\frac{1}{2}}}{\partial t}\right)^n, \\
\left(\overline{\boldsymbol{q}}^R_{j+\frac{1}{2}}\right)^{n+\frac{1}{2}} & = & \overline{\boldsymbol{q}}^n_{j+1} - \frac{\Delta x}{2}\left(\frac{\partial \overline{\boldsymbol{q}}_{j+1}}{\partial x}\right)^n & + \ \frac{1}{2}\Delta t\left(\frac{\partial \overline{\boldsymbol{q}}^R_{j+\frac{1}{2}}}{\partial t}\right)^n, \\
\left(\overline{\boldsymbol{q}}^L_{j-\frac{1}{2}}\right)^{n+\frac{1}{2}} & = & \overline{\boldsymbol{q}}^n_{j-1} + \frac{\Delta x}{2}\left(\frac{\partial \overline{\boldsymbol{q}}_{j-1}}{\partial x}\right)^n & + \ \frac{1}{2}\Delta t\left(\frac{\partial \overline{\boldsymbol{q}}^L_{j-\frac{1}{2}}}{\partial t}\right)^n, \\
\left(\overline{\boldsymbol{q}}^R_{j-\frac{1}{2}}\right)^{n+\frac{1}{2}} & = & \overline{\boldsymbol{q}}^n_j - \frac{\Delta x}{2}\left(\frac{\partial \overline{\boldsymbol{q}}_j}{\partial x}\right)^n & + \ \frac{1}{2}\Delta t\left(\frac{\partial \overline{\boldsymbol{q}}^R_{j-\frac{1}{2}}}{\partial t}\right)^n. \\
 & & \Downarrow & \qquad \Downarrow \\
 & & \text{DG part} & \text{GR correction}
\end{array} \tag{C.23}$$

With the original equation (2.8) we can write for the time derivatives:

$$\begin{aligned}
\frac{\partial \overline{\boldsymbol{q}}^L_{j+\frac{1}{2}}}{\partial t} &= -\boldsymbol{f}'\left(\overline{\boldsymbol{q}}^L_{j+\frac{1}{2}}\right)\frac{\partial \overline{\boldsymbol{q}}^L_{j+\frac{1}{2}}}{\partial x} = -\boldsymbol{f}'\left(\overline{\boldsymbol{q}}_j + \frac{\Delta x}{2}\frac{\partial \overline{\boldsymbol{q}}_j}{\partial x}\right)\frac{\partial\left(\overline{\boldsymbol{q}}_j + \frac{\Delta x}{2}\frac{\partial \overline{\boldsymbol{q}}_j}{\partial x}\right)}{\partial x}, &\text{(C.24)}\\
\frac{\partial \overline{\boldsymbol{q}}^R_{j+\frac{1}{2}}}{\partial t} &= -\boldsymbol{f}'\left(\overline{\boldsymbol{q}}^R_{j+\frac{1}{2}}\right)\frac{\partial \overline{\boldsymbol{q}}^R_{j+\frac{1}{2}}}{\partial x} = -\boldsymbol{f}'\left(\overline{\boldsymbol{q}}_{j+1} - \frac{\Delta x}{2}\frac{\partial \overline{\boldsymbol{q}}_{j+1}}{\partial x}\right)\frac{\left(\overline{\boldsymbol{q}}_{j+1} - \frac{\Delta x}{2}\frac{\partial \overline{\boldsymbol{q}}_{j+1}}{\partial x}\right)}{\partial x}, &\text{(C.25)}\\
\frac{\partial \overline{\boldsymbol{q}}^L_{j-\frac{1}{2}}}{\partial t} &= -\boldsymbol{f}'\left(\overline{\boldsymbol{q}}^L_{j-\frac{1}{2}}\right)\frac{\partial \overline{\boldsymbol{q}}^L_{j-\frac{1}{2}}}{\partial x} = -\boldsymbol{f}'\left(\overline{\boldsymbol{q}}_{j-1} + \frac{\Delta x}{2}\frac{\partial \overline{\boldsymbol{q}}_{j-1}}{\partial x}\right)\frac{\left(\overline{\boldsymbol{q}}_{j-1} + \frac{\Delta x}{2}\frac{\partial \overline{\boldsymbol{q}}_{j-1}}{\partial x}\right)}{\partial x}, &\text{(C.26)}\\
\frac{\partial \overline{\boldsymbol{q}}^R_{j-\frac{1}{2}}}{\partial t} &= -\boldsymbol{f}'\left(\overline{\boldsymbol{q}}^R_{j-\frac{1}{2}}\right)\frac{\partial \overline{\boldsymbol{q}}^R_{j-\frac{1}{2}}}{\partial x} = -\boldsymbol{f}'\left(\overline{\boldsymbol{q}}_j - \frac{\Delta x}{2}\frac{\partial \overline{\boldsymbol{q}}_j}{\partial x}\right)\frac{\left(\overline{\boldsymbol{q}}_j - \frac{\Delta x}{2}\frac{\partial \overline{\boldsymbol{q}}_j}{\partial x}\right)}{\partial x}. &\text{(C.27)}
\end{aligned}$$

It is clear that these expressions require the derivative of the flux function $\boldsymbol{f}$. When going to even higher orders, such as third-order accuracy, this approach even requires the second-order derivatives of the flux function. Given the computational complexity this approach is thus quite impractical. In the next section we will show that the GR approach is in fact not necessary for obtaining a second-order accurate discretization.

