Centrum voor Wiskunde en Informatica

REPORT*RAPPORT*

MAS

Modelling, Analysis and Simulation

Modelling, Analysis and Simulation

Numerical solver for compressible two-fluid flow

J. Naber

REPORT MAS-E0505 FEBRUARY 2005

CWI is the National Research Institute for Mathematics and Computer Science. It is sponsored by the Netherlands Organization for Scientific Research (NWO).
CWI is a founding member of ERCIM, the European Research Consortium for Informatics and Mathematics.

CWI's research has a theme-oriented structure and is grouped into four clusters. Listed below are the names of the clusters and in parentheses their acronyms.

Probability, Networks and Algorithms (PNA)

Software Engineering (SEN)

Modelling, Analysis and Simulation (MAS)

Information Systems (INS)

P.O. Box 94079, 1090 GB Amsterdam (NL)
Kruislaan 413, 1098 SJ Amsterdam (NL)
Telephone +31 20 592 9333
Telefax +31 20 592 4199

ISSN 1386-3703

Numerical solver for compressible two-fluid flow

ABSTRACT
This report treats the development of a numerical solver for the simulation of flows of two non-mixing fluids described by the two-dimensional Euler equations. A level-set equation in conservative form describes the interface. After each time step the deformed level-set function is transformed back to a real signed distance function using a PDE-based redistancing procedure. Interface smoothing is applied to prevent staircasing and possible unphysical oscillations. A finite-volume approximation is used for the numerical solver. The flow model is discretized using a three-stage time marching scheme together with the approximate Riemann solver of Roe. Quadratic sub-cell interpolation is obtained using the limiter by Koren. The combination of time and space discretization makes the method effectively second-order accurate although third-order accuracy can theoretically be reached. The redistancing equation is discretized using a two-stage time-marching scheme and a second-order accurate spatial interpolation. The spurious pressure oscillations due to the numerical inconsistency of conservative level-set methods are removed using a simple 'ghost-fluid like' fix. Several numerical tests are performed to test the solver for its performance. Standard one-dimensional shock tube problems prove the existence of the pressure oscillations and verify the simple fix. A convergence test verifies the numerical order of the scheme. Two-dimensional tests are performed using the shock-bubble interaction problem, the Kelvin-Helmholtz instability and the supersonic free jet.

2000 Mathematics Subject Classification: 65M60,76N15,76T10
Keywords and Phrases: gas dynamics, interface capturing, compressible two-gas flows, level-set method, interface pressure-error, simple ghost-fluid method
Note: This research was performed at the University of Michigan in Ann Arbor. The work was carried out under CWI-project MAS2.1 "Computational Fluid Dynamics and Computational Electromagnetics".

Numerical Solver for Compressible Two-Fluid Flow

Jorick Naber
CWI
P.O. Box 94079, 1090 GB Amsterdam, The Netherlands

December 2004

ABSTRACT

This report treats the development of a numerical solver for the simulation of flows of two non-mixing fluids described by the two-dimensional Euler equations. A level-set equation in conservative form describes the interface. After each time step the deformed level-set function is transformed back to a real signed distance function using a PDE-based redistancing procedure. Interface smoothing is applied to prevent staircasing and possible unphysical oscillations. A finite-volume approximation is used for the numerical solver. The flow model is discretized using a three-stage time marching scheme together with the approximate Riemann solver of Roe. Quadratic sub-cell interpolation is obtained using the limiter by Koren. The combination of time and space discretization makes the method effectively second-order accurate although third-order accuracy can theoretically be reached. The redistancing equation is discretized using a two-stage time-marching scheme and a second-order accurate spatial interpolation. The spurious pressure oscillations due to the numerical inconsistency of conservative level-set methods are removed using a simple 'ghost-fluid like' fix. Several numerical tests are performed to test the solver for its performance. Standard one-dimensional shock tube problems prove the existence of the pressure oscillations and verify the simple fix. A convergence test verifies the numerical order of the scheme. Two-dimensional tests are performed using the shock-bubble interaction problem, the Kelvin-Helmholtz instability and the supersonic free jet.

Acknowledgments

This is the technical report treating the work done during my internship at the W.M. Keck Laboratory for Computational Fluid Dynamics of the University of Michigan in Ann Arbor, United States of America. The focus here lies on the details regarding the development of a numerical solver for the simulation of *two-fluid flows*. Together with the report treating the preparation for this internship [25] this report is an excellent starting point for students interested in finite-volume methods for the Euler equations and the level-set method.

For the less technical and more personal experiences I would like to refer to my website which I kept up-to-date during my stay in the USA: 'http://juruk.waarbenjij.nu' (in Dutch only).

I would like to thank Professor van Leer for the opportunities he gave me and the support he provided during this internship. It was a great pleasure working with a person with such a profound knowledge in the area of Computational Fluid Dynamics (CFD) and maybe even more important with someone that has such joy in sharing it with his students. I sincerely believe that these four months have contributed to a much better understanding of not only the technical issues involved with CFD, but also of the issues involved with doing research in general.

Further I owe much gratitude to Professor Koren for his involvement in this internship and the month of research prior to the internship. Without his help this internship would not have been possible. I am looking forward to continuing the present research under his supervision at the Center for Mathematics and Computer Science (CWI) in Amsterdam.

Last but not least I would like to thank all the people I met during these four months in Ann Arbor. From new friends to new professional contacts, all of them contributed in some way to this great period. Thanks a lot for all the fun I had in these four months. Hopefully we meet up someday in the future!

Financial support for this internship came from the 'STIR fonds' of the Delft University of Technology (TU Delft), the 'Universiteitsfonds' of the TU Delft, the 'Professor van der Maas Fonds' of the Faculty of Aerospace Engineering of the TU Delft, and from the Department of Aerospace Engineering of the University of Michigan. My sincere regards to those who made this funding possible.

Ann Arbor, December 2004,

Jorick Naber

Table of 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. A thorough understanding of these flows is therefore of utmost importance. This is the reason that the simulation of two-fluid flows using numerical methods has been the subject of elaborate research in recent years. Though, in contrast to the simulation of single-fluid flows, there is reason to believe that there is still no sufficiently accurate and efficient simulation method available for two-fluid flows. This is mainly due to difficulties arising when treating the interface between the two fluids.

The different numerical methods employed so far can generally be divided into *Lagrangian* and *Eulerian* methods (for a complete discussion of both types see [4]). The former type, in the case of two-fluid flows also referred to as *interface-fitting*, makes use of the flow equations reduced to equations for solely the interface (explicit formulation of the interface). This approach allows for a clear representation of the interface and its surroundings, thus preventing smearing of the interface. However, Lagrangian methods are less useful for large deformations which are characteristic for almost all flow problems.

In Eulerian methods, also known as *interface-capturing* methods, the full flow equations are solved in the domain made up by both fluids. Often an extra transport equation for a parameter describing the properties of the fluid mixture (i.e. the mass fraction or the ratio of specific heats) is added to the system of flow equations. This approach requires no further treatment of the interface since it implicitly follows from the flow solution. Although these *conservative* methods allow for large deformations, and are therefore especially useful for the treatment of fluid problems, it has been shown in [1] that they suffer from spurious pressure oscillations due to the inconsistency of the discretized equations near the interface. Several fixes for this problem have been proposed, see for example [2, 8, 14, 15, 18, 30], 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 (see [1] for a complete overview of the methods mentioned). Another approach is to use a model based on the separate flow equations for both fluids, as was done in [11] and more recently in [36, 37]. Although these models give good oscillation-free results, the governing flow equations are rather complicated, thus requiring complex numerical techniques to be solved.

Another frequently used type of method is known as *interface-tracking*. These methods use the full conservative flow equations, as do the Eulerian methods, but an extra equation is added to describe the evolution of the interface (directly or indirectly), which is typical for a Lagrangian approach. Earlier versions of these tracking methods are known as the *marker and cell* (MAC) and *volume of fluid* (VOF) methods, see also [13]. A more recent development is known as the *level-set* method, which was first presented in [22] and extended to the Euler equations in [24]. Unfortunately these methods suffer from the same spurious pressure oscillations near the interface as the conservative capturing methods described above. Of all these methods the level-set method shows most competence. Its implementation in existing solvers is rather simple and straightforward and makes it therefore a good starting point for the search for a new and efficient method for solving two-fluid problems.

This report focusses on the application of the level-set approach for solving the two-dimensional Euler equations for compressible, inviscid, unsteady two-fluid flow. An extra transport equation for the level-set function is added to the flow equations to keep track of the interface. The set of conservative equations is discretized using a finite-volume method. Time discretization is done using a three-stage method. Roe's approximate Riemann solver is used for the discretization of the interface fluxes.

Several test cases are treated to investigate the behavior of the solver. Due to the inconsistency in the conservative formulation of the level-set model the known pressure oscillations occur. As a fix for these oscillations the simple algorithm proposed in [1] is applied. Tests with this simple method prove its capabilities.

In chapter 2 the flow model is discussed. The Euler equations are derived, the level-set method is discussed and the characteristics analysis is performed. Chapter 3 focusses on the discretization of flow equations and the implementation of this finite-volume discretization into a numerical solver. Chapter 4 treats the spurious pressure oscillations and discusses a simple fix for them. Finally chapter 5 discusses several test results of the numerical solver with and without fix.

Chapter 2
Flow model

In this chapter the two-dimensional Euler equations describing compressible, inviscid, unsteady flow are discussed. The simulation of two-fluid flows is done using a level-set method. The interface evolution is described by an additional level-set equation, which is added to the system of flow equations. For a correct implementation of this level-set method several properties of this method have to be treated. Using the complete flow model a characteristics analysis is performed, resulting in the eigenvalues and eigenvectors necessary for implementation in the approximate Riemann solver of Roe (see next chapter).

2.1. Euler equations

The Euler equations for compressible, inviscid, unsteady flow can be derived from the Navier-Stokes equations when neglecting friction and heat conduction (this shall not be shown here). Because we desire to simulate flow problems with discontinuities such as shock waves, the Euler equations in strictly conservative form are used. When further gravity is neglected, the equations in differential form can be written as:

$$\frac{\partial \boldsymbol{q}}{\partial t} + \frac{\partial \boldsymbol{f}(\boldsymbol{q})}{\partial x} + \frac{\partial \boldsymbol{g}(\boldsymbol{q})}{\partial y} = \mathbf{0}. \tag{2.1}$$

Here $\boldsymbol{q}$ is the conservative state vector and $\boldsymbol{f}$ and $\boldsymbol{g}$ the flux vectors in the x and y directions, respectively. For the two-dimensional version of the Euler equations (for a discussion of a numerical solver for the one-dimensional equations see [25]) these vectors each have four components, representing the equations for conservation of mass, momentum in x and y directions and energy, respectively:

$$\boldsymbol{q} = \begin{pmatrix} \rho \\ \rho u \\ \rho v \\ \rho E \end{pmatrix} \quad , \quad \boldsymbol{f} = \begin{pmatrix} \rho u \\ p + \rho u^2 \\ \rho uv \\ \rho uH \end{pmatrix} \quad \text{and} \quad \boldsymbol{g} = \begin{pmatrix} \rho v \\ \rho uv \\ p + \rho v^2 \\ \rho vH \end{pmatrix}. \tag{2.2}$$

Their components include the four primary state variables being, ρ the density, u the velocity in x direction, v the velocity in y direction 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:

$$p = \rho e \left(\gamma - 1\right), \tag{2.3}$$

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

$$E = e + \frac{1}{2}\left(u^2 + v^2\right), \tag{2.4}$$

where e follows from (2.3). The total enthalpy H is defined as:

$$H = E + \frac{p}{\rho}. \tag{2.5}$$

A major advantage of the level-set method is that no further model is required to describe the different fluids. The standard Euler equations can be used in every point of the domain that is treated. The ratio of specific heats

is the only variable that explicitly depends on the fluid one is treating. Solving for the primary thermodynamic variables thus means using the correct value of γ instead of solving a complete flow model for both gasses.

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 primary variables. For convenience's sake therefore the so-called primary state vector $\boldsymbol{w} = (\rho, u, v, p)^T$ is introduced.

2.2. Level-set method

The level-set method is an implicit method for describing the evolution of the interface. 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 an interface [1]. This function is chosen 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 (see figure 2.1). For a more detailed discussion of the level-set function see [31] and [9].

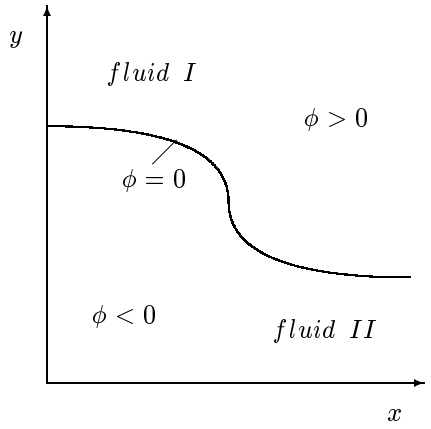

Figure 2.1: Two fluids separated by an interface. The level-set function ϕ is a signed distance function, being zero at the interface, positive in one fluid and negative in the other.

2.2.1 Level-set equation

The level-set 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. The well-known advection or transport equation can therefore be used to describe its motion, and thus the evolution of the interface, in time. For the two-dimensional case this can be written as:

$$\frac{\partial \phi}{\partial t} + \boldsymbol{V} \cdot \nabla \phi = \frac{\partial \phi}{\partial t} + u\frac{\partial \phi}{\partial x} + v\frac{\partial \phi}{\partial y} = 0. \tag{2.6}$$

To ease the numerical treatment of this equation it is often written as a conservation equation. In this way it can be added to the system of conservation laws (the Euler equations) and can thus be solved with the same numerical techniques [2]. Multiplying equation (2.6) with the density ρ and adding it to the equation for

[1] Besides a linear representation for ϕ there are other possibilities. The advantage of ϕ being a linear function of x is that it is independent of second order derivatives that occur in most numerical discretizations (numerical diffusion).

[2] Although this is the most common approach it is not the only option. Another frequently used method is to use a separate discretization method for the level-set advection equation. This requires approximately the same amount of computational power as using a conservative equation and solving it with the Euler equations. Recent results have shown that the 'conservative' approach used here leads to numerical

conservation of mass (2.1) multiplied with the level-set function ϕ gives after some rewriting:

$$\frac{\partial \rho\phi}{\partial t} + \nabla \cdot \rho \boldsymbol{V}\phi = \frac{\partial \rho\phi}{\partial t} + \frac{\partial \rho u\phi}{\partial x} + \frac{\partial \rho v\phi}{\partial y} = 0, \tag{2.7}$$

which can be interpreted as an equation for the conservation of the distance function times the mass. Adding this conservative form of the level-set equation to the system of conservation laws given by equation (2.1) and the vectors (2.2) leads to the following new versions of the state and flux vectors:

$$\boldsymbol{q} = \begin{pmatrix} \rho \\ \rho u \\ \rho v \\ \rho E \\ \rho\phi \end{pmatrix} \quad , \quad \boldsymbol{f} = \begin{pmatrix} \rho u \\ p + \rho u^2 \\ \rho uv \\ \rho uH \\ \rho u\phi \end{pmatrix} \quad \text{and} \quad \boldsymbol{g} = \begin{pmatrix} \rho v \\ \rho uv \\ p + \rho v^2 \\ \rho vH \\ \rho v\phi \end{pmatrix}. \tag{2.8}$$

2.2.2 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-homogeneous 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 ($\phi = 0$). Due to this non-homogeneous velocity field the value $\phi = 0$ moves faster than the other values for ϕ. This means that the distance between the interface and the point with $\phi = 1$ 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 $\phi = 0$, 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 level-set function will, besides maybe at time zero, never be exactly equal to zero in a cell. To be able to give an accurate representation of the interface it is therefore necessary that near the interface the level-set function is the real distance function and not the distorted version. This requires for a *redistancing* procedure of the level-set 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 in [33]. The idea is to solve the following differential equation [3] until its steady state is reached:

$$\frac{\partial \phi}{\partial \tau} = S\left(\phi_{\tau=0}\right)\left(1 - |\nabla\phi|\right), \tag{2.9}$$

where τ is an artificial time scale only used within the redistancing procedure and $S\left(\phi_{\tau=0}\right)$ the sign function of ϕ_0. A steady state, i.e. $\frac{\partial \phi}{\partial \tau} = 0$, 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 following chapter. [4]

2.2.3 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 we treat here only ideal gasses,

errors near contact discontinuities. Using a separate 'non-conservative' approach prevents this. A more elaborate discussion will follow in the Master's thesis of the author.

[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.

[4] A present subject of investigation is whether this rather elaborate PDE-approach is necessary. The basic idea of this research is to look for a new level-set equation with an additional source term which takes care of the redistancing. If such an equation can be found is not clear yet. There is reason to believe though that such an approach can be less computationally intensive since it does not need a separate update procedure.

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:

$$\gamma = \begin{cases} \gamma_1 & \text{if} \quad \phi > 0, \\ ? & \text{if} \quad \phi = 0, \\ \gamma_2 & \text{if} \quad \phi < 0. \end{cases} \tag{2.10}$$

Unfortunately this rather simplistic approach leads to some difficulties. It is unknown what to take for γ at the interface. 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. Using the model defined above yields the *'staircasing'* of the interface such that it coincides with the cell boundaries. This can lead to spurious waves in the solution (see appendix 1 for a discussion of these oscillations due to staircasing).

