Centrum voor Wiskunde en Informatica

REPORT*RAPPORT*

MAS

Modelling, Analysis and Simulation

Modelling, Analysis and Simulation

Improved capturing of contact discontinuities for two-fluid flows

G.F. Duivesteijn

NOTE MAS-N0202 DECEMBER 31, 2002

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

Improved Capturing of Contact Discontinuities for Two-Fluid Flows

G.F. Duivesteijn
CWI
P.O. Box 94079, 1090 GB Amsterdam, The Netherlands
(July 2000 – September 2002)

ABSTRACT

Past years, there has been much research in extending and applying approximate Riemann solvers to immiscible two-fluid flows, more and more often in combination with a level-set technique to improve the resolution of the interface(s) between the two fluids. The interfaces are contact discontinuities. Accurately capturing contact discontinuities is harder than capturing shock waves, because of the lack of a steepening mechanism. Moreover, a known difficulty in computing flows with contact discontinuities in the usual conservative formulation is that zeroth-order pressure errors may arise. (This interesting numerical feature is fully understood; it appears to be independent of the monotonicity and accuracy properties of the time and space discretization.) Several remedies against the pressure errors have been proposed already. This MSc work consists of computations of a two-fluid interface moving in a tube: first without pressure-oscillation fix (to see how serious the pressure oscillations really are) and next, with fixes. The used approximate methods are the ghost-fluid method and a new method, called the mass-fraction method. Also, an exact two-fluid Riemann solver has been derived and implemented in a software program, called "Visual Shock Tube Solver".

2000 Mathematics Subject Classification: 65M60, 76N99, 76T10.
1998 ACM Computing Classification System: D.1.5, D.2.2.
Keywords and Phrases: two-fluid flows, level-set, ghost-fluid, mass-fraction, two-fluid Riemann solver, visual shock tube solver.
Note: This research was carried out under CWI-project MAS2.1 "Computational Fluid Dynamics".

Table of Contents

List of Figures

Chapter 1
Introduction

Past years, there has been much research in extending and applying approximate Riemann solvers to immiscible two-fluid flows, more and more often in combination with a level-set technique to improve the resolution of the interface(s) between the two fluids. The interfaces are contact discontinuities. Accurately capturing contact discontinuities is harder than capturing shock waves, because of the lack of a steepening mechanism. Moreover, a known difficulty in computing flows with contact discontinuities in the usual conservative formulation is that zeroth-order pressure errors may arise. (This interesting numerical feature is fully understood; it appears to be independent of the monotonicity and accuracy properties of the time and space discretization.) Several remedies against the pressure errors have been proposed already. Chapter 2 of this MSc-work consists of computations of a two-fluid interface moving in a tube: first without pressure-oscillation fix (to see how serious the pressure oscillations really are) and next, with fixes.

All fixes have in common that near the contact discontinuity, they somehow neglect the conservation requirements, e.g., by locally switching (near the interface) to a convection formulation. At contact discontinuities, this is allowed; contact discontinuities are convection phenomena. However, if a shock wave hits the contact discontinuity, a conservative discretization is required again. Then, fixes such as those proposed in the referenced papers may no longer be applied in a straightforward manner without making large errors.

Recently, it has been discovered that one can use a conservative two-fluid formulation throughout the entire flow field without suffering from zeroth-order pressure errors near and at the contact discontinuity. (It has been proven that the statement from [1] that "any Godunov-type scheme which is fully conservative will not be able to maintain pressure equilibrium and will develop a pressure oscillation across material fronts", is not correct!) Chapter 3 goes deeper in this new conservative method.

The two-fluid capturing schemes discussed in this MSc-thesis require two-fluid flux algorithms. For this purpose, a two-fluid exact and an approximate Riemann solver have been derived and tested. The numerical two-fluid flow solver is called Linearized Godunov scheme (section 2.3). Simple 1D two-fluid flow problems can be treated as series of Riemann problems, including complicated flow patterns like shock and contact-discontinuity interaction. This method is discussed and tested in chapter 4.

An overview of methods, used for this MSc-work, is given in figure 1.1.

The multi-fluid computation techniques are applicable to a large area of flow problems, like ship hydrodynamics, multi-fluid tube flows, shallow under-water blast waves, exhaust flow from a nozzle, dam burst, under-water rocket launch, etc. The titlepage shows a photo made at the

BAKER test at Bikini in July 1946. In this test, a nuclear weapon of approximately 20-kilotons was detonated well below the surface of the lagoon which was about 200 feet deep. These conditions may be regarded as corresponding to a shallow underwater explosion.

1.1. Free surface flows

In order to compute multi-material flows, two basic approaches are available, known as Lagrangean schemes and Eulerian schemes. With Lagrangean schemes, the grid is attached to the free surface. The grid deforms with the movement of the surface. Lagrangean schemes work well on interfaces, since they do not blur the interface. With Eulerian schemes, the grid is fixed. The surface is usually not aligned with the grid.

1.1.1 Moving grid (Lagrangean schemes)

The position of the material interface can be described by a height function, in which height is a function of position and time. One can choose to compute the bulk-flow and height of the interface simultaneously, or to execute the bulk-flow computation separately from the interface height computation. The latter choice has the advantage of simplicity. The amount of initial conditions and boundary conditions on the free surface is equal to the amount of conditions on a fixed boundary, augmented by the conditions for the height function. For example, in case of a combined aerodynamic-hydrodynamic problem, possible augmented boundary conditions are zero mass transport through the free surface and zero stress at the free surface.

One of the advantages of this method is that the free surface is always represented sharply. However, accuracy of the location and topology of the free surface should be questioned. The moving grid method is only suitable for flows with a low degree of deformation of the free surface and is not capable of modelling bifurcations. The height function cannot be multi-valued. Even for strongly distorted free surfaces, which are non-bifurcating, the method may be unsuited as well.

1.1.2 Fixed grid (Eulerian schemes)

1.1.2.1 Tracking versus Capturing Fixed grid techniques can be applied in two different ways. One method is to apply boundary conditions on the free surface. This method is called tracking. But applying conditions on the free surface, limits the freedom of this surface and is a contradiction in terms. The other method is to let the free boundary truly free by not imposing any boundary conditions on it. This is called capturing. For capturing, it is necessary to use a two-fluid flux formula at cell face level. A problem of capturing is the possible occurrence of large errors of the state variables near the interface. This is known as "pressure oscillations" and does not occur with tracking and fitting methods. In literature, some remedies and fixes are proposed to overcome the pressure oscillations. In this thesis, the capturing technique will be used.

1.1.2.2 Marker And Cell To locate the interface, three well known methods and one brandnew method are available. A classical method is called the Marker And Cell (MAC) method. In order to distinguish the two fluids, one of the media is represented by massless particles. These marker particles are only introduced for the purpose of indicating the flow topology. They move according to the velocity components in their vicinities and show which cells contain the marked medium and especially, which cells lie along the free surface. Flow bifurcations are not a problem with this method. But one has to be aware of not using too few particles in order to avoid numerical cavitation. When cavitation occurs, the computation has to be done with more particles in order to distinguish the difference between physical cavitation and numerical cavitation. The MAC is known to be very computation intensive, especially for 3D flow problems, involving a large amount of particles.

1.1.2.3 Volume Of Fluid Another method for distinguishing the two media is by defining a function F whose value is one at any point occupied by medium one and zero at any point occupied by medium two. The average value of F in a cell would then represent the fractional

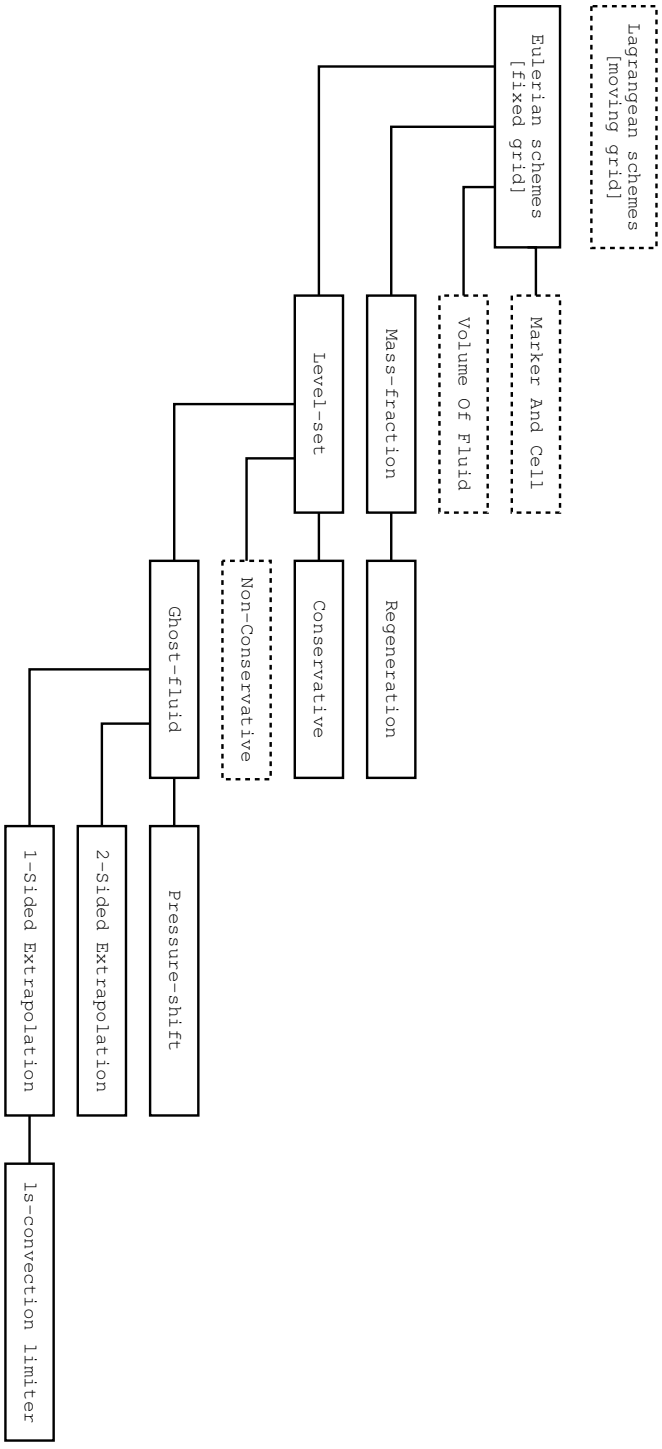

Figure 1.1: Method scheme, showing the relations between the different methods, fixes and limiters. Methods represented with a solid-line box have been derived and/or tested in this MSc-work.

volume of a cell occupied by medium one. For $0 < F < 1$, the cell contains a free surface. The value F is transported by the convection equation

$$\frac{\partial F}{\partial t} + u\frac{\partial F}{\partial x} = 0.$$

The fact that F is a step function, causes many discretization methods to smear the F function and interfaces to lose their definition. The Volume-of-Fluid method is less computation intensive. Only one scalar per cell has to be stored and updated, instead of a large quantity of particles. The method is very capable of computing bifurcations and cavitation.

1.1.2.4 Level-set A logical sequel is to define a smooth function ϕ as a replacement of the VOF step function F, for instance a signed distance function. This method is known as the level-set method. The free surface is now defined as the zero of the distance function. The value ϕ is transported by the convection equation

$$\frac{\partial \phi}{\partial t} + u\frac{\partial \phi}{\partial x} = 0.$$

In contrast to the VOF, differencing the smooth level-set function is not a problem anymore. However, in order to ensure accuracy of the zero levelset location (and with that the free surface location), the level-set function should remain to be a signed distance function and the slope of the level-set function should not be too steep or too flat. During computation, the function may need to be reinitialized. As with the VOF method, the level-set method does a good job with bifurcating flow problems and physical cavitation. Chapter 2 gives a detailed description and some improvements of the method, completed with numerical results.

1.1.2.5 Mass-fraction The mass-fraction method uses a different approach. Instead of a Volume-of-Fluid fraction or distance to the free surface function, the mass fraction β of medium one is stored. The mass-fraction of medium two is defined as $(1 - \beta)$. The major advantage of this method is that it is proven to be conservative in all circumstances, in contrast to aforementioned free-surface location methods. But the prize for that is the loss of a crisp interface. The interface location smears out during computation. Chapter 3 gives a description of the method, a new post process to improve the free surface resolution and some numerical results.

Chapter 2
Numerical two-fluid flow calculation with use of the level-set approach

2.1. Introduction

In this chapter, some numerical calculation methods will be discussed for computation of immiscible two-fluid flows. The methods will be illustrated by test cases and their numerical results. All the described methods are based on the Euler equations, which model the dynamics of compressible, inviscid fluids. These are non-linear hyperbolic partial differential equations, describing the conservation of mass, momentum and, if necessary, energy. If the fluid consists of several components, species equations are added which describe the conservation of the species.

2.2. Euler equations

In conservative differential form, the Euler equations reads

$$\frac{\partial q}{\partial t} + \frac{\partial f_1(q)}{\partial x} + \frac{\partial f_2(q)}{\partial y} + \frac{\partial f_3(q)}{\partial z} = g(x), \tag{2.2.1}$$

with

$$f_1(q) = \begin{pmatrix} \rho u \\ \rho u^2 + p \\ \rho uv \\ \rho uw \\ \rho u\left(e + \frac{p}{\rho}\right) \end{pmatrix}, \quad f_2(q) = \begin{pmatrix} \rho v \\ \rho uv \\ \rho v^2 + p \\ \rho vw \\ \rho v\left(e + \frac{p}{\rho}\right) \end{pmatrix}, \quad f_3(q) = \begin{pmatrix} \rho w \\ \rho uw \\ \rho vw \\ \rho w^2 + p \\ \rho w\left(e + \frac{p}{\rho}\right) \end{pmatrix}, \tag{2.2.2}$$

and

$$q = \begin{pmatrix} \rho \\ \rho u \\ \rho v \\ \rho w \\ \rho e \end{pmatrix}, \tag{2.2.3}$$

and g a possible source term, describing, e.g., the influence of gravity. The total energy e is given by

$$e = \frac{1}{\gamma - 1}\frac{p}{\rho} + \frac{1}{2}\left(u^2 + v^2 + w^2\right). \tag{2.2.4}$$

Here ρ, u, v, w, p and γ denote respectively density, velocity components in x-, y- and z-direction, static pressure and ratio of specific heats. The system of 5 equations has 6 unknowns. One additional equation is required. The density ρ can be written in terms of pressure p with, e.g., Tait's stiffened equation of state

$$\frac{p + Bp_{\text{ref}}}{p_{\text{ref}}(1+B)} = \left(\frac{\rho}{\rho_{\text{ref}}}\right)^{\gamma}. \tag{2.2.5}$$

where B is a constant depending on the type of fluid. The subscript ref refers to an arbitrary reference state. When substituting $B = 0$ in equation (2.2.5), the barotropic equation of state for a perfect gas is obtained

$$\frac{p}{p_{\text{ref}}} = \left(\frac{\rho}{\rho_{\text{ref}}}\right)^{\gamma}. \tag{2.2.6}$$

For a two-fluid mixture, the density in an arbitrairy volume is unknown due to the lack of knowledge of the material-interface location. The density ρ can be the density of fluid 1, fluid 2 or a volume-weighted average of the densities of fluid 1 and 2; the so-called compound or bulk density. In other words, the exact location of the material interface is needed in order to compute the density in a given volume. To specify the location of the free-surface, some different methods have been developed, like the so-called "Marker and Cell method" [10], "Volume-of-Fluid method" [13] and the "Level-set method" [26]. A brief description of these methods has been given in section 1.1.2. For the bulk density the following formula is introduced

$$\rho(\alpha, p) = \alpha\rho_{\text{I}}(p) + (1-\alpha)\,\rho_{\text{II}}, \qquad \alpha \in [0,1]. \tag{2.2.7}$$

The variable α represents the volume fraction which is filled with fluid 1. The determination of the value α, can be done by different kinds of techniques, like VOF or level-set techniques. In this chapter the main focus will be the use of level-set techniques.

The test models discussed in this document are restricted to one dimension in space, but there is no difference to problems with two or three dimensions. The conservation laws of mass and momentum for inviscid 1D flow of two perfect fluids may be written as

$$\frac{\partial q}{\partial t} + \frac{\partial f}{\partial x} = g(x), \tag{2.2.8a}$$

where

$$q = \begin{pmatrix} \rho \\ \rho u \end{pmatrix}, \quad f = \begin{pmatrix} \rho u \\ \rho u^2 + p \end{pmatrix}. \tag{2.2.8b}$$

The energy equation is not needed if pressure p is given by a barotropic equation of state. The bulk density ρ is determined according to (2.2.7) in terms of pressure p and Volume-of-Fluid fraction α. For the numerical tests in this chapter, we used the level-set method for specifying the interface location.

2.3. Linearized Godunov scheme

To determine the flux vectors across the cell faces, a flux-difference splitting scheme will be used. Due to the two-fluid character of the flow, i.e., large denstiy ratio ρ_2/ρ_1 at the interface, a robust flux-difference splitting scheme is desirable, derived from a Riemann problem. Besides good capturing of the contact discontinuity and shock waves, the Riemann-based approach is expected to yield robustness and a good boundary-condition treatment. For this reason, a first-order approximation of the full 1D Riemann problem is used to evaluate the flux across a cell face. The flux in or out of a cell is defined as the difference of the individual fluxes at the left cell face and the right cell face. A full derivation of the exact solution of the two-fluid Riemann problem is given in section 4.2. Regard a cell face $\partial\Omega_{i+\frac{1}{2}}$ separating two different states q_i and q_{i+1} in Ω_i and Ω_{i+1}, as seen in figure 2.1. Anticipating to chapter 4, q_i and q_{i+1} will be named q_1 and q_4 respectively. For

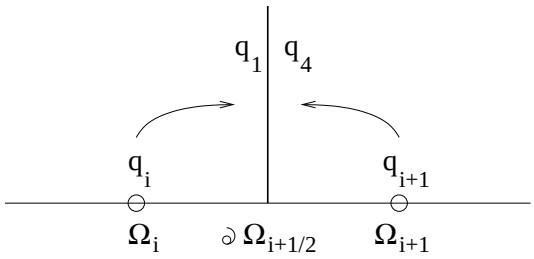

Figure 2.1: Situation at cell face $\partial\Omega_{i+\frac{1}{2}}$.

our first order flux difference splitting method, the states at nodes i and $i+1$ are directly copied on the cell face $\partial\Omega_{i+\frac{1}{2}}$, without any interpolation techniques. Recall the Riemann invariants $J^{\pm}$ in differential form [3]:

$$dJ^{+} = du + \frac{dp}{\rho a} = 0 \qquad \text{along } \Gamma^{+}, \tag{2.3.1}$$

$$dJ^{-} = du - \frac{dp}{\rho a} = 0 \qquad \text{along } \Gamma^{-}. \tag{2.3.2}$$

In non-homentropic flow without shock discontinuities, the entropy s remains constant when following a particle. However, s may differ for different fluid particles. Due to this non-uniform entropy the Riemann invariants cannot be integrated directly. Using the property that the Riemann invariants J^{+} and J^{-} stay constant along the characteristic Γ^{+} and Γ^{-} respectively, and linearizing (2.3.1) and (2.3.2) around state 1 and state 4 gives

$$u_2 - u_1 = \frac{p_1 - p_2}{\rho_1 a_1} \qquad \text{along } \Gamma_1^{+}, \tag{2.3.3}$$

$$u_3 - u_4 = -\frac{p_4 - p_3}{\rho_4 a_4} \qquad \text{along } \Gamma_4^{-}. \tag{2.3.4}$$

Note that equations (2.3.3) and (2.3.4) can be interpreted as the tangent of the Hugoniot - and/or Poisson curve in the (u,p)-plane at states 1 and 4. Adding and subtracting equations (2.3.3) and (2.3.4) yields

$$u_{2,3} = \frac{\rho_1 a_1 u_1 + \rho_4 a_4 u_4 - (p_4 - p_1)}{\rho_1 a_1 + \rho_4 a_4}, \tag{2.3.5}$$

$$p_{2,3} = \frac{\rho_4 a_4 p_1 + \rho_1 a_1 p_4 + \rho_1 a_1 \rho_4 a_4 (u_1 - u_4)}{\rho_1 a_1 + \rho_4 a_4}, \tag{2.3.6}$$

with $u_2 = u_3 = u_{2,3}$ and $p_2 = p_3 = p_{2,3}$. The given solutions $u_{2,3}$ and $p_{2,3}$ can be found at the

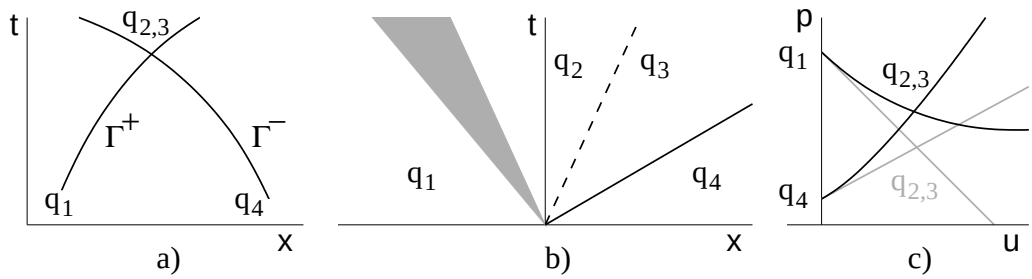

Figure 2.2: a) Characteristic curves in physical plane, b) Shock, contact discontinuity and expansion wave in physical plane, c) Hugoniot curve and Poisson curve intersecting at exact solution. The tangents of the two curves intersect at first order accurate solution.

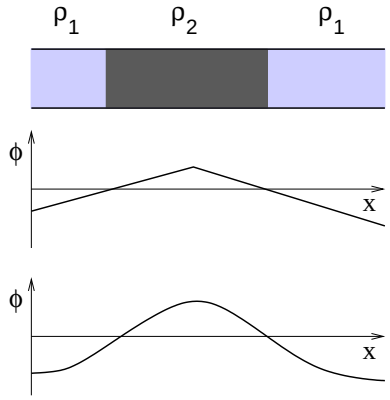

Figure 2.3: Possible choices for the level-set function

intersection of the tangents of the exact solution curves. This is shown in figure 2.2c. The flux vector $f_{i+\frac{1}{2}}$yields

$$\begin{aligned} f_{i+\frac{1}{2}} &= \begin{pmatrix} \rho_{i+\frac{1}{2}} u_{i+\frac{1}{2}} \\ \rho_{i+\frac{1}{2}} u^2_{i+\frac{1}{2}} + p_{i+\frac{1}{2}} \end{pmatrix} \\ &= \begin{pmatrix} \rho_{i+\frac{1}{2}} u_{2,3} \\ \rho_{2,3} u^2_{2,3} + p_{2,3} \end{pmatrix}. \end{aligned} \tag{2.3.7}$$

The density $\rho_{i+\frac{1}{2}}$ is given by, using equation of state (2.2.5):

$$\rho_{i+\frac{1}{2}} = \begin{cases} \alpha_1 \rho_{\mathrm{I}}(p_{2,3}) + (1-\alpha_1)\rho_{\mathrm{II}}(p_{2,3}) & \text{if } \mathrm{u}_{2,3} \geq 0 \\ \alpha_4 \rho_{\mathrm{I}}(p_{2,3}) + (1-\alpha_4)\rho_{\mathrm{II}}(p_{2,3}) & \text{if } \mathrm{u}_{2,3} \leq 0 \end{cases} \tag{2.3.8}$$

The flux for given control volume Ω_i is given by the difference of the two flux vectors at the cell faces:

$$\int_{\Omega_i} \frac{\partial f}{\partial x} dx = f_{i+\frac{1}{2}} - f_{i-\frac{1}{2}}. \tag{2.3.9}$$

Similar expressions can be found when linearizing the Osher scheme. For a detailed description of the Linearized Godunov scheme, the reader is invited to read [19].

2.4. Level-set method

An early level-set method for two-fluid flows was proposed by Mulder, Osher and Sethian [24] where, in 2D, a level-set function ϕ is used to capture the 1D interface. The two fluids are named medium 1 and medium 2, and state properties change discontinuously across the interface. So, within this method, mixture of fluids does not occur. Initialization of the level-set function is done by taking a function which is smooth at the interface and which is positive for medium 1 and negative for medium 2. The zero of the function indicates the location of the interface. A commonly used function is the so-called "signed-distance function". In case of a 1D problem, the signed-distance function is a linear function, which in case of a single interface is differentiable infinitely many times. In case of more interfaces, for instance the case with two interfaces sketched in figure 2.1, differentiability is lost, though not at the important interfaces. To still restore the uniform differentiability, another good choice for ϕ is an exponential function (figure 2.3). This function is also differentiable for infinite times, but does not become discontinuous after differentiating. The level-set function satisfies the transport equation, which may be recast in the conservation form

$$\frac{\partial \rho\phi}{\partial t} + \frac{\partial \rho u \phi}{\partial x} = 0. \tag{2.4.1}$$

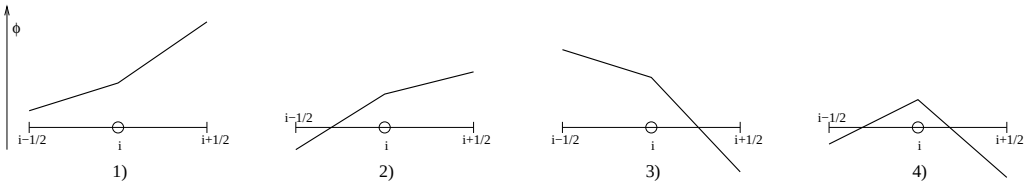

Figure 2.4: Four possible combinations of signs of $\phi_{i-\frac{1}{2}}$ and $\phi_{i+\frac{1}{2}}$ for $\phi_i > 0$.

Yet, note that $\rho\phi$ does not need to be conserved. By definition, $\phi(x,t) = 0$ tracks the interface at all latter times $t > 0$. The big advantage of the level-set method above the Volume-of-Fluid method is that the level-set function is smooth at the interface. The discontinuous VOF function may easily smear out during convection, degenerating a sharp interface into an erroneous, finite-width transition zone between the two fluids (blocks become blobs). The level-set function however, is smooth at the interface location and can be advected exactly as long as it is a signed-distance function. The level-set equation can be easily added to the system of fluid-flow equations (2.2.8a) and can be included in an exact or approximate Riemann solver. When writing (2.2.8a) in integral form, the natural discretization is a finite-volume technique. For convenience, we consider cell-centered finite volumes with constant mesh size h. With this choice, we can immediately calculate the volume fraction α of fluid 1 as a function of ϕ; $\alpha = \alpha(\phi)$. For a finite volume Ω_i with the cell faces $\partial\Omega_{i-\frac{1}{2}}$ and $\partial\Omega_{i+\frac{1}{2}}$, the level-set values at the cell faces can be defined as

$$\phi_{i-\frac{1}{2}} = \frac{1}{2}\left(\phi_{i-1} + \phi_i\right), \tag{2.4.2a}$$

$$\phi_{i+\frac{1}{2}} = \frac{1}{2}\left(\phi_i + \phi_{i+1}\right). \tag{2.4.2b}$$

When the level-set function at the cell center is positive, there are four possibilities for the Volume-of-Fluid-1 fraction, depending on the values of the level-set function at the cell faces (see figure 2.4). The following expressions for α_i are proposed:

$$\phi_{i-\frac{1}{2}} \geq 0,\ \phi_{i+\frac{1}{2}} \geq 0: \qquad \alpha_i = 1, \tag{2.4.3a}$$

$$\phi_{i-\frac{1}{2}} < 0,\ \phi_{i+\frac{1}{2}} \geq 0: \qquad \alpha_i = \frac{1}{2}\left(\frac{\phi_i}{\phi_i - \phi_{i-\frac{1}{2}}} + 1\right), \tag{2.4.3b}$$

$$\phi_{i-\frac{1}{2}} \geq 0,\ \phi_{i+\frac{1}{2}} < 0: \qquad \alpha_i = \frac{1}{2}\left(1 + \frac{\phi_i}{\phi_i - \phi_{i+\frac{1}{2}}}\right), \tag{2.4.3c}$$

$$\phi_{i-\frac{1}{2}} < 0,\ \phi_{i+\frac{1}{2}} < 0: \qquad \alpha_i = \frac{1}{2}\left(\frac{\phi_i}{\phi_i - \phi_{i-\frac{1}{2}}} + \frac{\phi_i}{\phi_i - \phi_{i+\frac{1}{2}}}\right). \tag{2.4.3d}$$

2.5. Discretization

For our numerical approach, a finite volume method is chosen. The finite volume method uses the integral form of the conservation equations as its starting point. The solution domain Ω is subdivided into a finite number of contiguous control volumes Ω_i, and the conservation equations are applied to each control volume. The cell faces of a control volume will be defined by $\partial\Omega_{i-\frac{1}{2}}$ and $\partial\Omega_{i+\frac{1}{2}}$. The flow solution is considered to be constant in each volume.

Allowing for a source term, for a small control volume Ω_i, the system of equations in conservative integral form considered reads

$$\int_{\Omega_i} \frac{dq}{dt} dx + \left(f(q)\right)_{\partial\Omega_{i+\frac{1}{2}}} - \left(f(q)\right)_{\partial\Omega_{i-\frac{1}{2}}} = \int_{\Omega_i} g(x,t) dx, \tag{2.5.1}$$

with

$$q = \begin{pmatrix} \rho \\ \rho u \\ \rho \phi \end{pmatrix}, \qquad f(q) = \begin{pmatrix} \rho u \\ \rho u^2 + p \\ \rho u \phi \end{pmatrix}, \tag{2.5.2}$$

for the fully conservative method. When using the ghost-fluid method, the level-set function is simply advected and the vectors q and f read

$$q = \begin{pmatrix} \rho \\ \rho u \end{pmatrix}, \qquad f(q) = \begin{pmatrix} \rho u \\ \rho u^2 + p \end{pmatrix}. \tag{2.5.3}$$

The bulk density in every cell is determined with aid of (2.2.7) where α is computed according to (2.4.2) and (2.4.3), and ρ_{I} and ρ_{II} according to Tait's equation of state (2.2.5). The determination of the fluxes is done by the first order accurate Linearized Godunov scheme, as described in section 2.3. Both static and dynamic problems in this MSc-work are integrated in time by using the forward Euler scheme. Discretization in time and space of (2.5.1) gives

$$\frac{q_i^{n+1} - q_i^n}{\Delta t}\Delta x + (f(q))_{\partial\Omega_{i+\frac{1}{2}}} - (f(q))_{\partial\Omega_{i-\frac{1}{2}}} = g_i \Delta x. \tag{2.5.4}$$

For numerical test cases with $g(x) = 0$, the new state vector becomes

$$q_i^{n+1} = q_i^n + \frac{\Delta t}{\Delta x}\left((f(q))_{\partial\Omega_{i-\frac{1}{2}}} - (f(q))_{\partial\Omega_{i+\frac{1}{2}}}\right). \tag{2.5.5}$$

The source term stiffens the differential equation. This implies that a small perturbation can cause large changes in the solution. If so, to improve both convergence and numerical stability, the source term needs special treatment. A two step procedure, known as a splitting method, is used as described in [18]. A brief description of this method is given below. The source term $g(x)$, modelling gravity, is given by

$$g(x,t) = Qq(x,t), \quad Q = \begin{pmatrix} 0 & 0 \\ -\mathrm{Fr}^{-2} & 0 \end{pmatrix},$$

where Fr is the Froude number, $\mathrm{Fr} = \frac{U}{\sqrt{gL}}$, with U a reference speed, g the acceleration of gravity and L a reference length. In the splitting method, first (2.5.1) is solved for a time step Δt, without a source term by using (2.5.5). Second, the newly obtained solution vector q is multiplied with $(I + \Delta t Q)$. After close inspection of this splitting method, it can be seen that the resulting set of equations uses an implicit scheme for evaluating the source term:

$$q_i^{n+1} = q_i^n + \frac{\Delta t}{\Delta x}\left((f(q))_{\partial\Omega_{i-\frac{1}{2}}} - (f(q))_{\partial\Omega_{i+\frac{1}{2}}}\right) + \Delta t \begin{pmatrix} 0 \\ -\rho_i^{n+1}\mathrm{Fr}^{-2} \end{pmatrix}. \tag{2.5.6}$$

Another consequence of the stiffening property induced by the source term is the necessity of lowering the CFL number. Normally the Courant Friedrichs Lewy (CFL) condition, defined as

$$\sigma = u_{\max}\frac{\Delta t}{\Delta x}, \tag{2.5.7}$$

where $u_{\max}$ is defined as the highest perturbation traveling speed, $u_{\max} = \max(|\mathrm{u}|, |\mathrm{u} - \mathrm{a}|, |\mathrm{u} + \mathrm{a}|)$, can be 1 or less when using forward Euler time integration. However, when the source term is added to the equation set, numerical stability is only obtained by much smaller CFL values. When the CFL value is chosen, the maximally allowed time step, used for forward Euler time integration, can be determined by simply isolating Δt from (2.5.7).