C.3 Runge-Kutta for the Euler equations

Here we will show that a second-order accurate Runge-Kutta time-marching method together with the standard Riemann approach suffices to obtain a discretization that is second-order accurate in space and time.

C.3.1 Discretization

We consider the following semi-discrete equation:

$$\Delta x\frac{\partial \overline{\boldsymbol{q}}_j}{\partial t} + \left[\tilde{\boldsymbol{f}}\left(\boldsymbol{q}^L_{j+\frac{1}{2}}, \boldsymbol{q}^R_{j+\frac{1}{2}}\right)^n - \tilde{\boldsymbol{f}}\left(\boldsymbol{q}^L_{j-\frac{1}{2}}, \boldsymbol{q}^R_{j-\frac{1}{2}}\right)^n\right] = 0. \tag{C.28}$$

Again we look for the left and right cell-face values, only now there is no GR correction present since we try to obtain second-order accuracy by choosing a suitable discretization for the time derivative only:

$$\begin{aligned}
\left(\overline{\boldsymbol{q}}^L_{j+\frac{1}{2}}\right)^{n+\frac{1}{2}} &= \overline{\boldsymbol{q}}^n_j + \tfrac{\Delta x}{2}\left(\tfrac{\partial \overline{\boldsymbol{q}}_j}{\partial x}\right)^n,\\
\left(\overline{\boldsymbol{q}}^R_{j+\frac{1}{2}}\right)^{n+\frac{1}{2}} &= \overline{\boldsymbol{q}}^n_{j+1} - \tfrac{\Delta x}{2}\left(\tfrac{\partial \overline{\boldsymbol{q}}_{j+1}}{\partial x}\right)^n,\\
\left(\overline{\boldsymbol{q}}^L_{j-\frac{1}{2}}\right)^{n+\frac{1}{2}} &= \overline{\boldsymbol{q}}^n_{j-1} + \tfrac{\Delta x}{2}\left(\tfrac{\partial \overline{\boldsymbol{q}}_{j-1}}{\partial x}\right)^n,\\
\left(\overline{\boldsymbol{q}}^R_{j-\frac{1}{2}}\right)^{n+\frac{1}{2}} &= \overline{\boldsymbol{q}}^n_j - \tfrac{\Delta x}{2}\left(\tfrac{\partial \overline{\boldsymbol{q}}_j}{\partial x}\right)^n.
\end{aligned} \tag{C.29}$$

For the time integration we now use a two-stage Runge-Kutta (RK) scheme, which looks as follows:

$$\begin{aligned}
\overline{\boldsymbol{q}}^{(0)}_j &= \overline{\boldsymbol{q}}^n_j,\\
\overline{\boldsymbol{q}}^{(1)}_j &= \overline{\boldsymbol{q}}^{(0)}_j - \tfrac{1}{2}\tfrac{\Delta t}{\Delta x}\left[\tilde{\boldsymbol{f}}\left(\boldsymbol{q}^L_{j+\frac{1}{2}}, \boldsymbol{q}^R_{j+\frac{1}{2}}\right)^{(0)} - \tilde{\boldsymbol{f}}\left(\boldsymbol{q}^L_{j-\frac{1}{2}}, \boldsymbol{q}^R_{j-\frac{1}{2}}\right)^{(0)}\right],\\
\overline{\boldsymbol{q}}^{(2)}_j &= \overline{\boldsymbol{q}}^{(0)}_j - \tfrac{\Delta t}{\Delta x}\left[\tilde{\boldsymbol{f}}\left(\boldsymbol{q}^L_{j+\frac{1}{2}}, \boldsymbol{q}^R_{j+\frac{1}{2}}\right)^{(1)} - \tilde{\boldsymbol{f}}\left(\boldsymbol{q}^L_{j-\frac{1}{2}}, \boldsymbol{q}^R_{j-\frac{1}{2}}\right)^{(1)}\right],\\
\overline{\boldsymbol{q}}^{n+1}_j &= \overline{\boldsymbol{q}}^{(2)}_j.
\end{aligned} \tag{C.30}$$

Note that this is not the only second-order accurate RK scheme. Any other second-order RK scheme would do too.