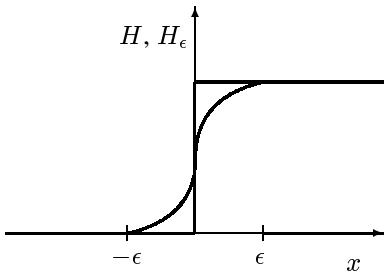

Figure 2.2: The Heaviside function $H(\phi)$ is smoothed to the transformed $H_\epsilon(\phi)$ to avoid spurious oscillations due to staircasing of the interface.

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 $H(\phi)$ for the determination of the local value of γ, a smooth function is used (see figure 2.2). This smooth step function $H_\epsilon \in [0,1]$ is defined as:

$$H_\epsilon(\phi) = \begin{cases} 0 & \text{if} \quad \phi < -\epsilon, \\ \frac{1}{2}\left[1 + \frac{\phi}{\epsilon} + \frac{1}{\pi}\sin\left(\frac{\pi\phi}{\epsilon}\right)\right] & \text{if} \quad -\epsilon \leq \phi \leq \epsilon, \\ 1 & \text{if} \quad \phi > \epsilon, \end{cases} \tag{2.11}$$

where ϵ is a small distance often taken equal to one and a half cell size [5] ($\epsilon = \frac{3\Delta x}{2}$). This allows for the following equation for the determination of γ:

$$\gamma = H_\epsilon \gamma_1 + (1 - H_\epsilon)\gamma_2. \tag{2.12}$$

Because the interface is no longer sharp but smeared out to some extent, the sign function $S(\phi_0)$ used in the redistancing equation (2.9) is no longer correct. Therefore a new sign function S_ϵ based on H_ϵ is defined:

$$S_\epsilon(\phi) = 2H_\epsilon(\phi) - 1. \tag{2.13}$$

2.3. Integral formulation

When performing the characteristics analysis for the Euler equations in two dimensions it is useful to start from the integral formulation of the set of flow equations. Another well-known reason for writing the Euler

[5] This specific value of ϵ is used since it is the smallest distance possible such that smoothing is always symmetrically applied. 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.

equations in integral form is that the differential form does not allow for discontinuities in the solution. Using the integral, or weak, formulation the *Rankine-Hugoniot equations* can be derived which describe the jump over such a discontinuity. It is actually the differential formulation that follows from the integral formulation of the Euler equations by neglecting these discontinuous solutions. To get back to the integral formulation we consider a fluid element Ω in two dimensions as indicated in figure 2.3. Integration of the differential equations (2.1) over the volume of this fluid element results in:

$$\int_\Omega \frac{\partial \boldsymbol{q}}{\partial t} dV + \int_\Omega \left(\frac{\partial \boldsymbol{f}}{\partial x} + \frac{\partial \boldsymbol{g}}{\partial y} \right) dV = \boldsymbol{0}. \tag{2.14}$$

Combining both $\boldsymbol{f}$ and $\boldsymbol{g}$ into one vector $\boldsymbol{F} = (\boldsymbol{f}, \boldsymbol{g})^T$ allows for the following expression:

$$\int_\Omega \frac{\partial \boldsymbol{q}}{\partial t} dV + \int_\Omega \nabla \cdot \boldsymbol{F} dV = \boldsymbol{0}. \tag{2.15}$$

Using *Gauss' divergence theorem* the second integral can be written as an integral over the surface Γ of the fluid element instead of an integral over its volume Ω:

$$\int_\Omega \frac{\partial \boldsymbol{q}}{\partial t} dV + \oint_\Gamma \boldsymbol{F} \cdot \boldsymbol{n} dA = \boldsymbol{0}, \tag{2.16}$$

where $\boldsymbol{n}$ is the unit vector normal to the surface of the fluid element. Writing this normal vector in components in x and y directions respectively, $\boldsymbol{n} = (n_x, n_y)^T$, allows for the second integral to finally be written as:

$$\int_\Omega \frac{\partial \boldsymbol{q}}{\partial t} dV + \oint_\Gamma \left(\boldsymbol{f} n_x + \boldsymbol{g} n_y \right) dA = \int_\Omega \frac{\partial \boldsymbol{q}}{\partial t} dV + \oint_\Gamma \boldsymbol{\Psi} dA = \boldsymbol{0}, \tag{2.17}$$

where we used $\boldsymbol{\Psi} = \boldsymbol{f} n_x + \boldsymbol{g} n_y$ for the normal flux through the surface of Ω. Using the definition of the normal velocity $v_n = \boldsymbol{V} \cdot \boldsymbol{n} = u n_x + v n_y$ this vector can be written as:

$$\boldsymbol{\Psi} = \begin{pmatrix} \rho v_n \\ \rho v_n u + p n_x \\ \rho v_n v + p n_y \\ \rho v_n H \\ \rho v_n \phi \end{pmatrix}. \tag{2.18}$$

Note that the formulation of the Euler equations derived at (2.17) expresses the time derivatives of the state variables as a function of the fluxes through the surface of the fluid element instead of the directional derivatives of the state variables themselves. The expression found above is the starting point for the finite-volume approximation to be derived in the following chapter.

2.4. Characteristics analysis

The application of Roe's approximate Riemann solver requires knowledge about the eigensystem, the eigenvalues and eigenvectors, of the flow equations. Therefore a characteristics analysis is performed for the flow model described by (2.1) and (2.8). The approach followed is the one described in [29].

2.4.1 Transformation matrix

The eigenproblem to be solved requires the *transformation matrix* or *Jacobian* of the conservative flow equations (2.1) and (2.8). This transformation matrix is defined as:

$$\boldsymbol{J} = \frac{\partial \boldsymbol{\Psi}}{\partial \boldsymbol{q}} = \begin{bmatrix} \frac{\partial \Psi_1}{\partial q_1} & \frac{\partial \Psi_1}{\partial q_2} & \cdots \\ \frac{\partial \Psi_2}{\partial q_1} & \frac{\partial \Psi_2}{\partial q_2} & \cdots \\ \vdots & \vdots & \ddots \end{bmatrix}. \tag{2.19}$$

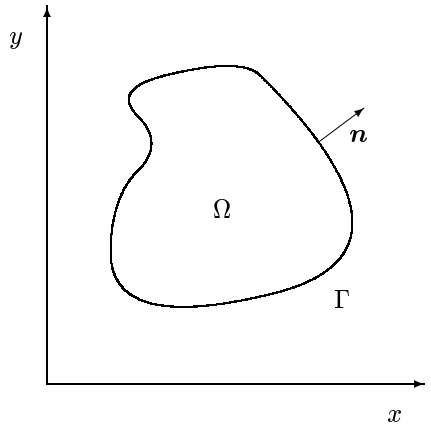

Figure 2.3: A fluid element Ω with surface Γ. The outward unit normal is indicated by $\boldsymbol{n}$.

Before this matrix can be calculated the flux vector $\boldsymbol{\Psi}$ has to be written in terms of the conservative variables $\boldsymbol{q} = (\rho, \rho u, \rho v, \rho E, \rho\phi)^T$ instead of the primary variables $\boldsymbol{w} = (\rho, u, v, p, \phi)^T$ as was done in (2.18):

$$\boldsymbol{\Psi} = \begin{pmatrix} q_2 n_x + q_3 n_y \\ \frac{q_2^2}{q_1} n_x + \frac{q_2 q_3}{q_1} n_y + n_x (\gamma - 1) \left(q_4 - \frac{q_2^2 + q_3^2}{2 q_1} \right) \\ \frac{q_2 q_3}{q_1} n_x + \frac{q_3^2}{q_1} n_y + n_y (\gamma - 1) \left(q_4 - \frac{q_2^2 + q_3^2}{2 q_1} \right) \\ \frac{q_2}{q_1} n_x \left(\gamma q_4 - (\gamma - 1) \frac{q_2^2 + q_3^2}{2 q_1} \right) + \frac{q_3}{q_1} n_y \left(\gamma q_4 - (\gamma - 1) \frac{q_2^2 + q_3^2}{2 q_1} \right) \\ \frac{q_2 q_5}{q_1} n_x + \frac{q_3 q_5}{q_1} n_y \end{pmatrix}. \tag{2.20}$$

Where the ratio of specific heats is a function of the level-set function: $\gamma = \gamma(\frac{\rho\phi}{\rho})$. Using this expression the calculation of the transformation matrix $\boldsymbol{J}$ is rather straightforward and shall therefore not be elaborated here (see [24] for a more detailed discussion). One finally ends up with the following result:

$$\boldsymbol{J} = \begin{bmatrix} 0 & n_x & n_y & 0 & 0 \\ [(\gamma - 1) e_k - \phi X] n_x - u v_n & v_n - (\gamma - 2) u n_x & u n_y - (\gamma - 1) v n_x & (\gamma - 1) n_x & n_x X \\ [(\gamma - 1) e_k - \phi X] n_y - v v_n & v n_x - (\gamma - 1) u n_y & v_n - (\gamma - 2) v n_y & (\gamma - 1) n_y & n_y X \\ [(\gamma - 1) e_k - H - \phi X] v_n & H n_x - (\gamma - 1) u v_n & H n_y - (\gamma - 1) v v_n & \gamma v_n & v_n X \\ -\phi v_n & \phi n_x & \phi n_y & 0 & v_n \end{bmatrix}, \tag{2.21}$$

where $e_k = \frac{1}{2}\left(u^2 + v^2\right)$ is the kinetic energy of the flow and $X = \frac{\gamma}{\gamma - 1}\frac{p}{\rho}$.

2.4.2 Eigensystem

The eigenvalues of the transformation matrix represent the wave speeds of the different types of information containing waves in the flow. The corresponding eigenvectors give the directions in which these different waves travel. The eigenvalues can be obtained from $\boldsymbol{J}$ by determining the roots λ_i $(i = 1, \ldots, 5)$ of the characteristic equation:

$$\det\left(\boldsymbol{J} - \lambda \boldsymbol{I}\right) = 0, \tag{2.22}$$

where $\boldsymbol{I}$ is the identity matrix. Evaluating the characteristic equation leads to the following eigenvalues:

$$\begin{aligned} \lambda_1 &= v_n - a, \\ \lambda_{2,3,4} &= v_n, \\ \lambda_5 &= v_n + a, \end{aligned} \tag{2.23}$$

where a is the speed of sound. From this it follows that waves travel either with the speed of sound relative to the flow velocity, or with the flow velocity itself. This is a well-known property of the Euler equations. Given these eigenvalues λ_i it is possible to determine the eigenvectors $\boldsymbol{r}_i$ of $\boldsymbol{J}$ using the following expression:

$$\boldsymbol{J}\boldsymbol{r}_i = \lambda_i \boldsymbol{r}_i. \tag{2.24}$$

Note that this equation holds for the *right eigenvectors*, while a similar equation for the *left eigenvectors* could also be used. Using straightforward matrix manipulations, which shall not be shown here, one ends up with the following matrix of right eigenvectors:

$$\boldsymbol{R} = \begin{bmatrix} \boldsymbol{r}_1 & \boldsymbol{r}_2 & \boldsymbol{r}_3 & \boldsymbol{r}_4 & \boldsymbol{r}_5 \end{bmatrix} = \begin{bmatrix} 1 & 1 & 0 & 0 & 1 \\ u - an_x & u & n_y & 0 & u + an_x \\ v - an_y & v & -n_x & 0 & v + an_y \\ H - av_n & \frac{1}{2}\left(u^2 + v^2\right) & v_t & -\frac{X}{\gamma - 1} & H + av_n \\ \phi & \phi & 0 & 1 & \phi \end{bmatrix}, \tag{2.25}$$

where v_t is the velocity in the direction of the vector tangential to the surface of Ω given by $v_t = \boldsymbol{V} \cdot \boldsymbol{t} = ut_x + vt_y$. Because $\boldsymbol{t}$ is perpendicular to $\boldsymbol{n}$ we can also write the velocity in terms of the normal vector components: $v_t = -un_y + vn_x$. The eigenvectors given in (2.25) are not distinct. Any other vector in the subspace formed by the eigenvectors $\boldsymbol{r}_{2,3,4}$, belonging to the same eigenvalue v_n could also be used.

Chapter 3
Flow solver

The flow model derived in the previous chapter will be implemented in a numerical flow solver. A finite-volume approximation is derived from the integral formulation of the flow equations. A three-stage time-marching method is used for the discretization. Roe's approximate Riemann solver is used for the calculation of the interface fluxes. Quadratic sub-cell interpolation is used for obtaining the cell-face values of the state variables. Further the discretization of the PDE used for the redistancing of the level-set function is treated.

3.1. FINITE-VOLUME APPROXIMATION

The derivation of the finite-volume approximation starts with the flow equations in integral form given by (2.17) and (2.18). The flow domain $\Omega \in \mathbb{R}^2$ is discretized with a finite number of quadrilateral grid cells $\Omega_{i,j}$, where the indices $i, j = 1, \ldots, N_{i,j}$ indicate the cell centers. The boundaries between the grid cells are indicated by Γ_l with $l = 1, \ldots, 4$. At the mid-point of each cell face an outward unit normal vector $\boldsymbol{n}_l = (n_x, n_y)_l^T$ is defined. Such a finite volume is drawn in figure 3.1.

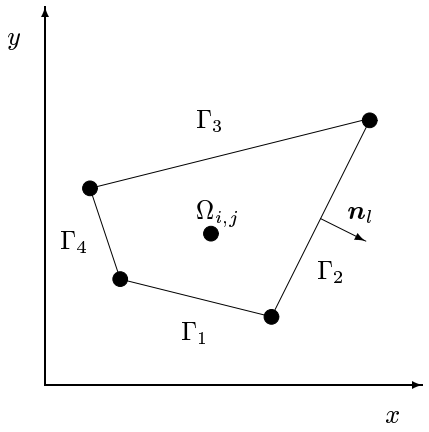

Figure 3.1: A quadrilateral finite volume $\Omega_{i,j}$ with cell interfaces Γ_1, Γ_2, Γ_3 and Γ_4. At each interface mid-point the outward unit normal vector $\boldsymbol{n}_l$ is defined, where l indicates the interface number (1 to 4).

Taking $\Omega_{i,j}$ as the integration domain and $\sum_{l=1}^{4} \Gamma_l$ as the boundary of this domain and applying the integral form of the flow equations (2.17) gives:

$$\int_{\Omega_{i,j}} \frac{\partial \boldsymbol{q}}{\partial t} dV + \oint_{\sum_{l=1}^{4} \Gamma_l} \left(\boldsymbol{f} n_x + \boldsymbol{g} n_y \right)_l dA = \int_{\Omega_{i,j}} \frac{\partial \boldsymbol{q}}{\partial t} dV + \oint_{\sum_{l=1}^{4} \Gamma_l} \boldsymbol{\Psi_l} dA = \boldsymbol{0}. \tag{3.1}$$

Due to the discrete nature of the cell faces it is allowed to write the integral over the boundary of $\Omega_{i,j}$ as a sum over the different cell faces, such that (3.1) becomes:

$$\int_{\Omega_{i,j}} \frac{\partial \boldsymbol{q}}{\partial t} dV = - \left[\sum_{l=1}^{4} \left(\boldsymbol{\Psi} \Delta s \right)_l \right]_{i,j}, \tag{3.2}$$

where Δs_l is the length of cell face l. Because the finite volume does not change its size in time it is allowed to interchange the time differentiation and the integration in the right-hand side of (3.2), such that:

$$\frac{\partial}{\partial t}\int_{\Omega_{i,j}} \boldsymbol{q} dV = -\left[\sum_{l=1}^{4} (\boldsymbol{\Psi}\Delta s)_l\right]_{i,j}. \tag{3.3}$$

The finite-volume approximation now follows when the discrete value of the state vector at the cell center $\boldsymbol{q}_{i,j}$ is taken equal to the spatial average of the exact state vector $\boldsymbol{q}$ over the finite volume $\Omega_{i,j}$:

$$\boldsymbol{q}_{i,j} = \frac{1}{A_{i,j}}\int_{\Omega_{i,j}} \boldsymbol{q} dV, \tag{3.4}$$

where $A_{i,j}$ is the area of the finite volume (in three dimension this would be the volume itself). We thus finally end up with the following finite-volume approximation:

$$\frac{\partial \boldsymbol{q}_{i,j}}{\partial t} = -\frac{1}{A_{i,j}}\left[\sum_{l=1}^{4} (\boldsymbol{\Psi}\Delta s)_l\right]_{i,j}. \tag{3.5}$$

Note that this approximation is still exact when (3.4) is correct, which is the case when the area $A_{i,j}$ vanishes. Equation (3.5) states that the time-rate of change of the state variables $\boldsymbol{q}_{i,j}$ in a cell follows from the in- and out-flowing fluxes over the cell faces.

Implementing this finite-volume approximation in a numerical solver requires the discretization of the time derivative and the interface fluxes, which shall be the subject of the following discussion.