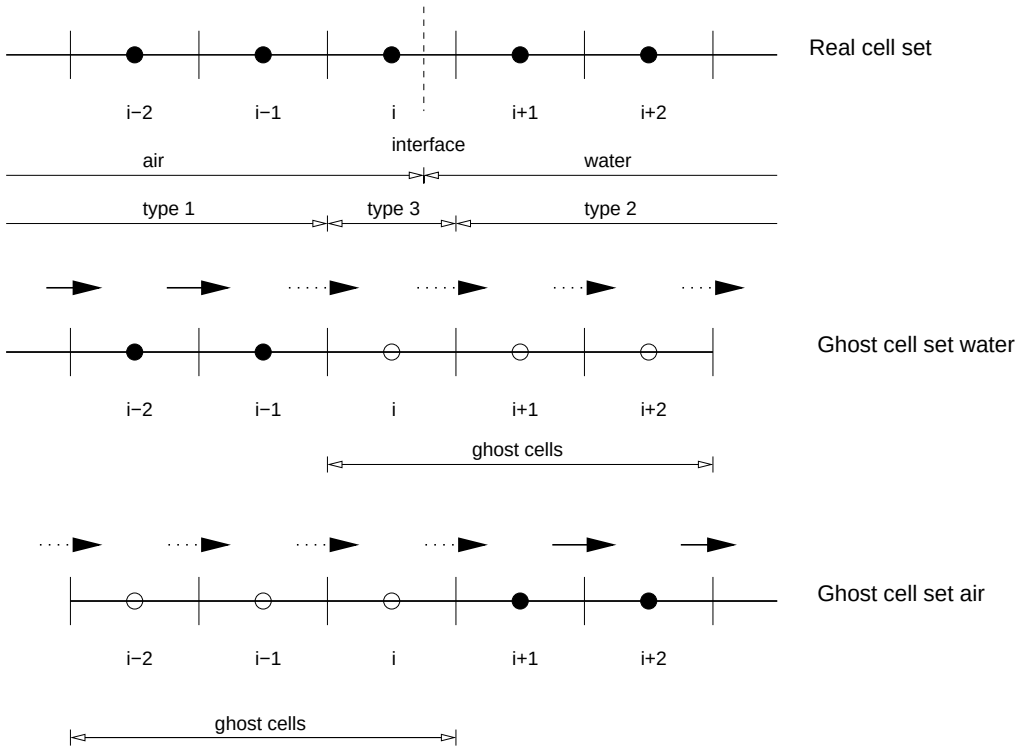

Figure 2.5: Definition of a narrow strip of ghost cells near the interface.

2.6. The ghost-fluid method

In [8] a new method is proposed for treating non-homentropic Eulerian equations in multi-fluid environments. The level-set algorithm is used for computing the interface as a moving internal boundary and is simply advected with (2.7.14). Near the interface, two sets of virtual cells are introduced, called "ghost cells". The real fluid state variables are kept in the real cells. In the ghost cells, we extrapolate the state variables from the real cells beyond the interface and copy those values node by node into the ghost cells as shown in figure 2.5.

For definition of the ghost nodes in one spatial dimension, three quantities must be defined in the ghost region. In [8] the recommended variables to copy to the ghost cells are u, p, and the entropy s. However, for our simplified model, only the two variables u and p will be copied into the ghost cells. So, near the interface, three sets of state variables are stored: the real values in the real cells and the extrapolated state values in the 2 sets of ghost cells. In each set of ghost cells an updated solution is obtained, one for fluid I and one for fluid II. This ambiguity must be removed. The state variables are made unique with use of the formula

$$\begin{pmatrix} u \\ p \end{pmatrix} = \alpha \begin{pmatrix} u_{\mathrm{I}} \\ p_{\mathrm{I}} \end{pmatrix} + (1-\alpha) \begin{pmatrix} u_{\mathrm{II}} \\ p_{\mathrm{II}} \end{pmatrix}. \tag{2.6.1}$$

There are no physical or mathematical arguments whatsoever for applying this equation. Other choices are possible. The fluxes (real and ghost) are computed with the single-fluid version of the linearized Godunov scheme. But the ghost-fluid method enables us to use other schemes as well, like Osher's scheme, Roe's scheme or Van Leer's flux vector splitting scheme, thanks to the single-fluid approach of the ghost-fluid method. Let's take a closer look at the ghost-fluid method for our particular test case. First determine for each cell if the cell is filled with fluid I, fluid II or contains both. The cells containing fluid I only are called cells of type 1. The cells containing fluid II only are called cells of type 2. Cells containing both fluids and thus having an interface, are called cells of type 3. For our test case, the set of ghost-fluid cells contains only cells of type 3. That limits the total bandwidth of ghost cells to one. Only by exception will the bandwidth of

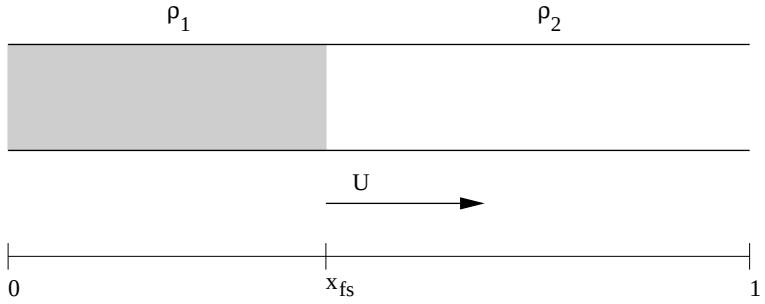

Figure 2.6: Model 1, a 1D open tube filled with, e.g., water and air.

ghost cells grow to two cells. This is the case when the interface is located at a cell face.

2.7. Test model 1: 1D two-fluid flow

To test the fully conservative level-set computation method, we consider a 1D tube with unit length, $x \in [0,1]$, inflow at $x = 0$, outflow at $x = 1$ and with as initial solution:

$$u(x, t=0) = U > 0, \tag{2.7.1}$$

$$p(x, t=0) = P, \tag{2.7.2}$$

$$\rho(x, t=0) = \begin{cases} \rho_{\mathrm{I}} = x \leq (x_{\mathrm{fs}})_{t=0}\,, \\ \rho_{\mathrm{II}} = x \geq (x_{\mathrm{fs}})_{t=0}\,, \end{cases} \tag{2.7.3}$$

where U and P are constant, and where x_{fs} is the location of the free surface, the two-fluid interface. For the boundary conditions we choose at the left hand side and right hand side a Dirichlet condition:

$$u(0,t) = U, \tag{2.7.4a}$$

$$p(1,t) = P_{\mathrm{init}}. \tag{2.7.4b}$$

For simplicity, we do not add external forces, like gravity. It's a very trivial test case, ideal for validation. The problem is sketched in figure 2.6.

2.7.1 Numerical results

2.7.1.1 Fully conservative method Due to the simplicity of this model, the exact solution is known for every $t > 0$. Both pressure p and velocity u remain constant for every t. The location of the interface reads $x_{\mathrm{fs}}(t) = (x_{\mathrm{fs}})_{t=0} + Ut$. With the knowledge of the interface location, the density of the flow reads $\rho = \rho_{\mathrm{I}}$ for $x < x_{\mathrm{fs}}(t)$ and $\rho = \rho_{\mathrm{II}} > x_{\mathrm{fs}}(t)$. At the interface the Dirichlet boundary condition $p_{\mathrm{I}}(x_{\mathrm{fs}}) = p_{\mathrm{II}}(x_{\mathrm{fs}})$ holds implicitly. It is not imposed to the capturing method! Recalling the conservative, semi-discrete Euler equations for finite volume Ω_i in integral form:

$$\int_{\Omega_i} \frac{dq}{dt} dx + f(q)_{\partial\Omega_{i+\frac{1}{2}}} - f(q)_{\partial\Omega_{i-\frac{1}{2}}} = 0,$$

$$q = \begin{pmatrix} \rho \\ \rho u \\ \rho\phi \end{pmatrix}, \qquad f(q) = \begin{pmatrix} \rho u \\ \rho u^2 + p \\ \rho u \phi \end{pmatrix}, \tag{2.7.5}$$

with ρ_i, the bulk density given by

$$\rho_i(\phi, p_i) = \alpha_i(\phi)\rho_{\mathrm{I}}(p_i) + (1 - \alpha_i(\phi))\rho_{\mathrm{II}}(p_i), \qquad 0 \leq \alpha_i(\phi) \leq 1, \tag{2.7.6}$$

in which $\alpha_i(\phi)$ is given by (2.4.2) and (2.4.3) and in which $\rho_{\mathrm{I}}(p_i)$ and $\rho_{\mathrm{II}}(p_i)$ are given by Tait's equation of state (2.2.5). So α_i is the volume of water fraction in finite volume Ω_i. If $\alpha_i = 1$, Ω_i is completely filled with water. If $0 < \alpha_i < 1$, it is partially filled with water and air, and if $\alpha_i = 0$

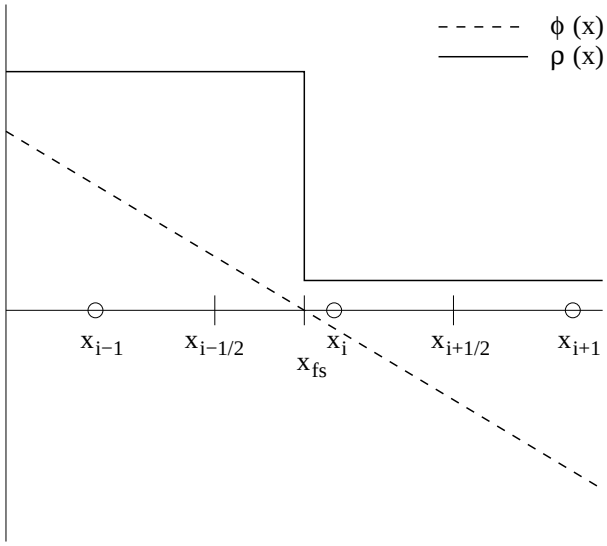

Figure 2.7: Density jump, level-set function and interface location in a 1D flow

entirely with air. For, e.g., water, it holds at sea level conditions: $\gamma = 7$, $B = 3000$, $\rho_{\text{ref}} = 1000$ kg/m^3 and for air $\gamma = 1.4$, $B = 0$, $\rho_{\text{ref}} = 1$ kg/m^3.

In order to compute this problem numerically, we make a space discretization by considering an equidistant finite-volume grid with mesh size h. For the time integration, we take the forward Euler scheme. The space discretization is taken first-order accurate. With this, from (2.7.5) it follows

$$\frac{\rho_i^{n+1} - \rho_i^n}{\Delta t} h + (\rho u)^n_{i+\frac{1}{2}} - (\rho u)^n_{i-\frac{1}{2}} = 0, \tag{2.7.7a}$$

$$\frac{(\rho u)_i^{n+1} - (\rho u)_i^n}{\Delta t} h + (\rho u^2 + p)^n_{i+\frac{1}{2}} - (\rho u^2 + p)^n_{i-\frac{1}{2}} = 0, \tag{2.7.7b}$$

$$\frac{(\rho \phi)_i^{n+1} - (\rho \phi)_i^n}{\Delta t} h + (\rho u \phi)^n_{i+\frac{1}{2}} - (\rho u \phi)^n_{i-\frac{1}{2}} = 0. \tag{2.7.7c}$$

The following states can be derived directly from the equations (2.7.7a), (2.7.7b) and (2.7.7c)

$$\rho_i^{n+1}, \tag{2.7.8a}$$

$$u_i^{n+1} = \frac{(\rho u)_i^{n+1}}{\rho_i^{n+1}}, \tag{2.7.8b}$$

$$\phi_i^{n+1} = \frac{(\rho \phi)_i^{n+1}}{\rho_i^{n+1}}. \tag{2.7.8c}$$

It is easy to check these numerical values with the true physics. We apply a first-order accurate Godunov scheme for the flux evaluation and consider the solution q_i^n to be such that the interface is precisely at the cell face $\partial\Omega_{i-\frac{1}{2}}$ (figure 2.7). Then, the density in finite volume Ω_i at the new time level becomes:

$$\rho_i^{n+1} = \rho_i^n + \frac{U \Delta t}{h} \left(\rho_{\text{I}}(p) - \rho_{\text{II}}(p) \right). \tag{2.7.9}$$

For the momentum at the new time level it follows:

$$(\rho u)_i^{n+1} = U \left(\rho_{\text{II}}(p) \right) + \frac{U \Delta t}{h} \left(\rho_{\text{I}}(p) - \rho_{\text{II}}(p) \right). \tag{2.7.10}$$

Introducing the notation $\sigma = \frac{U\Delta t}{h}$, with (2.7.8c), (2.7.9) and (2.7.10), for the velocity in cell Ω_i at the new time level we can thus write:

$$u_i^{n+1} = \frac{U\left(\sigma\rho_{\mathrm{I}} + (1-\sigma)\rho_{\mathrm{II}}\right)}{\left(\sigma\rho_{\mathrm{I}} + (1-\sigma)\rho_{\mathrm{II}}\right)} = U, \tag{2.7.11}$$

which is in agreement with the true physics. For the update of $\rho\phi$ in cell Ω_i, with ϕ defined as the signed-distance function, it follows:

$$\begin{aligned}(\rho\phi)_i^{n+1} &= \rho_{\mathrm{II}}\phi_i^n + \sigma\left(\rho_{\mathrm{I}}\phi_{i-1}^n - \rho_{\mathrm{II}}\phi_i^n\right),\\ &= -\rho_{\mathrm{II}}\frac{h}{2} + \sigma\left(\rho_{\mathrm{I}} + \rho_{\mathrm{II}}\right)\frac{h}{2},\\ &= \left(\sigma\rho_{\mathrm{I}} - (1-\sigma)\,\rho_{\mathrm{II}}\right)\frac{h}{2}.\end{aligned} \tag{2.7.12}$$

Using equation (2.7.8c), (2.7.9) and (2.7.12) gives

$$\phi_i^{n+1} = \frac{\sigma\rho_{\mathrm{I}} + (\sigma-1)\rho_{\mathrm{II}}}{\sigma\rho_{\mathrm{I}} + (1-\sigma)\rho_{\mathrm{II}}}\frac{h}{2}. \tag{2.7.13}$$

Here the trouble starts. The (non-conservative) convection equation for ϕ reads

$$\frac{\partial\phi}{\partial t} + u\frac{\partial\phi}{\partial x} = 0. \tag{2.7.14}$$

Using the standard first-order upwind discretization, this equation gives

$$\begin{aligned}\phi_i^{n+1} &= \sigma\phi_{i-1}^n + (1-\sigma)\phi_i^n,\\ &= -\frac{h}{2} + \sigma h.\end{aligned} \tag{2.7.15}$$

The conservative solution (2.7.12) and the non-conservative solution (2.7.15) differ from each other. Solution (2.7.15) is the desired one; the level-set function is convected with the flow without errors. For the error introduced by the conservative calculation of the level-set function,

$$\Delta\phi_i^{n+1} = \left(\phi_i^{n+1}\right)_{\mathrm{conservative}} - \left(\phi_i^{n+1}\right)_{\mathrm{non-conservative}}, \tag{2.7.16}$$

we get

$$\begin{aligned}\Delta\phi &= \frac{\sigma\rho_{\mathrm{I}} + (\sigma-1)\rho_{\mathrm{II}}}{\sigma\rho_{\mathrm{I}} + (1-\sigma)\rho_{\mathrm{II}}}\frac{h}{2} - \left(-\frac{h}{2} + \sigma h\right),\\ &= \frac{\left(-\frac{h}{2} + \sigma h\right)\left(\sigma\rho_{\mathrm{I}} + (1-\sigma)\rho_{\mathrm{II}}\right) - \left(\sigma\rho_{\mathrm{I}} + (\sigma-1)\rho_{\mathrm{II}}\right)\frac{h}{2}}{\left(\sigma\rho_{\mathrm{I}} + (1-\sigma)\rho_{\mathrm{II}}\right)},\\ &= \frac{-h\sigma\rho_{\mathrm{I}} + \sigma h\left(\sigma\rho_{\mathrm{I}} + (1-\sigma)\rho_{\mathrm{II}}\right)}{\left(\sigma\rho_{\mathrm{I}} + (1-\sigma)\rho_{\mathrm{II}}\right)},\\ &= \frac{\sigma(\sigma-1)(\rho_{\mathrm{I}} - \rho_{\mathrm{II}})}{\left(\sigma\rho_{\mathrm{I}} + (1-\sigma)\rho_{\mathrm{II}}\right)}h.\end{aligned} \tag{2.7.17}$$

So, the error introduced by calculating in conservative form behaves $O(h)$, except for $\sigma = 0$ (trivial), $\sigma = 1$ and $\rho_{\mathrm{I}} = \rho_{\mathrm{II}}$. Unfortunately, this $O(h)$ error in ϕ induces an $O(1)$ error in α, which on its turn induces an $O(1)$ error in p [20]. Some numerical experiments have been made. For this experiment, the following values have been used: $\rho_{\mathrm{I}}/\rho_{\mathrm{II}} = 10$, $\sigma = 0.01$, $u_{\mathrm{init}} = 1$ and $p_{\mathrm{init}} = 1$ (see figures 2.8 to 2.13). A possible remedy in order to increase the accuracy and minimize the error could be a higher-order discretization of the function ϕ. Let us recall equation (2.7.12). Instead of a first-order accurate space discretization, a second-order central discretization will be used. For time integration, again first order forward-Euler is used,

$$\begin{aligned}\left(\phi_i^{n+1}\right)_{\mathrm{conservative}} &= \frac{(\rho\phi)_i^{n+1}}{\rho_i^{n+1}}\\ &= \frac{(\rho\phi)_i^{n+1} + \sigma\left((\rho\phi)_{i-\frac{1}{2}}^n - (\rho\phi)_{i+\frac{1}{2}}^n\right)}{\rho_i^n + \sigma\left(\rho_{i-\frac{1}{2}}^n - \rho_{i+\frac{1}{2}}^n\right)},\end{aligned} \tag{2.7.18}$$

with

$$\rho^n_{i-\frac{1}{2}} = \frac{1}{2}\left(\rho_{\mathrm{I}} + \rho_{\mathrm{II}}\right), \tag{2.7.19a}$$

$$\rho^n_{i+\frac{1}{2}} = \rho_{\mathrm{II}}, \tag{2.7.19b}$$

$$\phi^n_{i-\frac{1}{2}} = \frac{1}{2}\left(\phi^n_{i-1} + \phi^n_i\right) = 0, \tag{2.7.19c}$$

$$\phi^n_{i+\frac{1}{2}} = \frac{1}{2}\left(\phi^n_i + \phi^n_{i+1}\right) = -h, \tag{2.7.19d}$$

which yields

$$\begin{aligned}\left(\phi^{n+1}_i\right)_{\text{conservative}} &= \frac{-\frac{h}{2} + \sigma\left(\frac{1}{2}\left(\rho_{\mathrm{I}} - \rho_{\mathrm{II}}\right)\frac{1}{2}\left(0\right) - \rho_{\mathrm{II}}\frac{1}{2}\left(-2h\right)\right)}{\rho_{\mathrm{II}} + \sigma\left(\frac{1}{2}\left(\rho_{\mathrm{I}} + \rho_{\mathrm{II}}\right) - \rho_{\mathrm{II}}\right)}, \\ &= \left(-\frac{h}{2} + \sigma h\right)\frac{\rho_{\mathrm{II}}}{\rho_{\mathrm{II}} + \sigma\frac{1}{2}(\rho_{\mathrm{I}} - \rho_{\mathrm{II}})}. \end{aligned} \tag{2.7.20}$$

The non-conservative form of ϕ^{n+1}_i reads

$$\begin{aligned}\left(\phi^{n+1}_i\right)_{\text{non}-\text{conservative}} &= \phi^n_i + \sigma\frac{\phi^n_{i-1} - \phi^n_{i+1}}{2}, \\ &= -\frac{h}{2} + \sigma h. \end{aligned} \tag{2.7.21}$$

For a linear distribution of ϕ^n, the non-conservative formula is again exact. So, the error can again be written as the difference between the conservative and non-conservative formulation, i.e. as (2.7.16). For (2.7.20) and (2.7.21), we find

$$\begin{aligned}\Delta\phi^{n+1}_i &= \frac{1}{4}\sigma\frac{\rho_{\mathrm{I}} - \rho_{\mathrm{II}}}{\rho_{\mathrm{II}} + \frac{1}{2}\sigma\left(\rho_{\mathrm{I}} - \rho_{\mathrm{II}}\right)}\left(\left(\phi^n_{i-1} - \phi^n_i\right) + \sigma\left(\phi^n_{i+1} - \phi^n_{i-1}\right)\right), \\ &= \left(-\frac{h}{2} + \sigma h\right)\frac{\sigma\frac{1}{2}(\rho_{\mathrm{I}} - \rho_{\mathrm{II}})}{\rho_{\mathrm{II}} + \sigma\frac{1}{2}(\rho_{\mathrm{I}} - \rho_{\mathrm{II}})}. \end{aligned} \tag{2.7.22}$$

This error obtained with the second-order central discretization is unfortunately not an improvement as compared to the error (2.7.17) obtained with the first-order upwind discretization. It is still of $O(h)$ except - again - for $\sigma = 0$ and $\rho_{\mathrm{I}} = \rho_{\mathrm{II}}$. ($\sigma = 1$ is no such special case here!) In the same way as for the first-order upwind discretization, it can be shown now that the $O(h)$ error (2.7.22) in ϕ induces an $O(1)$ error in α and hence in p. Through the bulk density formula (2.7.6), the error carries over into a pressure error Δp^{n+1}_i. A proof is given in [20]. So, high-order discretization is no fix for the zeroth-order pressure error.

So, even the most trivial two-fluid flow case seems impossible to solve with use of the conservative level-set approach. Other methods have to be applied to get numerical stability and a valid solution.