C.3.2 Accuracy analysis

We shall now perform a time accuracy analysis of the RK scheme. Therefore we start with a general analysis of the expression for $\overline{\boldsymbol{q}}_j^{n+1}$ where we have not taken into account the RK scheme yet. A Taylor series expansion around the point n allows us to write the following expressions:

$$\begin{aligned}
\overline{\boldsymbol{q}}_j^{n+1} &= \overline{\boldsymbol{q}}_j^n + \Delta t \left(\frac{\partial \overline{\boldsymbol{q}}_j}{\partial t}\right)^n + \frac{\Delta t^2}{2}\left(\frac{\partial^2 \overline{\boldsymbol{q}}_j}{\partial t^2}\right)^n + \mathcal{O}\left(\Delta t^3\right), && \text{(C.31)}\\
\left(\frac{\partial \overline{\boldsymbol{q}}_j}{\partial t}\right)^n &= -\frac{1}{\Delta x}\left[\tilde{\boldsymbol{f}}\left(\boldsymbol{q}^L_{j+\frac{1}{2}}, \boldsymbol{q}^R_{j+\frac{1}{2}}\right)^n - \tilde{\boldsymbol{f}}\left(\boldsymbol{q}^L_{j-\frac{1}{2}}, \boldsymbol{q}^R_{j-\frac{1}{2}}\right)^n\right] = \boldsymbol{F}\left(\overline{\boldsymbol{q}}_j^n\right), && \text{(C.32)}\\
\left(\frac{\partial^2 \overline{\boldsymbol{q}}_j}{\partial t^2}\right)^n &= \frac{d\boldsymbol{F}\left(\overline{\boldsymbol{q}}_j^n\right)}{d\overline{\boldsymbol{q}}_j}\left(\frac{\partial \overline{\boldsymbol{q}}_j}{\partial t}\right)^n = \frac{d\boldsymbol{F}\left(\overline{\boldsymbol{q}}_j^n\right)}{d\overline{\boldsymbol{q}}_j}\boldsymbol{F}\left(\overline{\boldsymbol{q}}_j^n\right). && \text{(C.33)}
\end{aligned}$$

Note that we introduced $\boldsymbol{F}\left(\overline{\boldsymbol{q}}_j^n\right)$ in expression (C.32). Although this notation suggests that $\boldsymbol{F}$ is only a function of $\overline{\boldsymbol{q}}_j^n$ it is clearly not. The use of a Riemann solver introduces also the information from the neighboring cells such that a more mathematically correct notation would be $\boldsymbol{F}\left(\ldots, \overline{\boldsymbol{q}}_j^n, \ldots\right)$. For convenience we shall use the somewhat shorter notation as defined in (C.32). Using these we can write for the solution at the new time level:

$$\overline{\boldsymbol{q}}_j^{n+1} = \overline{\boldsymbol{q}}_j^n + \Delta t \boldsymbol{F}\left(\overline{\boldsymbol{q}}_j^n\right) + \frac{\Delta t^2}{2}\frac{d\boldsymbol{F}\left(\overline{\boldsymbol{q}}_j^n\right)}{d\overline{\boldsymbol{q}}_j}\boldsymbol{F}\left(\overline{\boldsymbol{q}}_j^n\right) + \mathcal{O}\left(\Delta t^3\right). \tag{C.34}$$

This result indicates that for a scheme to be second-order accurate it has to give an identical expression. We will show that this is the case for the two-stage RK scheme. Using the second stage we can write for the solution at the new time level:

$$\overline{\boldsymbol{q}}_j^{n+1} = \overline{\boldsymbol{q}}_j^n + \Delta t \boldsymbol{F}\left(\overline{\boldsymbol{q}}_j^{(1)}\right). \tag{C.35}$$

The expression $\boldsymbol{F}\left(\overline{\boldsymbol{q}}_j^{(1)}\right)$ we find by Tayloring around the point n:

$$\boldsymbol{F}\left(\overline{\boldsymbol{q}}_j^{(1)}\right) = \boldsymbol{F}\left(\overline{\boldsymbol{q}}_j^n\right) + \left(\overline{\boldsymbol{q}}_j^{(1)} - \overline{\boldsymbol{q}}_j^n\right)\frac{d\boldsymbol{F}\left(\overline{\boldsymbol{q}}_j^n\right)}{d\overline{\boldsymbol{q}}_j} + \mathcal{O}\left(\left(\overline{\boldsymbol{q}}_j^{(1)} - \overline{\boldsymbol{q}}_j^n\right)^2\right). \tag{C.36}$$

From the first RK stage we know that:

$$\overline{\boldsymbol{q}}_j^{(1)} - \overline{\boldsymbol{q}}_j^n = \frac{1}{2}\Delta t \boldsymbol{F}\left(\overline{\boldsymbol{q}}_j^n\right). \tag{C.37}$$

Substitution of (C.36) and (C.37) into (C.35) finally gives us the modified equation:

$$\overline{\boldsymbol{q}}_j^{n+1} = \overline{\boldsymbol{q}}_j^n + \Delta t \boldsymbol{F}\left(\overline{\boldsymbol{q}}_j^n\right) + \frac{\Delta t^2}{2}\frac{d\boldsymbol{F}\left(\overline{\boldsymbol{q}}_j^n\right)}{d\overline{\boldsymbol{q}}_j}\boldsymbol{F}\left(\overline{\boldsymbol{q}}_j^n\right) + \mathcal{O}\left(\Delta t^3\right). \tag{C.38}$$

It is clear that this modified equation is identical to (C.34). The two stage RK scheme that we are using is thus second-order accurate although we make use of the standard Riemann approach. The multi-stage nature of the RK method makes the discretization sensitive to the highly structured solution in a cell. A GR correction is thus not necessary for obtaining a sufficiently high order of accuracy.