3.2. Three-stage scheme

The finite-volume approximation (3.5) is discretized using a three-stage scheme [1]. Using three intermediate stages the solution for the state variables $\boldsymbol{q}_{i,j}$ *'marches'* from a certain discrete time level n to the next, $n+1$. This procedure is repeated as many times as necessary to reach the final time level $n = N$. The complete scheme looks as follows:

$$\begin{aligned}
\boldsymbol{q}_{i,j}^{(0)} &= \boldsymbol{q}_{i,j}^{n}, \\
\boldsymbol{q}_{i,j}^{(1)} &= \boldsymbol{q}_{i,j}^{(0)} - \tfrac{\Delta t}{3A_{i,j}}\left[\textstyle\sum_{l=1}^{4}\left(\boldsymbol{\Psi}^{(0)}\Delta s\right)_l\right]_{i,j}, \\
\boldsymbol{q}_{i,j}^{(2)} &= \boldsymbol{q}_{i,j}^{(0)} - \tfrac{\Delta t}{2A_{i,j}}\left[\textstyle\sum_{l=1}^{4}\left(\boldsymbol{\Psi}^{(1)}\Delta s\right)_l\right]_{i,j}, \\
\boldsymbol{q}_{i,j}^{(3)} &= \boldsymbol{q}_{i,j}^{(0)} - \tfrac{\Delta t}{A_{i,j}}\left[\textstyle\sum_{l=1}^{4}\left(\boldsymbol{\Psi}^{(2)}\Delta s\right)_l\right]_{i,j}, \\
\boldsymbol{q}_{i,j}^{n+1} &= \boldsymbol{q}_{i,j}^{(3)}.
\end{aligned} \tag{3.6}$$

For the calculations of the intermediate solutions (1), (2) and (3) the sum of the fluxes through the cell faces are required at each intermediate stage. These numerical fluxes are calculated using the Riemann solver of Roe as shall be shown in the next section. The accuracy of this scheme is treated after the discussion of the sub-cell interpolation.

3.3. Roe's Riemann solver

Application of the numerical scheme for time marching (3.6) requires knowledge of the in- and out-flowing fluxes at the cell faces of each grid-cell. Unfortunately the state variables required for the calculation are only known in the cell centers and not at the interfaces between the cells. Godunov proposed in [10] a method that solved a Riemann problem at each interface to obtain the state variables and thus the fluxes at these interfaces. Solving a Riemann problem means, given a left and right initial state separated by a contact-discontinuity,

[1] Another frequently used numerical scheme is that of Hancock. Having second-order accuracy in time makes this a very suitable method. See [25] for a one-dimensional version of this scheme.

finding the resulting state-variable distribution in time. For a complete discussion of the Riemann problem see [25].

The numerical flux at the interface can thus be written as a function of the left and right initial values at the interface:

$$\mathbf{\Psi}_l^{(m)} = \mathbf{\Psi}\left(\boldsymbol{q}_{l,L}^{(m)}, \boldsymbol{q}_{l,R}^{(m)}\right) \qquad \text{for} \qquad l = 1, 2, \ldots, 4, \tag{3.7}$$

where (m) indicates the different stages in the time marching scheme. The advantage of solving a Riemann problem at each interface is that all physical information about the flow is used.

Since Godunov proposed his *exact Riemann solver* many variations on this method have been proposed. Widely used are the *approximate Riemann solvers*. These methods use also a left and right initial state to calculate the flux at the interface, but instead of solving the complete Riemann problem they approximate the solution. Some of the best-known approximate solvers are the ones of *Roe*, *Osher* and of *Harten*, *Lax* and *Van Leer* (HLL). Here the solver of *Roe* is used because of its accuracy and ease of implementation.

3.3.1 The interface flux

Roe's method [28] is a *flux differencing method* for the calculation of the interface fluxes based on averaged flow characteristics described by the eigensystem (the eigenvalues and eigenvectors) and the jump relations. It calculates the interface fluxes $\mathbf{\Psi}_l$ using the following approximation:

$$\mathbf{\Psi}_l = \frac{1}{2}\left(\mathbf{\Psi}_{l,L} + \mathbf{\Psi}_{l,R}\right) - \frac{1}{2}\sum_{k=1}^{5}\left|\hat{\lambda}_k\right| \Delta v_k \hat{\boldsymbol{r}}_k, \tag{3.8}$$

where $\mathbf{\Psi}_{l,L}$ and $\mathbf{\Psi}_{l,R}$ are the fluxes calculated using the left and right initial conditions respectively. Note that the superscript (m) is suppressed for convenience's sake. 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*. In the single-fluid case, so no equation for the level-set function ϕ and only one value for γ (thus $\hat{\gamma} = \gamma$) , these quantities can be written as:

$$\hat{\rho} = \omega\rho_L, \tag{3.9}$$

$$\hat{u} = \frac{u_L + \omega u_R}{1+\omega}, \tag{3.10}$$

$$\hat{v} = \frac{v_L + \omega v_R}{1+\omega}, \tag{3.11}$$

$$\hat{H} = \frac{H_L + \omega H_R}{1+\omega}, \tag{3.12}$$

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

$$\omega = \sqrt{\frac{\rho_R}{\rho_L}}. \tag{3.13}$$

Using (3.10) and (3.11) 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:

$$\hat{a} = \sqrt{(\hat{\gamma}-1)\left(\hat{H} - \frac{1}{2}\left(\hat{u}^2 + \hat{v}^2\right)\right)}. \tag{3.14}$$

In the case of two different fluids and thus two different values for γ this becomes more difficult, since Roe averaged values for ϕ, γ and X are now required. Mulder, Osher and Sethian have shown in [24] that a

satisfactory accurate choice for the Roe matrix (approximate Jacobian), and thus the Roe averaged eigenvalues and eigenvectors can be found by introducing:

$$\hat{\phi} = \frac{\phi_L + \omega\phi_R}{1+\omega}, \tag{3.15}$$

$$\hat{\gamma} = \frac{\left(\hat{\phi} - \phi_L\right)\gamma_R + \left(\phi_R - \hat{\phi}\right)\gamma_L}{\phi_R - \phi_L}, \tag{3.16}$$

$$\hat{X} = \frac{(p_R - p_L) - (\hat{\gamma}-1)\left[p_R/\left(\gamma_R - 1\right) - p_L/\left(\gamma_L - 1\right)\right]}{\hat{p}\left(\phi_R - \phi_L\right)}, \tag{3.17}$$

where $\hat{p} = \hat{\rho}\hat{a}^2/\hat{\gamma}$. Note that when $\gamma_R = \gamma_L = \gamma$, (3.16) and (3.17) reduce to: $\hat{\gamma} = \gamma$ and $\hat{X} = 0$, such that we indeed obtain the single-fluid situation.

The jump relations Δv_k in (3.8) 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 (2.25) 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 $\Delta\left[\cdots\right] = \left[\cdots\right]_R - \left[\cdots\right]_L$. See [24] 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 $\boldsymbol{\Psi}_{l,L}^{(m)}$ and $\boldsymbol{\Psi}_{l,R}^{(m)}$. 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 (3.8)

3.3.2 Entropy fix

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:

$$\begin{array}{rcll} \frac{\delta\lambda_k}{2} & = & 0 & \text{for} \quad k = 2,3,4, \\ \frac{\delta\lambda_k}{2} & = & \min\left[\hat{a}, \max\left(0, 2\left(\lambda_{kR} - \lambda_{kL}\right)\right)\right] & \text{for} \quad k = 1,5. \end{array} \tag{3.18}$$

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

$$\begin{array}{rcll} \left|\hat{\lambda}_k\right|^* & = & \left|\hat{\lambda}_k\right| & \text{for} \quad \left|\hat{\lambda}_k\right| \geq \frac{\delta\lambda_k}{2}, \\ \left|\hat{\lambda}_k\right|^* & = & \frac{\left(\hat{\lambda_k}\right)^2}{\delta\lambda_k} + \frac{\delta\lambda_k}{4} & \text{for} \quad \left|\hat{\lambda}_k\right| < \frac{\delta\lambda_k}{2}. \end{array} \tag{3.19}$$

3.4. SUB-CELL INTERPOLATION

Solving for the interface flux using a Riemann solver requires a left and right initial state as was depicted in (3.7). A first choice for these interface values would be to take them equal to the values of $\boldsymbol{q}$ in the cell centers. Though in the case of large gradients in the state variables this results in a very poor approximation. A

better choice would be to use an interpolation method. To make full use of the accuracy of the chosen scheme it is preferred to use a quadratic interpolation method. Such an interpolation method also allows for the use of a *limiter* to reduce spurious *'wiggles'* in the solution.

The sub-cell interpolation is performed using the limiter derived by Koren [16], which embeds the so-called κ-scheme with $\kappa = \frac{1}{3}$ in it. Cast in the *Sweby*-type form, the sub-cell interpolation can be written as follows:

$$\boldsymbol{w}^*_{i+\frac{1}{2},L} = \boldsymbol{w}^*_i + \frac{\psi\left(r_i\right)}{2}\left(\boldsymbol{w}^*_i - \boldsymbol{w}^*_{i-1}\right), \tag{3.20}$$

$$\boldsymbol{w}^*_{i-\frac{1}{2},R} = \boldsymbol{w}^*_i - \frac{r_i\psi\left(\frac{1}{r_i}\right)}{2}\left(\boldsymbol{w}^*_i - \boldsymbol{w}^*_{i-1}\right), \tag{3.21}$$

with:

$$r_i = \frac{\boldsymbol{w}^*_{i+1} - \boldsymbol{w}^*_i}{\boldsymbol{w}^*_i - \boldsymbol{w}^*_{i-1}}, \tag{3.22}$$

$$\psi\left(r_i\right) = \frac{2r_i^2 + r_i}{2r_i^2 - r_i + 2}, \tag{3.23}$$

$$r_i\psi\left(\frac{1}{r_i}\right) = \frac{r_i^2 + 2r_i}{2r_i^2 - r_i + 2}. \tag{3.24}$$

Note that here $\boldsymbol{w}^*$ is used, which is defined as the vector containing the state variables per unit mass: $\boldsymbol{w}^* = \left(\rho, \rho u, \rho v, p, \rho\phi\right)^T$. This is done to increase the accuracy of the interpolation. Further note that instead of the subscript l here the indices $i \pm \frac{1}{2}$ are used to indicate the left and right (or bottom and top) cell face. This implies that for solving the four interface fluxes the sub-cell interpolation in i and j directions (for Cartesian grids x and y) should be separated.

It can be shown that the full time-space method, the numerical scheme (3.6) combined with the quadratic sub-cell interpolation and the Riemann solver, is theoretically *third-order accurate* for systems of linear equations. Unfortunately the Euler equations are non-linear which means that obtaining true third-order accuracy is computationally expensive. This is due to the fact that for real third-order accuracy the flux at the interface should be averaged by solving two Riemann problems in the Gaussian integration points on the cell face. The method that we will use assumes the value of the flux to follow from the average of the state quantities over the cell face, which is not the case because the flux depends non-linearly on the state variables. The required average of the flux is not equal to the flux of the average. This assumption decreases the accuracy of the scheme to second-order with increased resolution.

3.5. Grid

Although the problems solved in this report allow for the use of uniform Cartesian grids only, the numerical solver developed here is able to treat non-Cartesian grids because the finite-volume approximation is formulated in the coordinate system aligned with the cell faces (normal and tangential to the interface).

3.5.1 Quadrilateral cells

A general grid consists of $N_i \times N_j$ quadrilateral cells. Each quadrilateral cell or finite volume $\Omega_{i,j}$, and thus the entire grid, is defined using the locations of its four node points (corners) $\boldsymbol{x}_{i,j}^{a,b,c,d}$. Knowledge of the coordinates of these node points allows for the calculation of all required data for the numerical solver.

Cell center The location of the cell center is only required for data processing since the finite-volume approach assumes the state-variables to be continuous over the whole cell (no distinct location in the cell is required in the numerical calculations). The cell center is defined as the point where the two lines connecting the mid points of the cell faces intersect (see figure 3.2 [2]). Analytically the coordinates of the cell center

[2] Although the figure shows the intersecting lines to be perpendicular this is in general not the case.

$\boldsymbol{x}_{i,j} = (x, y)_{i,j}$ follow from taking the average of the locations of the corner points $\boldsymbol{x}^a_{i,j}$, $\boldsymbol{x}^b_{i,j}$, $\boldsymbol{x}^c_{i,j}$ and $\boldsymbol{x}^d_{i,j}$:

$$\boldsymbol{x}_{i,j} = \frac{\boldsymbol{x}^a_{i,j} + \boldsymbol{x}^b_{i,j} + \boldsymbol{x}^c_{i,j} + \boldsymbol{x}^d_{i,j}}{4}. \tag{3.25}$$

cell faces The cell faces $l = 1, \ldots, 4$ (respectively bottom, right, top and left) are defined as the lines connecting the node points. Each cell face has length Δs_l which can be calculated using the length in x- and y-direction:

$$\Delta s_l = \sqrt{(\Delta x_l)^2 + (\Delta y_l)^2}. \tag{3.26}$$

Here the lengths Δx_l and Δy_l simply follow from the coordinates of the corner points connected by the specific cell face. Using this information it is possible to calculate the cell-face normal and tangential vectors. It is not difficult to see that these vectors can be written as follows:

$$\boldsymbol{n}_l = \left(\frac{\Delta y_l}{\Delta s_l}, -\frac{\Delta x_l}{\Delta s_l}\right)^T, \tag{3.27}$$

$$\boldsymbol{t}_l = \left(\frac{\Delta x_l}{\Delta s_l}, \frac{\Delta y_l}{\Delta s_l}\right)^T. \tag{3.28}$$

Cell area The area $A_{i,j}$ of cell $\Omega_{i,j}$ follows from the two cross-vectors connecting the node points $a - c$ and $d - b$. Taking half of the outer-product of the vectors $\boldsymbol{r}_{db}$ and $\boldsymbol{r}_{ac}$ gives:

$$A_{i,j} = \frac{1}{2}\left|\boldsymbol{r}_{db} \times \boldsymbol{r}_{ac}\right| = \frac{1}{2}\left|\left(x^b - x^d\right)\left(y^c - y^a\right) - \left(x^c - x^a\right)\left(y^b - y^d\right)\right|. \tag{3.29}$$

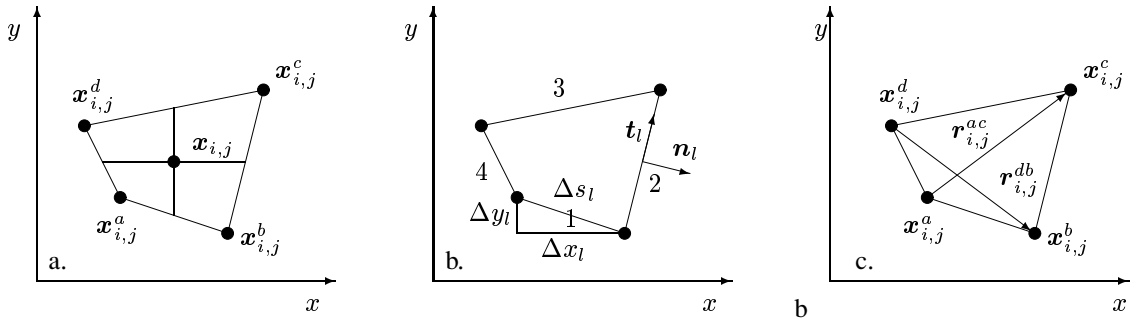

Figure 3.2: A quadrilateral finite volume $\Omega_{i,j}$ defined by its four node points $\boldsymbol{x}^{a,b,c,d}_{i,j}$. *(a)* The cell center $\boldsymbol{x}_{i,j}$ follows from the two lines connecting the mid points of the cell faces. *(b)* At each cell face $l = 1, \ldots, 4$ the outward unit normal vector $\boldsymbol{n}_l$ and the tangential unit vector $\boldsymbol{t}_l$ follow from the length and the orientation of the cell face. *(c)* The area $A_{i,j}$ of the cell follows from the vectors $\boldsymbol{r}_{db}$, $\boldsymbol{r}_{ac}$ connecting the node points.

3.5.2 Boundary conditions

The boundary conditions are applied by making use of extra rows of grid cells, also called *virtual cells*, at the boundary of the flow region. The state-variable distribution in the cell centers are either copies or mirrors, depending on the type of boundary condition, of the cells they are bordering. The different types used in this report are treated here in more detail.

Soft boundary The soft boundary condition is applied when waves are allowed to run out of the flow region without creating reflections. A typical situation for using the soft boundary is in the case of boundaries extending to infinity. In the case of soft boundaries the virtual cells are exact copies of their real *neighbor* cells; the state-variables are exactly the same. Because these virtual cells have only one neighbor, sub-cell interpolation can not be applied here. Therefore another approach has to be followed to obtain the cell-face values of the state-variables at the interface separating the virtual cell and its real neighbor. Because the interface fluxes between the virtual cells and their neighboring real cells have to be zero to prevent reflections to occur, the cell-face values in the virtual cells can be directly copied from the cell-face values in their real copy. Obviously sub-cell interpolation is not required now.

Although the level-set function is also referred to as a state variable, its treatment near boundaries requires some special attention. Because the level-set function is actually a distance function it may not always be physical to just copy the level-set value of a real cell into a virtual cell. Fortunately the soft boundary condition implies that no flux should exist over the final cell face such that taking the neighboring value for the level-set function will result in no problems.