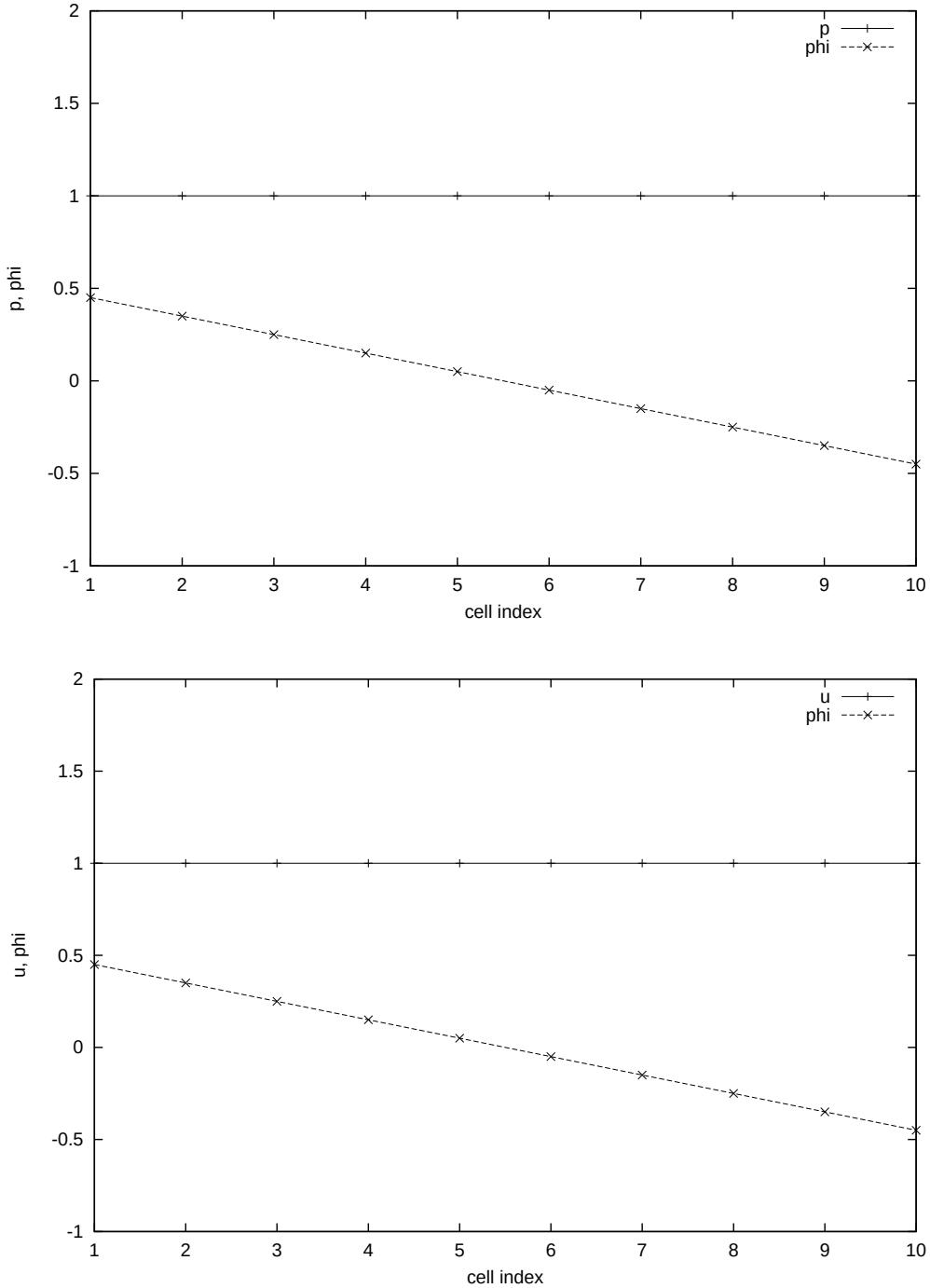

Figure 2.8: Pressure, level-set and velocity at $t = 0.0$

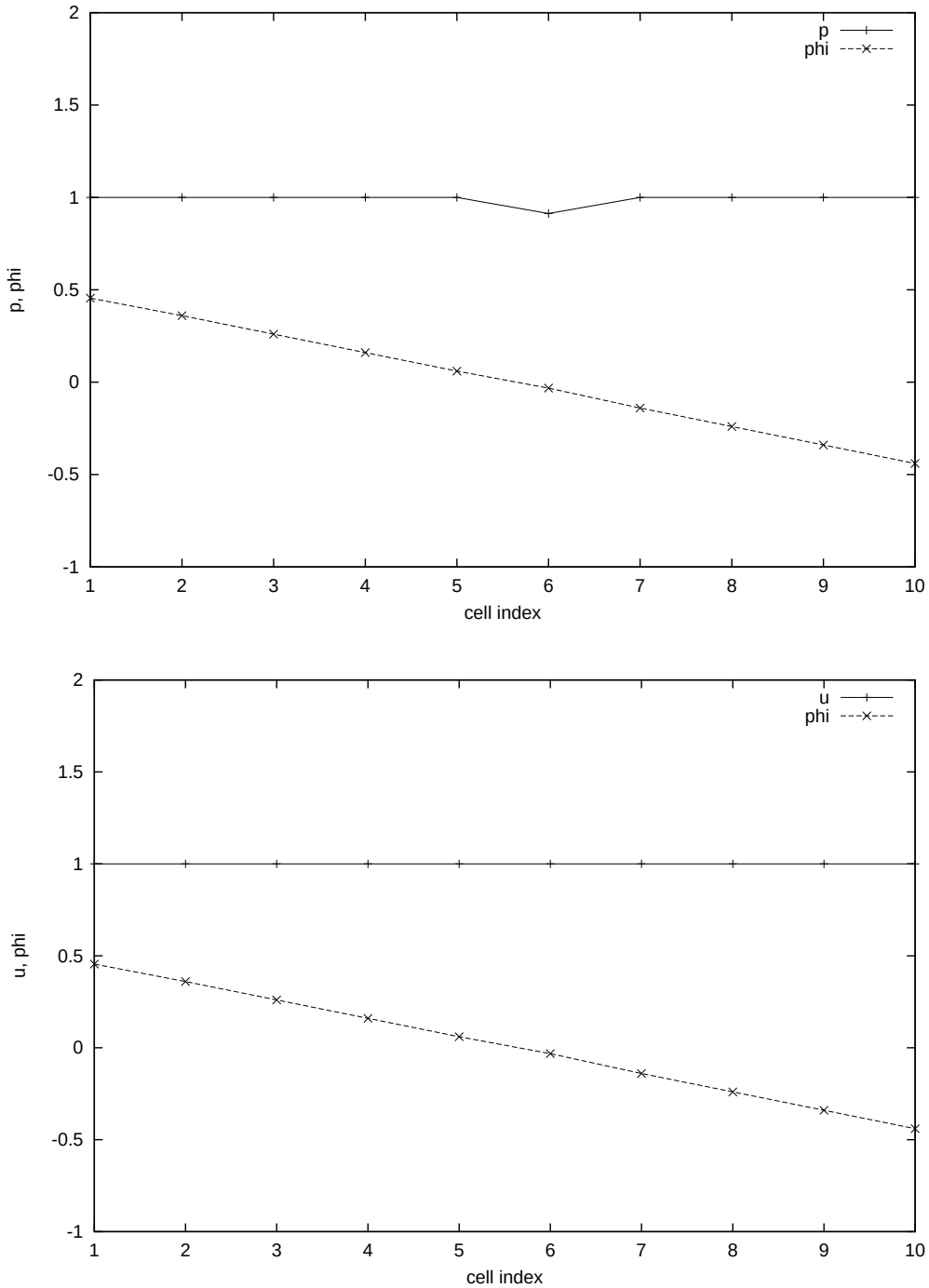

Figure 2.9: Pressure, level-set and velocity at $t = 0.001$

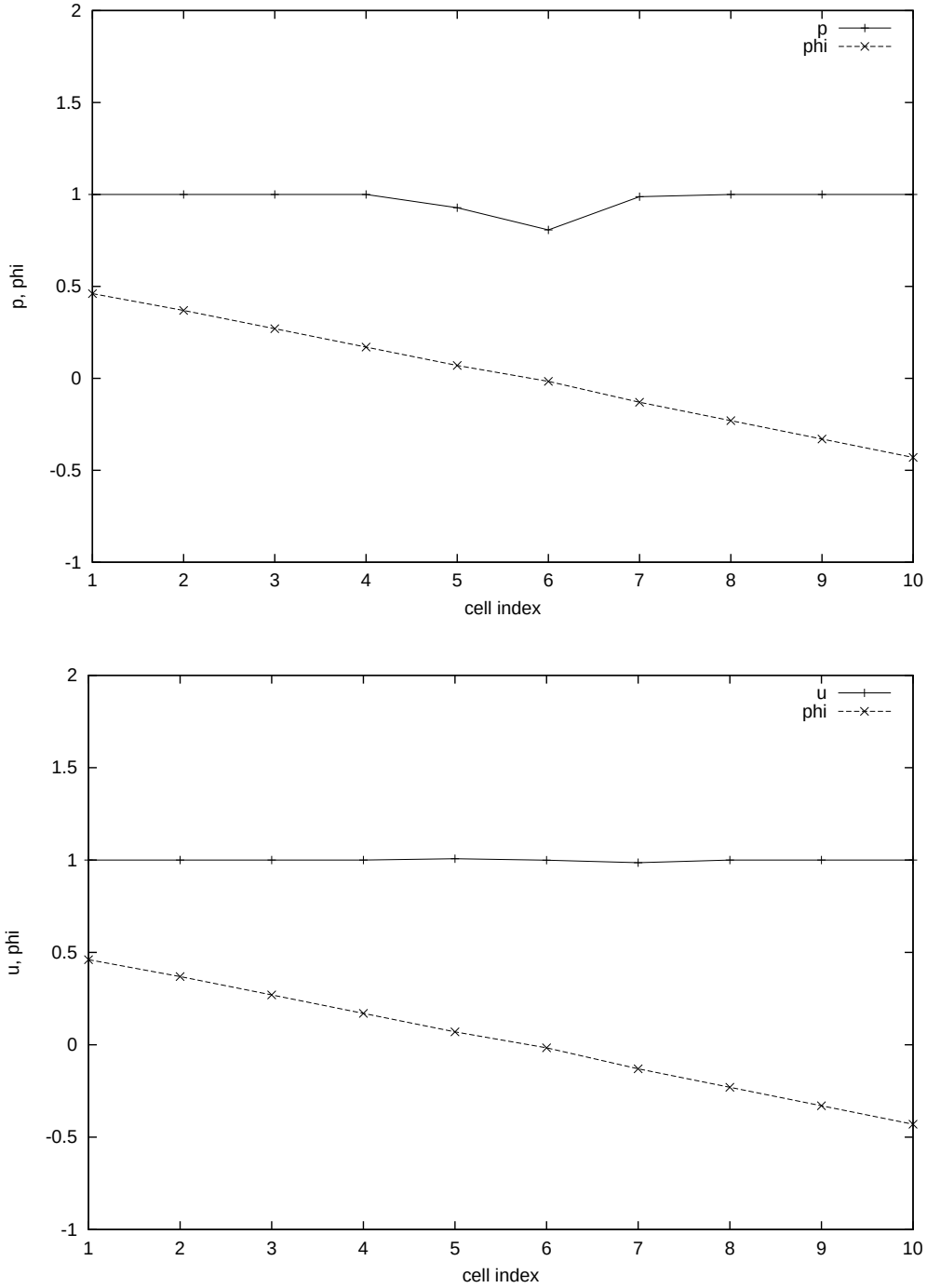

Figure 2.10: Pressure, level-set and velocity at $t = 0.002$

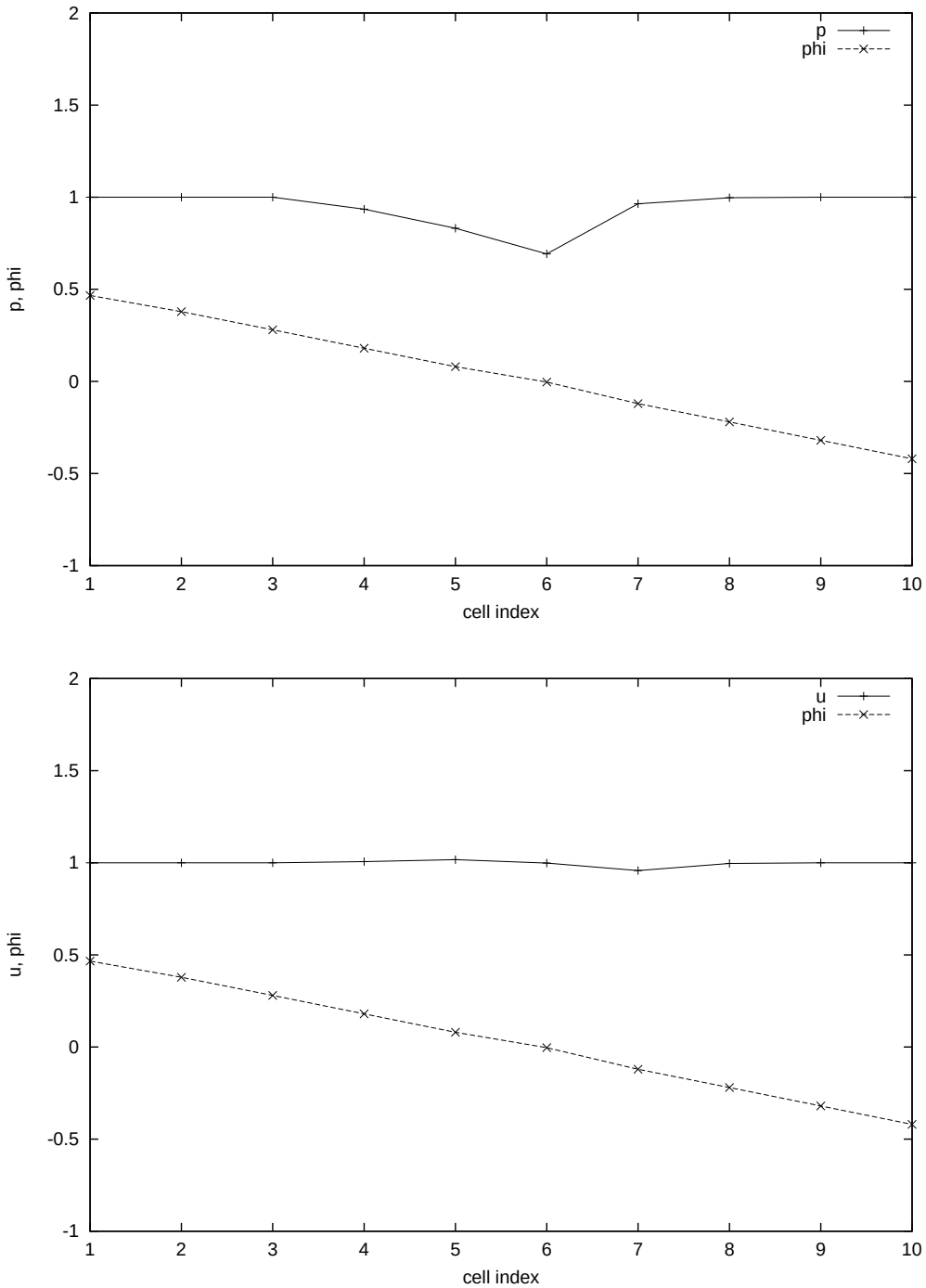

Figure 2.11: Pressure, level-set and velocity at $t = 0.003$

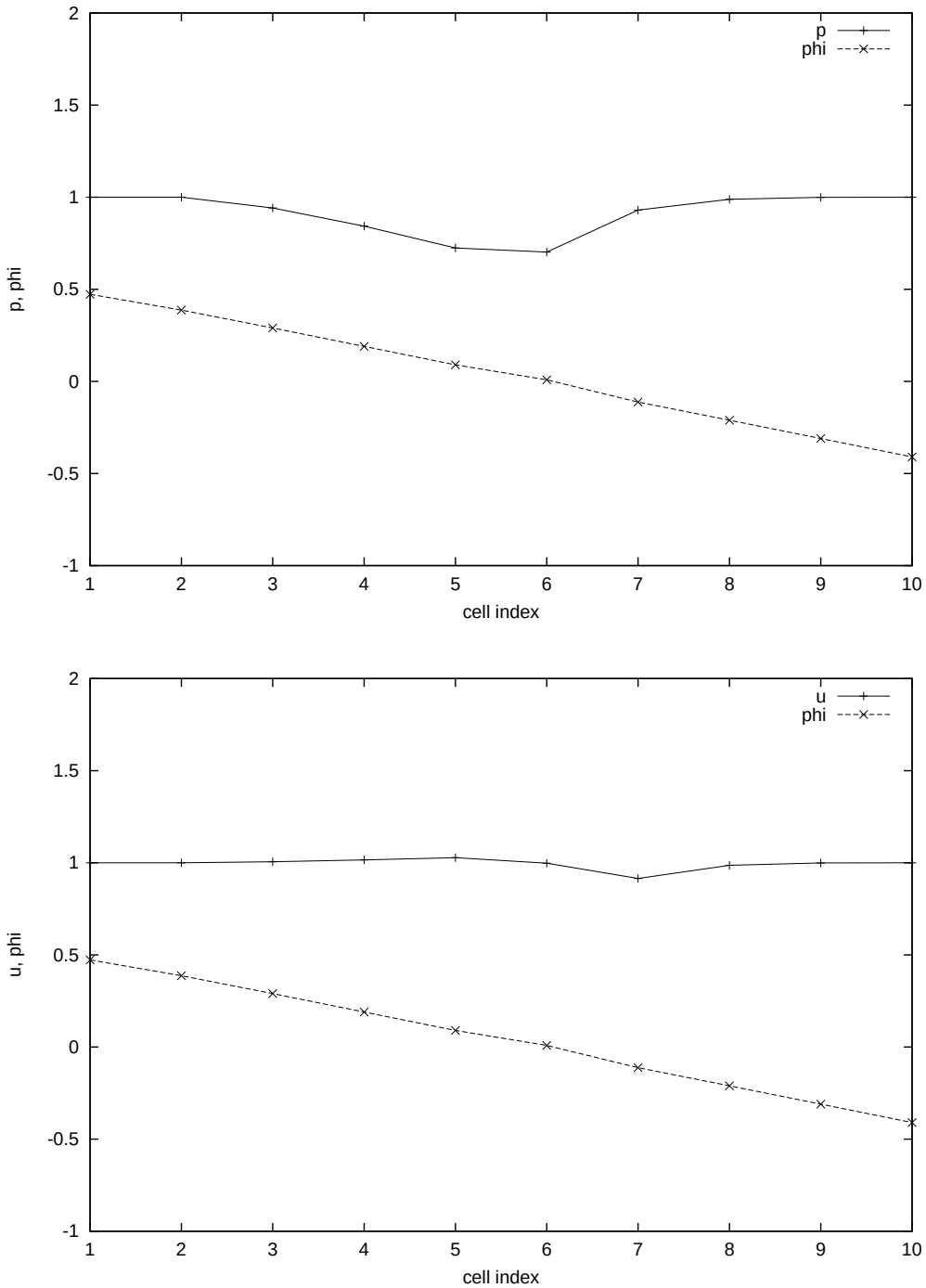

Figure 2.12: Pressure, level-set and velocity at $t = 0.004$

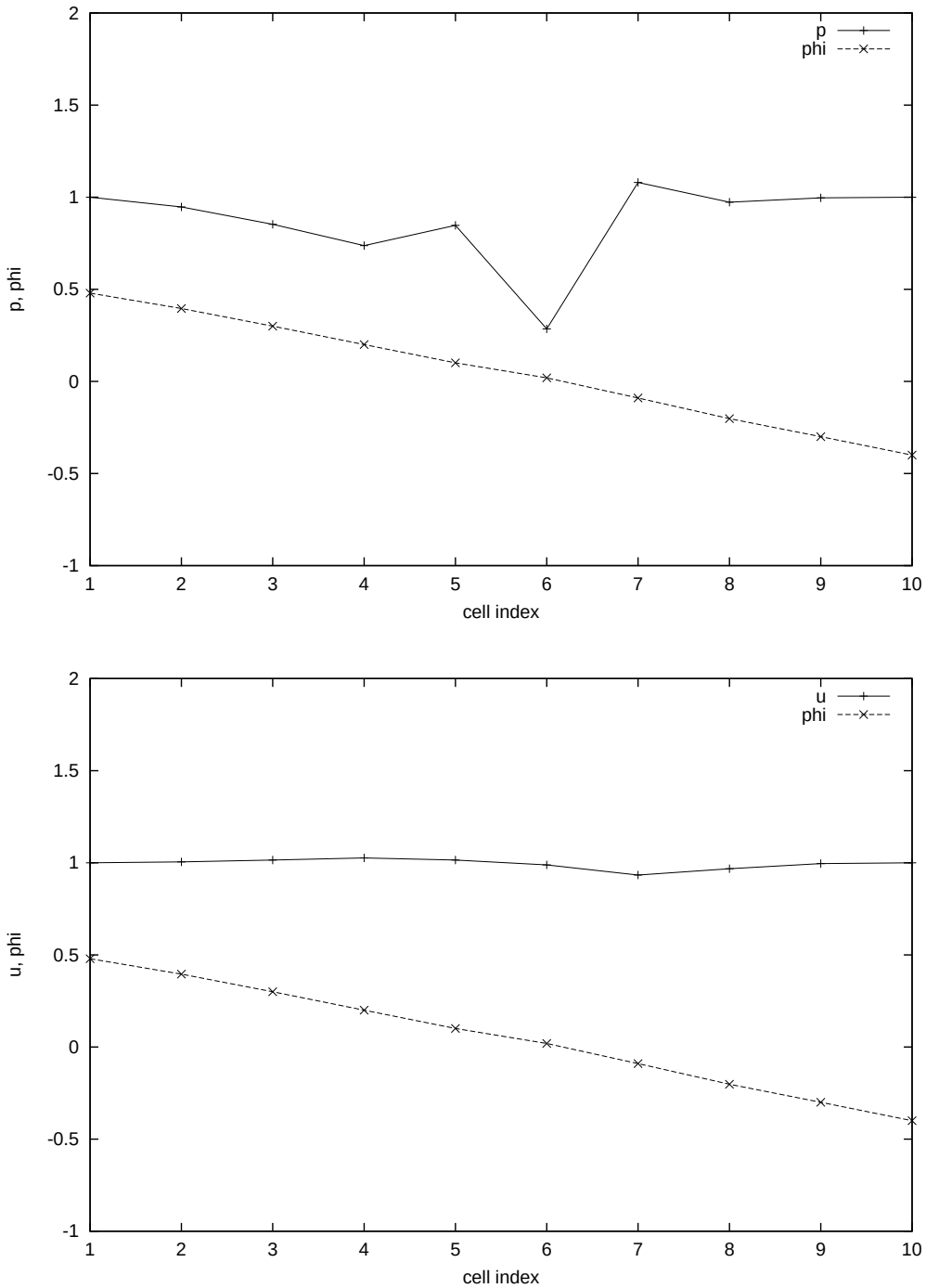

Figure 2.13: Pressure, level-set and velocity at $t = 0.005$

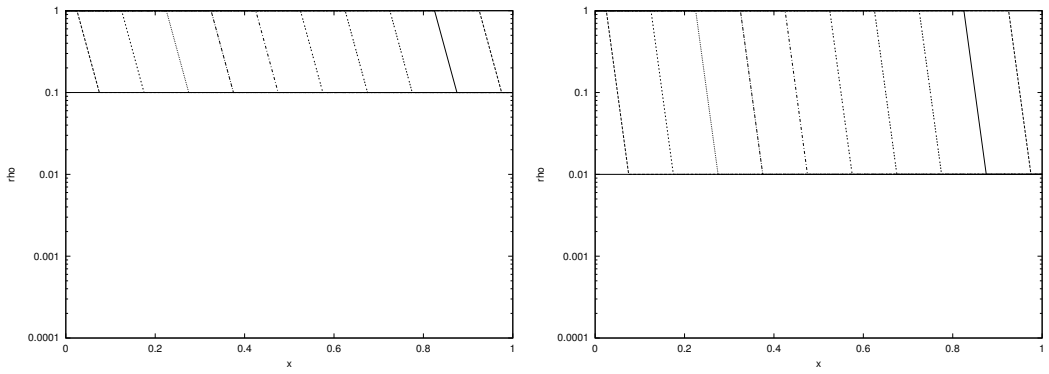

Figure 2.14: Ghost-fluid method applied to test model 1 with $\frac{\rho_{\mathrm{I}}}{\rho_{\mathrm{II}}} = 10$ (left) and $\frac{\rho_{\mathrm{I}}}{\rho_{\mathrm{II}}} = 100$ (right).

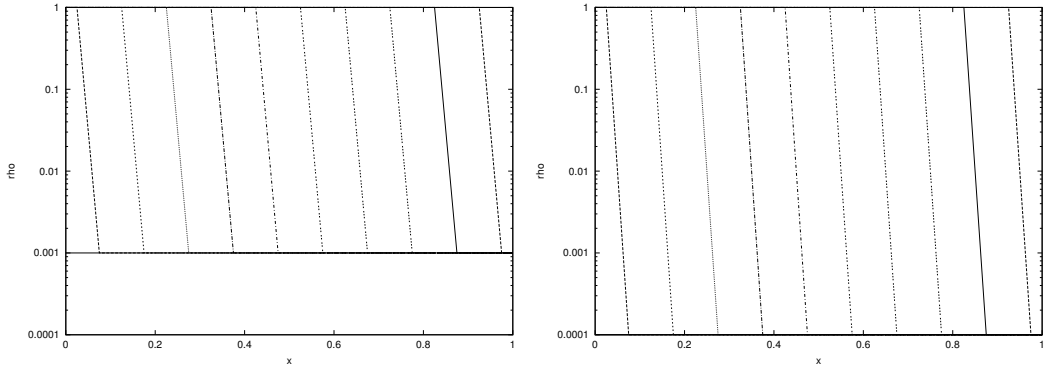

Figure 2.15: Ghost-fluid method applied to test model 1 with $\frac{\rho_{\mathrm{I}}}{\rho_{\mathrm{II}}} = 1000$ (left) and $\frac{\rho_{\mathrm{I}}}{\rho_{\mathrm{II}}} = 10000$ (right).

2.7.1.2 Ghost-fluid method The ghost-fluid method, described in section 2.6 is numerically tested. In the cell containing the material interface (cell type 3), both water and air fluxes are computed separately. Then, the solution is made unique with equation (2.6.1). Results of the tests are shown in figures 2.7.1.2 to 2.7.1.2 for $t = 0.0$, 0.1, 0.2,..., 1.0. In contrary to the conservative level-set method, described in the former section, this method proves to work very well for the test model 1, even with extremely large density ratios. Pressure oscillations, as seen with the fully conservative method, do not occur. The computation and solution does not suffer from pressure oscillations anymore. The interface is represented as a sharp discontinuity and is nicely advected through the tube with constant speed, as expected from the exact solution.

2.8. Test model 2: 1D two-fluid flow with gravity

Again, we test the level-set and ghost-fluid method, by taking a 1D problem. However, in this test case we add a source term, modelling gravity. Recall equation (2.2.8a). For the source term, we choose

$$g(x) = \begin{pmatrix} 0 \\ -\frac{\rho}{\mathrm{Fr}^2} \end{pmatrix}. \tag{2.8.1}$$

The dimensionless parameter $\mathrm{Fr} = \frac{U}{\sqrt{gL}}$ is the Froude number in which U and L act as the characteristic speed and length. For this test case, the unit length of the tube has been choosen for L and a typical subsonic speed 1 has been choosen for U. Another possible choice for L could be the column height of fluid I and for U the highest perturbation speed of the fluids. The quantity

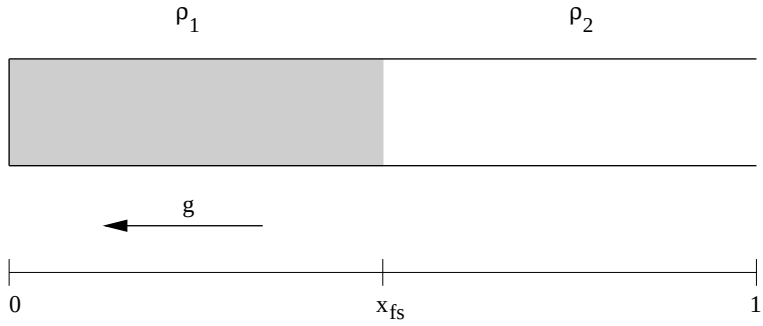

Figure 2.16: Model 2, one side closed 1D tube filled with water and air, gravity force acting from right to left.

g is the acceleration of gravity. Consider a 1D tube with unit length, $x \in [0,1]$, with as initial solution:

$$u(x,t=0) = 0, \tag{2.8.2a}$$
$$p(x,t=0) = P, \tag{2.8.2b}$$
$$\rho(x,t=0) = \begin{cases} \rho_{\mathrm{I}} = x \leq (x_{\mathrm{fs}})_{t=0}\,, \\ \rho_{\mathrm{II}} = x \geq (x_{\mathrm{fs}})_{t=0}\,, \end{cases} \tag{2.8.2c}$$

where U and P are constant, and where x_{fs} is the location of the free surface, the water-air interface. The left side of the tube is closed, the right side of the tube is open (figure 2.16). For the boundary conditions we choose for both left side and right boundary a Dirichlet condition:

$$u(0,t) = 0, \tag{2.8.3a}$$
$$p(1,t) = P_{\mathrm{init}}. \tag{2.8.3b}$$

As with the first test case, the interface is captured. The condition $p_{\mathrm{I}}(x_{\mathrm{fs}}) = p_{\mathrm{II}}(x_{\mathrm{fs}})$ is satisfied implicitly; we do not impose this condition. An approximate equation for the pressure at the bottom of the tube $x = 0$ is

$$p(0) = p(1) + \frac{\rho_{\mathrm{I}}(p_{\mathrm{init}})(l_{\mathrm{I}})_{\mathrm{init}} + \rho_{\mathrm{II}}(p_{\mathrm{init}})(l_{\mathrm{II}})_{\mathrm{init}}}{\mathrm{Fr}^2}, \tag{2.8.4}$$

where $(l_{\mathrm{I}})_{\mathrm{init}}$ and $(l_{\mathrm{II}})_{\mathrm{init}}$ are the initial lengths of the water- and air columns. The discretization and numerical approach is done as described in section 2.5. For this dynamic problem, the forward Euler time integration is used. The two fluid columns oscillate in time. Since no physical damping exists, only the numerical diffusion is responsible for damping of the oscillating fluid columns. For this test case, we are only interested in the steady solution and convergence has been reached when the steady solution has been obtained.

2.8.1 Numerical results

2.8.1.1 Ghost-fluid method In contrast to the excellent results with the ghost-fluid method for a two-fluid flow without gravity, the ghost-fluid method fails for a two-fluid flow with gravity. After several time steps, a zeroth-order pressure error starts to develop near the interface, until finally the computation bursts. This is clearly visible in figure 2.17. Let us take a closer look to find out what causes the pressure oscillations by performing a qualitative timestep analysis. Consider an exact solution with $x_{\mathrm{fs}} = 0.5$, $U(x) = 0$ and the pressure distribution as given in figure 2.18. For convenience, discretization is done in such a way that the free-surface is located on a cell face. Cells with index $i-1$ and i are bounded by x_{fs} and defined as ghost cells. If the system is stable, one timestep can be taken without changing the state variables. Now, we perform one time step.

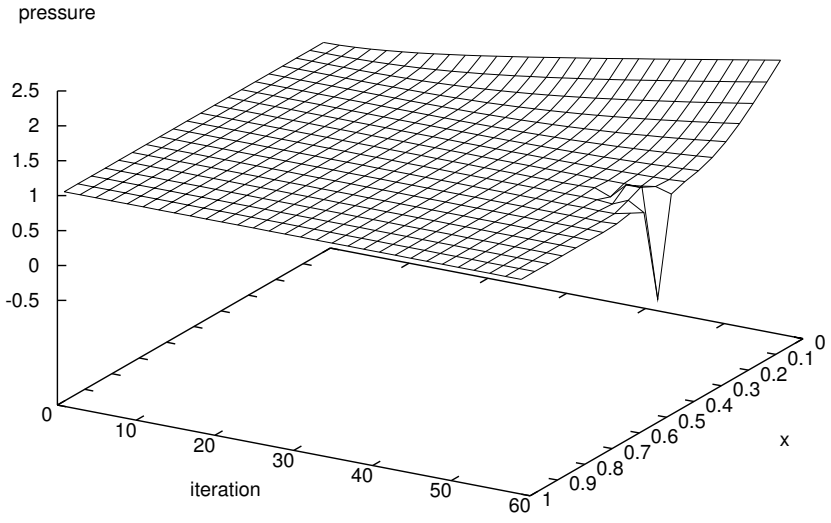

Figure 2.17: First iteration steps using the ghost-fluid method. When the pressure buildup reaches air, the computation collapses.

First, the level-set function is advected by the local velocity:

$$\begin{aligned} \phi_i^{n+1} &= \phi_i^n + \frac{\Delta t}{h}(u^n_{i-\frac{1}{2}}\phi^n_{i-\frac{1}{2}} - u^n_{i+\frac{1}{2}}\phi^n_{i+\frac{1}{2}}) \\ &= \phi_i^n. \end{aligned} \tag{2.8.5}$$

Velocity is zero, so nothing is changed here. Now we perform a time step for ghost cell i. Notice that only air has to be evaluated. The Volume-of-Fluid fraction $\alpha = 0$ due to the absence of water in the cell. We apply a first-order accurate Godunov scheme for the flux evaluation. The density in finite volume Ω_i at the new time level becomes:

$$\begin{aligned} \rho_i^{n+1} &= \rho_i^n + \frac{\Delta t}{h}(\rho^n_{i-1}u^n_{i-1} - \rho^n_i u^n_i), \\ &= \rho_i^n. \end{aligned} \tag{2.8.6}$$

Again, no change here. For the momentum at the new time level, it follows:

$$\begin{aligned} (\rho u)_i^{n+1} &= (\rho u)_i^n + \frac{\Delta t}{h}\left(\left(\rho u^2 + p\right)^n_{i-1} - \left(\rho u^2 + p\right)^n_i\right) - \Delta t\frac{\rho_i^{n+1}}{\mathrm{Fr}^2}, \\ &= \frac{\Delta t}{h}(p^n_{i-1} - p^n_i) - \Delta t\frac{\rho_i^{n+1}}{\mathrm{Fr}^2}. \end{aligned} \tag{2.8.7}$$

At this point, the pressure oscillation is initiated. At time n, the momentum $(\rho u)_i^n = 0$. But with $p^n_{i-1} > p^n_i$ and $\frac{\Delta t}{h}(p^n_{i-1} - p^n_i) \neq -\Delta t\frac{\rho_i^{n+1}}{\mathrm{Fr}^2}$, the momentum $(\rho u)_i^{n+1}$, and with that u_i^{n+1}, returns a non-zero erroneous value. This error continues into the next timesteps and has a diverging character. The density, and with that pressure, will be affected. The problem is caused by a sudden change in pressure at volume Ω_{i-1}, which is induced by gravity acting on water, not on air, with $\rho_{\mathrm{I}} \gg \rho_{\mathrm{II}}$. So, without taking precautions, a numerical solution cannot be found when computing a two-fluid flow with gravity for $\rho_{\mathrm{I}} \neq \rho_{\mathrm{II}}$, using the ghost-fluid method. Notice that pressure oscillations do not occur for the trivial case $\rho_{\mathrm{I}} = \rho_{\mathrm{II}}$.

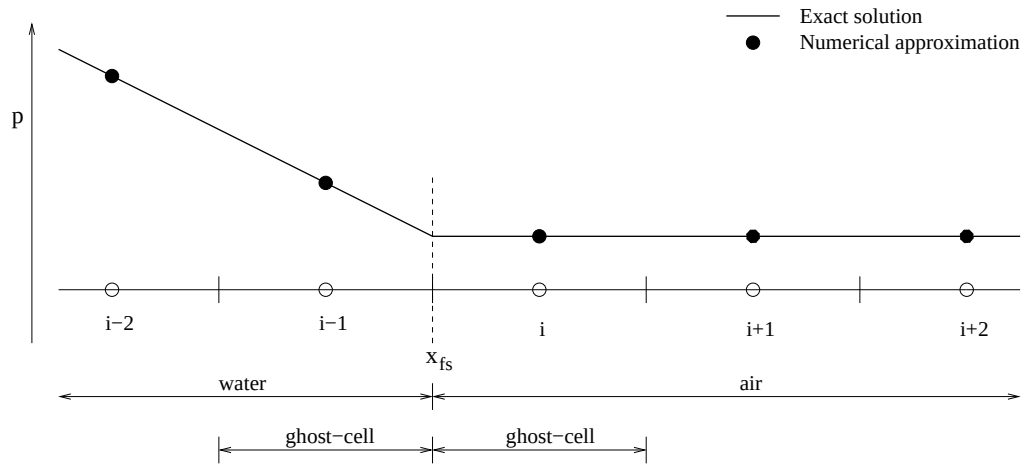

Figure 2.18: Converged pressure distribution near the interface.

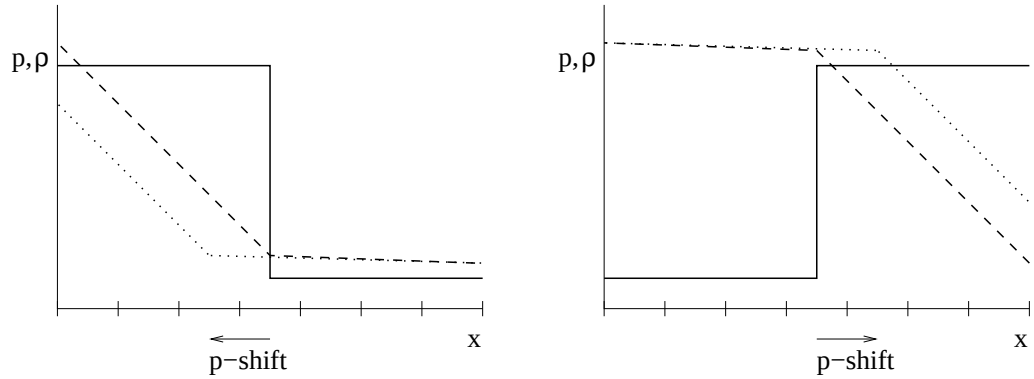

Figure 2.19: Pressure shift. The solid line denotes bulk-density, the dashed line physical pressure and the dotted line shifted pressure.

2.8.1.2 Ghost-fluid with pressure-shift To overcome the erroneous large flux of air in the cells near an interface, several fixes will be proposed. The first successes were made by the author with an on purpose made error in the computation of the pressure. The pressure distribution is shifted one cell from the low density area into the high density area, as shown in figure 2.19. With figure 2.16 in mind, pressure will be computed with:

$$p_i = \begin{cases} p(\rho_{i+1}) & \rho_{\mathrm{I}} > \rho_{\mathrm{II}}, \\ p(\rho_{i-1}) & \rho_{\mathrm{I}} < \rho_{\mathrm{II}}. \end{cases} \tag{2.8.8}$$

As seen in the former section, pressure oscillations are initiated when computing the momentum flux for the low-density medium near the interface. Recall (2.8.7). With the introduction of the pressure-shift, $p^n_{i-1} \approx p^n_i$ and $\frac{\Delta t}{h}(p^n_{i-1} - p^n_i) \approx -\Delta t \frac{\rho_i^{n+1}}{\mathrm{Fr}^2}$. Pressure oscillations do not occur anymore. Numerical tests have been done for a tube with $h = 1/10$, $h = 1/20$ and $h = 1/40$ using the following conditions: $u_{\mathrm{init}} = 0$, $p_{\mathrm{init}} = 1$, $\rho_{\mathrm{I}} = 1.0$, $\rho_{\mathrm{II}} = 0.001$ and $\mathrm{Fr} = 0.43$, using single precision numerical computation[1]. It has to be noted that the CFL criterion has to be less than or equal to 0.02 in order to get numerical stability. From the initial solution, the state values oscillate toward the static solution (figure 2.23). When converged, the pressure at the bottom of the tube is 3.1811 for $h = 1/10$, 3.4500 for $h = 1/20$ and 3.5332 for $h = 1/40$. The approximate equation (2.8.4) gives 3.7069. This result suggests that the accuracy increases for cell width $h \to 0$, which

[1]Single precision REAL in Fortran 77, without compiler optimizations.

is as expected. The order of accuracy with gridrefinement can be calculated with

$$\frac{\Delta_1}{\Delta_2} = \frac{\alpha h_1^{p_{1,2}}}{\alpha h_2^{p_{1,2}}}, \tag{2.8.9}$$

where Δ is the difference of the numerical and exact solution, and $p_{1,2}$ is the order of accuracy. The order of accuracy for going from cell size $h = \frac{1}{10}$ to $h = \frac{1}{20}$ is 1.0333 and for $h = \frac{1}{20}$ to $h = \frac{1}{40}$ is 0.5629. The velocity distribution does not tend to go to zero for $h \to 0$, but converges to a finite error after infinite grid refinement (figure 2.8.1.2). With that, the location of the interface as indicated by the level-set method should be questioned, for the level-set is advected by the erroneous velocity.

The residual of unsteadyness,

$$\sum_{i=0}^{n+2} \left(q_i^{n+1} - q_i^n \right), \quad q_i = \begin{pmatrix} u_i \\ p_i \end{pmatrix}, \tag{2.8.10}$$

does not drop to machine precision (figure 2.22). The on-purpose-made pressure-shift are expected to cause the numerical stability. The pressure-shift is also expected to cause the inability to converge to machine precision.

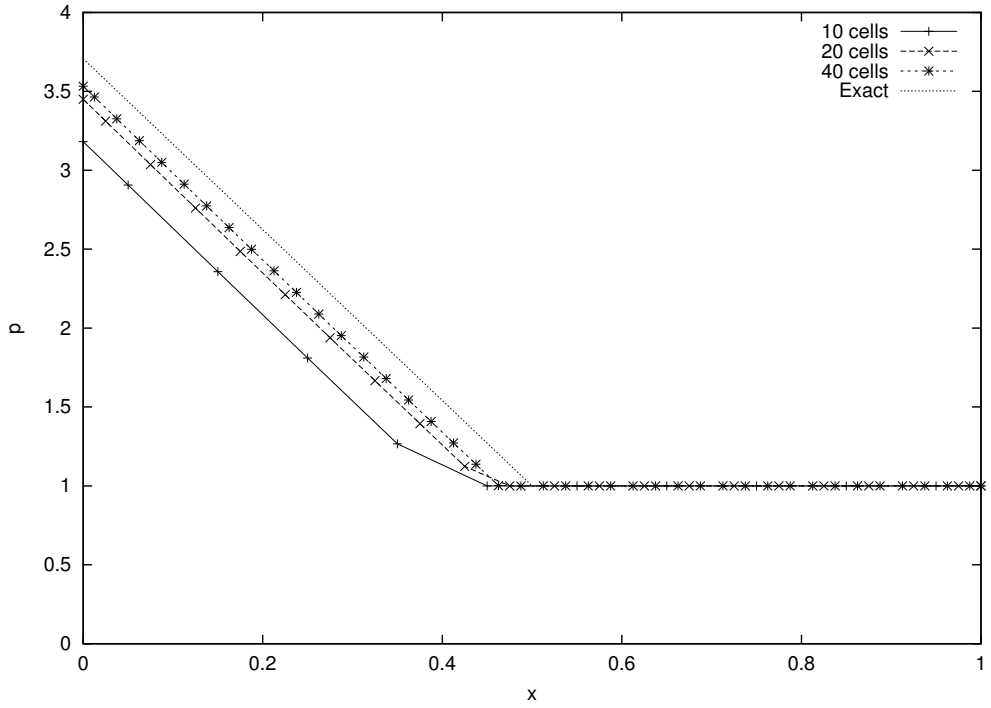

Figure 2.20: Test model 2: Converged solution of pressure p, ghost-fluid with pressure-shift. Grid refinement leads to increase of accuracy.

This pressure-shift method seems to be used to produce the results published in [18]. Although not mentioning the applied fix, the results look remarkably the same, having the same (error) properties. All velocities in the cell containing the interface have a change of sign, all pressures found at the bottom of the tube tend to be too low, comparing with the exact result and accuracy of the pressure at the bottom increases when increasing the amount of cells. The most remarkable similarity is the shift to the left in pressure distribution.

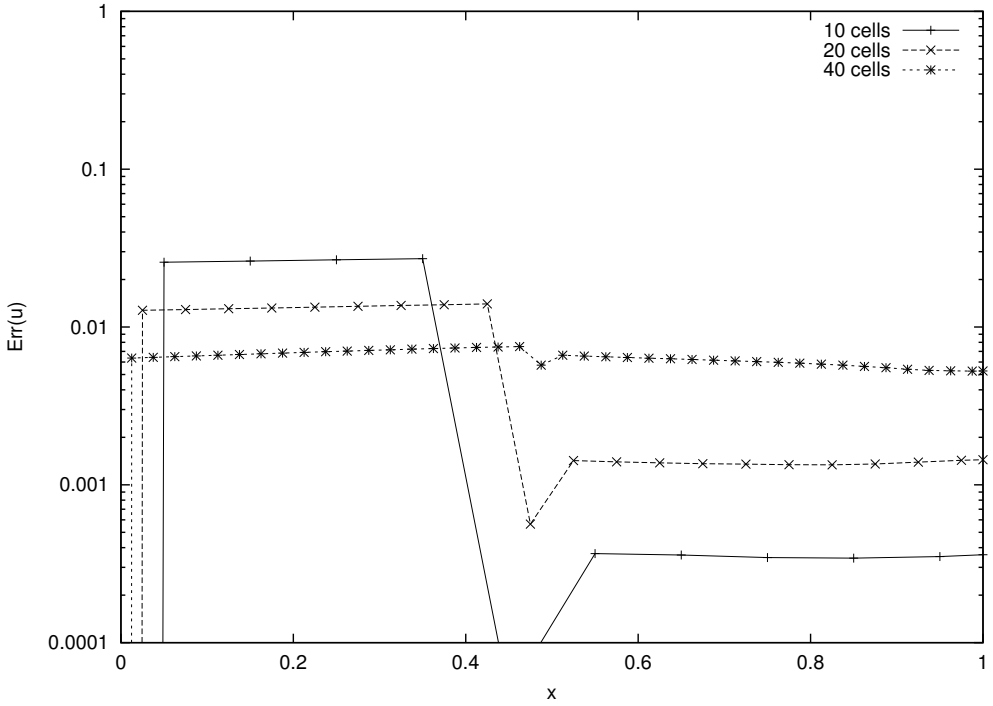

Figure 2.21: Test model 2: Error of the flow velocity u, ghost-fluid with pressure-shift. Grid refinement does not lead to decrease of error.