The redistancing procedure also requires boundary conditions. Using virtual cells suffices. Only now it is not allowed to simply copy the level-set function into the virtual cell since this means that the gradient over these cells is zero, while the redistancing procedure wants it to be one everywhere. Therefore a better, but in two dimensions still not very accurate choice is to take the value of the level-set function and add or subtract the local cell-length in x or y-direction. Since the level-set function has to be a distance function (linear) approach suffices. Although it would be better to use the local gradient to determine in which direction the gradient is pointing instead of solely taking the cell-length in one direction.

Hard boundary This type of boundary condition is used when waves should be reflected and continued as their mirror images. This is the case when a solid wall is present in the flow region or when symmetry is applied. The hard boundary condition requires every virtual cell to be the exact copy of the neighboring cell, except for the normal velocity, which should be mirrored instead of copied. The same holds for the cell-face values; all are copied from their real neighbors, instead of the normal velocity which is mirrored.

The level-set function now needs no further attention since a hard boundary implies mirroring (symmetry). The virtual cells are therefore required to contain exactly the same values of the level-set function as their neighboring cells.

Periodic boundary The period boundary condition is used when the flow domain treated is periodically repeated at this boundary. Periodic boundary conditions always require two opposite boundaries of equal shape. This boundary condition is applied by copying the first row of real cells into the virtual cells neighboring the last row of real cells, and by copying the last row of virtual cells in the row of virtual cells next to the row of first real cells. The same holds for the level-set function and the redistancing procedure.

Constant inflow boundary One of the test cases (supersonic free jet) requires a constant inflow condition. This means that these virtual cells have the same state-variable distribution during the whole calculation. The same holds for the redistancing procedure.

3.5.3 CFL condition

The stability of a numerical scheme plays an important role. A scheme is numerically stable if numerical errors in the solution remain bounded, i.e. do not grow. Whether a scheme is stable depends on the size of the temporal step and spatial cell sizes, Δt and $A_{i,j}$. One of these can be chosen freely but the other has to fulfill the so-called *CFL condition*, which secures stability by relating the two grid size parameters using the *CFL number*. For a two-dimensional scheme using quadrilateral cells the CFL number is defined as follows:

$$\nu_{i,j} = \frac{\Delta t}{2A_{i,j}} \left[\sum_{l=1}^{4} \left(|v_n| + a \right)_l \Delta s_l \right]_{i,j} . \tag{3.30}$$

For the three-stage scheme combined with the $\kappa = \frac{1}{3}$ scheme (in the limiter form), the CFL number has to fulfill the condition: $\nu \leq 1.35$. This approach implies that before choosing a temporal step size, the maximum wave speeds at all interfaces must be known. Furthermore it requires for each cell a different CFL number, to be updated after every time step. Because this procedure is computationally expensive and does not pay-off in most cases a different approach shall be followed. Based on the initial conditions, and the maximum initial wave speed, a comfortable choice will be made for the CFL number, which will be kept constant during all computations.

3.6. Redistancing of level-set function

The redistancing procedure that changes the distorted level-set function back into a true signed distance function is governed by the PDE equation (3.31) as explained in the previous chapter. Solving for this equation means looking for that distribution of ϕ such that its absolute gradient $|\nabla\phi| = 1$ everywhere. To avoid confusion between the real time t and the pseudo-time τ the procedure is written as follows:

$$\begin{array}{rcl} d\left(\boldsymbol{x}, \tau = 0\right) & = & \phi\left(\boldsymbol{x}, t\right), \\ \frac{\partial d}{\partial \tau} & = & S\left(\phi_0\right)\left(1 - |\nabla d|\right), \\ \phi\left(\boldsymbol{x}, t\right) & = & d\left(\boldsymbol{x}, \tau = \infty\right), \end{array} \tag{3.31}$$

where $S_\epsilon\left(\phi\right) = 2H_\epsilon\left(\phi\right) - 1$. 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 (3.31). Therefore a scheme is chosen that is second-order in both space and time. The update in time is chosen to be a two-stage scheme similar to (3.6). 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:

$$\begin{array}{rcl} d_{i,j}^{(0)} & = & d_{i,j}^{n_\tau}, \\ d_{i,j}^{(1)} & = & d_{i,j}^{(0)} + \frac{\Delta\tau}{2}\left[S\left(\phi_0\right)\left(1 - \left|\nabla d^{(0)}\right|\right)\right]_{i,j}, \\ d_{i,j}^{(2)} & = & d_{i,j}^{(0)} + \Delta\tau\left[S\left(\phi_0\right)\left(1 - \left|\nabla d^{(1)}\right|\right)\right]_{i,j}, \\ d_{i,j}^{n_\tau+1} & = & d_{i,j}^{(2)}. \end{array} \tag{3.32}$$

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:

$$|\nabla d|_{i,j} = \sqrt{\left(\frac{\Delta_x d}{\Delta x}\right)_{i,j}^2 + \left(\frac{\Delta_y d}{\Delta y}\right)_{i,j}^2}, \tag{3.33}$$

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

$$\left(\Delta_x^L d\right)_{i,j} = d_{i,j} - d_{i-1,j} + \frac{1}{2}\left|\min\right|\left(d_{i+1,j} - 2d_{i,j} + d_{i-1,j}, d_{i,j} - 2d_{i-1,j} + d_{i-2,j}\right), \tag{3.34}$$

$$\left(\Delta_x^R d\right)_{i,j} = d_{i+1,j} - d_{i,j} - \frac{1}{2}\left|\min\right|\left(d_{i+1,j} - 2d_{i,j} + d_{i-1,j}, d_{i+2,j} - 2d_{i+1,j} + d_{i,j}\right), \tag{3.35}$$

$$w_x^L = S_\epsilon\left(\phi_{i,j}\right)\left(\Delta_x^L d\right)_{i,j}, \tag{3.36}$$

$$w_x^R = S_\epsilon\left(\phi_{i,j}\right)\left(\Delta_x^R d\right)_{i,j}, \tag{3.37}$$

$$\left(\Delta_x d\right)_{i,j} = \begin{cases} \left(\Delta_x^L d\right)_{i,j} & \text{if} \quad w_x^L > 0 \quad \text{and} \quad w_x^L + w_x^R > 0, \\ \left(\Delta_x^R d\right)_{i,j} & \text{if} \quad w_x^R < 0 \quad \text{and} \quad w_x^L + w_x^R < 0, \\ 0 & \text{if} \quad w_x^L < 0 \quad \text{and} \quad w_x^R > 0. \end{cases} \tag{3.38}$$

A similar scheme can be derived for $\Delta_y d$, which shall not be done here. It is obvious that the requirement of a uniform Cartesian grid indeed holds for this specific scheme. In the case of a non-uniform grid the differences $\Delta_x d$ and $\Delta_y d$ can not directly be obtained from the differences in i and j directions as was done in (3.34)

and (3.35). This because the directions i, j are not the same as x, y. It can well be that i, j for certain cells are not orthogonal. Another approach should be followed in this case. Because only uniform Cartesian grids are treated this shall not be elaborated 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 (3.34) and (3.35) by removing the most right terms ($|\min|\,(\ldots)$). What remains is a stability condition relating $\Delta\tau$, Δx and Δy. A safe choice will be: $\Delta t = \frac{1}{2}\min\left(\Delta x, \Delta y\right)$.

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 exactly correct distance function, it is possible to take only a small amount of time steps in the redistancing procedure. It has been shown that 6 will do.

Chapter 4
Pressure Oscillations

One of the major problems of conservative methods for compressible [1] two-fluid flows is the occurrence of *spurious pressure oscillations* due to solely numerical errors. 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 and others [8] and the fixes by Karni [14, 15] are best known. In this chapter a locally non-conservative method, first developed by Abgrall and Karni in [1], is discussed. The method is comparable to the ghost-fluid method but is more simple to implement and requires less computational power. In the next chapter its capabilities are shown using several tests.

4.1. Origin of the oscillations

All numerical methods suffer from numerical errors. Depending on the stability of the scheme used these errors can result in oscillations or *wiggles* in the distributions of the state variables. Fortunately most errors are related to the size of the grid cells used; the smaller the grid cells the smaller the errors. Using a higher-order method makes that these grid-related errors often disappear. Unfortunately this is not the case for the pressure oscillations treated here. These oscillations, which do not occur in the case of single-fluid flows, i.e. a constant value of γ, do not disappear when choosing a higher-order scheme.

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 [1] 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 [18] 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 treat the one-dimensional version of the Euler equations discretized using a simple upwind scheme. Assume the interface to be located between the left cell face of cell i and its cell center (see figure 4.1). 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. Applying the upwind scheme, for a flow to the right, to the equations for conservation of mass, momentum and energy results in:

$$\rho_i^{n+1} - \rho_i^n = -\frac{\Delta t}{\Delta x}\left[(\rho u)_i^n - (\rho u)_{i-1}^n\right], \tag{4.1}$$

$$(\rho u)_i^{n+1} - (\rho u)_i^n = -\frac{\Delta t}{\Delta x}\left[\left(\rho u^2 + p\right)_i^n - \left(\rho u^2 + p\right)_{i-1}^n\right], \tag{4.2}$$

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

Assuming that both the velocity u and the pressure p are continuous over the interface (at $t = n$), 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 (4.1) and (4.2) 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 (4.3) as follows:

$$\left(\frac{p}{\gamma - 1}\right)_i^{n+1} - \left(\frac{p}{\gamma_R - 1}\right) = -\frac{u\Delta t}{\Delta x}\left[\left(\frac{p}{\gamma_R - 1}\right) - \left(\frac{p}{\gamma_L - 1}\right)\right], \tag{4.4}$$

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

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, and 2. $\gamma_i^{n+1} = \gamma_R = \gamma_i^n$, which means that the interface has not moved. Since in the chosen approach both situation are in general not the case, pressure oscillations are the undesired result.

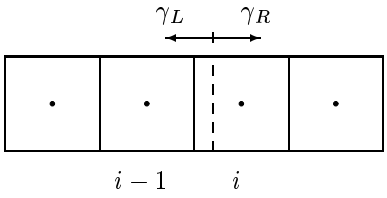

Figure 4.1: Grid cells containing an interface separating two fluids with $\gamma_L \neq \gamma_R$.

From expression (4.4) it follows that the presence of the factor $\frac{1}{\gamma-1}$, which jumps over an interface, is the origin of the 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. Though there is a way of reducing the magnitude of these errors; interface smoothing will reduce the local jumps in the γ distribution and thus 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 non-conservative method), a fix has to be applied. Because it has been shown that the errors occur due to the jump in the ratio of specific heats over the interface, it 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 used for all fluxes.

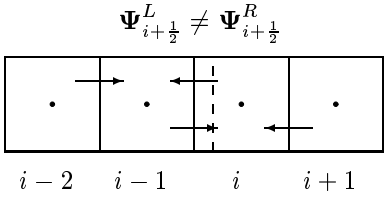

Figure 4.2: The simple fix applied to grid cell i containing an interface (dotted line) separating two fluids with $\gamma_L \neq \gamma_R$.

To clarify the chosen approach the situation from figure 4.1 is considered. The fluxes to be used at the interfaces are shown in figure 4.2. When updating the state variables in cell i two 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 left, right, top and bottom flux, indicated with $l = 1, \ldots, 4$). When no interface and thus no jump in γ is present the flux at $i - \frac{1}{2}$ updating cell

i, here indicated with $\mathbf{\Psi}^R_{i-\frac{1}{2}}$ 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, $\mathbf{\Psi}^L_{i-\frac{1}{2}}$ (only the normal vector is pointed in opposite direction so the contribution of these fluxes to the total flux will differ in sign for both cells). 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:

$$\mathbf{\Psi}^R_{i-\frac{1}{2}} = \mathbf{\Psi}\left(\gamma_R\right), \tag{4.5}$$

$$\mathbf{\Psi}^L_{i-\frac{1}{2}} = \mathbf{\Psi}\left(\gamma_L\right). \tag{4.6}$$

Because the fluids in cells i and $i+1$ are the same, the right and left *pointing* [2] 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 of to only one as figure 4.2 maybe suggests.

It has been shown that besides a jump in γ the oscillations are also caused by a possible difference between γ_i^{n+1} and γ_i^n. When the total energy $(\rho E)_i^{n+1}$ is known it is used to calculate the pressure at the new time level. This new pressure will only be error free when the *old* version of γ, i.e. γ_i^n, is used. This procedure is called *freezing* of the ratio of specific heats.

There are several advantages to this method. The first is the most trivial one; it removes the pressure oscillations because each cell only *sees* one γ. The second advantage is its ease of implementation. Because the interface flux calculation now only uses one value of γ, a simple and standard single-fluid Riemann solver can be used (this means that equation (3.17) can be used in its original form with a γ instead of a $\hat{\gamma}$). No extra interpolation of γ is necessary. This brings us directly to the third advantage, namely the fact that this method works only when necessary. When there is no jump in the distribution of γ between two cells, the flux at the interface separating both cells is the same for both update procedures. This means that only at those interfaces that separate cells with different values of γ two fluxes should be calculated, while all other cells still require only one flux solving procedure.

Although this approach seems similar to the ghost fluid method there are major differences. Instead of introducing *ghost cells* using extrapolation of state variables, only the choice of γ changes, which is a simple efficient procedure without extra computational effort.

4.2.1 Fix algorithm

The method described above will be implemented in the numerical solver in the following way: at first the location of the material interface is determined using the distribution of the level-set function ϕ (interface has $\phi = 0$). Using the smoothing procedure the smooth step-function H_ϵ is determined. Using this step-function the value of γ in each cell is calculated. Given the primary state variables $\boldsymbol{w}$ and γ the conservative state variables $\boldsymbol{q}$ can be calculated. Note that γ is only used in the transformation from the pressure to the total energy. Using the distribution of conservative state variables it is possible to calculate the interface fluxes. If two neighboring cells have different values of γ then at their common cell face two fluxes are calculated using different values of γ. If two neighboring cells contain the same fluid then only one flux is calculated using the Riemann solver. Given all interface fluxes the solution of $\boldsymbol{q}$ is updated to the following time level. A three-stage scheme is used for getting to the new time level. Using the *old* or *freezed* values for γ the primary state variables are calculated from the *new* conservative state variables. Finally the level-set function is redistanced to a real distance function. And the whole procedure is repeated.

4.2.2 Errors due to fix

The first type of error introduced by the fix presented above is that it locally abandons conservation of energy. The effect of locally not being conservative has been derived in [1] for a first-order scheme. The total

[2] 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).

accumulated error for the one-dimensional case after n time steps is equal to:

$$\epsilon_{\text{cons.}} = n\Delta t \begin{pmatrix} 0 \\ 0 \\ up\Delta\left(\frac{1}{\gamma-1}\right) \end{pmatrix}, \tag{4.7}$$

where Δ indicates the jump between cells i and $i-1$. Note that there is a dependency on the temporal accuracy.

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:

$$\Delta x \left(E|_{\gamma=\gamma_L} - E|_{\gamma=\gamma_R} \right) = -\Delta x \Delta \left(\frac{1}{\gamma - 1} \right) p. \tag{4.8}$$

It has been shown by Abgrall and Karni that the contributions of (4.7) and (4.8) have opposite effects on the solution.

Chapter 5
Numerical results

In the previous chapters the model describing the flow of two compressible, inviscid fluids was derived and the governing flow equations were discretized. In this chapter the derived flow solver is tested. First several one-dimensional tests are performed with both the solver with and without fix. The results verify the capabilities of the proposed fix. With the fixed numerical solver also several two-dimensional test cases are considered. For reference purposes both the one and two-dimensional test cases performed in [36] are used. The solver seems to produce excellent results.

5.1. 1D SHOCK-TUBE PROBLEMS

Three one-dimensional shock-tube problems are used to test the numerical solver and to prove the capabilities of the simple fix proposed in the previous chapter. A shock-tube problem is actually a Riemann problem since its goal is to find the solution in time of two initially separated states. At time $t = 0$ the membrane separating the two fluids is removed and several waves (either shocks, expansion fans or contact discontinuities) appear which will run into the flow domain and change the state variable distributions. Because the exact solutions of these shock-tube problems are known the accuracy of the numerical solver can be determined.

5.1.1 Translating interface

The first shock-tube problem treats a translating interface separating two fluids. Initially the velocity and pressure are continuous, while the density and the ratio of specific heats differ (see table 5.1). The ratio of specific heats will be smoothed using the smoothed step function with $\epsilon = \frac{3\Delta x}{2}$. Due to this initial distribution the interface will be advected to the right with the flow velocity. In the exact situation the velocity and pressure remain constant at all time.

From figure 5.1 it can be seen that this is not the case for the solver without fix. Oscillations appear in the pressure distribution which due to the coupling of the flow variables also make the density and velocity distributions to show oscillations (due to the chosen scale these oscillations are not visible in the density distribution but *zooming in* proves their presence). These oscillations are solely the result of the spurious pressure oscillations mentioned earlier.

The results obtained using the solver with the fix show none of this. Both the pressure and velocity distributions remain oscillation-free and continuous at the interface, as they should be. Although the density is smeared out to some extent near the interface, which is the case for all numerical methods, the location of the contact discontinuity is perfectly captured.