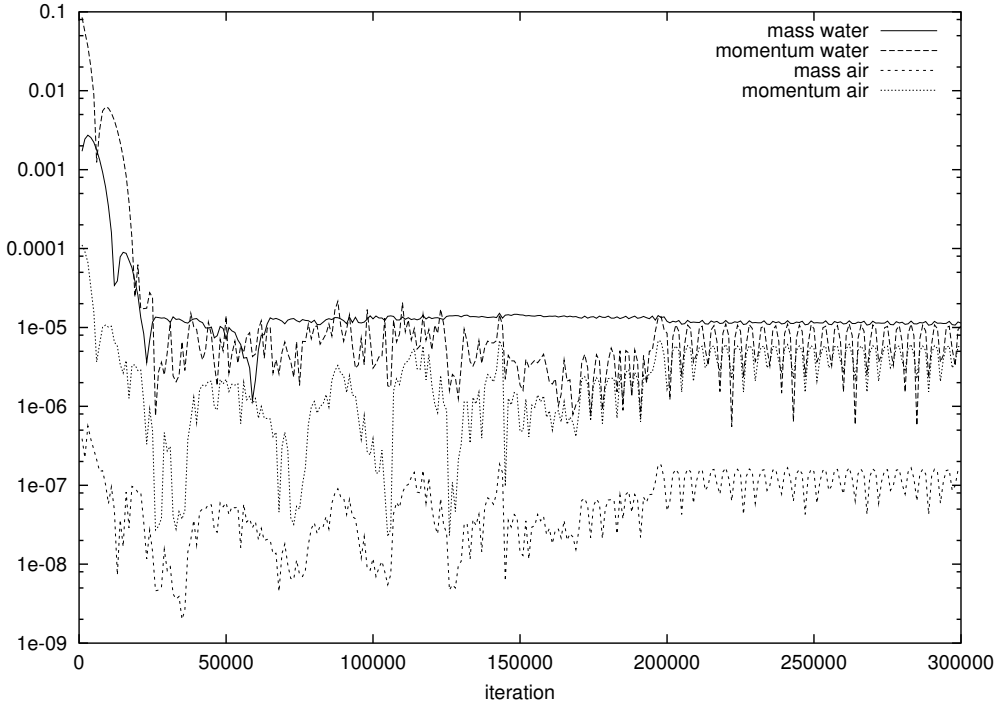

Figure 2.22: Time history of residual of unsteadyness, using ghost-fluid and pressure-shift.

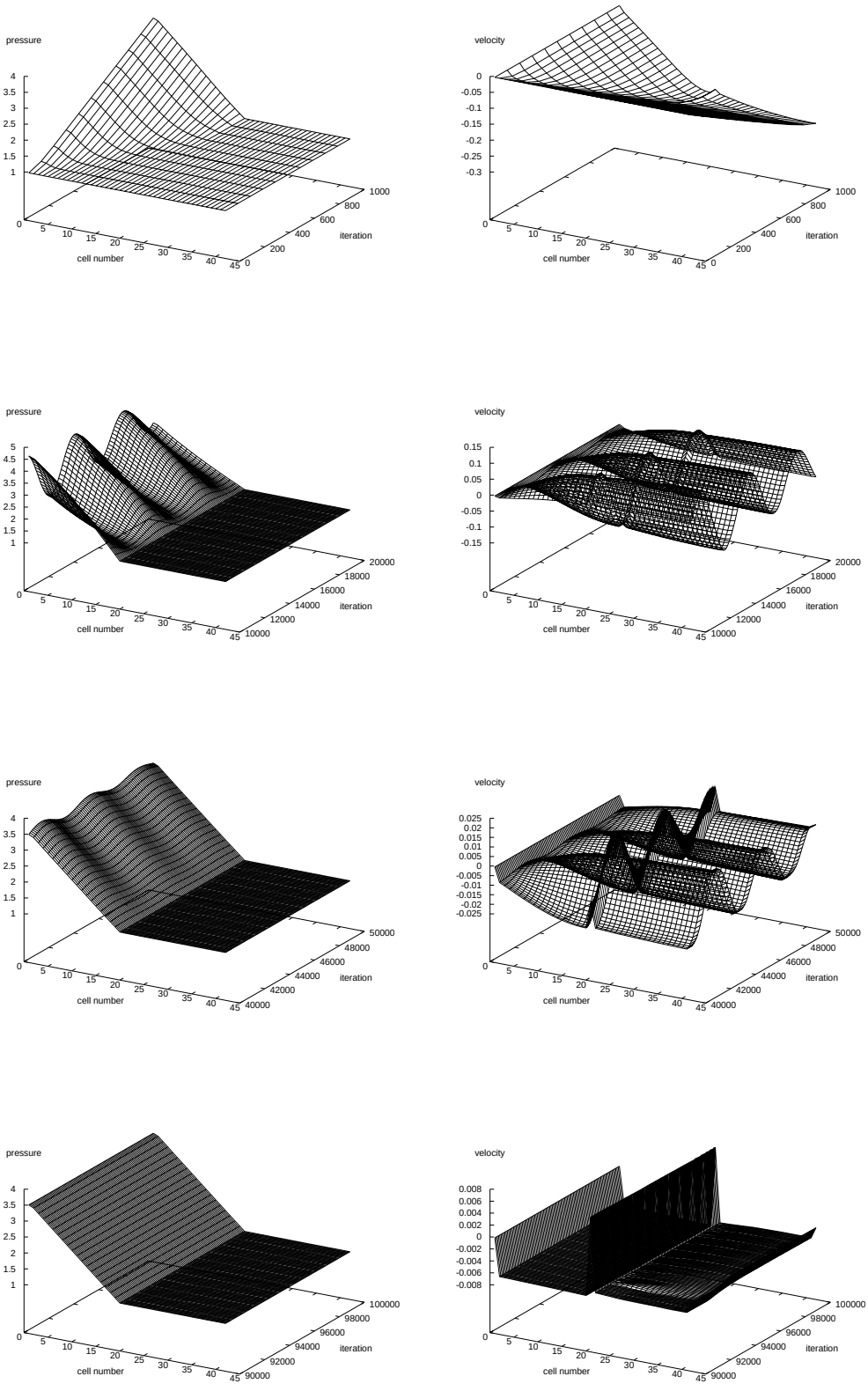

Figure 2.23: Pressure and velocity development.

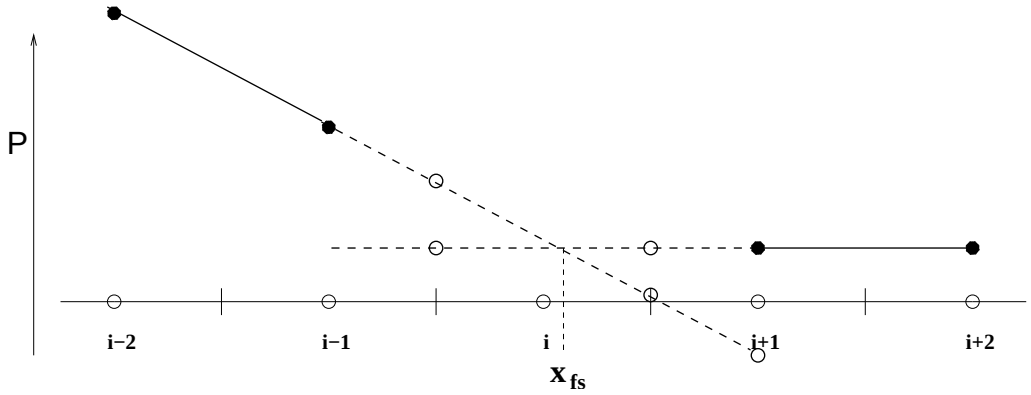

Figure 2.24: Pressure extrapolation of water and air in the cell containing an interface.

2.8.1.3 Ghost-fluid with extrapolation of water and air Another fix to overcome the too large pressure raise for air is by extrapolating the pressure for each medium to the other side of the free-surface. The individual fluxes of the cell faces will be computed not using the real pressures acting on the cell faces, but extrapolated pressure values. For each cell face which borders an interface-containing cell, two situations can be distinguished; the cell is either the left or the right face of that cell. As extrapolation technique, a simple linear extrapolation algorithm is chosen. The extrapolated water pressure values for the left and right cell faces read:

$$p_l = p_{i-1}, \tag{2.8.11}$$
$$p_r = 2p_{i-1} - p_{i-2}, \tag{2.8.12}$$

for the left face of a cell of type 3 and

$$p_l = 2p_{i-1} - p_{i-2}, \tag{2.8.13}$$
$$p_r = 3p_{i-1} - 2p_{i-2}, \tag{2.8.14}$$

for the right face of a cell of type 3. In a similar manner, the extrapolated air pressure values for the left and right cell faces read:

$$p_l = 3p_{i+1} - 2p_{i+2}, \tag{2.8.15}$$
$$p_r = 2p_{i+1} - p_{i+2}, \tag{2.8.16}$$

for the left face of a cell of type 3 and

$$p_l = 2p_{i+1} - p_{i+2}, \tag{2.8.17}$$
$$p_r = p_{i+1}, \tag{2.8.18}$$

for the right face of a cell with an interface. In figure 2.24, the cell with an interface is shown as cell i. The cell faces bordering a cell with an interface and the extrapolated pressure values (shown as open dots) are located at $i - \frac{1}{2}$ and $i + \frac{1}{2}$. All other cell faces, not bordering a cell with an interface and not affected by the ghost-fluid method, are treated normally without any extrapolation techniques or ghost-fluid method. Now the individual fluxes of each cell can be computed. When updating the flow field with use of the forward Euler scheme, the cells with an interface make use of the density values, calculated from the extrapolated pressures, with the equations of state (2.2.5) and (2.2.6). A numerical result is shown in figure 2.25. The first couple of iteration sweeps looks promising. However, after about 600 iteration sweeps, the water pressure starts to diverge. The pressure at the interface shows a jump and isn't continuous anymore. When continuing the iteration process, the pressure at the bottom diverges to infinity.

What's going wrong? By extrapolating both water and air, the pressure values of water and air are not necessarily the same anymore and with that the implicit Dirichlet condition $p_w(x_{\mathrm{fs}}) =$

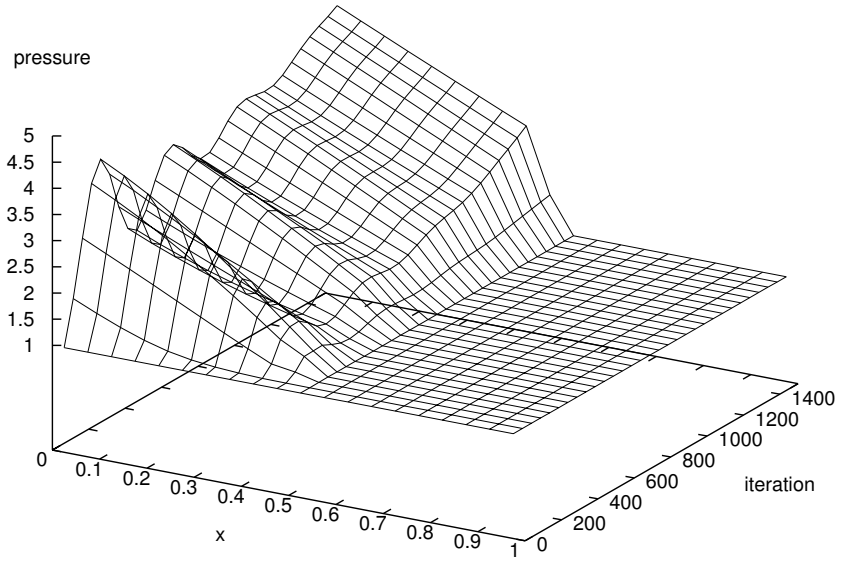

Figure 2.25: Pressure explosion when using extrapolation techniques on both water and air.

$p_a(x_{\text{fs}})$ is lost. But probably the biggest problem is having pressure at the left of the interface which is decoupled from the Dirichlet boundary condition $p(1,t) = 1$. In other words, the water pressure is set once by an initial condition, but isn't bounded by any boundary condition.

2.8.1.4 Ghost-fluid with extrapolation of air only To overcome the loss of pressure boundary condition through the material interface, the solution is simple. Looking back at the reason for concidering extrapolation techniques, in the first place this was to eliminate the great pressure jump which induces a large mass flux and momentum flux for air. The pressures at the cell faces, used for computing the water flux are computed with use of equation (2.8.11) and (2.8.14). The pressures at the cell faces, used for computing the air flux are computed with use of equations (2.8.15) to (2.8.18). At the time of computing the individual water and air fluxes in the ghost cells, the implicit Dirichlet condition is lost, but after the unique update, the implicit Dirichlet condition is restored and valid. But of major importance is that the whole flow field "feels" the boundary condition $p(1,t) = 1$ again. Note that the Dirichlet boundary condition $u(0,t) = 0$ was never affected, due to the fact that only pressure has been extrapolated, not velocity. When performing numerical tests, a new problem emerges. Known from the exact static solution, the velocity of the whole flow field should be exactly equal to zero. However, due to first order accuracy, the converged velocity in the whole flow field never reaches exactly zero. The level-set function will be advected by this non-zero velocity error. When computing on very coarse grids, the impact is tremendous. As seen before, the static solution has been reached when the oscillation of the pressure and velocity has stopped. However, due to the non-zero first-order velocity error, the interface is starting to move, because the level-set function is advected by this velocity error. An example is given in figure 2.26. The figure clearly shows the convection of the interface during computation. Every time the interface passes a cell face, the flow field starts to oscillate again, until a new unsteady (not a static!) equilibrium has been established. When the interface reaches the vicinity of the left boundary, the computation becomes unstable and the numerical solution starts to diverge. This unstable behaviour is clearly visible by examining the residual, shown in

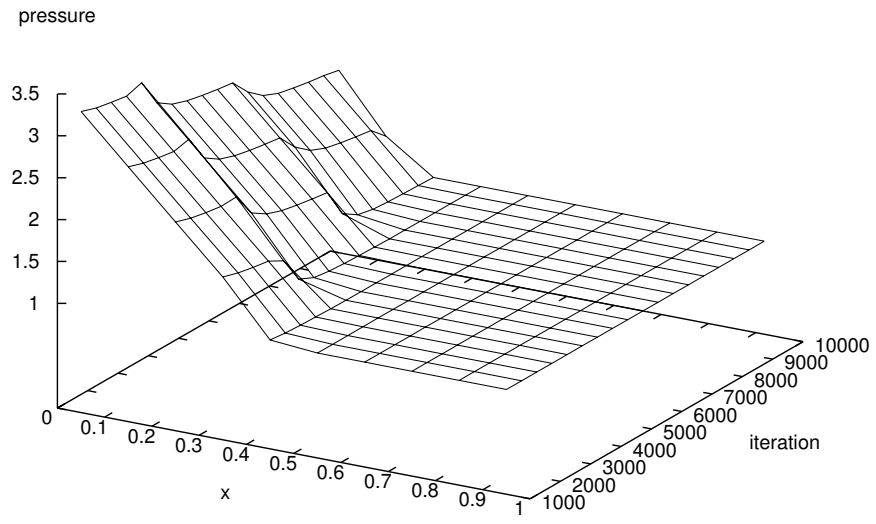

Figure 2.26: Endless convection of the interface due to first-order error of the velocity.

figure 2.27. A fix for this unfavourable convection is given in the next subsection.

2.8.1.5 Ghost-fluid with extrapolation of air only and level-set convection limiter Convection of the interface induced by real flow velocity is desirable. But as shown in the previous subsection, convection due to first-order errors is totally unacceptable. To overcome this problem, the velocity u_i, used for convection of the level-set function, will be truncated to $\widehat{u}_i$, containing as few significant numbers as first-order accuracy prescribes. Instead of using (2.4.1), the level-set function will be convected by

$$\frac{\partial \rho\phi}{\partial t} + \widehat{u}\frac{\partial \rho\phi}{\partial x} = 0. \tag{2.8.19}$$

This formulation allows physical movement of the interface, but limits erroneous movements. Numerical results are plotted in figures 2.28 to 2.31. Recalling for this test case the approximate exact solution at the bottom of the tube, $p(0) = 3.7069$, figure 2.28 exhibits good agreement with this exact solution. All pressure values points of the 10, 20 and 40 cell grids lay on the same line, except in the vicinity of the interface. This can be attributed to the resolution of the used grids. Grid refinement improves the accuracy. The level-set values for the 10, 20 and 40 cell grids lay on the same line (figure 2.30), and when the computation has fully converged, the residual has been reduced to the order of machine precision (figure 2.31). Still, point of concern is the large error for the velocity of the flow field (figure 2.29). Although grid refinement reduces the error, high-order or complete other methods could be necessary to improve the accuracy.

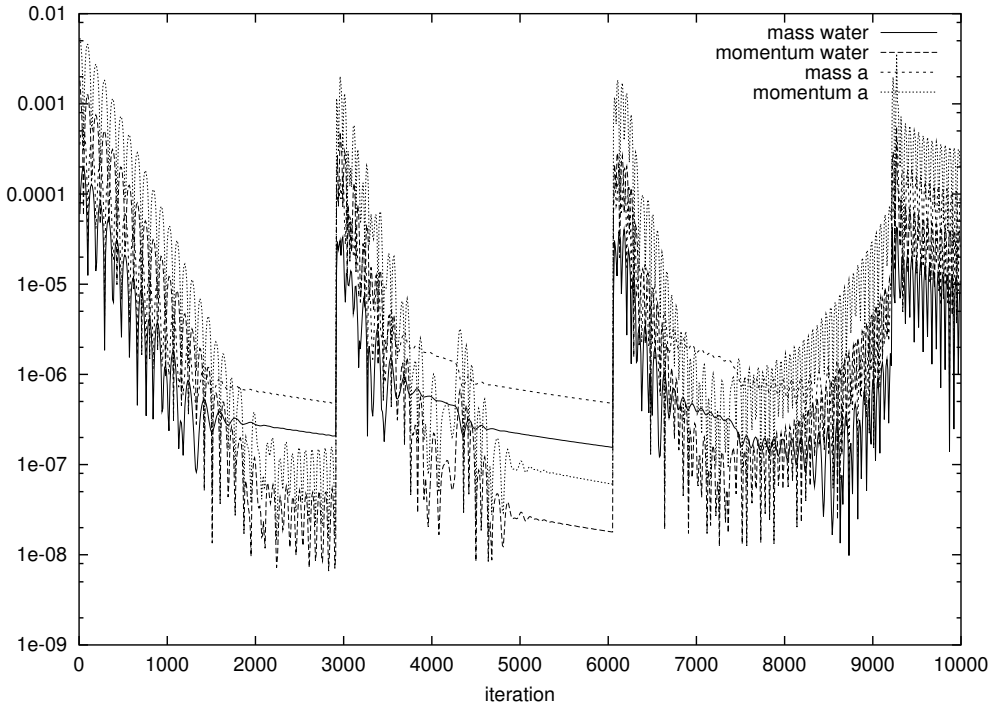

Figure 2.27: Residual during iteration process for $h = \frac{1}{10}$. When the interface passes a cell face, the flow field starts to oscillate again.

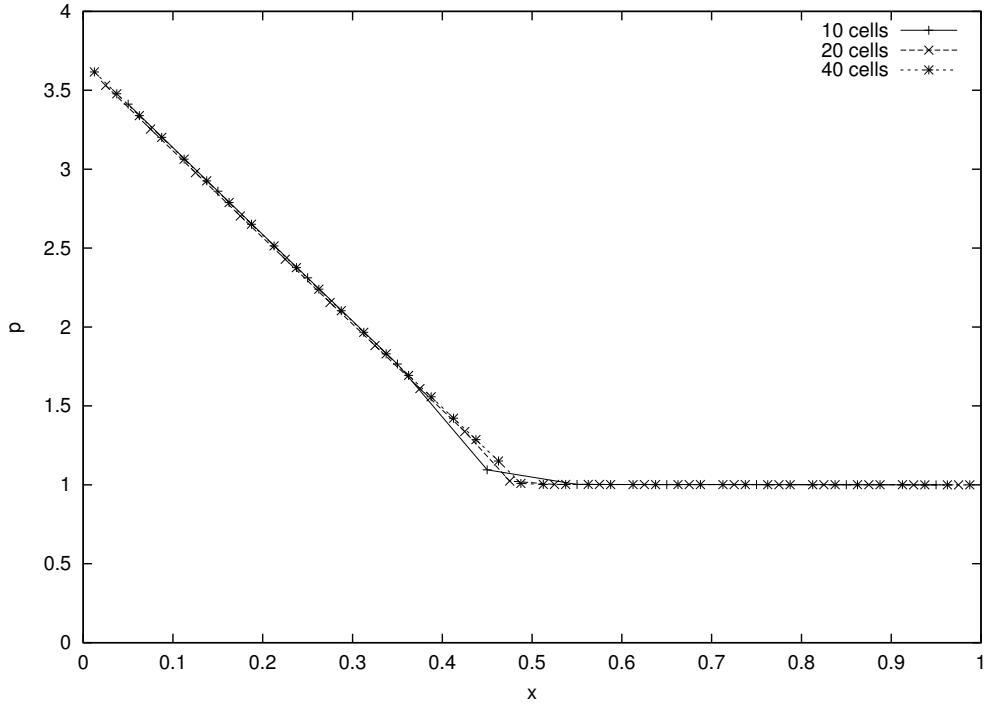

Figure 2.28: Test model 2: Converged solution of pressure p, ghost-fluid with extrapolation of air only and level-set convection fix.

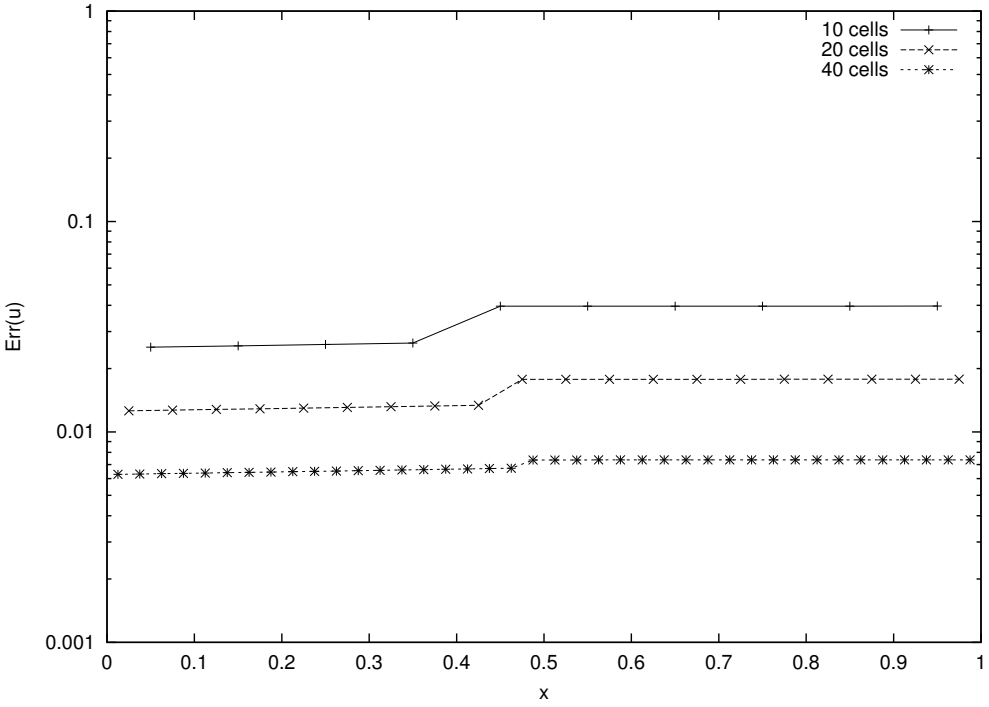

Figure 2.29: Test model 2: Error of the flow speed u, ghost-fluid with extrapolation of air only and level-set convection fix.

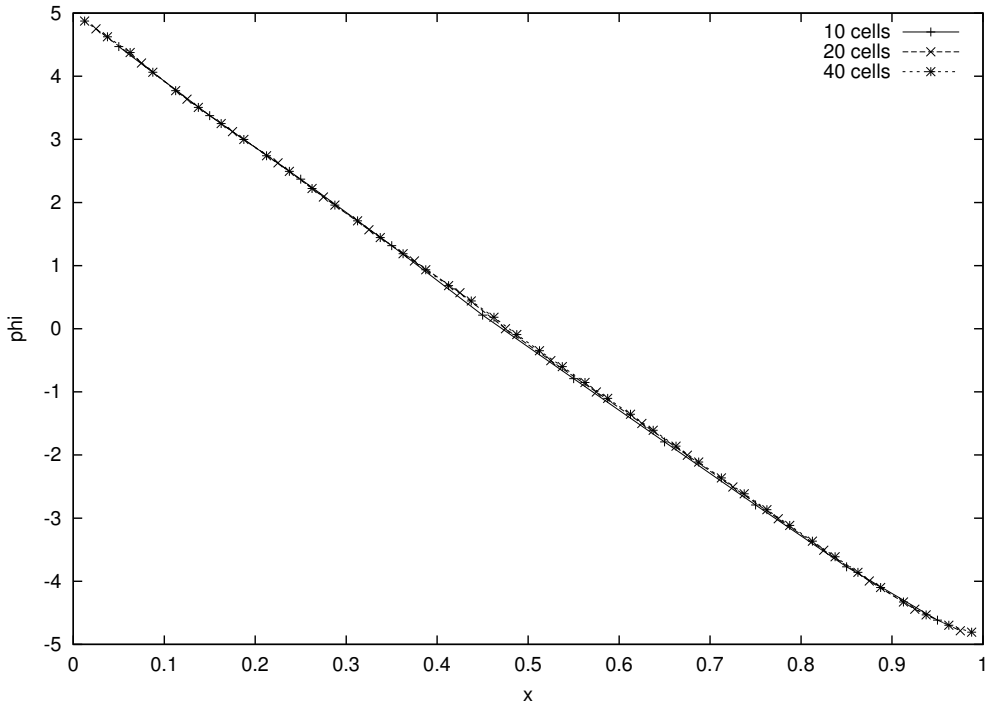

Figure 2.30: Test model 2: Converged solution of the level-set values ϕ, ghost-fluid with extrapolation of air only and level-set convection fix.

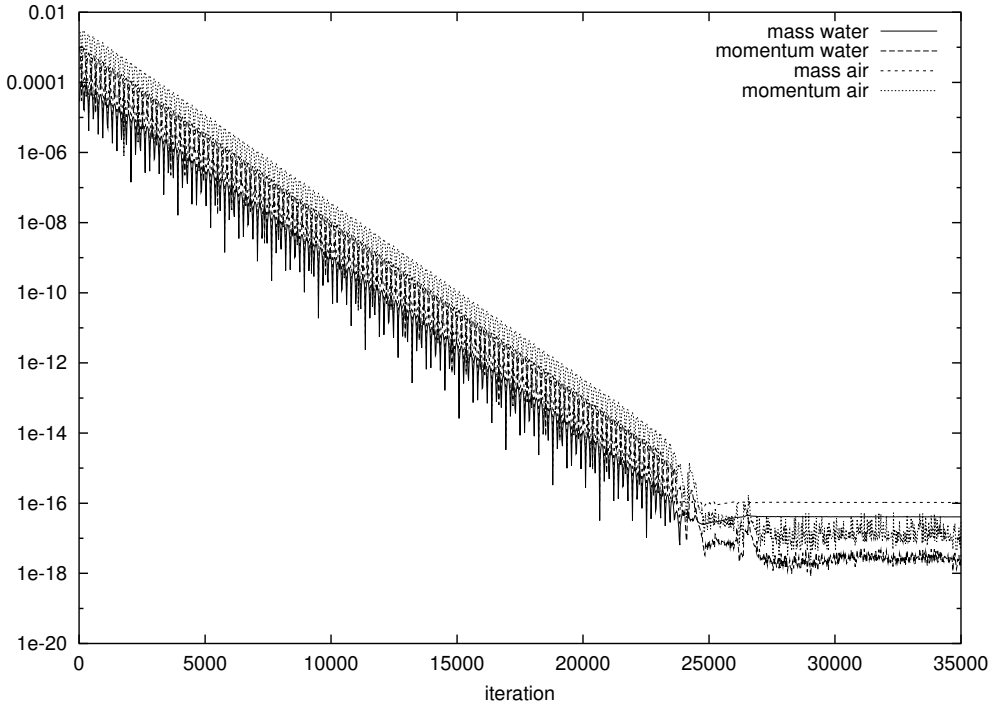

Figure 2.31: Test model 2: Residue during iteration process for the 20 cells model. The problem converges to machine zero, $L_1 < 10^{-16}$.

h	a	u	ε
0.1	10.0	1.0	0.05
0.05	10.0	1.0	0.025
0.025	10.0	1.0	0.0125

Table 2.1: Estimation of the numerical diffusion with h, a and ε as the cell width, speed of sound and the numerical diffusion factor respectively.

2.8.1.6 Diffusion (Navier Stokes) During the iteration process, the flow field shows an oscillatory behaviour until the static solution has been reached. Since we are only interested in the static solution, we want to find a method to decrease computation time. It is expected that the iteration process will be finished sooner by adding a diffusion term that acts as a damping term. The converged static solution should be the same as the one retrieved by using the Euler equations[2]. The extra diffusion term has to be dominant against the already present numerical diffusion, induced by the used upwind differencing scheme. An estimation of the numerical diffusion can be calculated by writing the test differential equation

$$\frac{\partial q}{\partial t} + u\frac{\partial q}{\partial x} = 0 \tag{2.8.20}$$

into the forward Euler upwind differencing equation

$$\frac{q_i^{n+1} - q_i^n}{\Delta t} + u\frac{q_i^n - q_{i-1}^n}{\Delta x} = 0, \tag{2.8.21}$$

and rewrite it into a central differencing equation

$$\frac{q_i^{n+1} - q_i^n}{\Delta t} + u\frac{q_{i+1}^n - q_{i-1}^n}{2\Delta x} + u\frac{2q_i^n - q_{i-1}^n - q_{i+1}^n}{2\Delta x} = 0. \tag{2.8.22}$$

Rewriting yields:

$$\frac{q_i^{n+1} - q_i^n}{\Delta t} + u\frac{q_{i+1}^n - q_{i-1}^n}{2\Delta x} = \frac{u\Delta x}{2}\frac{q_{i-1}^n - 2q_i^n + q_{i+1}^n}{(\Delta x)^2}. \tag{2.8.23}$$

Writing (2.8.23) into a differential form gives

$$\frac{\partial q}{\partial t} + u\frac{\partial q}{\partial x} = \varepsilon\frac{\partial^2 q}{\partial x^2}, \tag{2.8.24}$$

with $\varepsilon = \frac{u\Delta x}{2}$. The right hand side of (2.8.24) is called the numerical diffusion term or stabilizing term. When adding the physical diffusion term $\frac{1}{\mathrm{Re}}\frac{\partial^2 q}{\partial x^2}$ to extend the Euler equation into a Navier-Stokes equation, the term $1/\mathrm{Re}$ must be greater than ε in order to be effective. Estimation of ε is given in table 2.1. With the speed of sound defined as 10.0, the chosen characteristic low-speed value for u is estimated at 1.0. According to table 2.1, the Reynolds number shouldn't be higher than approximately 100. When examining the numerical results, some interesting aspects can be observed. Looking at table 2.2, the drop of cycles failed to materialize. For all grid spacings, a Reynolds number of 10 gave a slight advantage. The advantage becomes bigger, when the cell width h goes to zero. This can be attributed to the decrease of artificial viscosity. Adding a little viscosity is more noticeable. If the flow becomes too viscous however, the number of iterations increases. When looking at the pressure results, another phenomenon can be seen. For $\mathrm{Re} \geq 10$, all solutions for pressure values are the same. However, for $\mathrm{Re} = 1$, the pressure values are smoothed into a curve and diverge from the exact result. Figures 2.32 and 2.33 illustrates this behaviour. This might be explained by

Overall, adding viscosity as a damping term for speeding up the iteration process gives only a slight advantage and only when taking the right value of Re. In case the Reynolds number is taken too high, the damping term seems ineffective. When choosing the Reynolds too low, the final result is affected and interpreting the solution should be done with care.

[2]Without diffusion, a static solution will never be reached. The fluid columns remain oscillating.

Cells	10	20	40
Euler	5660	23710	94640
Re=100	5980	23760	93780
Re=10	5250	20750	70870
Re=1	5990	33870	211216

Table 2.2: Number of iterations performed to reach machine precision $L_1 < 10^{-16}$.

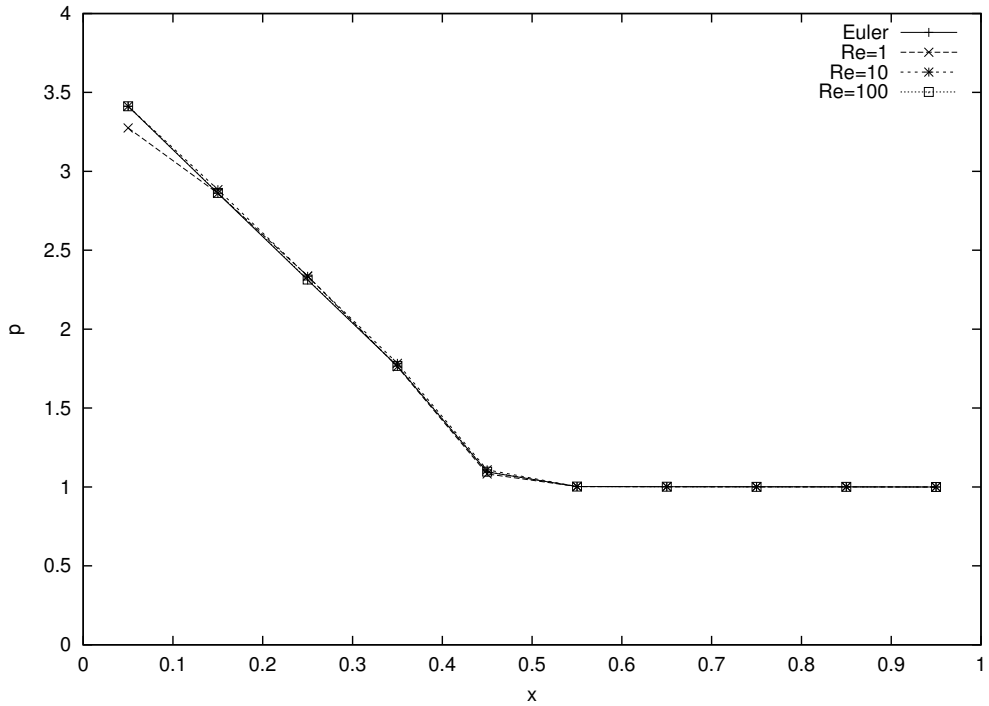

Figure 2.32: Model 2: Navier-Stokes results, compared with Euler results for pressure with a 10 cells grid.

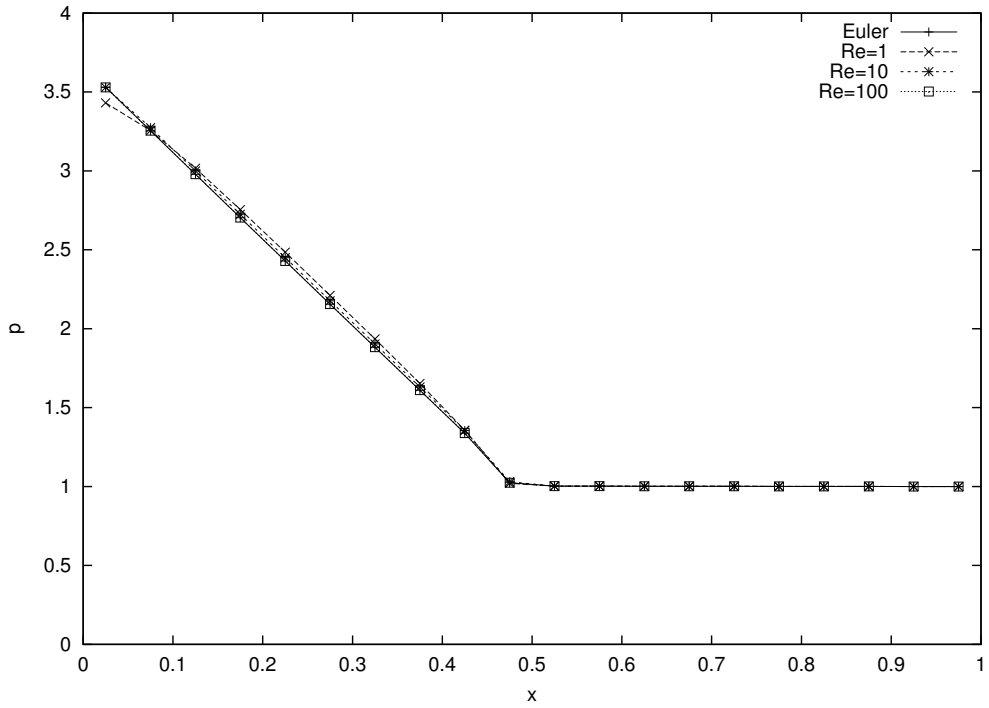

Figure 2.33: Model 2: Navier-Stokes results, compared with Euler results for pressure with a 20 cells grid.

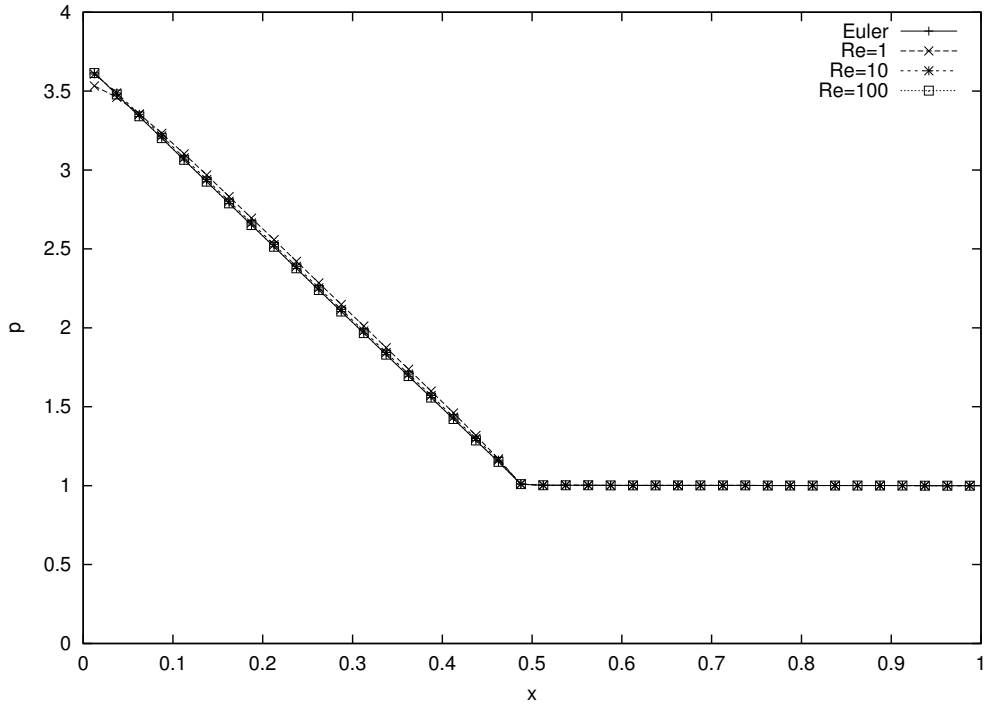

Figure 2.34: Model 2: Navier-Stokes results, compared with Euler results for pressure with a 40 cells grid.

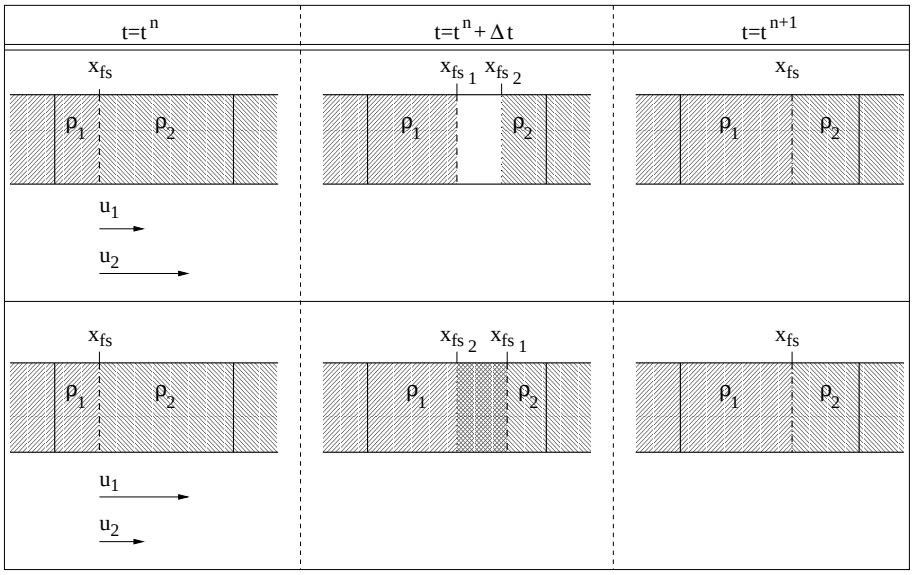

Figure 2.35: Non-conservative behaviour of the ghost-fluid method.

2.9. Ghost-fluid induced mass error

As stated before, in the ghost cells, the conservation laws are applied to virtual (ghost) amounts of water and air, not to the real physical amounts. So, conservation of the real amounts of mass and momentum is not guaranteed automatically. In fact, mass conservation is lost in case the velocities of the two fluids in the ghost cells are different. This is visualized in figure 2.35. Time t^n is taken as starting point. When performing one time step Δt, the solutions for both fluids are computed separately, according to the ghost-fluid method. In case $u_1 < u_2$, a void is created, and in case $u_1 > u_2$, the fluids tend to overlap. At the new time level t^{n+1}, the solutions are made unique with (2.6.1), resulting in a mass gain for $u_1 < u_2$ and a mass loss for $u_1 > u_2$. An expression for this type of mass error can be found in the following way. Recall the mass equation in integral form:

$$\int_{\Omega_i} \frac{\partial \rho}{\partial t} dx + (\rho u)_{i+\frac{1}{2}} - (\rho u)_{i-\frac{1}{2}} = 0. \tag{2.9.1}$$

Rewriting the mass equation (2.9.1), with the rate of change of mass inside control volume Ω_i written at the lefthand side, and the net mass flow into the control volume through surfaces $\partial\Omega_{i+\frac{1}{2}}$ and $\partial\Omega_{i-\frac{1}{2}}$ at the righthand side gives

$$\frac{\Delta m}{\Delta t} = (\rho u)_{i-\frac{1}{2}} - (\rho u)_{i+\frac{1}{2}} \,. \tag{2.9.2}$$

Substituting density and still non-unique ghost-fluid velocity in (2.9.2) for $t = t^n + \Delta t$, according to figure 2.35 gives

$$(\Delta m)_1 = \big((\rho_{\mathrm{I}} u_{\mathrm{I}})^n_{i-\frac{1}{2}} - (\rho_{\mathrm{II}} u_{\mathrm{II}})^n_{i+\frac{1}{2}} \big) \Delta t. \tag{2.9.3}$$

Substituting density and unique ghost-fluid velocity in (2.9.2) for $t = t^{n+1}$ gives

$$(\Delta m)_2 = \big((\rho_{\mathrm{I}} u)^n_{i-\frac{1}{2}} - (\rho_{\mathrm{II}} u)^n_{i+\frac{1}{2}} \big) \Delta t. \tag{2.9.4}$$

The error for the increase of mass for the cell containing an interface can be defined as the difference between the increase of mass inside the cell before and after making the velocity unique for one time step; i.e.

$$\mathrm{Err}_m = (\Delta m)_1 - (\Delta m)_2. \tag{2.9.5}$$

When substituting (2.6.1) for u into (2.9.5), the expression for the mass error yields

$$\mathrm{Err}_m = (u_\mathrm{I} - u_\mathrm{II}) \left(\rho_\mathrm{I} (1-\alpha) + \rho_\mathrm{II}\alpha\right)\Delta t. \tag{2.9.6}$$

It is obvious to see that no mass error due to unique-making of the velocity solution will be made in case $u_\mathrm{I} = u_\mathrm{II}$.

2.9.1 Numerical results

In order to see if this mass error, induced by making the solution unique, really occurs, the difference $u_\mathrm{I} - u_\mathrm{II}$ (figure 2.36) and the value Err_m (figure 2.37) is measured during the numerical computation of test model 2 of section 2.8. Figure 2.36 shows that in a cell containing an interface, the velocity solutions u_I and u_II are not equal. This error carries over into the mass equation. The difference between the increase of mass inside the cell before and after making the velocity unique is not equal to zero (figure 2.37). Conservation of bulk mass for the whole system should be questioned.

To see if grid refinement results in better solutions, the order of accuracy can be calculated with (2.8.9). For the velocity, the order of accuracy for going from cell size $h = \frac{1}{10}$ to $h = \frac{1}{20}$ is 0.6413 and for $h = \frac{1}{20}$ to $h = \frac{1}{40}$ is 0.7352. For the mass error Err_m, the order of accuracy for going from cell size $h = \frac{1}{10}$ to $h = \frac{1}{20}$ is 0.1561 and for $h = \frac{1}{20}$ to $h = \frac{1}{40}$ is 0.2911.

This illustrates a weak point of the ghost-fluid method. Summarizing, the conservative level-set method suffers from pressure oscillations. The ghost-fluid method appears to be a good fix for these oscillations as seen in section 2.7.1.2. The ghost-fluid method works outstanding in case the velocity of the whole flowfield is constant. But the Achilles' heel of the ghost-fluid method emerges when the velocity is not constant, for example, when the flowfield oscillates. Another example of mass loss due to the ghost-fluid method can be found in [20].

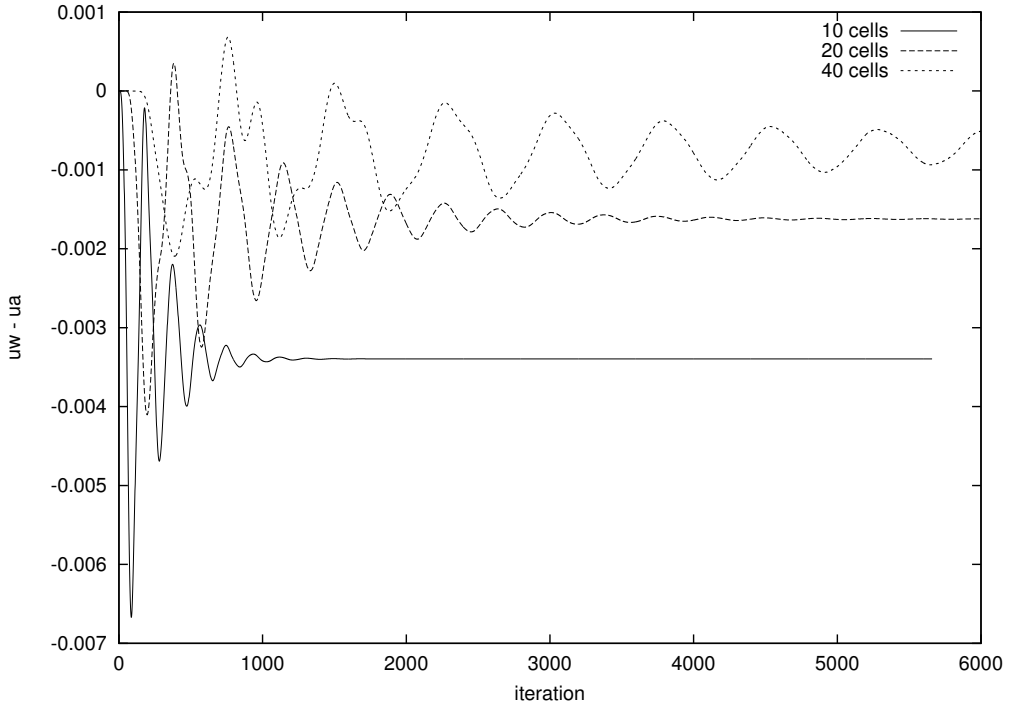

Figure 2.36: Difference of the updated solutions u_{I} and u_{II} from the ghost cells.

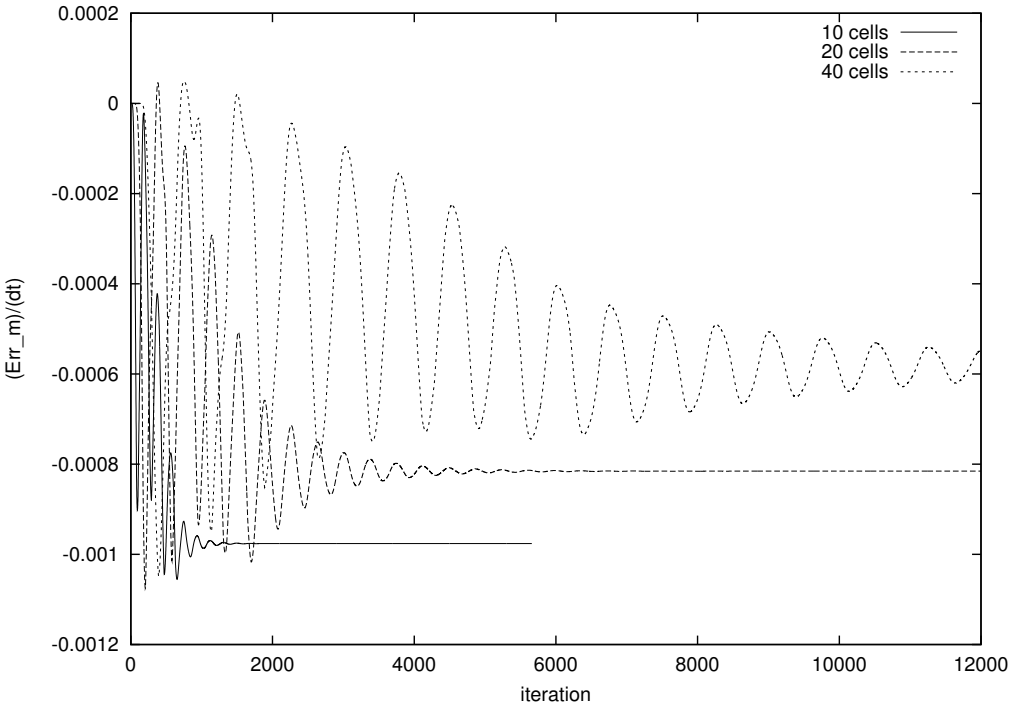

Figure 2.37: Difference between the rate of change of mass inside the cell before and after making the velocity solution unique.

Chapter 3
Mass-fraction method

3.1. Conservative method using mass of fluid fraction

As seen in chapter 2, the level-set method and the ghost-fluid method have some serious drawbacks. In short, the most important disadvantages of the level-set method are:

- The level-set values do not have a physical origin.
- Applying the level-set method in fully conservative way, the method shows to be numerically unstable, if no special precautions are taken.
- Applying the ghost-fluid method, conservation of mass is lost near an interface.

So, a conservative method which conserves the bulk mass and mass of each individual medium is highly desirable. According to Abgrall and Karni [1] it is not possible to compute a two-fluid flow in conservative manner without getting pressure oscillations. However, this section will show a method which computes the entire two-fluid flow with conservative equations without getting pressure oscillations. The method is first described in [4]. The equation in integral form is

$$\int_\Omega q_t dx + (f(q))_{\partial\Omega_{right}} - (f(q))_{\partial\Omega_{left}} = 0, \tag{3.1.1}$$

$$q = \begin{pmatrix} \rho \\ \beta\rho \\ \rho u \end{pmatrix}, \qquad f(q) = \begin{pmatrix} \rho u \\ \beta\rho u \\ \rho u^2 + p \end{pmatrix},$$

where β denotes the mass fraction of one of the two fluids. The exact definition will be described later. Written in differential form, the equation set becomes

$$\frac{\partial \rho}{\partial t} + \frac{\partial(\rho u)}{\partial x} = 0, \tag{3.1.2a}$$

$$\frac{\partial(\beta\rho)}{\partial t} + \frac{\partial(\beta\rho u)}{\partial x} = 0, \tag{3.1.2b}$$

$$\frac{\partial(\rho u)}{\partial t} + \frac{\partial}{\partial x}\left(\rho u^2 + p\right) = 0. \tag{3.1.2c}$$

In case of a water-air flow problem, the two equations for conservation of mass

$$\frac{\partial \rho}{\partial t} + \frac{\partial(\rho u)}{\partial x} = 0, \tag{3.1.3a}$$

$$\frac{\partial(\beta\rho)}{\partial t} + \frac{\partial(\beta\rho u)}{\partial x} = 0 \tag{3.1.3b}$$

are equivalent to the mass equations

$$\frac{\partial \rho_{\mathrm{I}}}{\partial t} + \frac{\partial (\rho_{\mathrm{I}} u)}{\partial x} = 0, \tag{3.1.4a}$$

$$\frac{\partial \rho_{\mathrm{II}}}{\partial t} + \frac{\partial (\rho_{\mathrm{II}} u)}{\partial x} = 0. \tag{3.1.4b}$$

Let's have a closer look at these equations. The mass fraction β is defined as

$$\beta = \alpha \frac{\rho_{\mathrm{I}}}{\rho}, \tag{3.1.5}$$

where α is the Volume-of-Fluid fraction. For this case, the Volume-of-Fluid is the volume of water. Likewise, it holds

$$(1-\beta) = (1-\alpha)\frac{\rho_{\mathrm{II}}}{\rho}. \tag{3.1.6}$$

To get an expression for the density in terms of mass fraction and pressure, the Volume-of-Fluid fraction α is eliminated from equations (3.1.5) and (3.1.6):

$$\frac{1}{\rho} = \frac{\beta}{\rho_{\mathrm{I}}} + \frac{1-\beta}{\rho_{\mathrm{II}}}. \tag{3.1.7}$$

Note that equation (3.1.7) can also be written as $V = \beta V_1 + (1-\beta)V_2$, where V denotes the volume, according to the definition $V = 1/\rho$. The difference between equation (2.7.6) and (3.1.6) is that the bulk density is not any more determined by the artificial level-set function, which itself is a function of x. Even so, the Volume-of-Fluid fraction α depends now on the mass fraction β instead of on the level-set function. Most important advantage of using the mass fraction approach is that with (3.1.3a) the total bulk mass is conserved and with (3.1.3b) the mass of one individual medium is conserved. For the two-fluid case, the second medium is implicitly conserved by taking the bulk mass minus the mass of medium 1. However, a drawback of this method is the loss of knowledge of the exact interface location.

3.1.1 Pressure invariance

As shown in section 2.7.1.1, the fully conservative level-set method suffers from pressure oscillations. However, when using the mass-fraction approach instead of the level-set approach, the pressure is invariant under all conditions. From (3.1.7), for the bulk density it follows

$$\rho(\beta, p) = \frac{\rho_{\mathrm{I}}(p)\rho_{\mathrm{II}}(p)}{\beta\rho_{\mathrm{II}}(p) + (1-\beta)\rho_{\mathrm{I}}}. \tag{3.1.8}$$

3.1.1.1 Error analysis The error analysis is done with (3.1.8) as a starting point. For variance calculation, a small perturbation term is added to the quantities ρ, $\beta\rho$ and p,

$$\rho \rightarrow \rho + \Delta\rho, \tag{3.1.9a}$$

$$\beta\rho \rightarrow \beta\rho + \Delta(\beta\rho), \tag{3.1.9b}$$

$$p \rightarrow p + \Delta p, \tag{3.1.9c}$$

demanding for pressure invariance, $\Delta p = 0$, i.e., $p = P$. For convenience, (3.1.8) is rewritten as

$$1 = \frac{\beta\rho}{\rho_{\mathrm{I}}(p)} + \frac{\rho - \beta\rho}{\rho_{\mathrm{II}}(p)}. \tag{3.1.10}$$

Substituting (3.1.9a), (3.1.9b) and (3.1.9c) into (3.1.10) gives

$$1 = \frac{\beta\rho + \Delta(\beta\rho)}{\rho_{\mathrm{I}}(P)} + \frac{\rho + \Delta\rho - \beta\rho - \Delta(\beta\rho)}{\rho_{\mathrm{II}}(P)} \Leftrightarrow \tag{3.1.11}$$

$$1 = \frac{\beta\rho}{\rho_{\mathrm{I}}(P)} + \frac{\rho - \beta\rho}{\rho_{\mathrm{II}}(P)} + \frac{\Delta(\beta\rho)}{\rho_{\mathrm{I}}(P)} - \frac{\Delta(\beta\rho)}{\rho_{\mathrm{II}}(P)} + \frac{\Delta\rho}{\rho_{\mathrm{II}}(P)}. \tag{3.1.12}$$

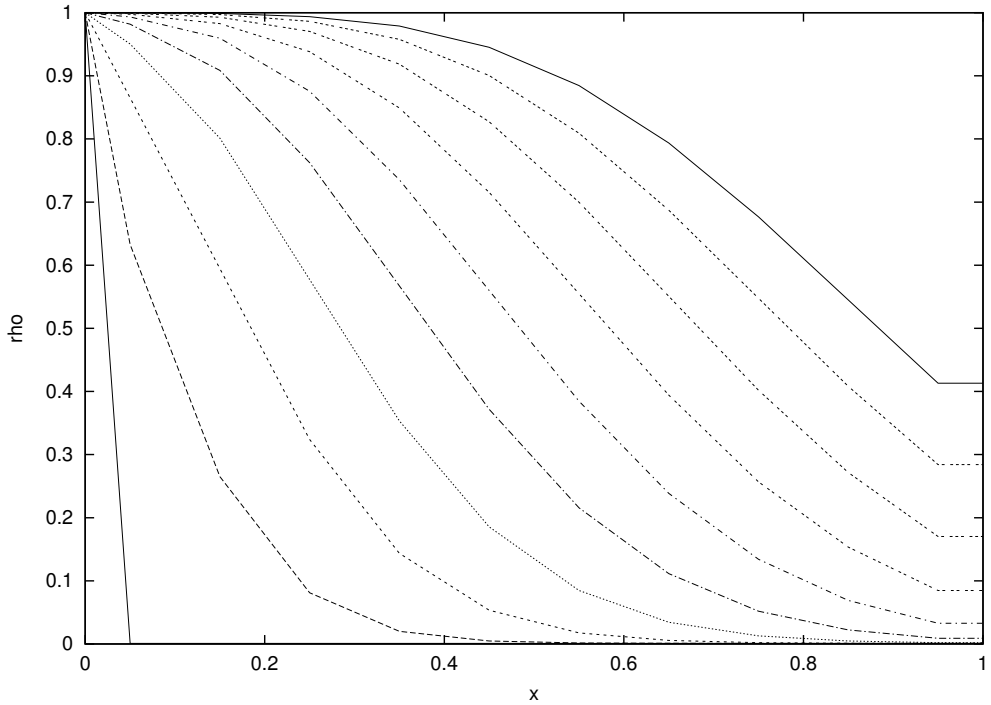

Figure 3.1: Model 1: Test case with use of the conservative mass-fraction method on a grid of 10 cells.

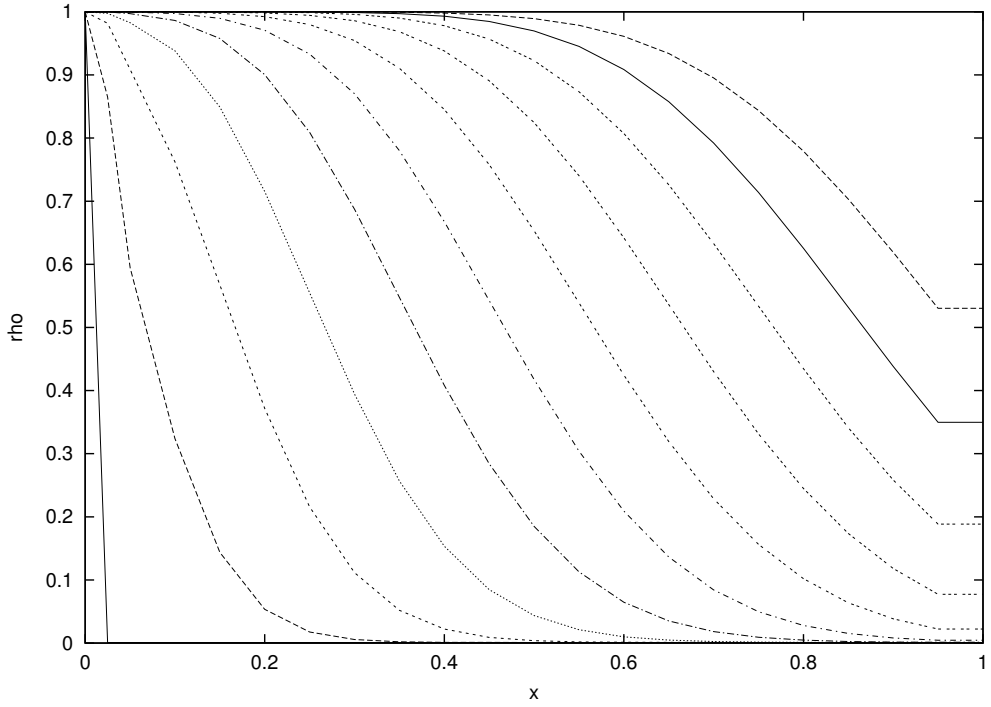

Figure 3.2: Model 1: Test case with use of the conservative mass-fraction method on a grid of 20 cells.

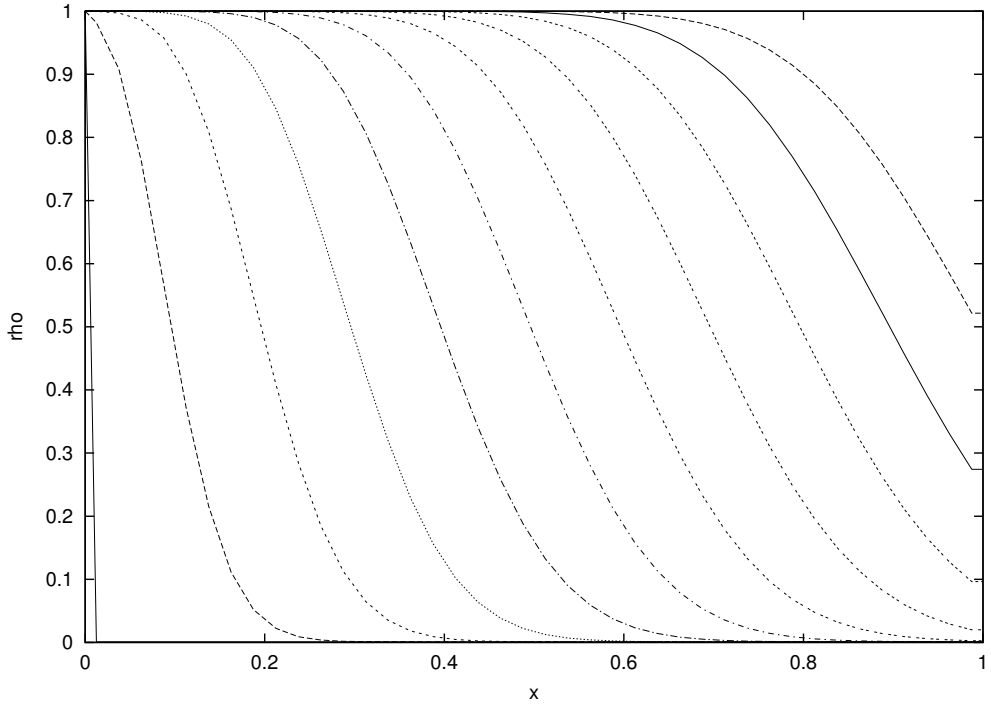

Figure 3.3: Model 1: Test case with use of the conservative mass-fraction method on a grid of 40 cells.

Using expression (3.1.10) and multiplying the whole equation by $\rho_{\mathrm{I}}\rho_{\mathrm{II}}$, (3.1.12) becomes

$$(\Delta\rho)\rho_{\mathrm{I}}(P) = \Delta(\beta\rho)\left(\rho_{\mathrm{I}}(P) - \rho_{\mathrm{II}}(P)\right). \tag{3.1.13}$$

So, in order to meet the pressure invariancy, equation (3.1.13) must hold.

3.1.1.2 Time stepping Performing one time step with the discretized form of (3.1.1) gives now

$$\Delta\rho_i = -\frac{\Delta t}{h}\left(\rho_{i+\frac{1}{2}}u_{i+\frac{1}{2}} - \rho_{i-\frac{1}{2}}u_{i-\frac{1}{2}}\right), \tag{3.1.14}$$

$$\Delta(\beta_i,\rho_i) = -\frac{\Delta t}{h}\left(\beta_{i+\frac{1}{2}}\rho_{i+\frac{1}{2}}u_{i+\frac{1}{2}} - \beta_{i-\frac{1}{2}}\rho_{i-\frac{1}{2}}u_{i-\frac{1}{2}}\right). \tag{3.1.15}$$

In this test case the velocity is constant throughout the whole flow field, $u_{i-\frac{1}{2}} = u_{i+\frac{1}{2}} = U$. Substituting this into (3.1.15) and (3.1.14) gives

$$\Delta\rho_i = -\frac{\Delta t}{h}U\left(\rho_{i+\frac{1}{2}} - \rho_{i-\frac{1}{2}}\right), \tag{3.1.16}$$

$$\Delta(\beta_i\rho_i) = -\frac{\Delta t}{h}U\left(\beta_{i+\frac{1}{2}}\rho_{i+\frac{1}{2}} - \beta_{i-\frac{1}{2}}\rho_{i-\frac{1}{2}}\right). \tag{3.1.17}$$

The flow moves in positive i-direction. The pressure p is constant throughout the whole flow field, $p_{i-\frac{1}{2}} = p_{i+\frac{1}{2}} = P$. For the values on the cell faces, we can write

$$\beta_{i-\frac{1}{2}} = \beta_{i-1}, \tag{3.1.18a}$$

$$\beta_{i+\frac{1}{2}} = \beta_i, \tag{3.1.18b}$$

$$\rho_{i-\frac{1}{2}} = \rho(P,\beta_{i-1}) = \alpha_{i-1}\rho_{\mathrm{I}}(P) + (1-\alpha_{i-1})\rho_{\mathrm{II}}(P), \tag{3.1.18c}$$

$$\rho_{i+\frac{1}{2}} = \rho(P,\beta_i) = \alpha_i\rho_{\mathrm{I}}(P) + (1-\alpha_i)\rho_{\mathrm{II}}(P). \tag{3.1.18d}$$

Substituton of (3.1.16) to (3.1.18d) into (3.1.13) yields

$$\Big(\big(\alpha_i\rho_{\mathrm{I}}(P)+(1-\alpha_i)\rho_{\mathrm{II}}(P)\big)-\big(\alpha_{i-1}\rho_{\mathrm{I}}(P)+(1-\alpha_{i-1}\rho_{\mathrm{II}}(P)\big)\Big)\rho_{\mathrm{I}}(P)= \\ \Big(\beta_i\big(\alpha_i\rho_{\mathrm{I}}(P)+(1-\alpha_i)\rho_{\mathrm{II}}(P)\big)-\beta_{i-1}\big(\alpha_{i-1}\rho_{\mathrm{I}}(P)+(1-\alpha_{i-1})\rho_{\mathrm{II}}(P)\big)\Big)\big(\rho_{\mathrm{I}}(P)-\rho_{\mathrm{II}}(P)\big). \tag{3.1.19}$$

Elimination of β_i and β_{i-1} from (3.1.19), using definition (3.1.5), in the form

$$\beta=\frac{\alpha\rho_{\mathrm{I}}}{\alpha\rho_{\mathrm{I}}+(1-\alpha)\rho_{\mathrm{II}}}$$

yields

$$\Big(\big(\alpha_i\rho_{\mathrm{I}}\left(P\right)+(1-\alpha_i)\,\rho_{\mathrm{II}}\left(P\right)\big)-\big(\alpha_{i-1}\rho_{\mathrm{I}}\left(P\right)+(1-\alpha_{i-1})\,\rho_{\mathrm{II}}\left(P\right)\big)\Big)\rho_{\mathrm{I}}= \\ \Big(\alpha_i\rho_{\mathrm{I}}\left(P\right)-\alpha_{i-1}\rho_{\mathrm{I}}\left(P\right)\Big)\big(\rho_{\mathrm{I}}\left(P\right)-\rho_{\mathrm{II}}\left(P\right)\big), \tag{3.1.20}$$

which is an identity. Hence, this method is pressure-invariant indeed.