However, the solutions look better than they are. When looking at the plot of the ratio of specific heats, figure 5.2, it is obvious that the numerical position of the interface, where γ jumps, is completely wrong compared to the exact location of the interface (the contact discontinuity). The plot of the numerical and the exact (not redistanced) level-set distribution shows the cause of this off-set; at those points where the density has been discontinuous over a cell face (the density becomes smooth due to numerical diffusion) the level-set function has been advected with the wrong velocity resulting in a wrong location of its zero-value and thus a wrong distribution of the ratio of specific heats. Thus, although it is not visible in the distributions of the density, pressure and velocity, near the interface the wrong values of γ have been used.

Further research on this solely numerical error will be performed in the near future and will be one of the research subjects of the author for his Master's thesis. For now it should be mentioned that the errors are intrinsic of the conservative approach of the level-set equation. Using a non-conservative approach will not produce them. Further the error is of order one (Δx) and it depends on the severity of the jump in the density.

Especially this dependence on the density makes the other test cases discussed in this report still valid. In particular the two-dimensional test cases treat small density differences and are therefore not subjected to this error.

	left	*right*
ρ	1000.0	1.0
u	1.0	1.0
p	1.0	1.0
γ	1.4	1.6

Table 5.1: Initial data for 1D translating interface problem.

5.1.2 High-pressure Sod

The second one-dimensional test case treats the well-known Sod problem. A two-fluid version with a stronger discontinuity is used. See table 5.2 for the initial data. The exact solution contains a left-running expansion fan, a right-running shock and a right-running contact discontinuity separating the two fluids (the interface).

Figure 5.3 shows the numerical results. Again the spurious pressure oscillations occur in the solution obtained when no fix is used. The solutions obtained with fix show again no oscillations. The results comply extremely well with the expected exact results. The expansion, the shock and the interface are positioned in the correct locations. Some smearing is present near the leading and trailing edges of the expansion fan, the interface and the shock, though this is expected.

	left	*right*
ρ	10.0	0.125
u	0.0	0.0
p	10.0	0.1
γ	1.4	1.6

Table 5.2: Initial data for 1D high-pressure Sod problem.

5.1.3 No-reflection problem

The final test treats a strong shock hitting the two-fluid interface from the left at time $t = 0$, resulting in a shock moving to the right without creating a reflection. The initial conditions for this Riemann problem are given in table 5.3 and the numerical results together with the exact solution are presented in figure 5.4.

When comparing the solutions obtained from the solver without and with fix, it appears that there are two different sources of errors present in the numerical solution. The first is due to the pressure oscillations mentioned several times already. Near the interface, the first jump in the density distribution, the numerical *unfixed* solution shows large spurious oscillations. These oscillations are not present when the fix is applied.

Though there is a second source of errors which is visible as a small *hump* in the state variable distributions. Straightforward calculations show that this small wiggle moves with the local left-running wave speed, i.e. the speed of sound subtracted from the local velocity (although this wave is referred to as *left-running* it is actually moving to the *right* because the local velocity is supersonic). This indicates that although the initial conditions are chosen in such a fashion that no reflection wave should occur, small errors appear due to the initial discontinuity and are transported with the local wave speeds.

The no-reflection problem is used to obtain the order of convergence of the numerical scheme used in the solver. Running the calculations for several grids with different cell sizes and calculating the L_1-error results in table 5.4. It has been mentioned earlier that the scheme used is approximately second order. Unfortunately for problems with strong shocks this order will never be obtained. The obtained results are therefore what

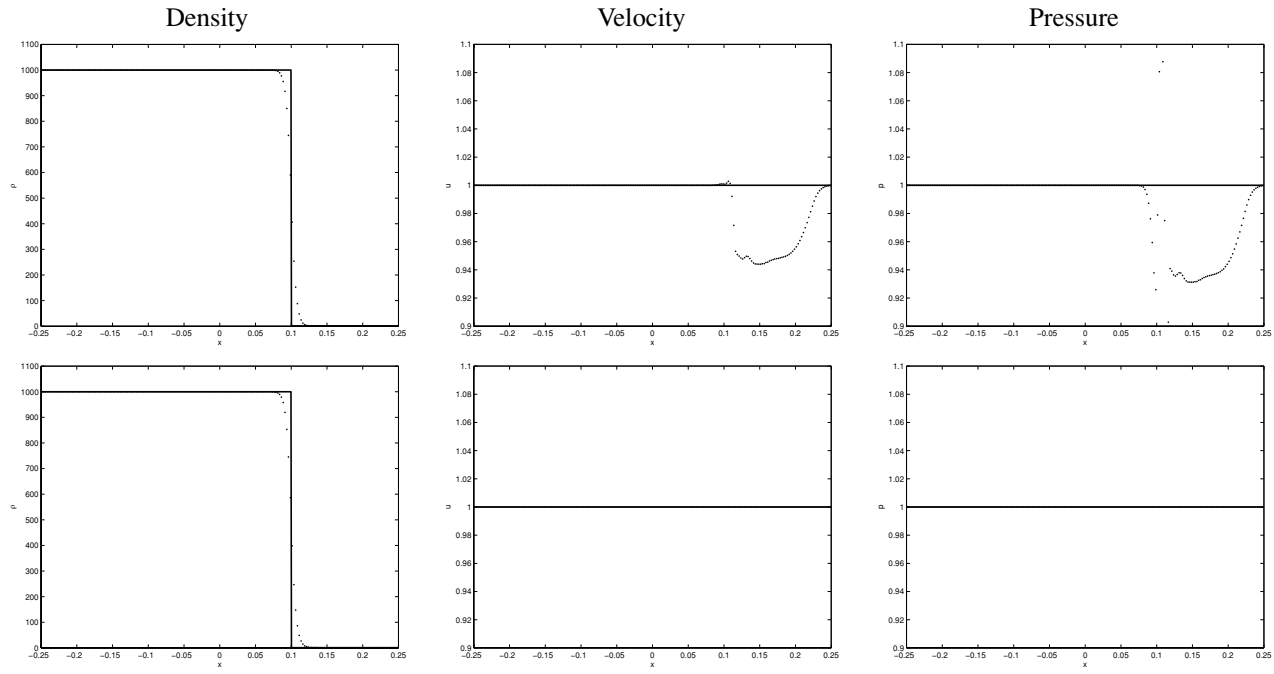

Figure 5.1: 1D translating interface problem. (ρ, u, p) at $t = 0.1$. Grid has 200 cells and $\Delta t/\Delta x = 0.5$. Solid lines are exact solutions, dotted lines are numerical solutions. *Top:* solver without fix. *Bottom:* solver with fix.

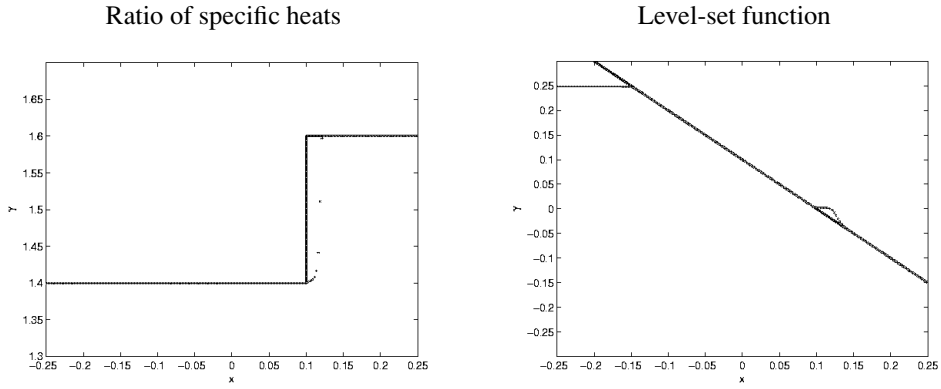

Figure 5.2: 1D translating interface problem with off-set of level-set interface due to numerical errors. Solid lines are exact solutions, dotted lines are numerical solutions. Note that level-set function has not been redistanced for this plot, resulting in a non-straight line.

	left	*right*
ρ	3.1748	1.0
u	9.4350	0.0
p	100.0	1.0
γ	1.667	1.2

Table 5.3: Initial data for 1D no-reflection problem.

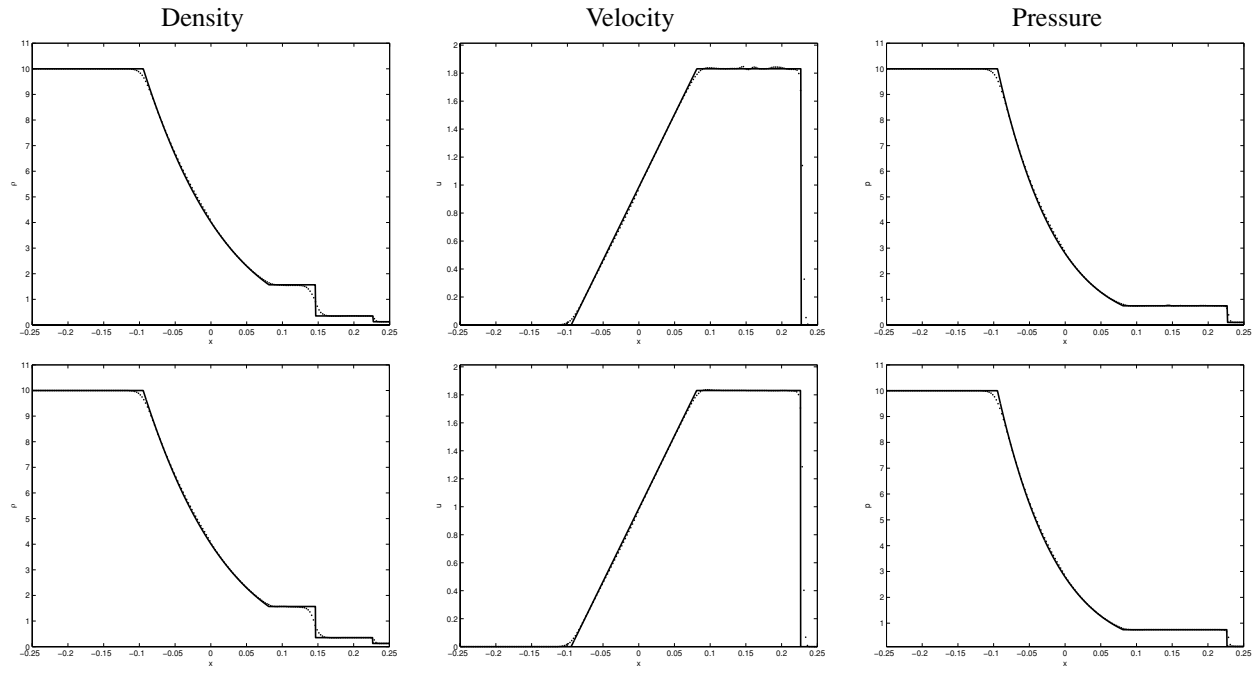

Figure 5.3: 1D high-pressure Sod problem. (ρ, u, p) at $t = 0.08$. Grid has 200 cells and $\Delta t/\Delta x = 0.2$. Solid lines are exact solutions, dotted lines are numerical solutions. *Top:* solver without fix. *Bottom:* solver with fix.

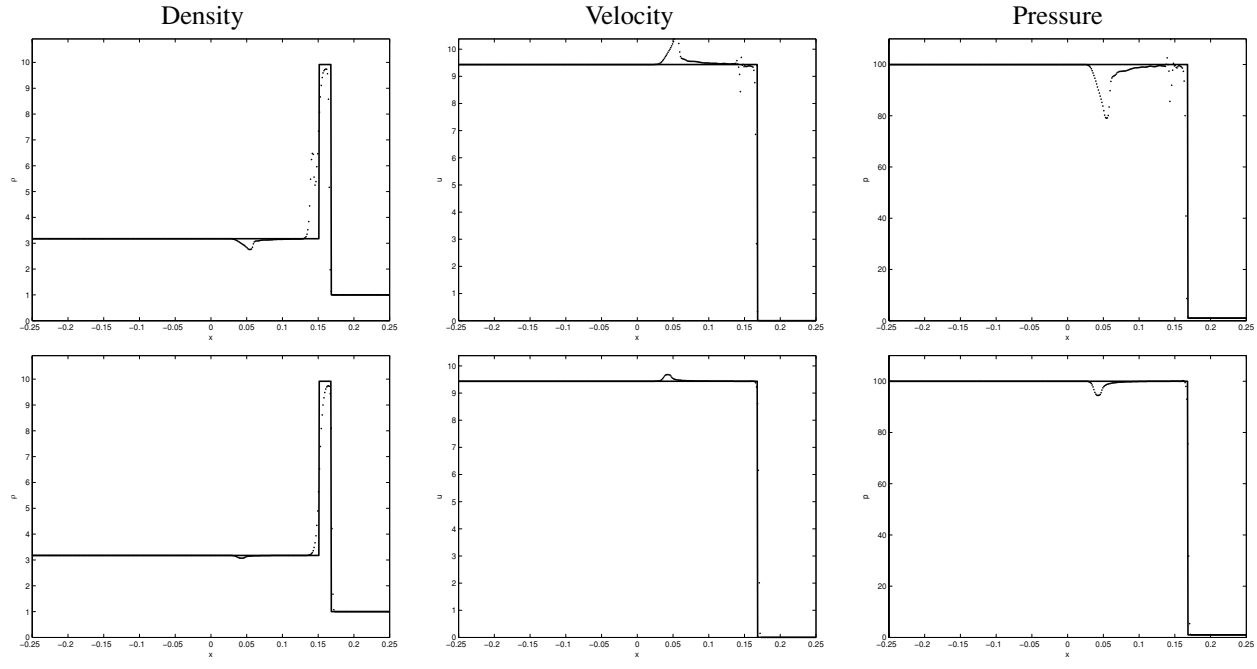

Figure 5.4: 1D no-reflection problem. (ρ, u, p) at $t = 0.016$. Grid has 400 cells and $\Delta t/\Delta x = 0.04$. Solid lines are exact solutions, dotted lines are numerical solutions. *Top:* solver without fix. *Bottom:* solver with fix.

one expects for these types of problems. Note that using a *fractional error norm* will result in a higher order because the influence of the errors due to the strong discontinuity are given less influence on the total error. See [25] for a discussion of these special norms.

J	ϵ_ρ	ϵ_u	ϵ_p
100	0.216326	0.136530	1.429744
200	0.121349	0.059584	0.739179
400	0.068219	0.032254	0.371704
800	0.039209	0.018549	0.197539
1600	0.024585	0.012426	0.121374
p	0.790460	0.859912	0.902024

Table 5.4: Results for convergence test for 1D no-reflection problem.

5.2. 2D 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. This problem is taken from the experiments of Haas and Sturtevant [12]. Numerical treatment of this problem has among others been done by Quirk and Karni [27] and more recently by Wackers and Koren [36]. The experimental (and numerical) 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 (not necessary in the numerical case). 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 (shock passes 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 two-fluid approach followed in this report can be used.

In these tests the circular bubble in air contains either helium (lighter than air) or the refrigerant gas R22 (heavier than air). In the helium case the speed of sound in the bubble is higher than the speed of sound in the air resulting in the *refracted wave*, the shock continuing in the bubble, to travel 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 wave to lag behind the shock in the air. After a while the wave interaction becomes more complex producing characteristic patterns for both cases.

For the numerical computations the speed of sound in the air is taken equal to one (in reality this was 343 m/s so every speed should be scaled with this number), such that the shock moves with a Mach number equal to 1.22. The initial conditions for both the helium and the air case are given in table 5.5. 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.

The numerical flow domain is halved by making use of symmetry properties (only one half of the bubble is treated) to reduce the amount of computations. Due to computational limitations the grid is limited to 300 cells in x direction and 150 cells in y direction (note that Wackers used a 400×200 grid in [36]). To fulfill the CFL condition a time step of $\Delta t = 1.25 \times 10^{-5}$ is used for the helium test and $\Delta t = 2.5 \times 10^{-5}$ for the R22 case. To allow the shock to develop before it hits the bubble, its initial position is taken 5 cells right of the bubble-air interface. The time the shock takes to reach the bubble is 2.24×10^{-3} seconds. 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 appendix I for a complete discussion of this procedure.

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

5.2.1 Helium bubble

The first test uses a bubble filled with helium. The helium used in this test (contaminated with air) has a sound speed that is much higher than the sound speed of the air ($a_{\text{helium}} = 2.44$ and $a_{\text{air}} = 1.0$). This makes that the *refracted shock*, which is the original shock continuing in the bubble, moves faster than the shock in air. The point where this refracted shock hits the interface therefore lies ahead of the point where the original shock hits the interface. At this point the refracted shock continues outside the bubble as the *transmitted shock* and moves towards the original shock. After intersecting this incoming shock it continues to the right until it is finally bent by the expansion that moves away downstream from the point where the original shock hits the interface. When the refracted shock passes through the bubble interface at the left it continues outside the bubble as the earlier mentioned transmitted shock. This shock now has the shape of a bow-shock. For a more extensive description and an analytical solution of this interaction see [36].

The wave pattern described above is clearly visible in the plots of the density (figure 5.5) and the pressure (figure 5.6). The results obtained fully comply with similar tests. Although there is some smearing near the right bubble interface, the density and pressure distributions remain surprisingly sharp and the location of the waves is correct, indicating that there is no apparent loss of accuracy due to the method locally not being conservative.