3.2. Regeneration of the exact interface location

3.2.1 Regeneration in 1 dimension

As mentioned before, in contrast to the level-set method, this method does not give the exact interface location. The interface is smeared over several cells due to numerical diffusion. The sharp contact discontinuity transforms into a smooth transition between fluid I and fluid II. From here, we call this smooth transition region the vapour or foam region. In reality such a vapour or foam region does not exist. This section gives a proposal for reconstructing the true position of the interface and compute the real values of the density. The regeneration method is intended as a post-process, but it can also be used as a re-initialization process during computation. We assume the interface to be a point in 1 dimension, a line in two dimensions and a plane in 3 dimensions. That implies that in physics there is no vapour region. The basic idea behind this regeneration process is to translate the mass-fraction back into a Volume-of-Fluid fraction and eliminate the diffusion by reordering of the volumes of the fluids. The regeneration can be done by doing the following steps. First, determine the start and endpoint of the vapour region. This is done by examining the mass fraction β through the whole flow field. For $\beta = 1$ the cell is fully filled with fluid I, for $\beta = 0$ the cell is fully filled with fluid II. Walking from fluid I, the first boundary B_{I} is defined when a cell contains a $\beta < 1$. From the opposite direction, walking from fluid II, the other boundary B_{II} is defined when a cell contains a $\beta > 0$. Second, compute for each cell the corresponding volume of fluid, defined as

$$\alpha_i=\frac{\beta_i\rho_i}{\rho_{\mathrm{I}}(p_i)}. \tag{3.2.1}$$

With this step the Volume-of-Fluid of each cell, corresponding to the local density and pressure has been found (figure 3.4). The last step consists of redistributing the fluid I and fluid II volumes to the original physical position. All volumes in the vapour region containing fluid I will be moved to the boundary, bordering fluid I. The exact interface location is now given by

$$x_{\mathrm{fs}}=x_{B_{\mathrm{I}}}\pm\sum_{i=B_{\mathrm{I}}}^{B_{\mathrm{II}}}\alpha_i h_i, \tag{3.2.2}$$

where a plus-sign is for summing in positive i-direction and a minus-sign is for summing in negative i-direction. Finding both boundaries is important. Only when both left and right boundaries can

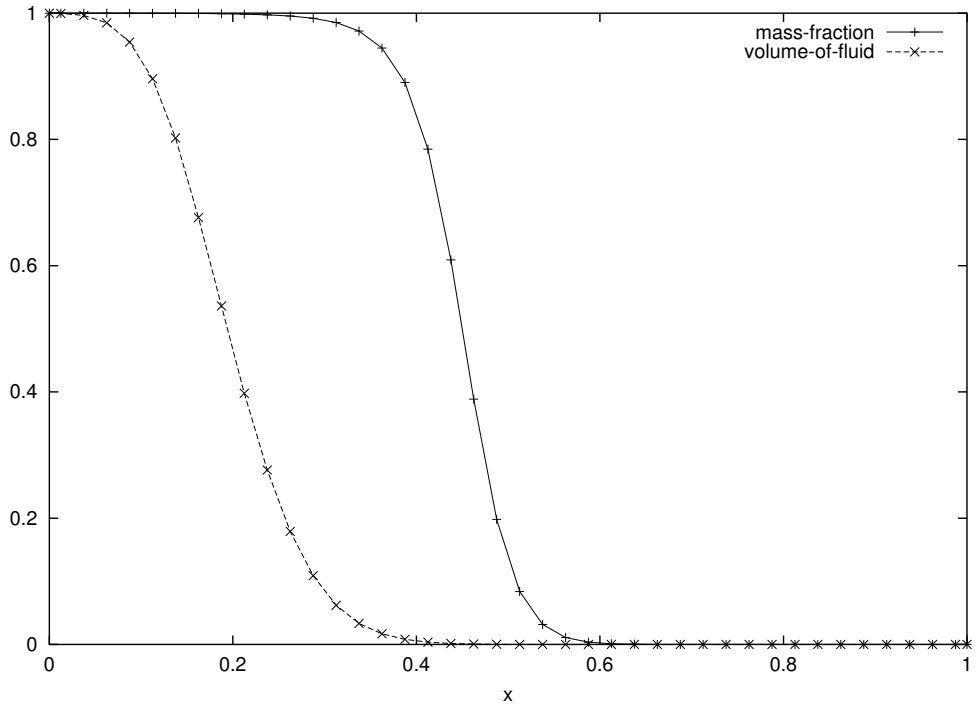

Figure 3.4: Mass-fraction and Volume-of-Fluid fraction for a water-air flow with $\rho_{\mathrm{I}} = 1$, $\rho_{\mathrm{II}} = 0.001$, $U = 1$, $(x_{\mathrm{fs}})_{\mathrm{init}} = 0.0$ and $P_{\mathrm{init}} = 1$ at $t = 0.2$.

be determined, the enclosure of the transition between fluid I and fluid II can be garanteed. Section 3.2.2 shows a numerical result. Unfortunately, this method works in 1D only. Due to the elliptic character of diffusion, exact regeneration in 2D and 3D is not possible. Stumbling block is the unknown diffusion direction of each particle.

3.2.2 Numerical results

Numerical results of this method is shown in figures 3.5 to 3.8. It's easy to see that the regenerated $\rho(x)$ matches the exact solution.

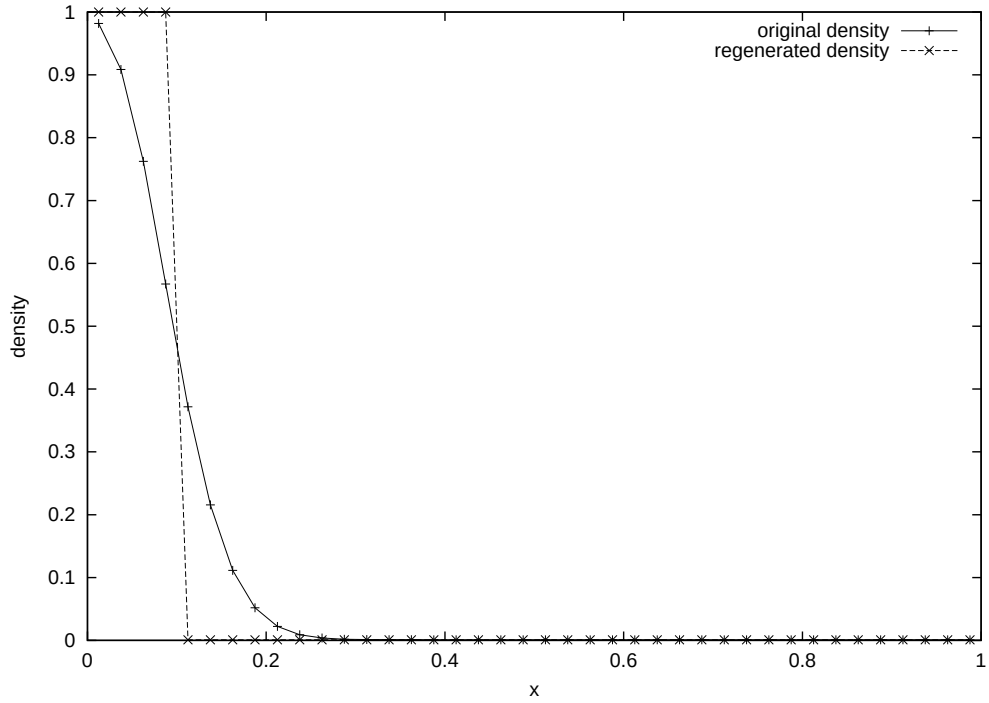

Figure 3.5: Density at $t = 0.1$

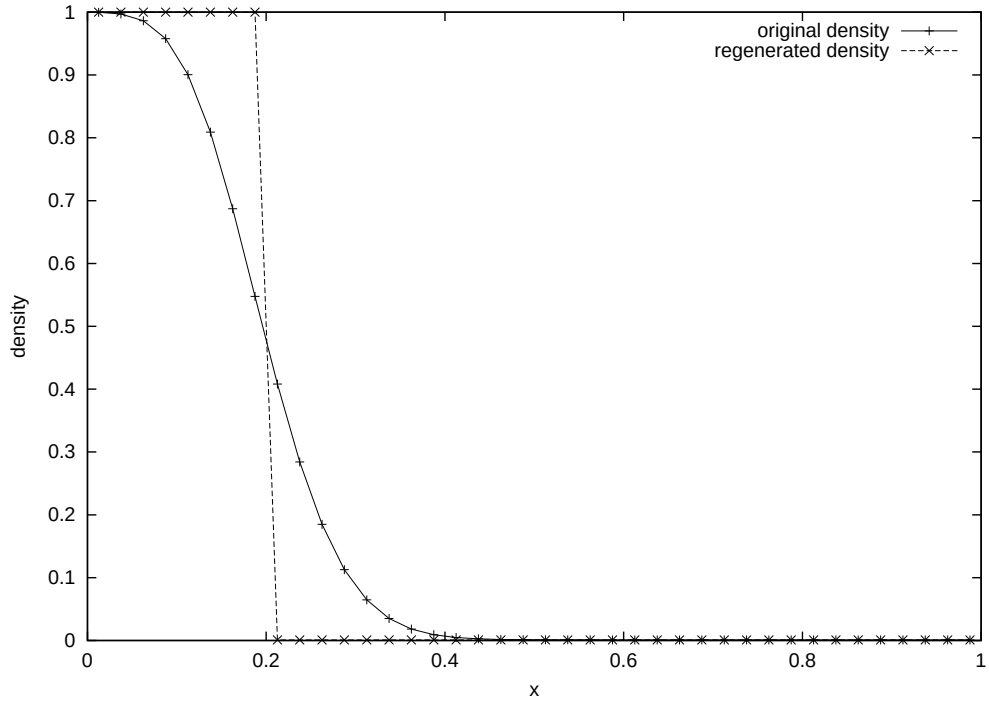

Figure 3.6: Density at $t = 0.2$

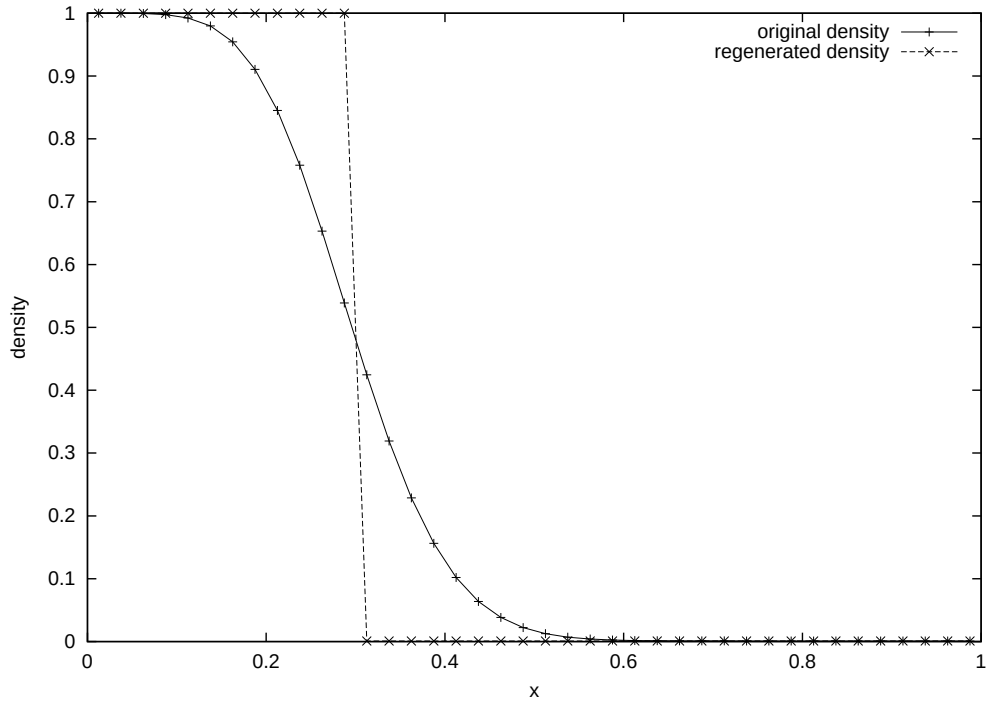

Figure 3.7: Density at $t = 0.3$

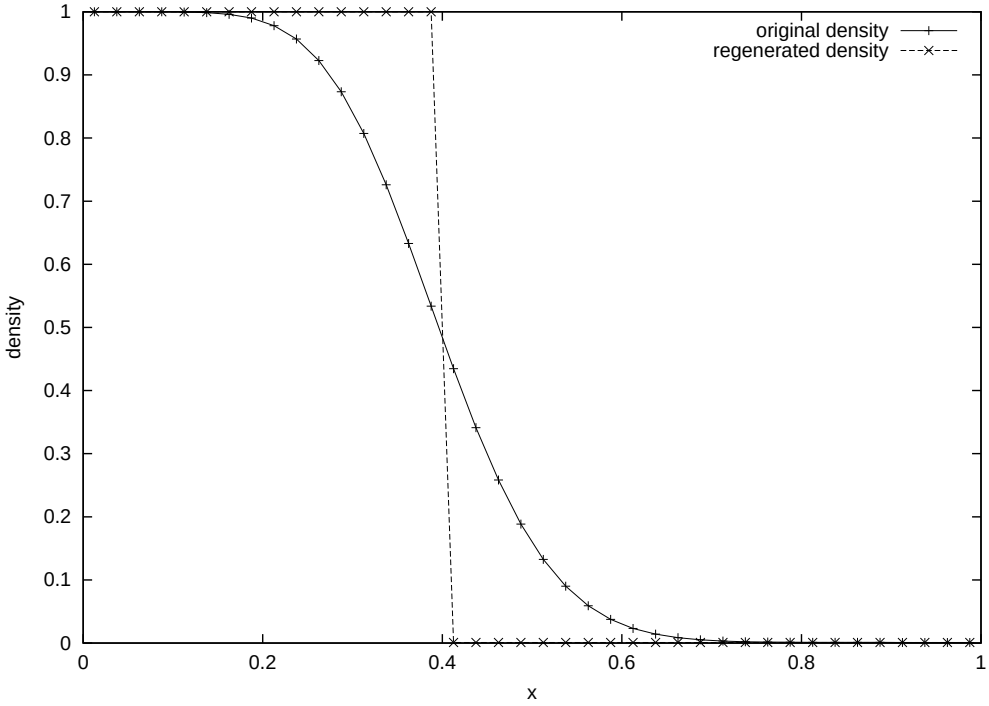

Figure 3.8: Density at $t = 0.4$

Chapter 4
Exact Method

4.1. Non-linear Riemann shock tube problem

A membrane separates two states of quiescent air until time $t = 0.0$. Then, the membrane ruptures. This gives 4 different possible pairs of shock and expansion waves. The exact solution of this problem was first obtained by Riemann, in the 19th century, and this problem is also often called the Riemann problem. Figure 4.1 shows the four possible wave pairs, traveling in space through time.

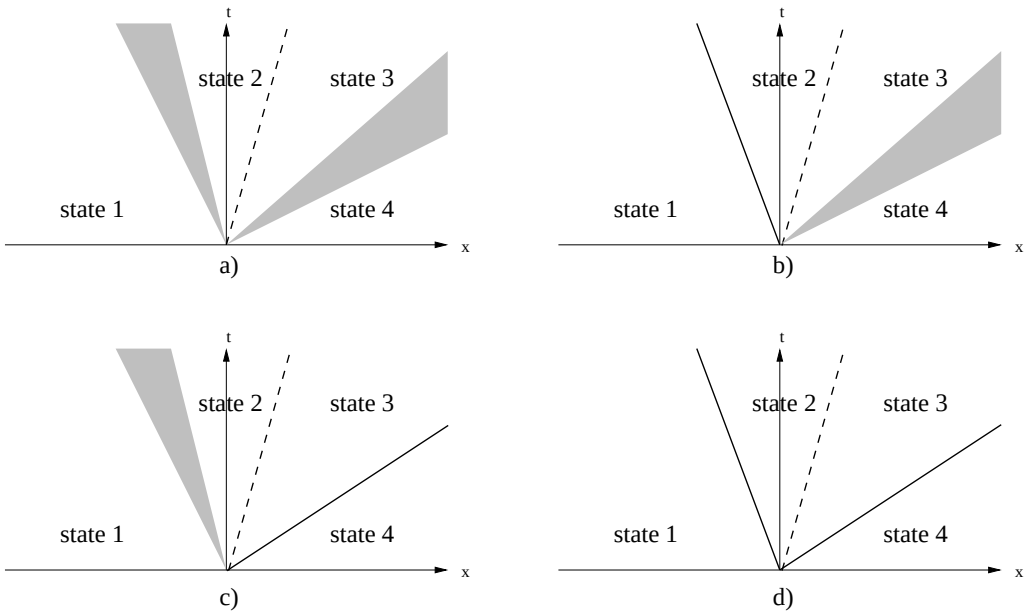

Figure 4.1: Wave combinations in physical space.

4.2. Non-isentropic single-fluid equations for the non-linear Riemann problem

For a non-isentropic single-fluid flow, the Euler equations in non-conservative form are given by

$$\frac{\partial \rho}{\partial t} + \nabla \cdot \rho \overrightarrow{V} = 0, \tag{4.2.1}$$

$$\frac{\partial \rho}{\partial t} + \nabla \cdot \rho \overrightarrow{V}\overrightarrow{V} + \nabla p = 0, \tag{4.2.2}$$

$$\frac{\partial \rho E}{\partial t} + \nabla \cdot \rho H \overrightarrow{V} = 0. \tag{4.2.3}$$

Further, reversibility (smooth flows) is assumed so that the entropy is constant when moving with a fluid particle. The energy equation (4.2.3) can be rewritten into

$$\frac{Ds}{Dt} = 0. \tag{4.2.4}$$

Using the relation $ds \propto dp - a^2 d\rho$, equation (4.2.4) may be written in terms of p and ρ:

$$\frac{\partial p}{\partial t} + u\frac{\partial p}{\partial x} - a^2\left(\frac{\partial \rho}{\partial t} + u\frac{\partial \rho}{\partial x}\right) = 0. \tag{4.2.5}$$

Using (4.2.1) to eliminate the ρ-derivatives and dividing by ρa gives

$$\frac{1}{\rho a}\frac{\partial p}{\partial t} + \frac{u}{\rho a}\frac{\partial p}{\partial x} + a\frac{\partial u}{\partial x} = 0. \tag{4.2.6}$$

Adding and subtracting (4.2.2) and (4.2.6) gives the set of characteristic equations

$$\frac{\partial J^+}{\partial t} = (u+a)\frac{\partial J^+}{\partial x}, \tag{4.2.7}$$

$$\frac{\partial J^-}{\partial t} = (u-a)\frac{\partial J^-}{\partial x}, \tag{4.2.8}$$

in which

$$dJ^+ = du + \frac{dp}{\rho a}, \tag{4.2.9}$$

$$dJ^- = du - \frac{dp}{\rho a}. \tag{4.2.10}$$

This relation is known as the differential definition of the Riemann invariants. In case of an ideal gas with $\frac{\rho}{\rho_{\text{ref}}} = \left(\frac{p}{p_{\text{ref}}}\right)^{\gamma}$, the term $\frac{dp}{\rho a}$ can be rewritten as $\frac{2da}{\gamma-1}$ and the Riemann invariants $J^{\pm}$can be integrated, yielding

$$J^{\pm} = u \pm \frac{2}{\gamma-1}a. \tag{4.2.11}$$

4.2.1 Shock waves

For a shock, running to the right with speed c_s, one can write the Rankine-Hugoniot relations for mass and momentum in a shock frame (see figure 4.2) as

$$\rho_{\text{pre}} c_s = \rho_{\text{post}}\left(c_s - u_{\text{post}}\right) \equiv m, \tag{4.2.12}$$

$$p_{\text{pre}} + \rho_{\text{pre}} c_s^2 = p_{\text{post}} + \rho_{\text{post}}\left(c_s - u_{\text{post}}\right)^2. \tag{4.2.13}$$

Inserting the mass flow variable into the momentum equation, gives

$$p_{\text{pre}} + m c_s = p_{\text{post}} + m\left(c_s - u_{\text{post}}\right), \tag{4.2.14}$$

or

$$\Delta p = m \Delta u. \tag{4.2.15}$$

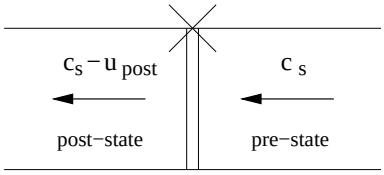

Figure 4.2: Shock view

Similarly, for a shock moving to the left we find

$$\Delta p = -m\Delta u. \tag{4.2.16}$$

The traveling speed of the shock can be determined in terms of the pressure rise $p_{\text{post}} - p_{\text{pre}}$, using the relation

$$\frac{p_{\text{post}}}{p_{\text{pre}}} = 1 + \frac{2\gamma}{\gamma+1}\left(M_{\text{pre}}^2 - 1\right). \tag{4.2.17}$$

When looking in shock frame, it can be written

$$M_{\text{pre}} = \frac{u_{\text{pre}}}{a_{\text{pre}}} = \frac{|c_s|}{a_{\text{pre}}}. \tag{4.2.18}$$

Substituting (4.2.18) into (4.2.17) and solving for c_s gives the shock speed equation

$$|c_s| = a_{\text{pre}}\sqrt{1 + \frac{\gamma+1}{2\gamma}\frac{\Delta p}{p_{\text{pre}}}}. \tag{4.2.19}$$

After substituting (4.2.19) into (4.2.12), the mass flux m is determined as

$$m = \rho_{\text{pre}} a_{\text{pre}}\sqrt{1 + \frac{\gamma+1}{2\gamma}\frac{\Delta p}{p_{\text{pre}}}}. \tag{4.2.20}$$

In case of a Riemann problem, where states 1 and 4 are the left and right pre-states and states 2 and 3 are the left and right post-states respectively, the equations for a right-running shock can be written as

$$p_3 - p_4 = m_{\text{right}}\left(u_3 - u_4\right), \tag{4.2.21}$$

$$m_{\text{right}} = \rho_4 c_s = \rho_4 a_4\sqrt{1 + \frac{\gamma+1}{2\gamma}\frac{p_3 - p_4}{p_4}}. \tag{4.2.22}$$

Combining (4.2.21) and (4.2.22) gives the Hugoniot equation, describing flow behaviour through a shock

$$p_3 - p_4 = \rho_4 a_4\sqrt{1 + \frac{\gamma+1}{2\gamma}\frac{p_3 - p_4}{p_4}}\left(u_3 - u_4\right). \tag{4.2.23}$$

The left-running wave of the Hugoniot equation changes sign and reads

$$p_2 - p_1 = -\rho_1 a_1\sqrt{1 + \frac{\gamma+1}{2\gamma}\frac{p_2 - p_1}{p_1}}\left(u_2 - u_1\right). \tag{4.2.24}$$

4.2.2 Expansion waves

Now, let's consider the right running expansion. As in the shock case, an expansion wave can be written as a relation of the pressure difference $\Delta p = p_3 - p_4$ to the velocity difference $u_3 - u_4$. Along a Γ^- characteristic that runs from pre-state 4 to post-state 3, one can write

$$u_3 - \frac{2}{\gamma-1}a_3 = u_4 - \frac{2}{\gamma-1}a_4, \tag{4.2.25}$$

or, with use of the isentropic relations

$$u_3 - u_4 = \frac{2}{\gamma - 1}\left(\left(\frac{p_3}{p_4}\right)^{\frac{\gamma-1}{2\gamma}} - 1\right) a_4. \tag{4.2.26}$$

Rearranging the terms, gives the so-called Poisson equation

$$\left(\frac{p_3}{p_4}\right)^{\frac{\gamma-1}{2\gamma}} = 1 + \frac{\gamma - 1}{2}\frac{u_3 - u_4}{a_4} \tag{4.2.27}$$

for the right-running expansion wave. The Poisson equation of the left running wave changes sign and reads

$$\left(\frac{p_2}{p_1}\right)^{\frac{\gamma-1}{2\gamma}} = -\left(1 + \frac{\gamma - 1}{2}\frac{u_2 - u_1}{a_1}\right). \tag{4.2.28}$$

The complete Riemann problem can be solved by drawing the two appropriate curves in the (u, p)-plane, a Hugoniot curve in case of a compression, a Poisson curve in case of an expansion. The intersection of two curves represents the post-state variables $u_{2,3}$ and $p_{2,3}$ (figure 2.2c).

4.3. Isentropic, two-fluid equations for the non-linear Riemann problem

As with the single-fluid case, the left and right Riemann states of the two-fluid case, (u_1, p_1) and (u_4, p_4), are also connected to the intermediate state $(u_{2,3}, p_{2,3})$ through a shock wave or an expansion wave (figure 4.1). The flow behaviour is described by a Hugoniot equation, while a Poisson equation describes the flow behaviour through an expansion fan. The specific forms of the Hugoniot - and Poisson equation depend on the equation of state considered. For our two-fluid model, Tait's equation of state is used.

4.3.1 Shock waves

Consider the shock frame in figure 4.2, showing a right running shock wave. The shock connects the pre-shock state (u_4, p_4) and the post-shock state (u_3, p_3). The flow moves from right to left. In the shock frame, the jump conditions for the right running shock read

$$\begin{bmatrix} u \\ p \end{bmatrix} = m \begin{bmatrix} -\frac{1}{\rho} \\ u \end{bmatrix}, \tag{4.3.1}$$

where m denotes the mass flow through the shock wave, $m = |\rho u|$. Combining the aforementioned jump condition yields

$$m = \sqrt{\frac{[p]}{-\left[\frac{1}{\rho}\right]}}. \tag{4.3.2}$$

With use of (4.3.2), the jump condition for the right running shock can be written as

$$p_3 - p_4 = \sqrt{\frac{p_3 - p_4}{\frac{1}{\rho_4} - \frac{1}{\rho_3}}}\,(u_3 - u_4). \tag{4.3.3}$$

In the two-fluid case, the density represents a bulk density $\rho(\alpha, p)$ given by (2.2.7) and (2.2.5). This makes the Hugoniot curve also a function of the Volume-of-Fluid fraction $\alpha \in [0, 1]$. Writing a generic formula, valid for every $\alpha \in [0, 1]$ is an extensive job. In case the fluid exists of 100% water, the Hugoniot equation for the rightrunning wave yields after some rewriting of the terms

$$p_3 - p_4 = (\rho_4 a_4)_w \sqrt{\frac{1}{\gamma_w}\frac{\frac{p_3 + B_w p_{\text{ref}}}{p_4 + B_w p_{\text{ref}}} - 1}{1 - \left(\frac{p_4 + B_w p_{\text{ref}}}{p_3 + B_w p_{\text{ref}}}\right)^{\frac{1}{\gamma_w}}}}\,(u_3 - u_4), \qquad \alpha_4 = 1. \tag{4.3.4}$$

For the leftrunning wave, equation (4.3.4) changes sign and becomes

$$p_2 - p_1 = -\left(\rho_1 a_1\right)_w \sqrt{\frac{1}{\gamma_w} \frac{\frac{p_2+B_w p_{\text{ref}}}{p_1+B_w p_{\text{ref}}} - 1}{1 - \left(\frac{p_1+B_w p_{\text{ref}}}{p_2+B_w p_{\text{ref}}}\right)^{\frac{1}{\gamma_w}}}} \left(u_2 - u_1\right), \qquad \alpha_1 = 1. \tag{4.3.5}$$

The infinity-subscript denotes a reference state. In case the fluid exists of 100% air, where $B = 0$, the equation set simplifies to

$$p_3 - p_4 = \left(\rho_4 a_4\right)_a \sqrt{\frac{1}{\gamma_a} \frac{\frac{p_3}{p_4} - 1}{1 - \left(\frac{p_4}{p_3}\right)^{\frac{1}{\gamma_a}}}} \left(u_3 - u_4\right), \qquad \alpha_4 = 0, \tag{4.3.6}$$

for the right running shock and

$$p_2 - p_1 = -\left(\rho_1 a_1\right)_a \sqrt{\frac{1}{\gamma_a} \frac{\frac{p_2}{p_1} - 1}{1 - \left(\frac{p_1}{p_2}\right)^{\frac{1}{\gamma_a}}}} \left(u_2 - u_1\right), \qquad \alpha_1 = 0, \tag{4.3.7}$$

for the leftrunning shock.

4.3.2 Expansion waves

Like the Hugoniot curve in the two-fluid case, the expansion wave is described by a family of curves, so called Poisson curves, due to the extra dependency caused by the Volume-of-Fluid fraction $\alpha \in [0, 1]$ in the given control volume. Along a characteristic, the Riemann invariants remain constant. We use equation (4.2.10) as a starting point. For rightrunning expansion waves, connecting (u_4, p_4) to (u_3, p_3), the family is determined by

$$u_4 - \int^{p_4} \frac{dp}{\rho a} = u_3 - \int^{p_3} \frac{dp}{\rho a}. \tag{4.3.8}$$

Again, finding the general formulation for the Poisson equation, valid for every value of α is a nasty job. Let's consider the $\alpha = 1$ situation, the control volumes are fully filled with water. Recalling Tait's equation of state and an expression for the speed of sound

$$\rho(p) = \left(\frac{p + B_w p_{\text{ref}}}{\left(1 + B_w\right) p_{\text{ref}}}\right)^{\frac{1}{\gamma_w}} \left(\rho_{\text{ref}}\right)_w, \tag{4.3.9}$$

$$a(p) = \left(\frac{p + B_w p_{\text{ref}}}{\left(1 + B_w\right) p_{\text{ref}}}\right)^{\frac{\gamma_w - 1}{2\gamma_w}} \left(a_{\text{ref}}\right)_w, \tag{4.3.10}$$

$$\left(a_{\text{ref}}\right)_w = \sqrt{\gamma_w \frac{\left(1 + B_w\right) p_{\text{ref}}}{\left(\rho_{\text{ref}}\right)_w}}. \tag{4.3.11}$$

Substitution of (4.3.9), (4.3.10) and (4.3.11) into (4.3.8) and integration, yields for the exact Poisson curve

$$u_3 - u_4 = \frac{2}{\gamma_w - 1} \left(\left(\frac{p_3 + B_w p_{\text{ref}}}{\left(1 + B_w\right) p_{\text{ref}}}\right)^{\frac{\gamma_w - 1}{2\gamma_w}} - \left(\frac{p_4 + B_w p_{\text{ref}}}{\left(1 + B_w\right) p_{\text{ref}}}\right)^{\frac{\gamma_w - 1}{2\gamma_w}} \right) \left(a_{\text{ref}}\right)_w, \tag{4.3.12}$$

for $\alpha_4 = 1$. The Poisson curve, describing the leftrunning expansion wave changes sign and reads

$$u_1 - u_2 = \frac{2}{\gamma_w - 1} \left(\left(\frac{p_2 + B_w p_{\text{ref}}}{\left(1 + B_w\right) p_{\text{ref}}}\right)^{\frac{\gamma_w - 1}{2\gamma_w}} - \left(\frac{p_1 + B_w p_{\text{ref}}}{\left(1 + B_w\right) p_{\text{ref}}}\right)^{\frac{\gamma_w - 1}{2\gamma_w}} \right) \left(a_{\text{ref}}\right)_w, \tag{4.3.13}$$

for $\alpha_1 = 1$.