The deformation of the bubble during the interaction is visible in figure 5.7 showing plots of the contours of the ratio of specific heats. Due to the level-set approach the interface remains very sharp (no more than five grid cells width), this in contrast to the five-equation model used by Wackers and Koren which shows some smearing of the interface. The location of the bubble is perfectly captured and remains accurate throughout the whole simulation. Besides the time periods used by Wackers and Koren figure 5.8 also shows the bubble at a later stage. Clearly visible is the process of *cutting-in* of the bubble along the symmetry axis, finally leading to the bubble breaking-up in two vortices. Because the initial density distribution is smeared over the interface no disastrous instabilities occur (see appendix 1 for a comparison between smeared and non-smeared solutions).

5.2.2 R22 bubble

In the second test case the bubble is filled with refrigerant gas R22, an HCFC. This gas is among others used in air conditioners and other similar applications. R22 is much heavier than air (density is three times larger) and the ratio of specific heats is somewhat lower. This makes that the sound speed in R22 is lower than the sound speed in air ($a_{\text{R22}} = 0.532$ and $a_{\text{air}} = 1.0$). This lower speed of sound results in the refracted wave (in the bubble) to move slower than the original wave (in the air). This causes the top of the refracted wave to bend forward when time passes. 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 [36].

Figures 5.9 and 5.10 show the density and pressure distributions. Again the results comply very well with known data. Clearly visible are the refracted and reflected shocks. Both the density and the pressure distributions show some smearing near the shocks which is as expected.

In figure 5.11 the distribution of the ratio of specific heats is plotted. Because the gas in the bubble is heavier it is less sensitive to compression as was the case for helium. The bubble therefore only deforms slightly. The interface between the bubble remains extremely sharp (intrinsic property of the level-set method).

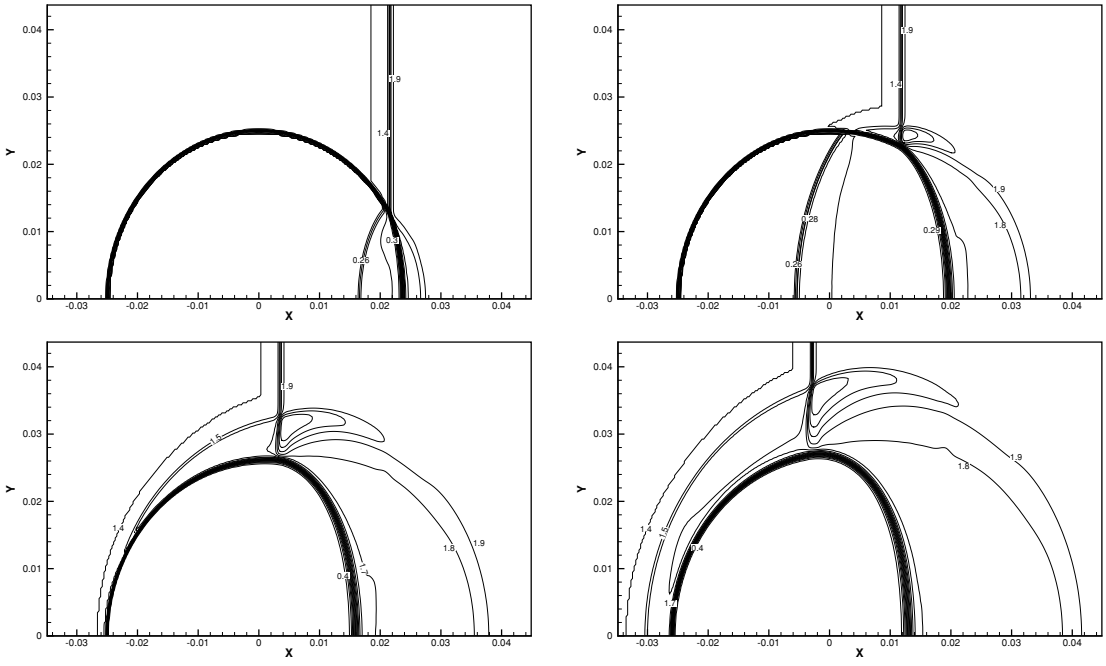

Figure 5.5: Shock hitting helium bubble. Density at $t = 5 \times 10^{-3}$, $t = 13 \times 10^{-3}$, $t = 19.8 \times 10^{-3}$ and $t = 25 \times 10^{-3}$. Grid has 300×150 cells and $\Delta t = 1.25 \times 10^{-5}$.

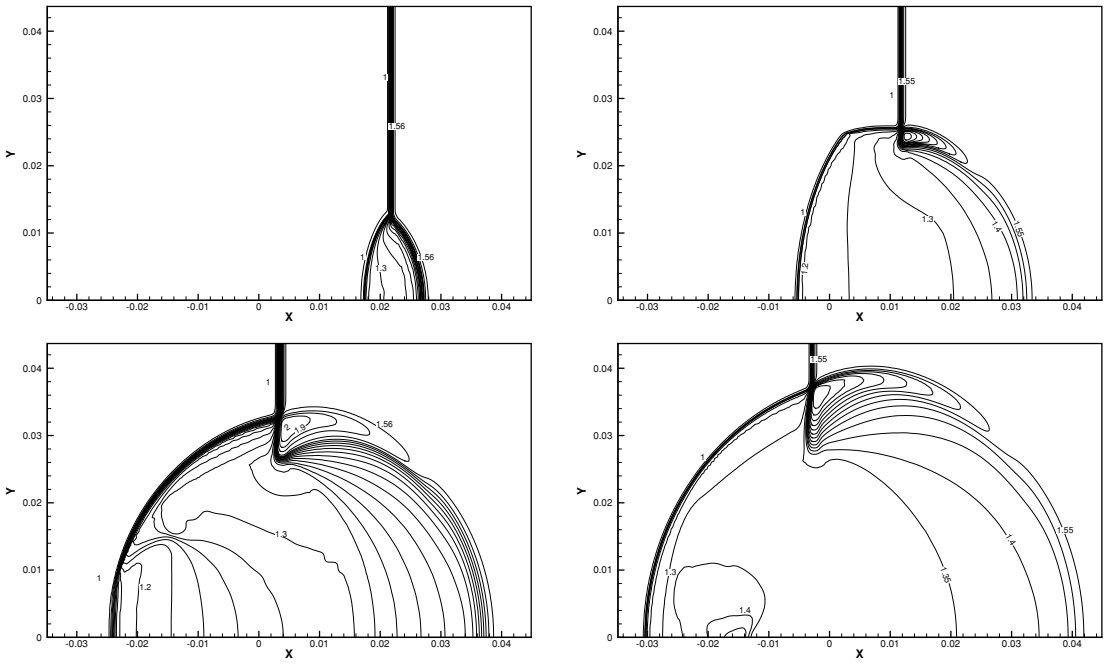

Figure 5.6: Shock hitting helium bubble. Pressure at $t = 5 \times 10^{-3}$, $t = 13 \times 10^{-3}$, $t = 19.8 \times 10^{-3}$ and $t = 25 \times 10^{-3}$. Grid has 300×150 cells and $\Delta t = 1.25 \times 10^{-5}$.

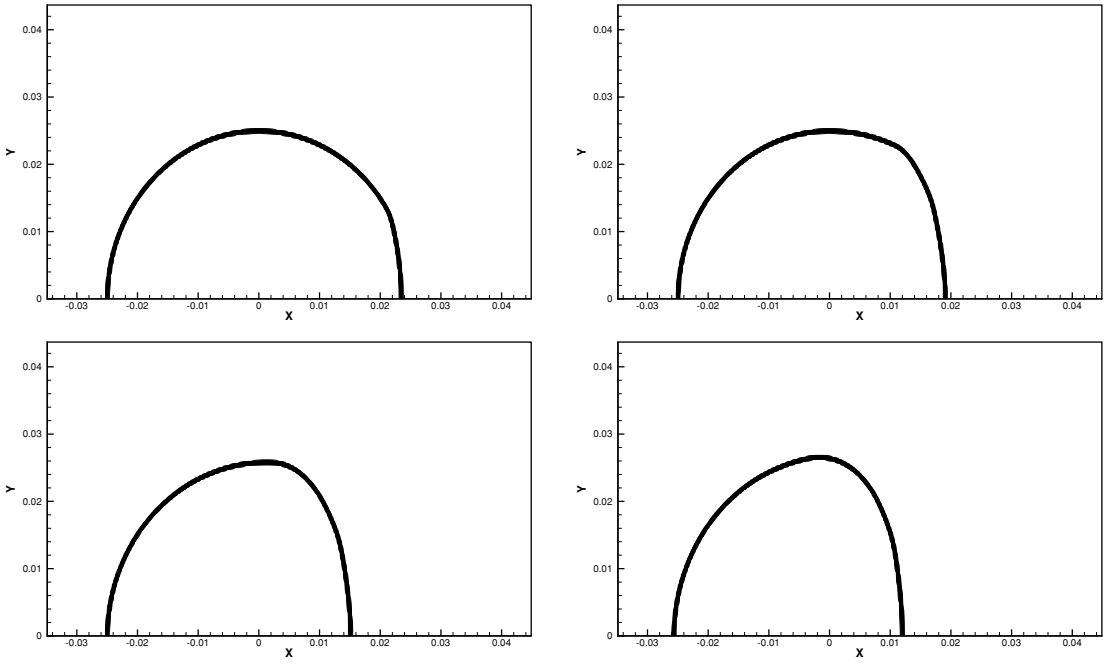

Figure 5.7: Shock hitting helium bubble. Ratio of specific heats at $t = 5\times10^{-3}$, $t = 13\times10^{-3}$, $t = 19.8\times10^{-3}$ and $t = 25 \times 10^{-3}$. Grid has 300×150 cells and $\Delta t = 1.25 \times 10^{-5}$.

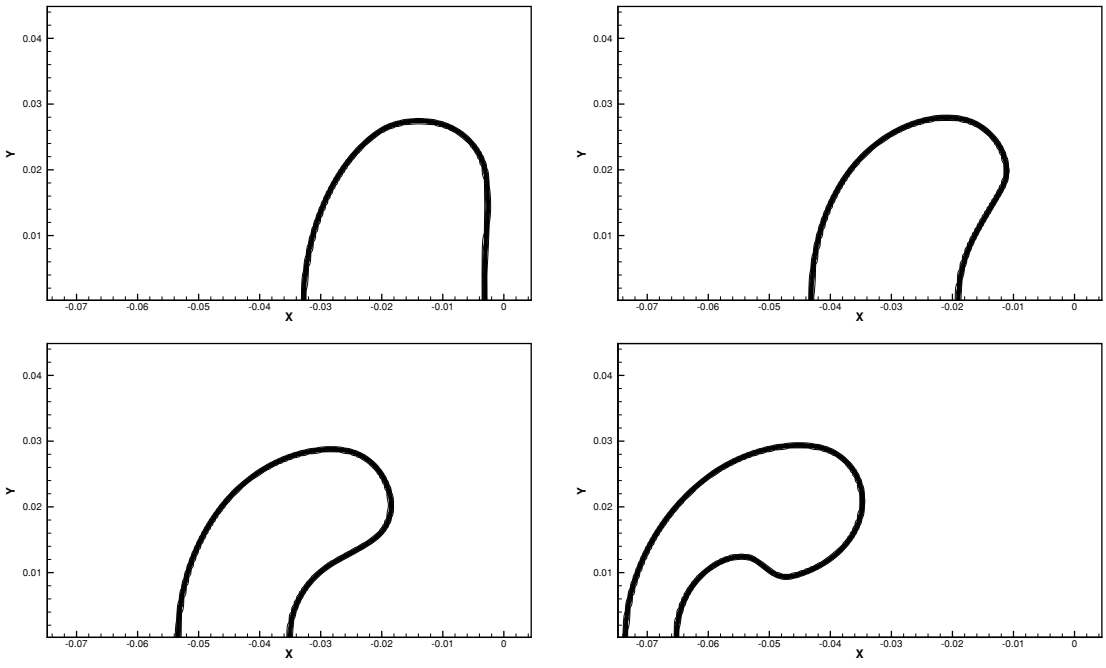

Figure 5.8: Shock hitting helium bubble. Ratio of specific heats at $t = 50 \times 10^{-3}$, $t = 75 \times 10^{-3}$, $t = 100 \times 10^{-3}$ and $t = 150 \times 10^{-3}$. Grid has 300×150 cells and $\Delta t = 1.25 \times 10^{-5}$.

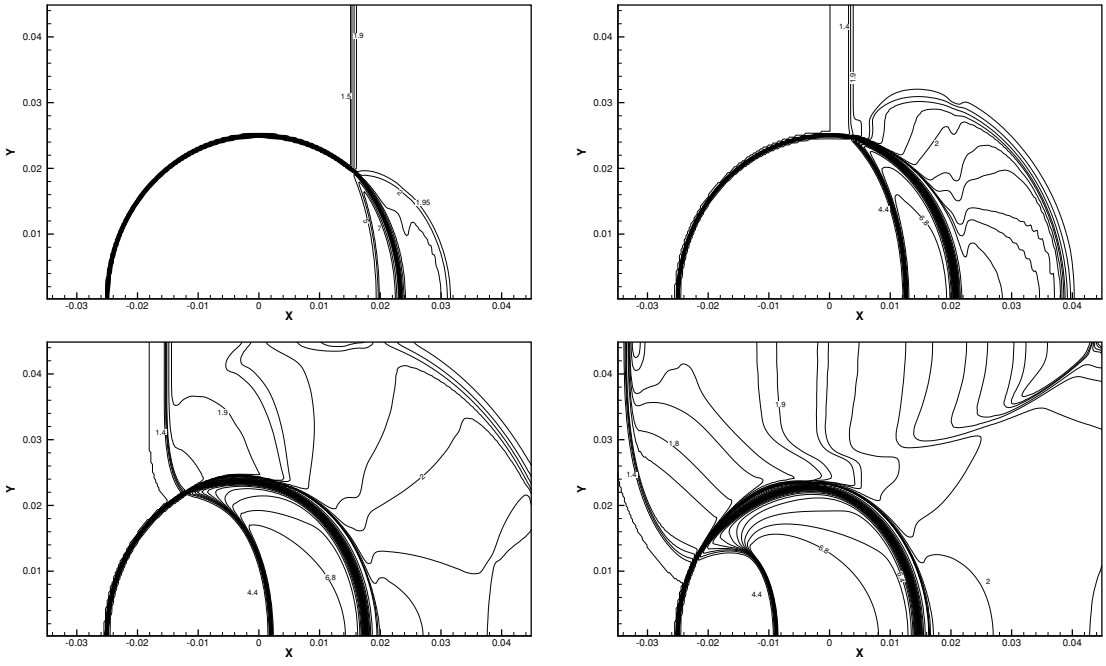

Figure 5.9: Shock hitting R22 bubble. Density at $t = 10 \times 10^{-3}$, $t = 20 \times 10^{-3}$, $t = 35 \times 10^{-3}$ and $t = 50 \times 10^{-3}$. Grid has 300×150 cells and $\Delta t = 2.5 \times 10^{-5}$.

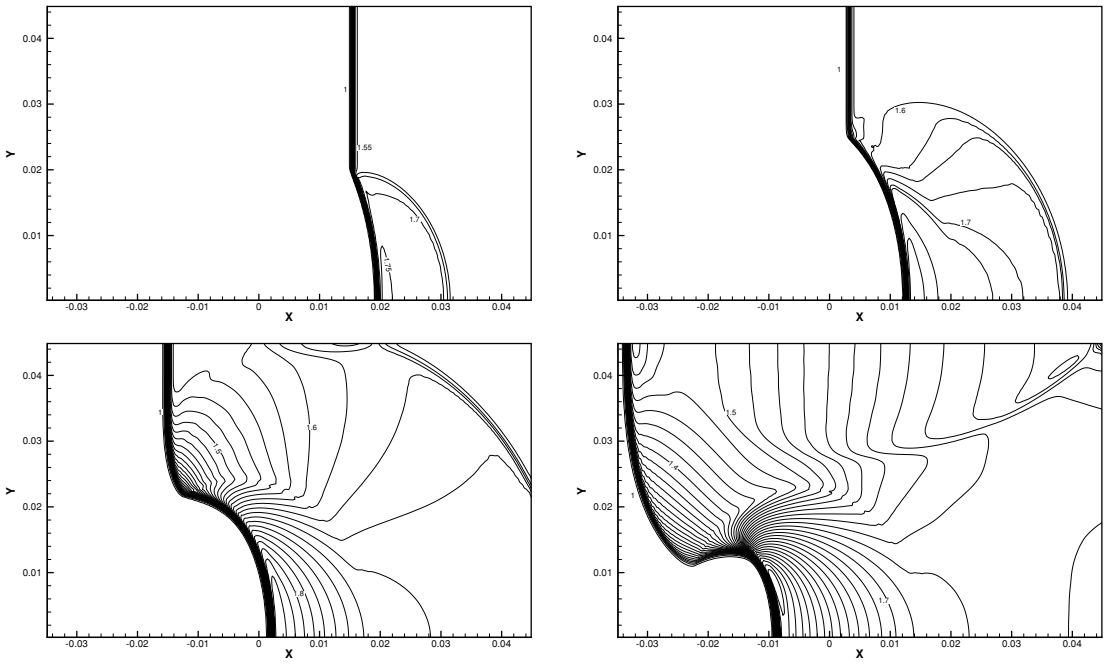

Figure 5.10: Shock hitting R22 bubble. Pressure at $t = 10 \times 10^{-3}$, $t = 20 \times 10^{-3}$, $t = 35 \times 10^{-3}$ and $t = 50 \times 10^{-3}$. Grid has 300×150 cells and $\Delta t = 2.5 \times 10^{-5}$.