4.4. Numerical approach for the exact solution

In order to find the solution for a given single-fluid Riemann problem or two-fluid Riemann problem, a numerical root finder algorithm is unavoidable. This chapter will describe what kind of solution technique is used in the program "Visual Shock Tube Solver". For a full description of the program, the reader is invited to read appendix IV. As shown in figure 4.1, 4 possible pairs of shock - and expansion waves are possible. This implies 4 possible pairs of Hugoniot curves and Poisson curves, independent of the amount of fluids involved. Before computing the exact solution, a first order estimation is computed, using the Hugoniot - and/or Poisson equations linearized around state 1 and state 4:

$$u_{2,3} - u_1 = \frac{p_1 - p_{2,3}}{\rho_1 a_1}, \tag{4.4.1a}$$

$$u_{2,3} - u_4 = -\frac{p_4 - p_{2,3}}{\rho_4 a_4}. \tag{4.4.1b}$$

Adding and extracting (4.4.1a) and (4.4.1b) gives two algebraic expressions for the post Riemann states 2 and 3

$$u_{2,3} = \frac{\rho_1 a_1 u_1 + \rho_4 a_4 u_4 - (p_4 - p_1)}{\rho_1 a_1 + \rho_4 a_4}, \tag{4.4.2a}$$

$$p_{2,3} = \frac{\rho_4 a_4 p_1 + \rho_1 a_1 p_4 + \rho_1 a_1 \rho_4 a_4 (u_1 - u_4)}{\rho_1 a_1 + \rho_4 a_4}, \tag{4.4.2b}$$

Now, the right choice regarding the connecting curves has to me made. Denote A_1 as the left connecting curve and A_2 as the right connecting curve. A_1 reads

$$p_1 < p_{2,3} \quad \Rightarrow \quad A_1 = H_1, \tag{4.4.3a}$$

$$p_1 > p_{2,3} \quad \Rightarrow \quad A_1 = P_1, \tag{4.4.3b}$$

$$p_4 < p_{2,3} \quad \Rightarrow \quad A_4 = H_4, \tag{4.4.3c}$$

$$p_4 > p_{2,3} \quad \Rightarrow \quad A_4 = P_4, \tag{4.4.3d}$$

where

$$H_1 = H(u_1, p_1, u_{2,3}, p_{2,3}), \tag{4.4.4a}$$

$$H_4 = H(u_4, p_4, u_{2,3}, p_{2,3}), \tag{4.4.4b}$$

$$P_1 = P(u_1, p_1, u_{2,3}, p_{2,3}), \tag{4.4.4c}$$

$$P_4 = P(u_4, p_4, u_{2,3}, p_{2,3}). \tag{4.4.4d}$$

For the single-fluid case, H_1 is represented by (4.2.24), H_4 by (4.2.23), P_1 by (4.2.28) and P_4 by (4.2.27). For the two-fluid case, in case of pure water, H_1 is represented by (4.3.5), H_4 by (4.3.4), P_1 by (4.3.13) and P_4 by (4.3.12). The exact state vector

$$q_{2,3} = \begin{pmatrix} u_{2,3} \\ p_{2,3} \end{pmatrix} \tag{4.4.5}$$

follows from the system of equations

$$A_1 = 0, \tag{4.4.6a}$$

$$A_4 = 0. \tag{4.4.6b}$$

The state vector $q_{2,3}$ is found with use of the Newton-Raphson method

$$q_{2,3}^{n+1} = q_{2,3}^n - \frac{A(q_{2,3}^n)}{A'(q_{2,3}^n)}, \tag{4.4.7}$$

where

$$A = \begin{pmatrix} A_4 \\ A_1 \end{pmatrix}, \quad A' = \begin{pmatrix} \frac{\partial A_4}{\partial u_{2,3}} & \frac{\partial A_4}{\partial p_{2,3}} \\ \frac{\partial A_1}{\partial u_{2,3}} & \frac{\partial A_1}{\partial p_{2,3}} \end{pmatrix}. \tag{4.4.8}$$

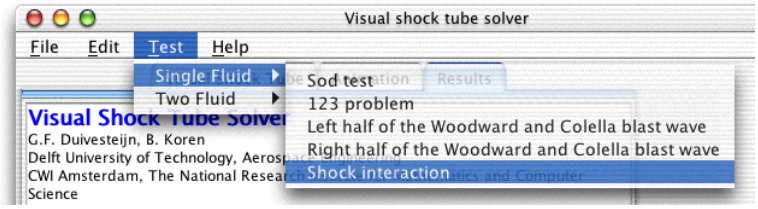

Figure 4.3: Test menu, showing the five single-fluid test cases

As starting point for the Newton-Raphson iteration, the linear solution of $q_{2,3}$, (4.4.2), is used for our educated guess. When $A\left(q^n_{2,3}\right) = 0$, the exact state $q_{2,3}$ has been found. However, sometimes the linear first guess returns negative pressures. This happens when for example two strong expansion waves occur[1]. In that case, a better educated guess for pressure $p_{2,3}$ is

$$p_{2,3} = \left(\frac{a_1 + a_4 - (\gamma - 1)(u_1 - u_4)}{\frac{a_1}{p_1^{\frac{\gamma-1}{2\gamma}}} + \frac{a_4}{p_4^{\frac{\gamma-1}{2\gamma}}}} \right)^{\frac{2\gamma}{\gamma-1}}, \tag{4.4.9}$$

for the single-fluid case and

$$p_{2,3} = \frac{1}{2} \left(\left(\frac{a_1 + a_4 - (\gamma_1 - 1)(u_1 - u_4)}{\frac{a_1}{p_1^{\frac{\gamma_1-1}{2\gamma_1}}} + \frac{a_4}{p_4^{\frac{\gamma_1-1}{2\gamma_1}}}} \right)^{\frac{2\gamma_1}{\gamma_1-1}} + \left(\frac{a_1 + a_4 - (\gamma_4 - 1)(u_1 - u_4)}{\frac{a_1}{p_1^{\frac{\gamma_4-1}{2\gamma_4}}} + \frac{a_4}{p_4^{\frac{\gamma_4-1}{2\gamma_4}}}} \right)^{\frac{2\gamma_4}{\gamma_4-1}} \right) \tag{4.4.10}$$

for the two-fluid case. Note that (4.4.9) represents an algebraic exact expression for the two expansion wave single-fluid case. There is no physical or mathematical reason for applying the mean value approach of (4.4.10). However, the obtained pressure from (4.4.10) proves to be remarkably close to the converged pressure, obtained from (4.4.7).

4.5. Numerical tests for single-fluid flows

To test the program "Visual Shock Tube Solver" for its single-fluid flow calculation capabilities, five Riemann problems have been selected. The five Riemann problems are extensively described section 4.3 of [29]. The present section will give a brief description of these problems.

The test can be easily reproduced by selecting the appropriate test from the menu item "test" (figure 4.3). Table 4.1 shows the initial values of all five test cases, table 4.2 shows the corresponding exact solutions for pressure $p_{2,3}$, $u_{2,3}$, ρ_2 and ρ_3.

4.5.1 Sod test

The first test is called the "Sod" test. The test is classified as an easy test containing all kind of waves; a leftrunning expansion wave, a rightrunning contact discontinuity and a rightrunning shock. Results are given in figure 4.4.

4.5.2 123 problem

The second test is called the "123 problem". The test consists of two strong expansion waves, a stationary contact discontinuity and a low initial pressure value. The linear solution returns

[1]The program "Visual Shock Tube Solver" demonstrates two strong expansion waves, named as the "123 problem". These cases can be selected from the "Test" menu item.

Test	ρ_1	u_1	p_1	ρ_4	u_4	p_4
1	1.0	0.0	1.0	0.125	0.0	0.1
2	1.0	-2.0	0.4	1.0	2.0	0.4
3	1.0	0.0	1000.0	1.0	0.0	0.01
4	1.0	0.0	0.01	1.0	0.0	100.0
5	5.99924	19.5975	460.894	5.99242	-6.19633	46.0950

Table 4.1: Data for the single-fluid Riemann problem tests.

Test	ρ_2	$u_{2,3}$	$p_{2,3}$	ρ_3
1	0.42632	0.92745	0.30313	0.26557
2	0.02185	0.0	0.00189	0.02185
3	0.57506	19.5975	460.894	5.99924
4	5.99242	-6.19633	46.0950	0.57511
5	14.2823	8.68975	1691.64	31.0426

Table 4.2: Results of the exact Riemann problem, computed with "Visual Shock Tube Solver".

a negative pressure. The educated guess, proposed in section 4.4, with (4.4.9), has to be taken as initial value for the Newton-Raphson iteration process. (This is automatically done by the program "Visual Shock Tube Solver".) Results are given in figure 4.5.

4.5.3 Left half of the Woodward and Colella blast wave
The third test case represents the left half of a blast wave, which consists of a left running expansion wave, a contact discontinuity to the right and a shock wave to the right. All waves travel at high speed. Results are given in figure 4.6.

4.5.4 Right half of the Woodward and Colella blast wave
The fourth Riemann problem is known as the right half of the Woodward and Colella blast wave. Its solution contains a left shock, a right moving contact discontinuity and a right expansion wave. Results are given in figure 4.7.

4.5.5 Shock interaction
Test 5 uses the shock solutions as found in test 3 and 4. After collision of these two strong shocks, the solution consists of a two shocks and a right moving contact discontinuity. Results are given in figure 4.8.

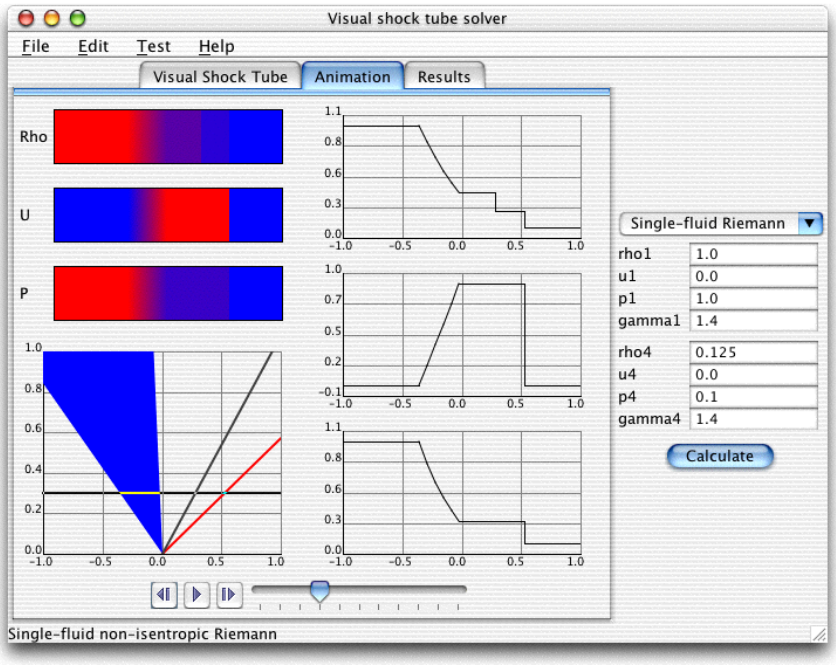

Figure 4.4: Test1: Exact solution for density, velocity and pressure at time $t = 0.4$.

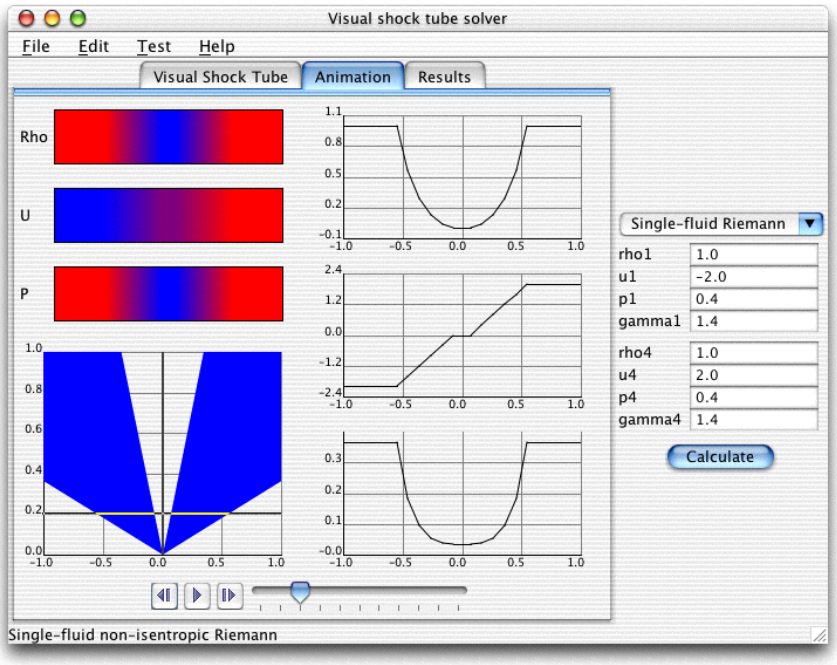

Figure 4.5: Test 2: Exact solution for density, velocity and pressure at time $t = 0.3$.

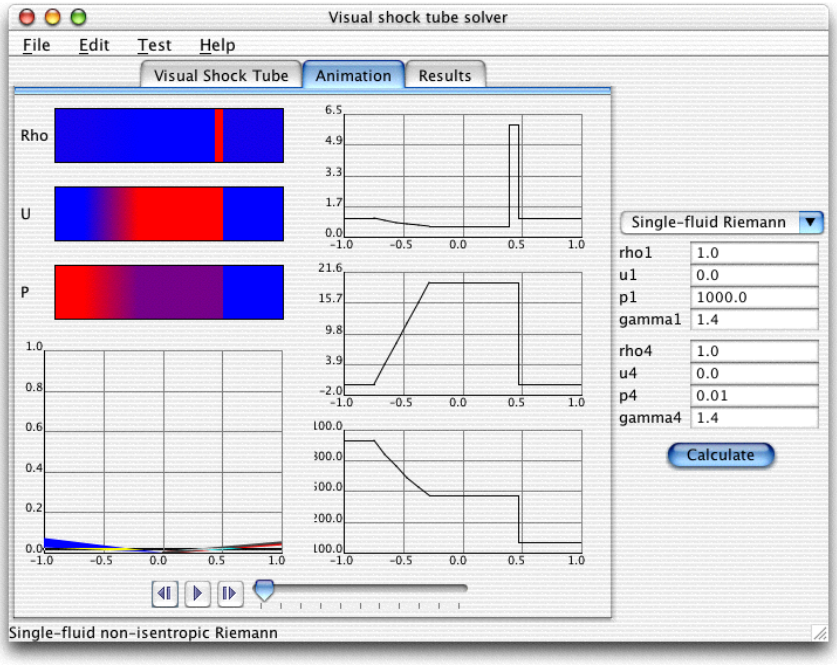

Figure 4.6: Test 3: Exact solution for density, velocity and pressure at time $t = 0.02$.

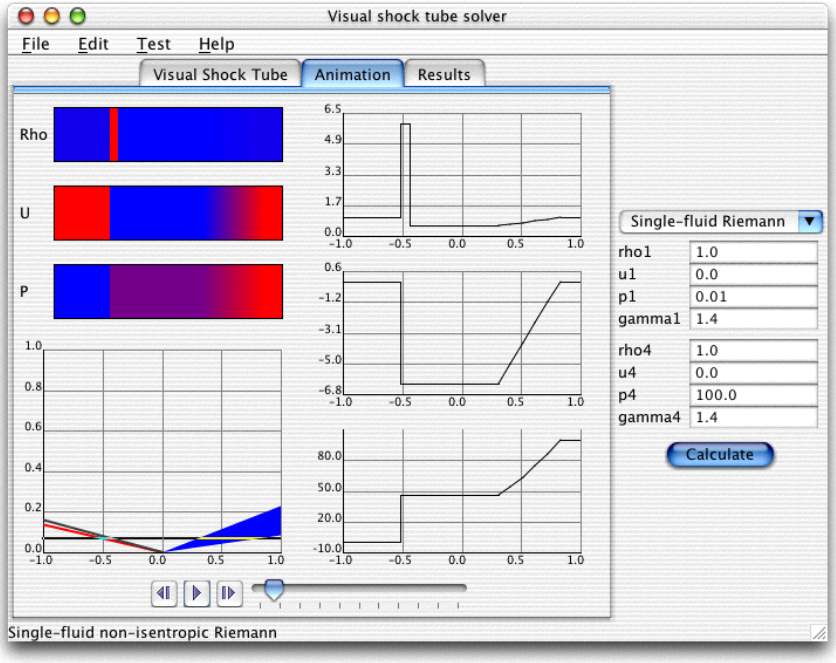

Figure 4.7: Test 4: Exact solution for density, velocity and pressure at time $t = 0.07$.

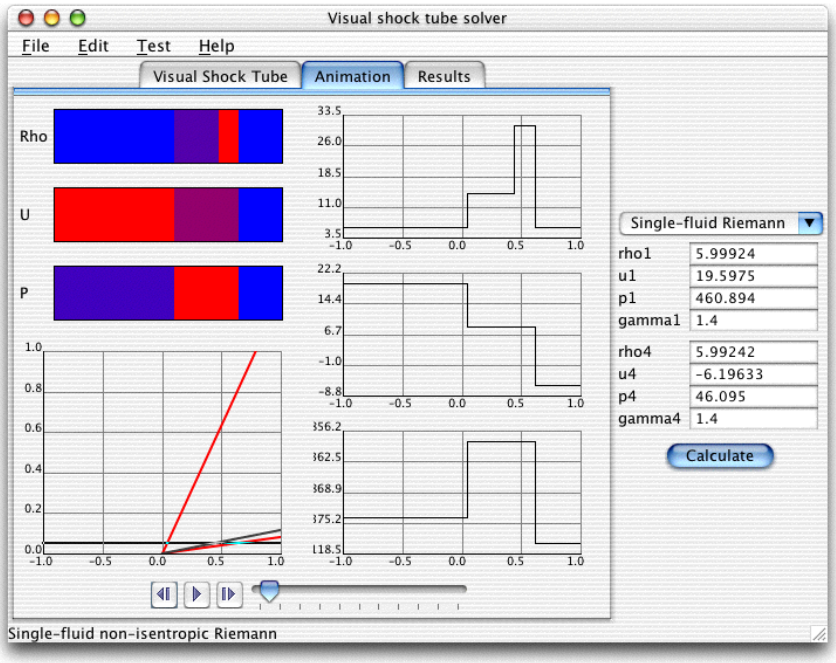

Figure 4.8: Test 5: Exact solution for density, velocity and pressure at time $t = 0.05$.

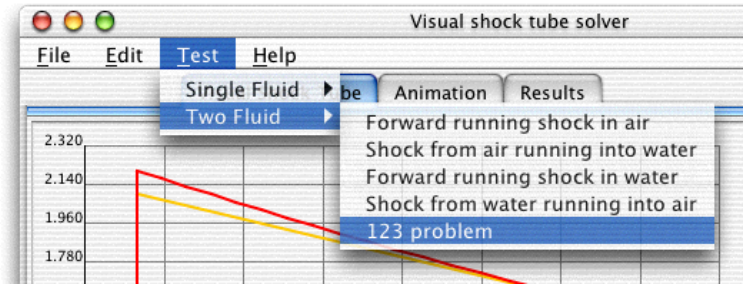

Figure 4.9: Test menu, showing the two-fluid test cases.

4.6. Numerical tests for two-fluid flows

To test the program "Visual Shock Tube Solver", 3 test cases have been selected. The tests can be easily reproduced by selecting the appropriate test from the menu item "test" (figure 4.9). Table 4.3 and 4.4 show the initial values and reference states of all two-fluid test cases, table 4.5 shows the corresponding exact solutions for pressure $p_{2,3}$, $u_{2,3}$, ρ_2 and ρ_3.

4.6.1 123-problem

Single-fluid test 3 has been chosen for the following reason. Knowing from physics, expansions are always isentropic. For a single-fluid test case with two expansions, the non-linear, single-fluid solution must be identical to the isentropic, two-fluid solution. In order to compare the 123-problem with the single-fluid case, we choose the reference state identical to state 1 and 4. As expected, both single-fluid and two-fluid solutions are the same.

4.6.2 Shock from air running into water

Consider a one dimensional tube, filled with air and water as shown in figure 4.10, where ρ_1 and ρ_2 represent air and water, respectively. At $x = 0$ a rightrunning shock emerges and hits the material interface at $t = 0.5$. At $t = 0.5$, a new Riemann problem can be distinguished. The solutions consist of a leftrunning reflected shock, a right moving contact discontinuity and a right running shock into water. Figure 4.11 shows the shock movements as function of time. Figure 4.12 shows the triangle of connecting Hugoniot curves, as computed with "Visual Shock Tube Solver".

4.6.3 Shock from water running into air

Again, consider a one dimensional tube filled with water and air as shown in figure 4.10. However, now ρ_1 and ρ_2 represent water and air, respectively. At $x = 0$ a rightrunning shock emerges and hits the material interface at $t = 0.5$. At $t = 0.5$, a new Riemann problem can be distinguished. The solutions consist of a leftrunning expansion wave, a right moving contact discontinuity and a right running transmitted weak shock into air. Figure 4.13 shows the shock movements as function of time. Figure 4.14 shows the triangle of connecting Hugoniot - and Poisson curves, as computed with "Visual Shock Tube Solver".

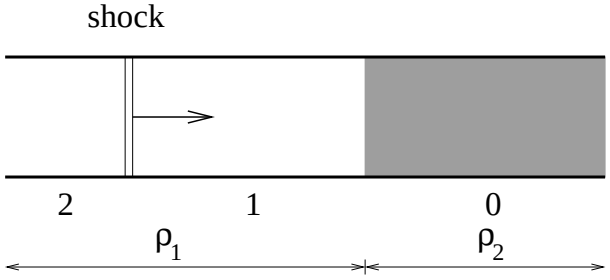

Figure 4.10: Model 3: Shock and contact discontinuity interaction.

Test	u_1	p_1	u_4	p_4
1	-2.0	0.4	2.0	0.4
2a	11.2127	1.5	0.0	1.0
2b	11.2127	1.5	0.0	1.0
3a	0.0034496	1.5	0.0	1.0
3b	0.0034496	1.5	0.0	1.0

Table 4.3: Data for the two-fluid Riemann problem tests.

Test	p_{ref}	ρ_{ref_1}	γ_1	B_1	ρ_{ref_4}	γ_4	B_4
1	0.4	1.0	1.4	0.0	1.0	1.4	0.0
2a	1.0	0.001	1.4	0.0	0.001	1.4	0.0
2b	1.0	0.001	1.4	0.0	1.0	7.0	3000
3a	1.0	1.0	7.0	3000	1.0	7.0	3000
3b	1.0	1.0	7.0	3000	0.001	1.4	0.0

Table 4.4: Reference states for the two-fluid Riemann problem tests.

Test	ρ_2	$u_{2,3}$	$p_{2,3}$	ρ_3
1	0.02185	0.0	0.00189	0.02185
2a	0.001336	11.21273	1.5	0.001
2b	0.001757	0.008281	2.2004	1.00006
3a	1.00003	0.0034496	1.5	1.0
3b	1.00003	0.006897	1.0003	0.001

Table 4.5: Results of the two-fluid Riemann problems, computed with “Visual Shock Tube Solver”.

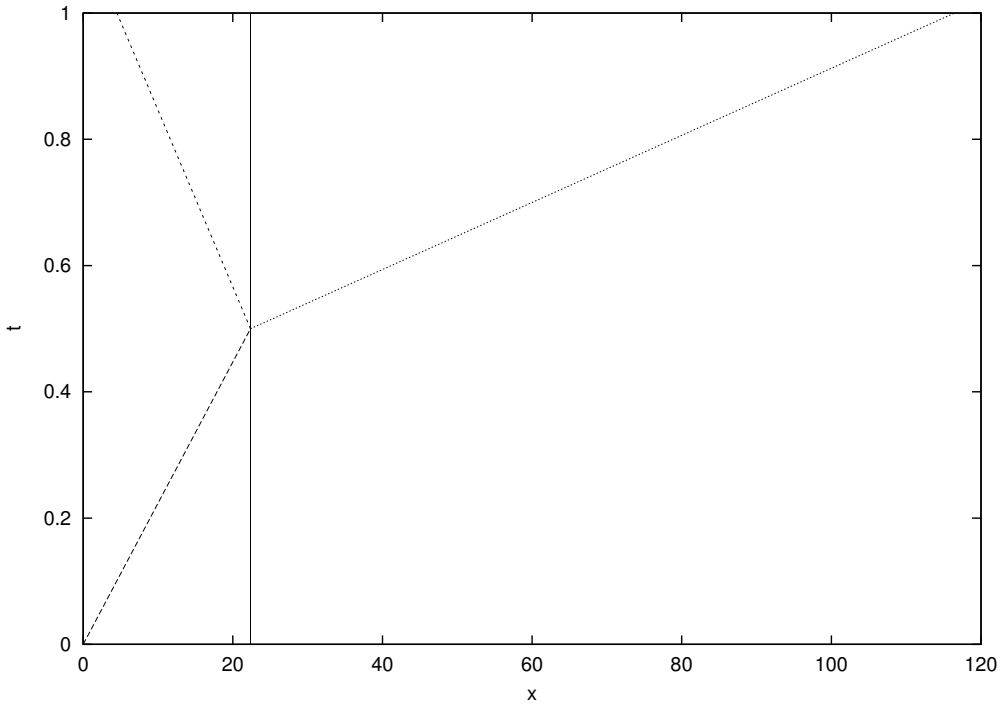

Figure 4.11: Model 3: Shock from air colliding against the interface, separating air and water.

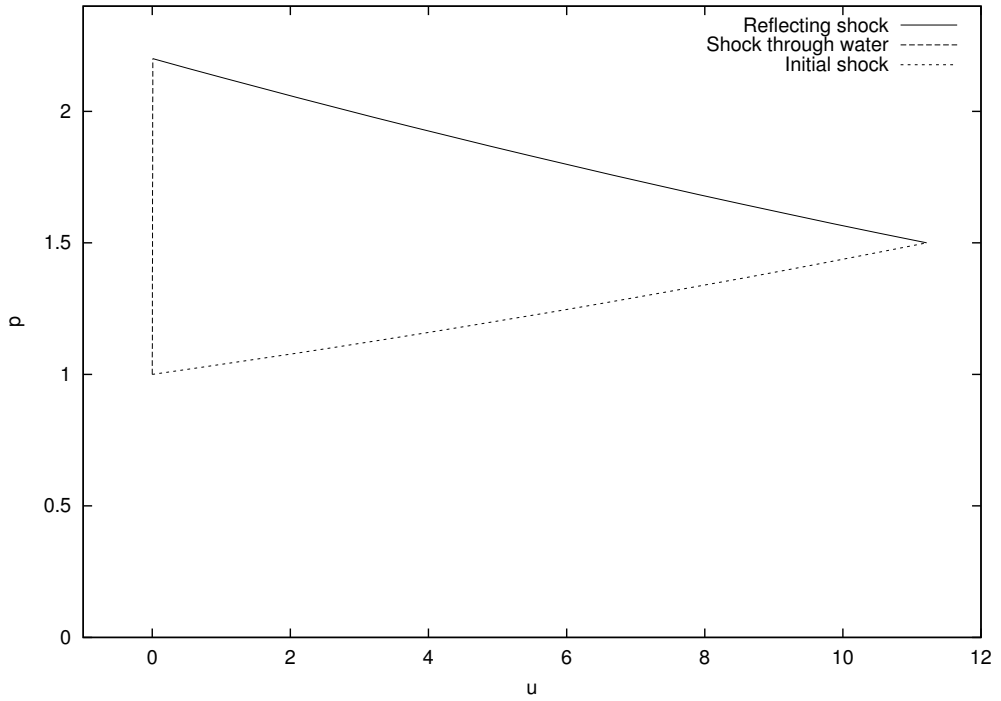

Figure 4.12: $u - p$ plot of the 3 Hugoniot curves, describing the flow behaviour.

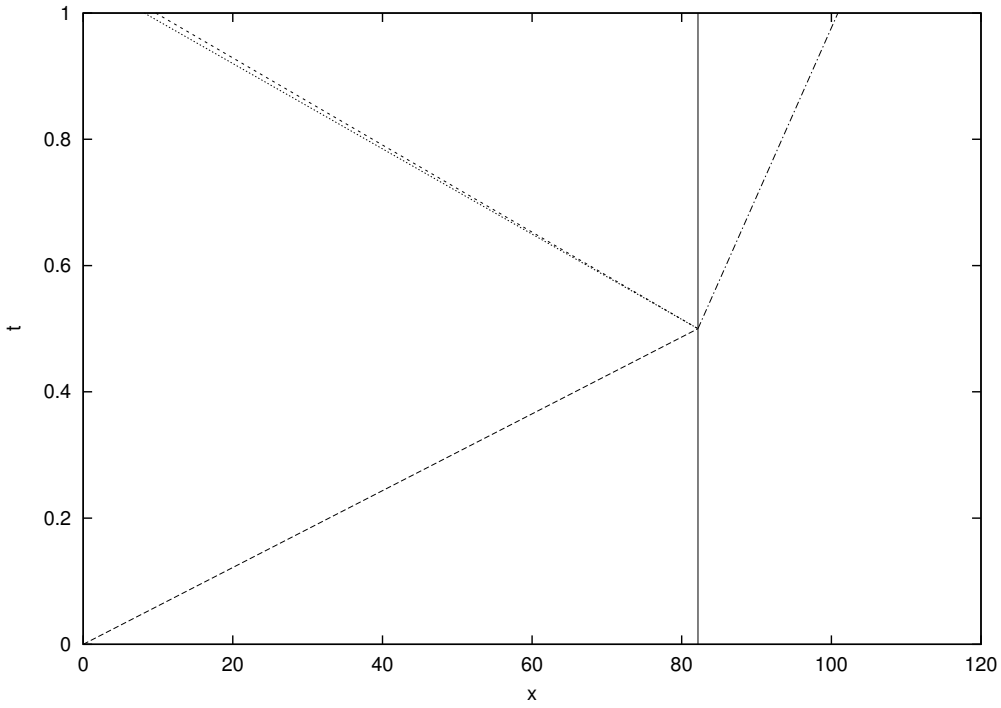

Figure 4.13: Model 3: Shock from water colliding against the interface, separating water and air.

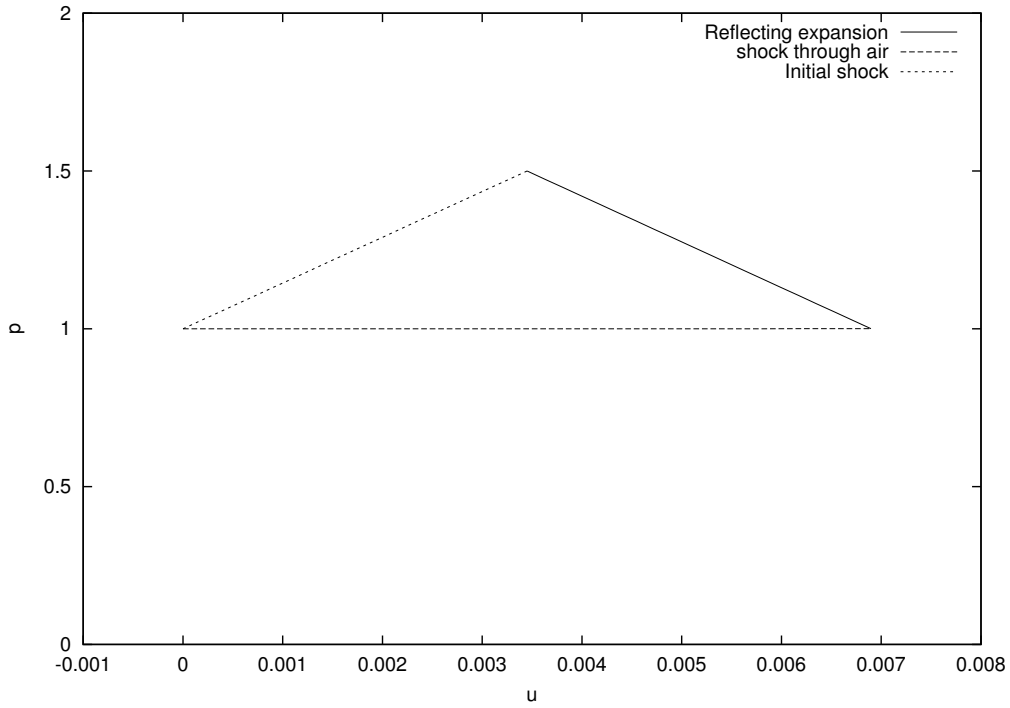

Figure 4.14: $u - p$ plot of the 3 Hugoniot curves, describing the flow behaviour.

Chapter 5
Conclusions

In this MSc-thesis, three numerical methods and one exact method for two-fluid flows have been discussed. The first numerical method, the fully conservative method, has been tested with a trivial case (two-fluid flow running through a tube at constant speed and pressure), called "test model 1". The second numerical method, the ghost-fluid method, has been tested with the trivial case and a non-trivial case (two-fluid column under the action of gravity), "test model 1" and "test model 2", respectively. The third numerical method, the mass-fraction method, has been tested with the trivial case, "test model 1".

5.1. Fully-conservative level-set method

When the level-set method is applied in fully conservative form with a finite volume technique and a linearized Godunov flux difference splitter, the method shows to be numerically unstable. Even test case 1 does not give a satisfactory solution due to pressure oscillations. This phenomenon is fully understood and in detail explained in section 2.7 and in reference [19].

5.2. Ghost-fluid / level-set method

5.2.1 Trivial test case (test model 1)

When the level-set method is applied with the ghost-fluid method, the aforementioned trivial case gives exact solutions, even for flows with extreme high density-ratios. Some successful solutions have been shown in section 2.7.1.2 and [20]. This test case shows the major advantage of the level-set method. The interface is represented as a sharp discontinuity.

5.2.2 Non-trivial test case (test model 2)

By adding gravity, the method seems to be numerical unstable due to pressure oscillations. This seems to be due to the flux in the vicinity of the interface, where sudden increase of pressure, induced by ρ_w affected by gravity, results in a too high momentum flux value for air. This has been proven in section 2.8.1.1. This proof shows that for a high-density ratio two-fluid flow with gravity, no solution can be found at all, when applying the ghost-fluid method, without taking precautions. To overcome this problem, two fixes are proposed and tested. The first fix, called the pressure-shift (section 2.8.1.2), gives reasonably good results:

- The pressure-shift eliminates pressure oscillations for a two-fluid flow computation with gravity.
- The accuracy of the pressure distribution improves with mesh refinement.

However,

- The error for the velocity distribution does not drop to zero for $h \rightarrow 0$. This makes the location of the free-surface, indicated by the level-set function, questionable.
- Time stepping has to be performed with a small CFL value, $\sigma < 0.02$. For these two reasons, the pressure-shift method is not recommended.
- The residual does not drop to machine precision during timestepping.
- Applying a pressure-shift is a violation to modelling the true physics.