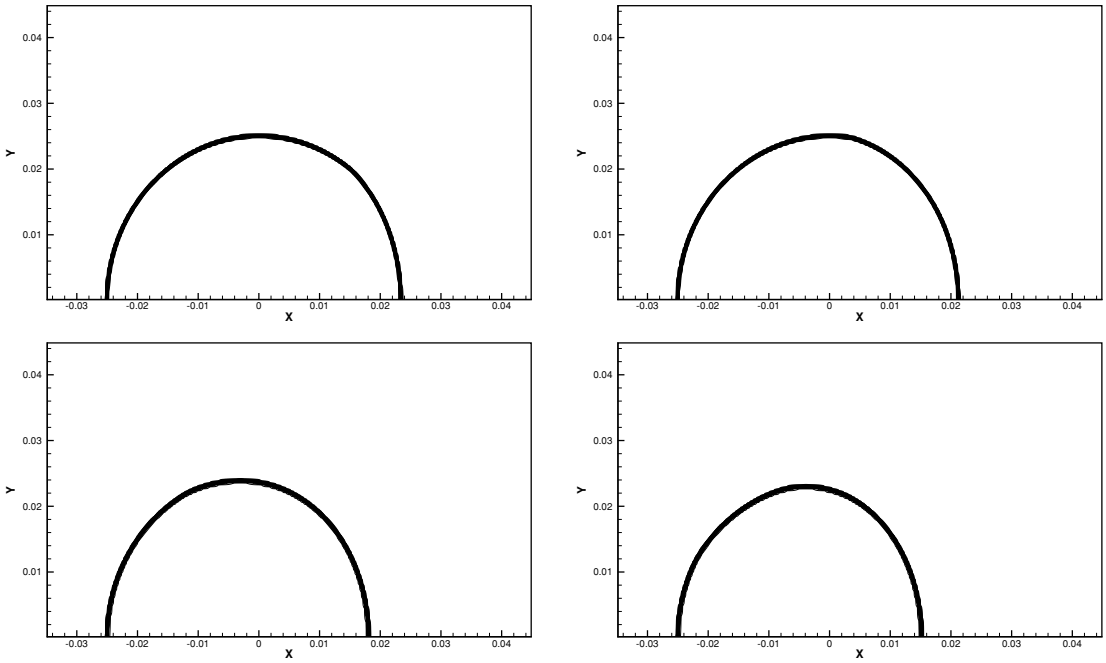

Figure 5.11: Shock hitting R22 bubble. Ratio of specific heats at $t = 10 \times 10^{-3}$, $t = 20 \times 10^{-3}$, $t = 35 \times 10^{-3}$ and $t = 50 \times 10^{-3}$. Grid has 300×150 cells and $\Delta t = 2.5 \times 10^{-5}$.

5.3. 2D Kelvin-Helmholtz instability

To investigate the possibilities of the level-set method further 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 disturbance in the interface will roll up in a characteristic vortex pattern. Because this vortex pattern will result in large distortions of the interface and possibly in topology changes of the interface the level-set approach is specifically useful since it allows for both.

The test case performed here is taken from [24]. 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.

5.3.1 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 $a = 1$). The computational domain has dimensions 1×2 in respectively x and y directions. The grid has 100×200 cells. The initial interface perturbation has amplitude 1.

The numerical results are depicted in figure 5.12. Clearly visible is the interface disturbance that grows in time. The final figure shows the topological change in the level-set contours. Some contours start to merge together while others split. Clearly the level-set method has no difficulties capturing these phenomena. Especially interesting is the behavior of the other level-set contours. Although the level-set function distorts due to the non-homogeneous flow field, the level-set contours that are shown remain real distances from the interface. This is solely the result of the redistancing procedure.

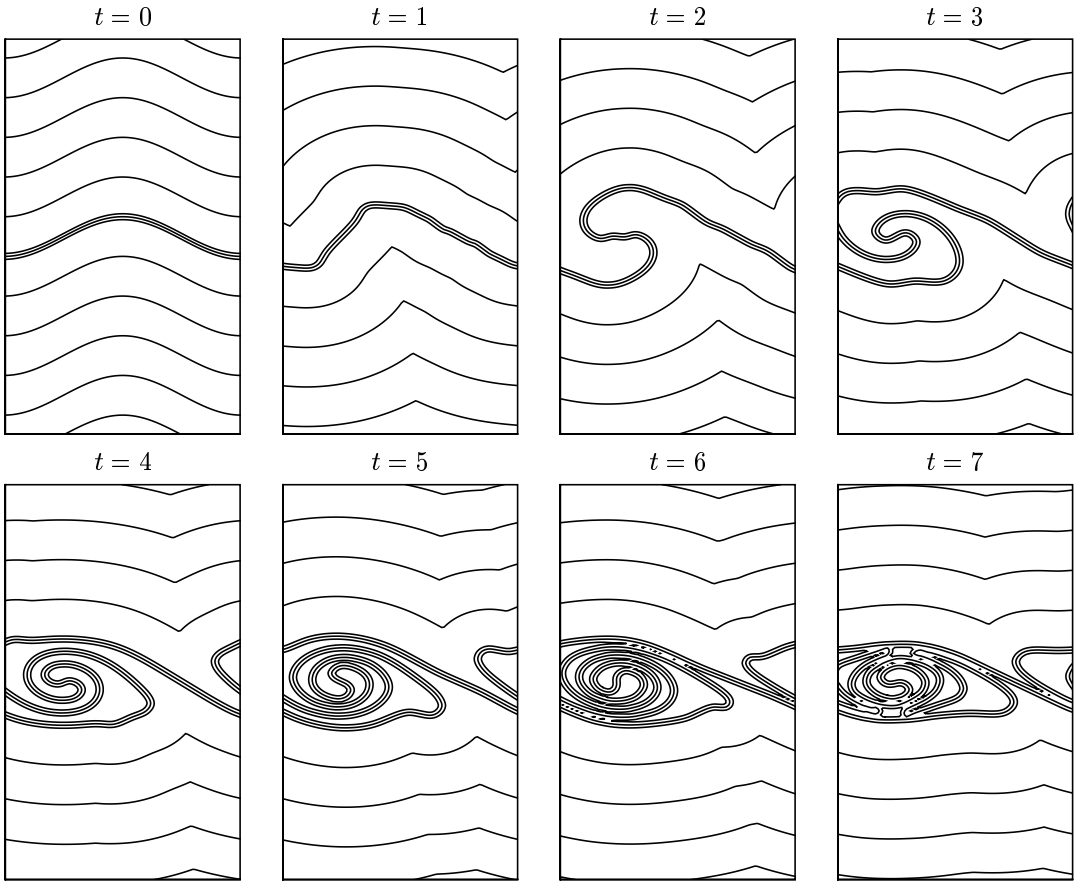

Figure 5.12: Kelvin-Helmholtz instability for air-air interface. Level-set function at different time levels. The three dense level-set contours are respectively $\phi = -\frac{3\Delta y}{2}, 0, \frac{3\Delta y}{2}$. Grid has 100×200 cells and $\Delta t = 1 \times 10^{-3}$.

5.3.2 Air-helium

In this test case the fluid above the interface is helium and the fluid below is air (heavier than helium)[1]. The properties of both air and helium are taken equal to the values used in the shock-bubble interaction problem (table 5.5). The helium used is thus contaminated with 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 used is the same as for the air-air case.

Figure 5.13 shows the numerical results. Depicted are the contours of the level-set function. The behavior of the interface clearly behaves different from 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 is comparable to waves in the ocean, with the difference that the fluid on top is air and the heavier fluid below is water.

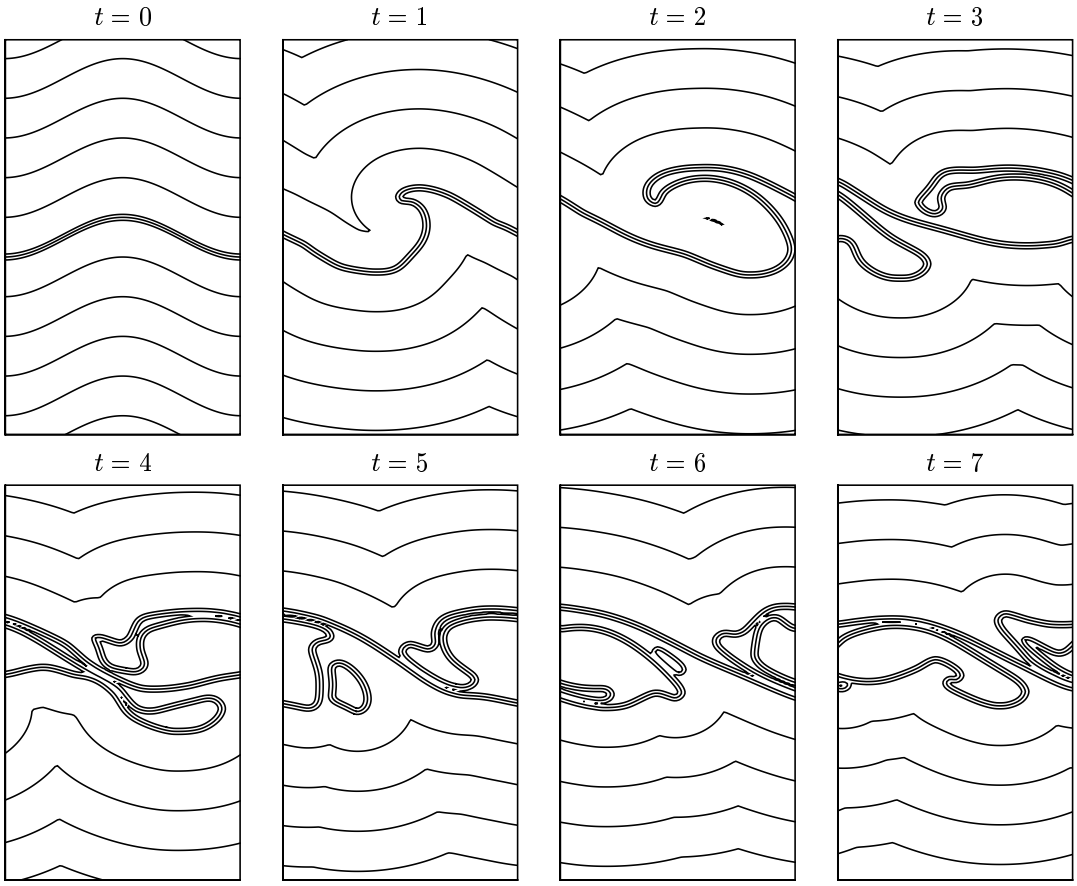

Figure 5.13: Kelvin-Helmholtz instability for air-helium interface. Level-set function at different time levels. The three dense level-set contours are respectively $\phi = -\frac{3\Delta y}{2}, 0, \frac{3\Delta y}{2}$. Grid has 100×200 cells and $\Delta t = 1 \times 10^{-3}$.

[1] Although we refer here to the difference in weight between air and helium, the Euler equations used contain no gravity terms. The differences between the air-air and air-helium cases are thus not due to the differences in weights but due to the differences in density and ratio of specific heats.

5.4. SUPERSONIC FREE JET

As a final two-dimensional test case the well-known supersonic free jet is treated. 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. 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). 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 trying 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 [7].

The initial conditions for the stagnant ambient air are taken as follows: $(\rho, u, v, p, \gamma) = (1.4, 0, 0, 1, 1.4)$. The conditions of the gas in the jet depend on the test case considered; for the underexpanded case the pressure is twice the ambient pressure and for the overexpanded case the pressure is half the ambient pressure. The density and the ratio of specific heats are the same for both the jet and the stagnant air. This makes that the Mach number in the underexpanded jet when exiting the nozzle is equal to $M = \sqrt{2}$ and the Mach number in the overexpanded jet $M = 2\sqrt{2}$.

The computational domain is chosen such that several periods of the jet are visible. Taking the domain equal to 3×1 in respectively x and y-directions and the width of the underexpanded jet to $d_{jet} = 0.25$ and that of the overexpanded jet to $d_{jet} = 0.42$ means that roughly two and a half period of both the underexpanded jet and the overexpanded jet will be captured. The grid is made-up of 120×40 cells in respectively x and y-directions (doing the same calculations on a finer grid is one of the future research interests).

5.4.1 Underexpanded jet

The results of the underexpanded supersonic jet are depicted in figure 5.14. The figure shows plots of the velocity, density, pressure and level-set function at 20 time units after the start of the calculations. Neglecting minor changes due to numerical errors, the jet can reasonably be assumed to be steady at this time level.

Clearly visible is the characteristic jet pattern occurring when an underexpanded jet enters in a stagnant gas. Because the pressure in the jet is higher than the ambient pressure two expansion fans originate from the nozzle exit (at respectively $y = \pm\frac{1}{8}$) 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). Because the compression fans (if the fans of one compression fan meet they form a shock) hit the jet boundary in this region, the boundaries curve outwards again, starting the process all over again.

The boundary remains very sharp during the simulation. Although not clearly visible in the plots, the jet boundary will lose its sine-shape further downstream of the nozzle.

5.4.2 Overexpanded jet

In figure 5.15 the results of the overexpanded jet are shown. Because the ambient pressure is now 2 times larger than the jet pressure two shocks originate from the nozzle exit to compress the jet to the outside pressure. When these shocks meet they deflect each other (sometimes forming compression fans). The jet boundary is initially *sucked* inside but starts to deflect towards the outside when the shocks hit 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*. Note however that in time the distinct diamond shape disappears since the sharp shocks make place for less sharp compression fans (only half of the first period is really diamond-shaped).

Clearly visible is the evolution of the jet further away from the nozzle. The first jet period still shows a sharp inward and outward curvature while later periods show a more *damped* behavior. The regions of lower and higher pressure and density become smaller and less severe until they finally damp out completely leaving a straight jet.

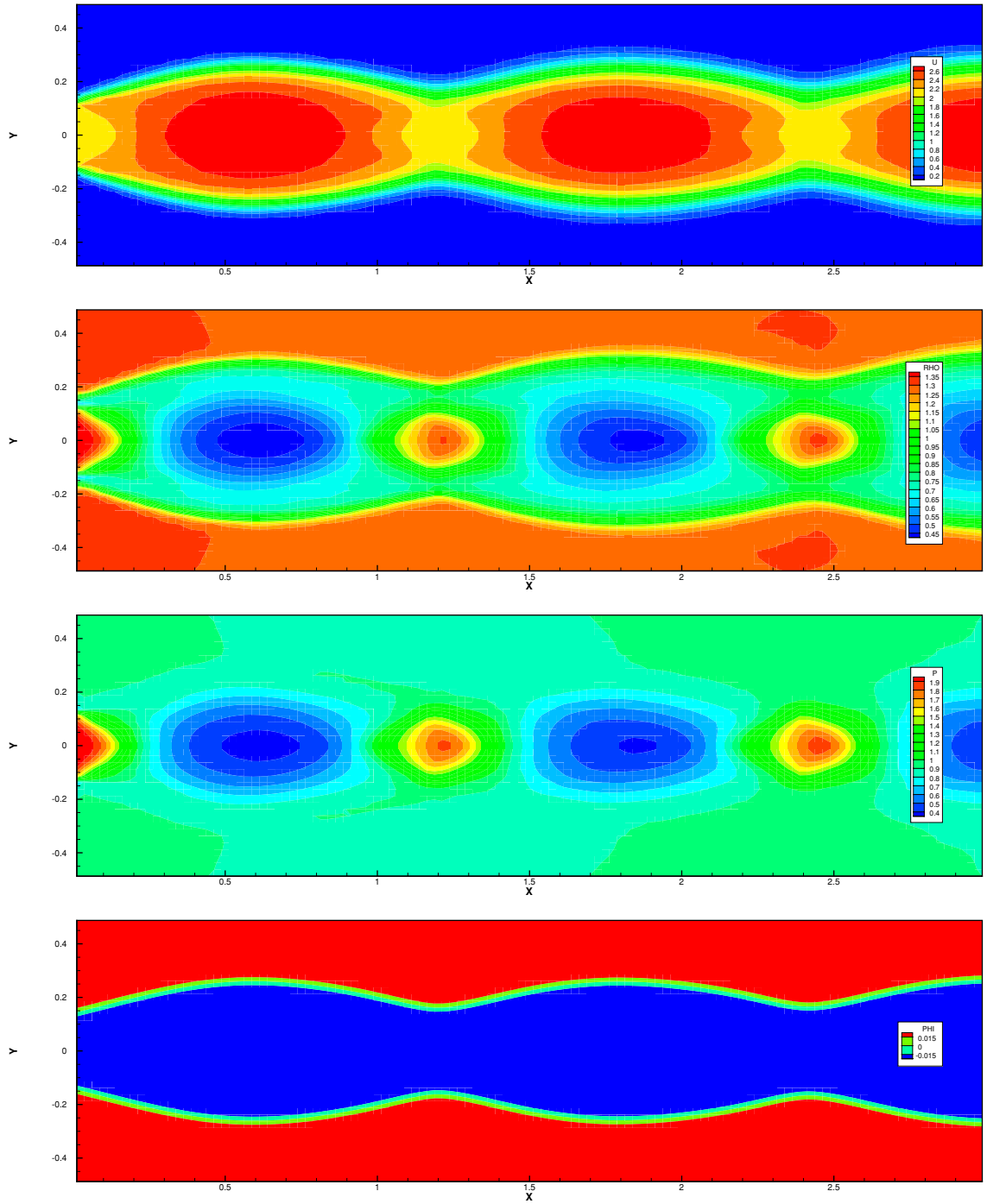

Figure 5.14: Supersonic underexpanded jet. *From top to bottom:* velocity, density, pressure and level-set function. Pressure in jet is 2 times larger than the ambient pressure. Grid has 120×40 cells and $\Delta t = 5 \times 10^{-4}$.

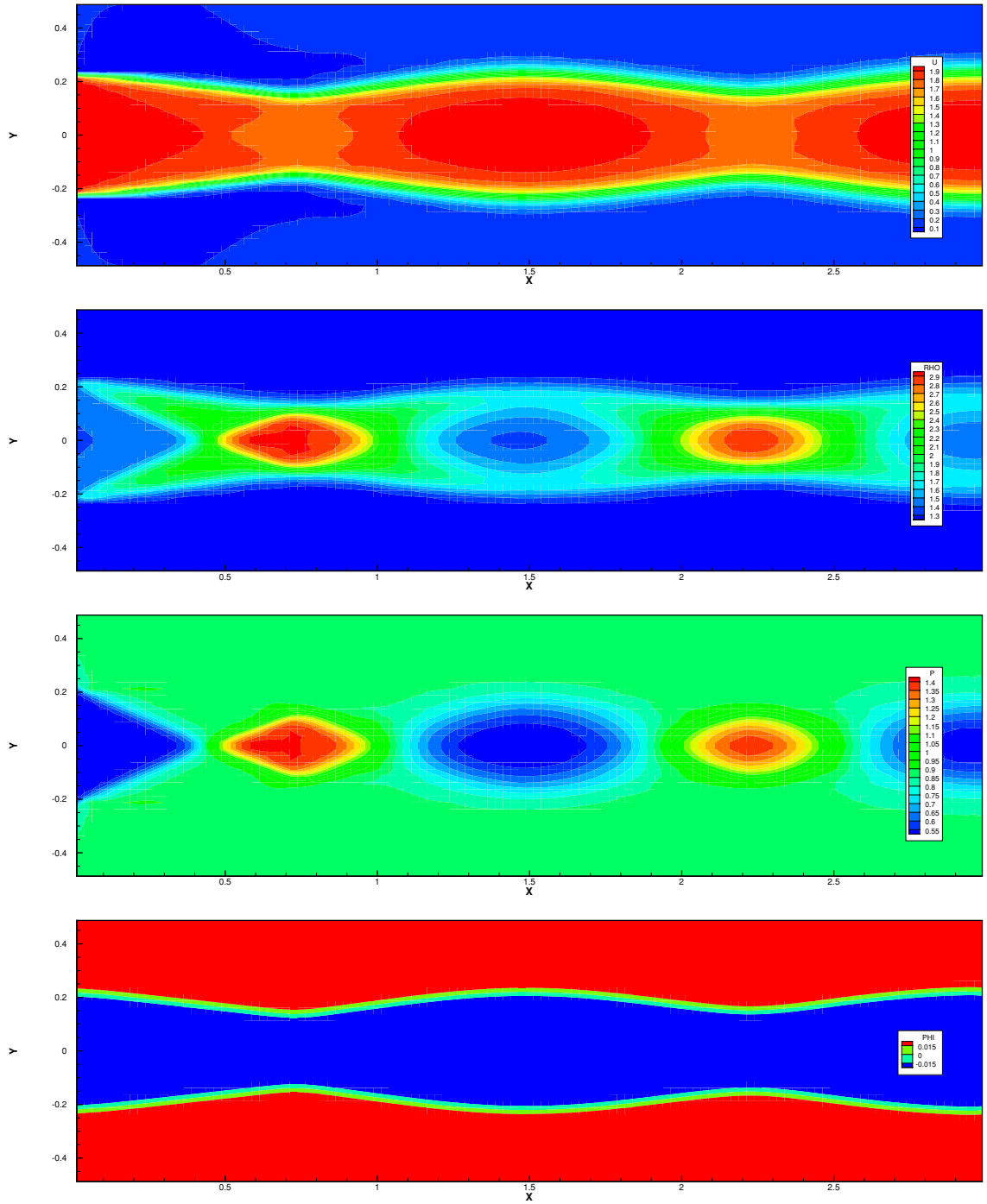

Figure 5.15: Supersonic overexpanded jet. *From top to bottom:* velocity, density, pressure and level-set function. Pressure in jet is 2 times smaller than the ambient pressure. Grid has 120×40 cells and $\Delta t = 5 \times 10^{-4}$.

Chapter 6
Conclusions

6.1. Current work

This report treats the development of a numerical solver for the simulation of flows consisting of two non-mixing fluids described by the two-dimensional Euler equations. A level-set equation in conservative form is added to describe the evolution of the interface. To prevent the level-set function from distorting a redistancing procedure is applied. This redistancing is done using a PDE-method which transforms the distorted level-set 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. A characteristics analysis resulted in the eigenvalues and eigenvectors of the flow model.

The flow equations are implemented in a numerical flow solver using a finite-volume approximation. A three-stage time marching scheme is used together with the approximate Riemann solver of Roe. Quadratic subcell interpolation is obtained using a limiter proposed by Koren. The combination of time and space discretization is theoretically third-order accurate but due to several simplifications this order will never be reached in reality, making the method effectively second-order accurate. The redistancing equation is discretized using a two-stage time-marching scheme and second-order accurate spatial interpolation.

The pressure oscillations present in all conservative level-set methods are discussed. Their origin is determined and a simple fix is treated that removes these spurious oscillations from the solutions. Although this fix makes the numerical method locally non-conservative the resulting errors are negligible compared to the previous pressure oscillations.

Several numerical tests are performed to test the solver for its performance. Standard one-dimensional shock-tube problems prove the existence of the pressure oscillations and confirm the proper working of the simple fix. Although not visible in the density, pressure and velocity distributions, a numerical error is present due to the conservative approach used for the level-set equation. A convergence test verifies the numerical order of the scheme. Two-dimensional tests are performed using the shock-bubble interaction problem, the Kelvin-Helmholtz instability and the supersonic free jet. All results indicate the excellent performance of the level-set method.

6.2. Future work

The work done acts as a starting point for further research in the area of level-set methods and two-fluid simulation. Interesting future research can focus on the following subjects:

Analysis of error due to conservative level-set approach. The one-dimensional test results with the conservative level-set approach show a numerical error due to the dependence of the level-set function on the distribution of the density. Further investigation of this error will be done to improve the available methods. A method that has been proven to solve the problem is to treat the level-set equation separate from the Euler equations.

Integrated redistancing. The redistancing of the level-set function is done by numerically solving a separate equation for the level-set function and its gradient. This approach seems to work fine but there is reason to believe that there is a more efficient approach. Including a redistancing term into the level-set equation is a possible option.

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.

Adaptive mesh refinement. To reduce the effect of the level-set method locally being 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.

Level-set for discontinuous Galerkin Especially interesting is the application of level-set methods to discontinuous Galerkin (finite-element) methods. Since DG methods use, besides information about the average value of the state variables in a cell, also information about the gradient of the state variables, they are specifically useful for level-set methods. Using the conservation equation for the gradient of the level-set function will automatically include the redistancing procedure in the method, without requiring a separate approach.

Level-set grid generation for complex geometries. The level-set method seems especially interesting for generating grids for complex geometries. Following the work of Sethian in [32] this approach can be used in future work.

Extension of numerical treatment of supersonic jet The supersonic free jet is an excellent problem where the analytical theories come short for a complete understanding of the gas dynamical processes taking place (the complex non-simple regions have no known exact solution). Numerical solutions provide therefore a more than useful tool for understanding these processes. Especially the jets involving multiple fluids are an interesting starting point for further research.

References

1. R. Abgrall and S. Karni. Computations of compressible multifluids. *J. Comp. Phys.* 169, pages 594-623, 2001.
2. R. Abgrall. How to prevent pressure oscillations in multicomponent flow calculations: A quasi conservative approach. *J. Comp. Phys.* 125, pages 150-160, 1996.
3. P.G. Bakker. Lecture Notes on Gasdynamics, AE4-140. Faculty of Aerospace Engineering, Delft University of Technology, 2001.
4. D.J. Benson. Computational methods for Lagrangian and Eulerian Hydroflow. *CMAME* 99, pages 235-394, 1992.
5. 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.
6. R.L. Burden and J.D. Faires. Numerical Analysis, 7^{th} edition. Brooks/Cole, 2001.
7. R. Courant and K.O. Friedrichs. Supersonic Flow and Shock Waves. Springer Verlag, New York, 1976.
8. 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.
9. S. Osher and F.P. Fedkiw. Level Set Methods and Dynamic Implicit Surfaces. Applied Mathematical Sciences 152, Springer-Verlag, 2002.
10. 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.
11. 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.
12. 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.
13. 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.
14. S. Karni. Multicomponent flow calculations by a consistent primitive algorithm. *J. Comp. Phys.* 112, pages 31-43, 1994.
15. S. Karni. Hybrid multifluid algorithms. *SIAM J. Sci. Comp.* 17, pages 1019-1039, 1996.
16. B. Koren. Multigrid and Defect Correction for the Steady Navier-Stokes Equations, Application to Aerodynamics, *CWI Tracts* 74, CWI, Amsterdam, 1991.
17. 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.
18. 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.
19. B. Koren and A.C.J. Venis. A fed back level-set method for moving material-void interfaces. *J. Comp. Appl. Math.* 101, pages 131-152, 1999.
20. B. van Leer. Computer Problem 2, version 1: Hypersonic flow over a cavity with fuel injection, A 2D Euler

code from 1D building blocks. Department of Aerospace Engineering, University of Michigan, 2003.

21. B. van Leer. Computer Problem 2, version 4: Incompressible free-surface flow over a bump. Department of Aerospace Engineering, University of Michigan, 2004.
22. 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.
23. M. Metcalf and J. Reid. FORTRAN 90/95 explained, 2^{nd} edition. Oxford University Press, 1999.
24. W. Mulder, S. Osher and J.A. Sethian. Computing interface motion in compressible gas dynamics. *J. Comp. Phys.* 100, pages 209-228, 1992.
25. J. Naber. Building your own shock tube. Report MAS-E0502, CWI, http://ftp.cwi.nl/CWIreports/MAS/MAS-E0502.pdf, Amsterdam, 2005.
26. S. Osher and J.A. Sethian. Fronts propagating with curvature-dependent speed: algorithms based on Hamilton-Jacobi formulations. *J. Comp. Phys.* 79(1), pages 12-49, 1988.
27. 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.
28. P.L. Roe. Approximate Riemann solvers, parameter vectors, and difference schemes. *J. Comp. Phys.* 43, pages 357-372, 1981.
29. A. Rohde. Eigenvalues and eigenvectors of the Euler equations in general geometries. AIAA Paper 2001-2609, American Institute of Aeronautics and Astronautics, 2003.
30. R. Saurel and R. Abgrall. A multiphase Godunov method for compressible multifluid and multiphase flows. *J. Comp. Phys.* 150, pages 425-467, 1999.
31. 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.
32. J.A. Sethian. Curvature flow and entropy conditions applied to grid generation. *J. Comp. Phys.* 115, pages 440-454, 1994.
33. 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.
34. 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(4), pages 1165-1191, 1999.
35. E.F. Toro. Riemann Solvers and Numerical Methods for Fluid Dynamics, A Practical Introduction, 2^{nd} edition. Springer-Verlag, Berlin, 1999.
36. 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.
37. J. Wackers and B. Koren. A fully conservative model for compressible two-fluid flow. *Int. J. Numer. Meth. Fluids* (to appear).
38. P. Woodward and P. Colella. The numerical simulation of two-dimensional fluid flow with strong shocks. Review article, *J. Comp. Phys.* 54, pages 115-173, 1984.

Appendix I
Density smoothing for shock-bubble interaction

The argument for the smoothing of the two-fluid interface was to prevent *staircasing* of the interface. 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 I.1).

This staircasing means that the location of the interface, and thus the local values of the ratio of specific heats, are used incorrectly when solving the Riemann problems on the cell faces. This can lead to numerical errors, which in turn can result in spurious oscillations of the interface.

The problem sketched above is not only restricted to the level-set function but also to the distribution of the other flow variables. Because the interface often separates two different fluids with different densities, the same problem can occur for the distribution of the density. In time the numerical errors can produce oscillations in the density distribution near the interface.

Unfortunately it is not possible to smooth the density distribution after each time update because the density in the flow almost never, besides maybe in some special situations, has two distinct values (there are no ρ_L and ρ_R). Except in the initial flow distribution! The bubble in the shock-bubble interaction problem has one density and the air has another. Smoothing of the density interface can thus be applied here. Using the smooth-step function H_ϵ given by (2.11) the density jump over the bubble can be smeared-out over several cells, thus preventing staircasing.

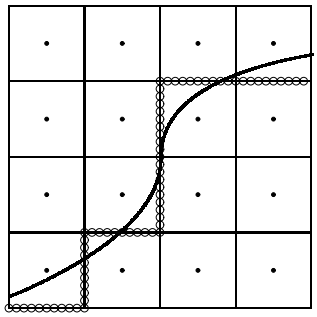

Figure I.1: Grid cells containing an interface separating two fluids (solid line). The interface is *pushed* to the cell faces (dotted line) leading to *staircasing* of the interface. This staircasing leads to numerical errors which can induce spurious oscillations in the state variable distributions.

Figure I.2 shows the results of computations with a non-smoothed helium bubble and a smoothed version. Although the density distribution is smeared-out over more grid cells than was the case without smoothing, resulting in a somewhat broader density interface, the spurious oscillations are not visible when smoothing is applied. This proves the use of this simple method.

Not only in the case of the shock-bubble interaction but every time an initial jump in a flow variable distribution is present this technique can be applied.

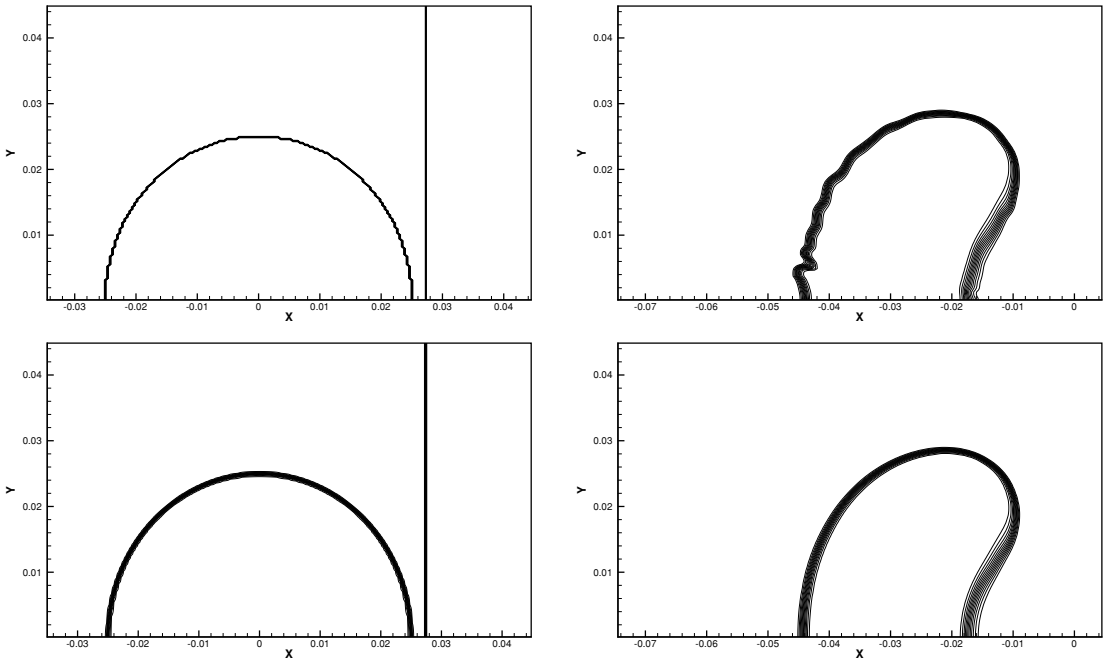

Figure I.2: Shock hitting helium bubble. Density at $t = 0$ and $t = 75 \times 10^{-3}$. Grid has 300×150 cells and $\Delta t = 1.25 \times 10^{-5}$. *Top:* Non-smoothed case. *Bottom:* Smoothed case.