The second proposed and tested fix, which gives numerical stability and a solution, is the one-sided extrapolation technique, described in section 2.8.1.4. Again, the high flux value for air, near the interface is avoided. In comparison to the pressure-shift method, numerical stability and accuracy of state variables has improved. The advantages of this method above the pressure-shift method are:

- Larger CFL values are allowed, $\sigma < 0.1$.
- Pressure distribution approximates the exact solution very accurate.
- Error for velocity distribution tends to go to zero for $h \rightarrow 0$

Although numerical stability has been obtained and improved, convergence to machine precision is still not possible. Because first-order discretization is used, the non-zero first order velocity error convects the level-set function. After this non-physical convection of the level-set function, the solution is made unique with aid of level-set values. This forces the state variables to a value that does not match the real physics anymore. As demonstrated in section 2.8.1.4, when using a coarse grid, which introduces a large first-order error, the flowfield suffers from endless convection. To avoid erroneous convection, a limiter has been introduced and demonstrated. The limiter allows true convection, but limits inaccurate first-order convection of the level-set function.

When using the combination of the level-set method, ghost-fluid method, one-sided extrapolation technique and a level-set convection limiter, the 1D water-air problem with gravity force gives a good result and approximates the analytical solution in an accurate way. This combination of methods allows the solution to converge to machine precision.

In case we are only interested in the static solution, section 2.8.1.6 has demonstrated that by adding the right amount of diffusion, the time-stepping iteration process toward the static solution can be accelerated. But when adding too much diffusion, the flow field "freezes" before the static solution has been reached.

5.2.3 Non-conservative behaviour of the ghost-fluid method

In section 2.9, a proof is given for the local non-conservative behaviour of the ghost-fluid method. When the difference of the rate of change of mass inside a control volume before and after unique-making of the solution shows a value unequal to zero, it is likely to assume that the ghost-fluid method neglets conservative behaviour. Eventually, this may lead to production or loss of bulk mass. For two-fluid flows with low-density ratio, the mass error is expected to be small. For high-density ratio multi-fluid flows however, the mass error cannot be neglected anymore. Grid refinement barely improves the accuracy. This error seems to be close to $O(1)$.

5.2.4 Ghost-fluid: Pros and Cons

Summarizing all benefits and disadvantages:

- Pro: The method makes computation of two-fluid flows possible.
- Pro: Interface between two fluids is represented as a sharp discontinuity.
- Con: Non-physical level-set equation requires extra computational effort.

- Con: In order to acquire numerical stability and an accurate solution for high-density ratio two-fluid flows with gravity, a series of fixes, tailored for the specific problem, has to be applied.
- Con: For non-trivial cases, the ghost-fluid method suffers from non-conservative behaviour.
- Con: Location of the sharp interface as given by the level-set function has to be questioned, due to erroneous convection of the level-set function.

The disadvantages make the method less suitable for generic water-air flow computations. It is expected that the non-conservative behaviour makes computations with shock-interface interaction impossible. It has to be noticed that computations described in [8] with the aid of the ghost-fluid method are performed with low-density ratio two-fluid flows, without gravity.

5.3. Mass-fraction method

The mass-fraction method is developed to overcome the disadvantages of the level-set method and the ghost-fluid method. The system of differential equations does not contain a non-physical equation anymore. The level-set function has been replaced by a mass-conservation equation. The method is proven to be fully conservative. Both bulk mass and mass of medium 1 are fully conserved under all conditions. With that, mass of medium 2 is implicitly fully conserved as well. However, the disadvantages of the mass-fraction method are:

- Diffusive behaviour of the contact-discontinuity. The interface is smeared out over several cells.
- Computing high-density ratio two-fluid flows, using the mass-fraction method, has to be performed with very high machine precision.

The quality of the interface can be improved by using a regeneration process or post process (as demonstrated in section 3.2), grid refinement or a higher-order accurate discretization.

The second disadvantage can be a pain in the neck for computations, which require a large amount of steps. When computing a water-air flow without double precision, the computational rounding error can easily grow to a multiple amount of the mass fraction in a given cell. So, extreme care has to be taken.

5.4. Exact two-fluid computational method

The derived exact two-fluid method, proves to work very well. Within the program Visual Shock Tube Solver, it enables the user to solve 1D two-fluid flow problems including contact-discontinuities, shock- and rarefaction waves. The linearized equations have proven to work well as approximate Riemann solver. However, the exact two-fluid Riemann solver, as implemented in Visual Shock Tube Solver, is expected to be very useful as replacement for approximate Riemann solvers applied in numerical codes. Although the method is expected to be expensive as numerical flux algorithm in terms of computation time, the method has a lot of advantages. The final results are exact and it handles interaction between shocks and contact discontinuities in a proper way.

Chapter 6
Suggestions for further research

The following subjects are suggested for future research:

- Test the ghost-fluid method with a high-density ratio two-fluid flow and shockwave - interface interaction. It is expected that computation of shock - interface interaction in a high-density ratio two-fluid flow problem is impossible, when using the ghost-fluid method. Numerical experiments have to confirm or deny this expectation.
- Investigation of the computation intensity when using exact two-fluid Riemann solvers, instead of approximate Riemann solvers. It is expected that the exact two-fluid Riemann solver is as computationally intensive as a two-fluid approximate Riemann solver, like an Osher FDS. Numerical experiments have to confirm or deny this expectation.
- Extend numerical experiments with mass-fraction method from 1D to 2D and 3D.
- Perform an error analysis, based on the relation between mass-fraction and required computational precision. In case of a water-air flow, the significant mass-fraction can be a very small number. Without taking care, the significance is beyond the machine precision, resulting in inaccurate results.
- Investigate the behaviour of the mass-fraction method for two-fluid flow cases with a source term, like gravity.

References

1. R. Abgrall and S. Karni, Computation of compressible multifluids, *J. Comput. Phys.*, **169**, 594-623 (2001).
2. J.D. Anderson, *Fundamentals of Aerodynamics*, McGraw-Hill, New York (1991).
3. P.G. Bakker, *Gasdynamics*, lecture notes AE4-140, Delft University of Technology (2001).
4. E.H. van Brummelen and B. Koren, A pressure-invariant conservative Godunov-type method for barotropic two-fluid flows, submitted for publication in *J. Comput. Phys.*
5. L. Cassady-Dorion, et al., *Industrial Strength Java*, New Riders Publishing, Indianapolis (1997).
6. R. Courant and K.O. Friedrichs, *Supersonic Flow and Shock Waves*, Springer, Berlin (1976).
7. B. Eckel, *Thinking in Java*, Prentice Hall, New Jersey (2000).
8. R.P. Fedkiw, T. Aslam, B. Merriman and S. Osher, A non-oscillatory Eulerian approach to interfaces in multimaterial flows (the ghost fluid method), *J. Comput. Phys.*, **152**, 457-492 (1999).
9. D.M. Geary, *Graphic Java 2, Mastering the JFC*, Sun Microsystem Press, Palo Alto (1999).
10. F.H. Harlow and J.E. Welch, Numerical calculation of time-dependent viscous incompressible flow of fluid with free surface, *Physics of Fluids*, **8**, 2182-2189 (1965).
11. C. Hirsch, *Numerical Computation of Internal and External Flows, volume 1: Fundamentals of Numerical Discretization*, Wiley, Chichester (1988).
12. C. Hirsch, *Numerical Computation of Internal and External Flows, volume 2: Computational Methods for Inviscid and Viscous Flows*, Wiley, Chichester (1990).
13. C.W. Hirt and B.D. Nichols, Volume of fluid (VOF) method for the dynamics of free boundaries, *J. Comput. Phys.*, **39**, 201-225 (1981).
14. C.S. Horstmann and G. Cornell, *Core Java, Volume I: Fundamentals*, Sun Microsystem Press, Palo Alto (1997).

15. C.S. Horstmann and G. Cornell, *Core Java, Volume II: Advanced Features*, Sun Microsystem Press, Palo Alto (1997).

16. P. Jenny, B. Muller and H. Thomann, Correction of conservative Euler solvers for gas mixtures, *J. Comput. Phys.*, **132**, 91-107 (1997).

17. S. Karni, Multicomponent flow calculations by a consistent primitive algoritm, *J. Comput. Phys.*, **112**, 31-43 (1994).

18. B. Koren and M.R. Lewis, Computation of compressible, immiscible water-air flows under the action of gravity, in Proceedings of *ECCOMAS CFD 2001.*

19. 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, Amsterdam (2001).

20. 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, Amsterdam (2001).

21. B. Koren and A.C.J. Venis, A fed back level-set mehod for moving material-void interfaces, *J. Comput. Appl. Math.*, **101**, 131-152 (1999).

22. A.M. Kuethe and C-Y. Chow, *Foundations of Aerodynamics*, Wiley, Chichester (1986).

23. B. van Leer, Towards the ultimate conservative difference scheme, V. A. second-order sequel to Godunov's method, *J. Comput. Phys.*, **32** 101-136 (1979), also *J. Comput. Phys.*, **135**, 229-248 (1997).

24. W. Mulder, S. Osher and J.A. Sethian, Computing interface motion in compressible gas dynamics, *J. Comput. Phys.*, **100**, 209-228 (1992).

25. S. Osher and F. Solomon, Upwind difference schemes for hyperbolic systems of conservation laws, *Math. Comput.*, **38**, 339-374 (1982).

26. J.A. Sethian, *Level Set Methods: Evolving Interfaces in Geometry, Fluid Mechanics, Computer Vision, and Material Science*, Cambridge University Press, Cambridge (1996).

27. M. Sussman, P. Smereka and S. Osher, A level set apporach for computing solutions to incompressible two-phase flow, *J. Comput. Phys.*, **114**, 146-159 (1994).

28. J.C. Tannehill, D.A. Anderson and R.H. Pletcher, *Computational Fluid Mechanics and Heat Transfer*, Taylor and Francis, Levittown (1997).

29. E.F. Toro, *Riemann Solvers and Numerical Methods for Fluid Dynamics, A Practical Introduction*, Springer, Berlin (1997).

Appendix I
The Jacobian matrix of the 3D Euler flow

The steady 3D Euler equations can be written on the domain $\Omega \subset \mathbb{R}^3$ as

$$\frac{\partial \vec{f_1}(\vec{q})}{\partial x} + \frac{\partial \vec{f_2}(\vec{q})}{\partial y} + \frac{\partial \vec{f_3}(\vec{q})}{\partial z} = \vec{0}, \tag{I.0.1}$$

with

$$\vec{f_1}(\vec{q}) = \begin{pmatrix} \rho u \\ \rho u^2 + p \\ \rho uv \\ \rho uw \\ \rho u \left(e + \frac{p}{\rho}\right) \end{pmatrix}, \quad \vec{f_2}(\vec{q}) = \begin{pmatrix} \rho v \\ \rho uv \\ \rho v^2 + p \\ \rho vw \\ \rho v \left(e + \frac{p}{\rho}\right) \end{pmatrix}, \quad \vec{f_3}(\vec{q}) = \begin{pmatrix} \rho w \\ \rho uw \\ \rho vw \\ \rho w^2 + p \\ \rho w \left(e + \frac{p}{\rho}\right) \end{pmatrix}, \tag{I.0.2a}$$

and

$$\vec{q} = \begin{pmatrix} \rho \\ \rho u \\ \rho v \\ \rho w \\ \rho e \end{pmatrix}. \tag{I.0.3}$$

Here $\vec{q}$ is the state vector of conservative quantities, and $\vec{f_1}$, $\vec{f_2}$ and $\vec{f_3}$ are the so-called flux vectors. The primitive quantities used here are the density ρ, the velocity u, v and w and the pressure p. For a perfect gas the total energy e is related to the primitive quantities as

$$e = \frac{1}{\gamma - 1}\frac{p}{\rho} + \frac{1}{2}(u^2 + v^2 + w^2), \tag{I.0.4}$$

where γ is the ratio of specific heats. To determine the Jacobian of the vectors, one has to rewrite (I.0.3) and substitute the result in (I.0.2a),(I.0.2a) and (I.0.2a). For the u-direction, $\vec{f_1}(\vec{q})$ and

vector $\vec{q}$ become

$$\vec{f}_1(\vec{q}) = \begin{pmatrix} q_2 \\ \frac{q_2^2}{q_1} + (\gamma-1)\left(q_5 - \frac{1}{2}\frac{1}{q_1}\left(q_2^2+q_3^2+q_4^2\right)\right) \\ \frac{q_2 q_3}{q_1} \\ \frac{q_3 q_4}{q_1} \\ \frac{q_2 q_5}{q_1} + (\gamma-1)\left(\frac{q_2 q_5}{q_1} - \frac{1}{2}\frac{q_2}{q_1^2}\left(q_2^2+q_3^2+q_4^2\right)\right) \end{pmatrix}, \qquad \vec{q} = \begin{pmatrix} q_1 \\ q_2 \\ q_3 \\ q_4 \\ q_5 \end{pmatrix} \tag{I.0.5}$$

The Jacobian $J = \frac{\mathrm{d}\vec{f}_1(\vec{q})}{\mathrm{d}\vec{q}}$ can be written as

$$\begin{pmatrix} 0 & 1 & 0 & 0 & 0 \\ j_{21} & (3-\gamma)\frac{q_2}{q_1} & -(\gamma-1)\frac{q_3}{q_1} & -(\gamma-1)\frac{q_4}{q_1} & (\gamma-1) \\ -\frac{q_2 a_3}{q_1^2} & \frac{q_3}{q_1} & \frac{q2}{q1} & 0 & 0 \\ -\frac{q_2 q_4}{q_1^2} & \frac{q_4}{q_1} & 0 & \frac{q_2}{q_1} & 0 \\ (2-\gamma)\frac{q_2 q_5}{q_1^2} & j_{52} & -(\gamma-1)\frac{q_2 q_3}{q_1^2} & -(\gamma-1)\frac{q_2 q_4}{q_1^2} & \gamma\frac{q_2}{q_1} \end{pmatrix} \tag{I.0.6}$$

with

$$\begin{aligned} j_{21} &= -\frac{q_2^2}{q_1^2} + \frac{1}{2}(\gamma-1)\frac{1}{q_1^2}\left(q_2^2+q_3^2+q_4^2\right) \\ j_{52} &= \gamma\frac{q_5}{q_1} - \frac{1}{2}\frac{1}{q_1^2}\left(3q_2^2+q_3^2+q_4^2\right) \end{aligned}$$

The eigenvalues of (I.0.6) are

$$\lambda_{1,2,3} = \frac{q_2}{q_1} \tag{I.0.7}$$

$$\lambda_4 = \frac{q_2}{q_1} - \frac{1}{2}\Big(4\gamma(\gamma-1)q_1^3 q_5 - 2\gamma(\gamma-1)q_1^2 q_4 - 2\gamma(\gamma-1)q_1^2 q_3 - 2\gamma(\gamma-1)q_1^2 q_2\Big)^{\frac{1}{2}} \tag{I.0.8}$$

$$\lambda_5 = \frac{q_2}{q_1} + \frac{1}{2}\Big(4\gamma(\gamma-1)q_1^3 q_5 - 2\gamma(\gamma-1)q_1^2 q_4 - 2\gamma(\gamma-1)q_1^2 q_3 - 2\gamma(\gamma-1)q_1^2 q_2\Big)^{\frac{1}{2}} \tag{I.0.9}$$

After substitution of the original values of $\vec{q}$, the eigenvalues can be written as

$$\lambda_{1,2,3} = u, \tag{I.0.10}$$

$$\lambda_4 = u + \frac{1}{2}\left(2\gamma(\gamma-1)\left(-u^2-2-v^2-w^2+2e\right)\right)^{\frac{1}{2}}, \tag{I.0.11}$$

$$\lambda_5 = u - \frac{1}{2}\left(2\gamma(\gamma-1)\left(-u^2-2-v^2-w^2+2e\right)\right)^{\frac{1}{2}}. \tag{I.0.12}$$

After substitution of (I.0.4) into the eigenvalues, the result is

$$\lambda_{1,2,3} = u, \tag{I.0.13}$$

$$\lambda_4 = u + \sqrt{\gamma\frac{p}{\rho}}, \tag{I.0.14}$$

$$\lambda_5 = u - \sqrt{\gamma\frac{p}{\rho}}, \tag{I.0.15}$$

where $\sqrt{\gamma\frac{p}{\rho}}$ is also known as the speed of sound. The eigenvalues represent the speed of a perturbation in an Euler flow. A similar calculation can be carried out for the v and w direction.

Appendix II
Object-oriented programming for CFD purposes

When practicing Computational Fluid Dynamics (CFD), three disciplines come in state. The most obvious two are the understanding of aerodynamics and mathematics. The third one is informatics (designing and developing software). To develop professional CFD software, a fair amount of coding skills is absolutely necessary. Without these skills, it is even difficult to have the necessary good sense for the computer-science aspects of different numerical methods.

Only coding the mathematics may not be sufficient in some cases. Being able to *see* what happens during a computation greatly improves understanding of the problem. Or, having a graphical user interface may be desirable for all kinds of reasons. This may put high demands to the programmer. Not only the graphical user interface may need to be taken care of. Planning the application, process time slicing, threading, taking care of usability and all other kind of things are key factors for a good application. When moving the console application to a GUI application, the amount of code increases. Structuring the code, protecting code and making components reusable is highly recommended. Object-oriented programming (in C++, Java or other object-oriented language) is the only way to ensure this, in contrast to procedural programming (in Fortran, Basic or other procedural language). But before the coding begins, it is essential to solve the object-oriented problem by constructing a model. Being able to visualize an application may be the most important step in the entire software development process, independently of the choice of used programming language. By fully planning an application, the coding process becomes smoother and, most importantly, the debugging time decreases significantly.

II.1. Java

Java is a relatively new programming language. The language has enormous appeal for many reasons. One major reason is that it is largely platform independent, meaning that an application written for one computer is very likely to run unchanged on another computer. Thus, a single application can be written, whitch can be executed on any of a company's / institute's computers, whether they are Unix workstations, Macs or PC's. A second major advantage is that Java has a C-like syntax, but drops many of the more obscure and messier features of C. Having a C-like syntax means that it is already partially familiar to millions of people. This fact aids its acceptance. An example of Java's improvement over C is its treatment of character strings. In Java, strings are treated as objects and are manipulated by a set of standard methods. In C, strings are manipulated with pointers, which is a much more error-prone process. A third major advantage of Java is that it is object oriented, which should make code written in Java more reusable between applications. With a little forethought, classes and methods written for one application are usable in another

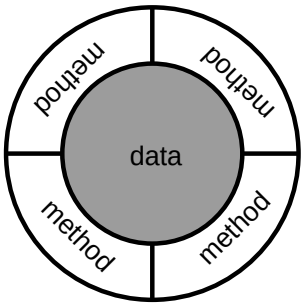

Figure II.1: An object with data encapsulation by its methods.

application without change, because the way that the data and methods are encapsulated in the objects prevents undesirable interactions among them. A fourth major advantage of Java is that it is possible to write device-independent graphics applications. Languages such as C and Fortran do not provide device-independent graphics, because the programmer must concern himself / herself with the specific details of the hardware being used to display the graphics. The language definitions do not include standard APIs for working with graphics at a higher level. In contrast, Java's AWT and Swing Graphics classes provide a higher-level abstraction that is the same across any Java implementation, making device-independent graphics practical. A final advantage of Java is that it is free. Sun provides a free Java Development Kit for download from its World Wide Web site (http://java.sun.com). This kit includes for free: Java compilers, development tools, and class libraries. Other vendors such as IBM offer free trial editions of their Java Integrated Development Environments, which include excellent debuggers. For many budgets, no price is the only right price.

The Unix operating system and C language reached their current strong positions because AT&T made Unix available essentially free to universities in the 1970's and 1980's, where generations of students were trained to use them, and went on to spread their use widely throughout the working world. Java is poised for a similar but even more rapid spread. With Java it is not only possible to make applications but also makes it easy to create applets that can be exchanged and executed freely across the Web.

The next sections describe briefly some properties of Java as an object-oriented program language.

II.2. Objects

An essential characteristic of objects is data encapsulation. An object consists of both data and procedures for manipulating data. Other objects or external code cannot access these data directly, but must send messages to the object requesting its data.

An object's procedures (called methods) respond to the message and may return data to the requesting object. As the symbol of figure II.1 suggests, an object's methods do the encapsulating, in effect mediating access to the object's data.

II.3. Inheritance, abstraction and polymorphism

Inheritance enables you to define a new class based on a class that already exists. The new class will be similar to the existing class, but will have some new characteristics. This makes programming easier, because you can build upon your previous work instead of starting from scratch. Inheritance (and other features of object oriented languages) is responsible for the enormous recent increase in features and complexity of modern software. Graphical user interfaces define each visual component (windows, buttons, sliders, and icons) by using inheritance with a "toolkit" of basic components. Polymorphism ("many forms") is the process by which objects are able to respond to different environments. An object-based example is an object that can produce a correct result in each situation.

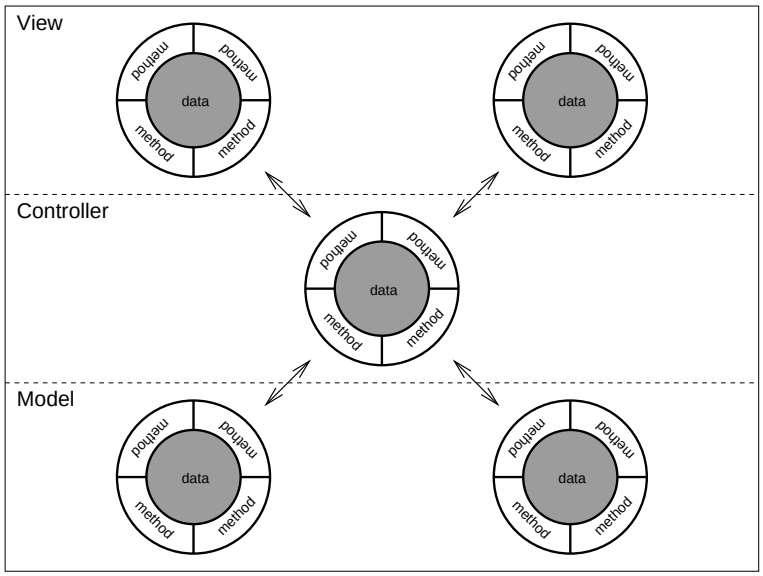

Figure II.2: Model-View-Controller

II.4. Multithreading

Multithreading is the ability of a program to do more than one thing at once, for example, computing a CFD related problem, plotting the results realtime on screen and storing data to disk. Multithreading extends the idea of multitasking by taking it one level lower: individual programs will appear to do multiple tasks at the same time. Each task is called a thread, which is short for thread of control. Programs that can run more than one thread at once are called multithreaded. Threads in Java can take advantage of multiprocessor systems if the base operating system does so (which happens with Sun Solaris, Apple OSX and Linux). The big difference between multiple processes and multiple threads is that while each process has a complete set of its own variables, threads share the same data segment. The advantage is that it takes much less overhead to create and destroy individual threads than it does to launch new processes. On the other hand, without taking precautions, synchronization problems can occur. This can happen when threads share data objects. A running thread can access any object to which it has a reference. Since it is possible for two threads to simultaneously have access to the same object, they can interfere with each other. For example, consider two computational threads working together to solve a problem. One thread is reading data from an object, while another thread is updating the data within the same object. Which data is read by the first object? This is usually called a synchronization problem. Java has the ability to synchronize access to shared objects. The program "Visual Shock Tube Solver" makes extensive use of synchronized threads.

II.5. Model-View-Controller

A simple but very effective way to structure an object-oriented program is to design it with the Model-View-Controller (MVC) model in mind. This design helps in the development of maintainable, extendible and above all, understandable systems. The MVC model proposes three types of objects in an application as shown in figure II.2. The object types are separated by abstract boundaries and communicating with each other across those boundaries.

II.5.1 Model Objects

This type of objects represents special knowledge and expertise. Model objects hold data and define the logic that manipulates these data. For example, a State3D object, common in CFD applications, is a Model object. It holds data describing the state in a cell with state variables like u, p, ρ, T, e, etc. Another Model might be one, containing the Osher scheme, a grid, a matrix

manipulation library, etc. Model objects are not directly displayed. They are often reusable, distributed, persistent and portable to a variety of platforms.

II.5.2 View Objects
This kind of objects represents something visible on the user interface, like for example buttons, windows and scrollbars. A view object is "ignorant" of the data it displays. In most cases, the development kit provides all the View objects. For example, Java provides View objects, called "Swing", shipped with the Java 2 Standard Development Kit (J2SDK). Another set of view objects is for example Qt, usable for C++ programs. In some cases it is desirable to create new View objects, like graph views. View objects, especially those in development kits, tend to be very reusable and so provide consistency between applications.

II.5.3 Controller objects
The Controller object acts as mediator between Model objects and View objects. Usually there is one Controller object per application or window. A Controller object communicates data back and forth between the Model objects and the View objects. The Controller object acts as the kernel of the application. Since what the Controller object does is very specific to an application, it is generally not reusable.

II.5.4 Hybrid models
MVC is not advisable in all circumstances. Sometimes it is best to combine roles. For instance, some graph objects (a View object), especially developed for "Visual Shock Tube Solver" are data aware. These objects communicate directly to some Model objects to gain a performance boost.

Appendix III
LevelSet1D

III.1. Introduction

The LevelSet1D program is especially developed for visualizing static water-air flow computation in a closed 1D tube. All implemented methods make use of the level-set method, which is discussed in chapter 2. The program LevelSet1D is designed with ease of use in mind. The graphical user interface enables/allows a user to test some newly developed level-set methods, based on Euler and Navier-Stokes equations.

III.2. Quick start

Start the application LevelSet1D by clicking on the appropriate icon or by typing **java -jar LevelSet1D.jar** in a console. The application is launched and shows the Desktop window of the application. LevelSet1D uses one window to do all preprocessing, computation and post processing analysis. Figure III.1 shows some of these functions in action. All functions, properties and possibilities of the program are explained in more detail in the next session. Now we take a quick tour and complete a computation. First make the Settings window visible by selecting "View" → "Settings". Select the tab "Computation" and click on the radio button "Extrapolate air only". Select "Round speed used for level-set convection" and set the amount of digits to "1". Optionally, you can minimize the Settings window. Second, make the windows "Solution", "Flux" and "Console" visible from the "View" menu. Third, hit the "Start" button from the button bar or select "Run" → "Start" from the main menu. The "Save" dialog pops up, asking you a name for the output files. Type "test.prj" and click the Save button on the dialog. (Don't forget the .prj extension. This is essential.) The computation is done real time. The plot windows show the intermediate results of the solution and fluxes. The console gives all kinds of useful messages. The computation stops when the solution has converged or when the maximum number of iterations has been reached. Congratulations! Your first CFD computation with the LevelSet1D program has been completed successfully.

III.3. Settings

Many different models, methods and fixes can be tested within the LevelSet1D program. However, it must be stated that not all combinations of settings do result in a useful answer. Most of the combinations of settings don't even converge. So, don't blame the program when the results look bad. The golden rule of numerical computation is valid: "When you put garbage in, you get garbage out."

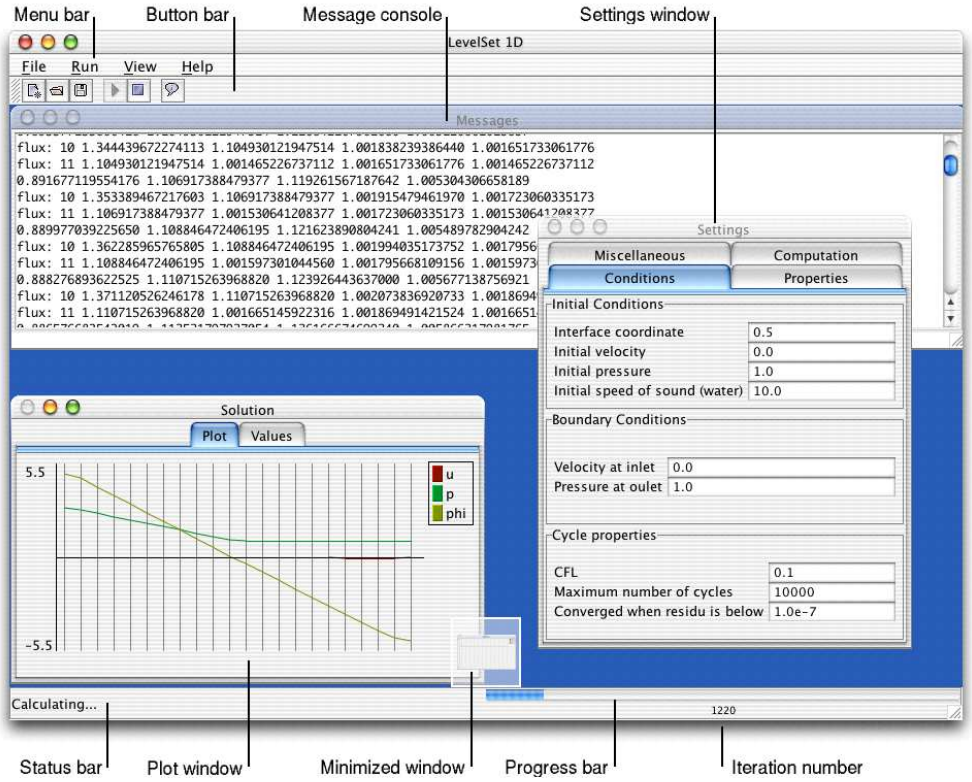

Figure III.1: LevelSet1D main window.

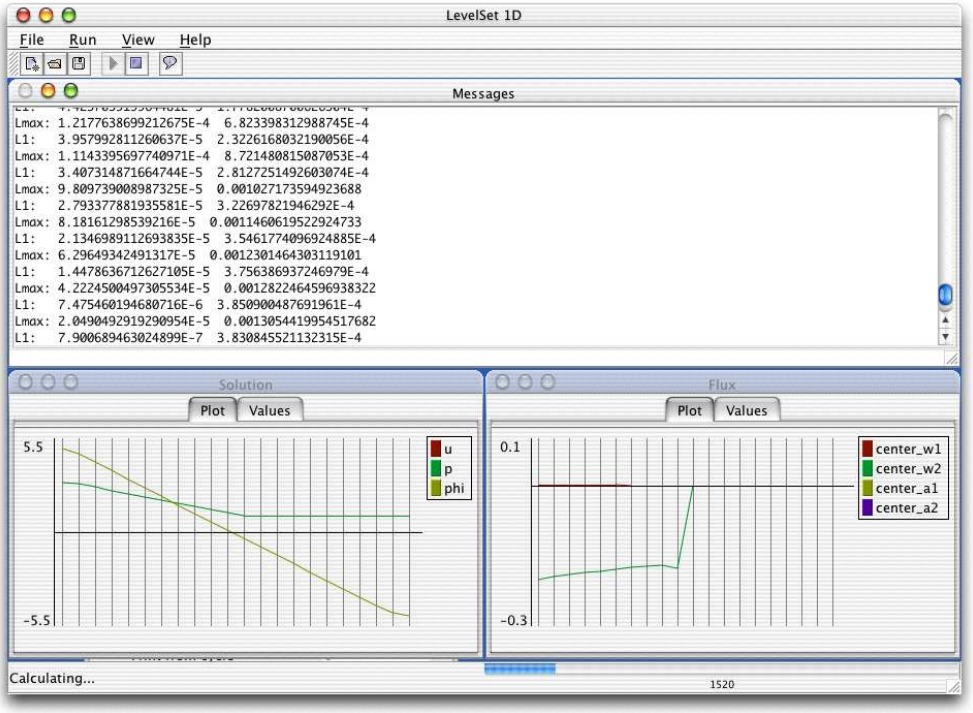

Figure III.2: LevelSet1D, running a computation with real-time evaluation of the values u, p, ϕ, mass flux and momentum flux values.

III.3.1 Computation
In this tab pane, the computation method is chosen. Because "Conservative" and "Non-conservative" are not fully implemented at the time of writing, the radio buttons are disabled and can't be selected. Within the ghost-fluid method, some options are possible:

- **Extrapolate none:** No extrapolation techniques are used in cells of type 3. Computation of fluxes and forward Euler updates are done directly with use of the actual state values.
- **Extrapolate water and air:** On both sides (water and air) extrapolated state values are used in cells of type 3 when computing the individual fluxes and forward Euler time update.
- **Extrapolate air only:** For computing the individual fluxes and forward Euler updates, only state values of air will be extrapolated in cells of type 3. Water uses the actual values.
- **Diffusion (Navier-Stokes):** Enable this option to make use of Navier-Stokes equations, rather than Euler equations. Note that the Reynolds number has to be greater then zero in order to make this option valid.
- **Round speed used for level-set convection:** In most cases, the computation won't converge to machine precision due to a first order error in the velocity u, causing the level-set function to be convected for ever. To prevent this, enable this option. For very coarse grids, the recommended amount of digits is 1. Refining the grid decreases the numerical error and allows more digits.

III.3.2 Conditions
In this tab pane, the conditions of the problem are set.

- **Interface coordinate:** The initial position of the interface in the tube. A valid range of this value is between 0 and 1.
- **Initial velocity:** The initial velocity of the whole flowfield.
- **Initial pressure:** The initial pressure of the whole flowfield.
- **Initial speed of sound:** The initial speed of sound for water.
- **Velocity at inlet:** Boundary condition at the left boundary.
- **Pressure at outlet:** Boundary condition at the right boundary.
- **CFL:** Courant Friedrichs Lewy (CFL) condition, defined as $\sigma = u\frac{\Delta t}{\Delta x}$, where u is defined as the highest perturbation traveling speed, can be 1 or less when using forward Euler time integration. However, when a source term like gravity is added to the equation set, numerical stability is first obtained for much smaller CFL values.
- **Maximum number of cycles:** This value indicates the maximum amount of iterations, allowed by the user. It is intended to avoid an infinite amount of iterations, when the problem does not tend to converge.

Converged when residual is below: The computation is defined as converged, when the residual of unsteadyness is below this given value. The residual is defined as the L1 norm of the vector $\sum_{i=0}^{N} q_i^{n+1} - q_i^n$, with i and n defined as the cell index and the time step, respectively.

III.3.3 Properties
This tab pane allows the user to specify the flow properties of each flow medium.

- **Density:** Density value ρ for each medium.
- **Gamma:** Ratio of specific heats, γ.

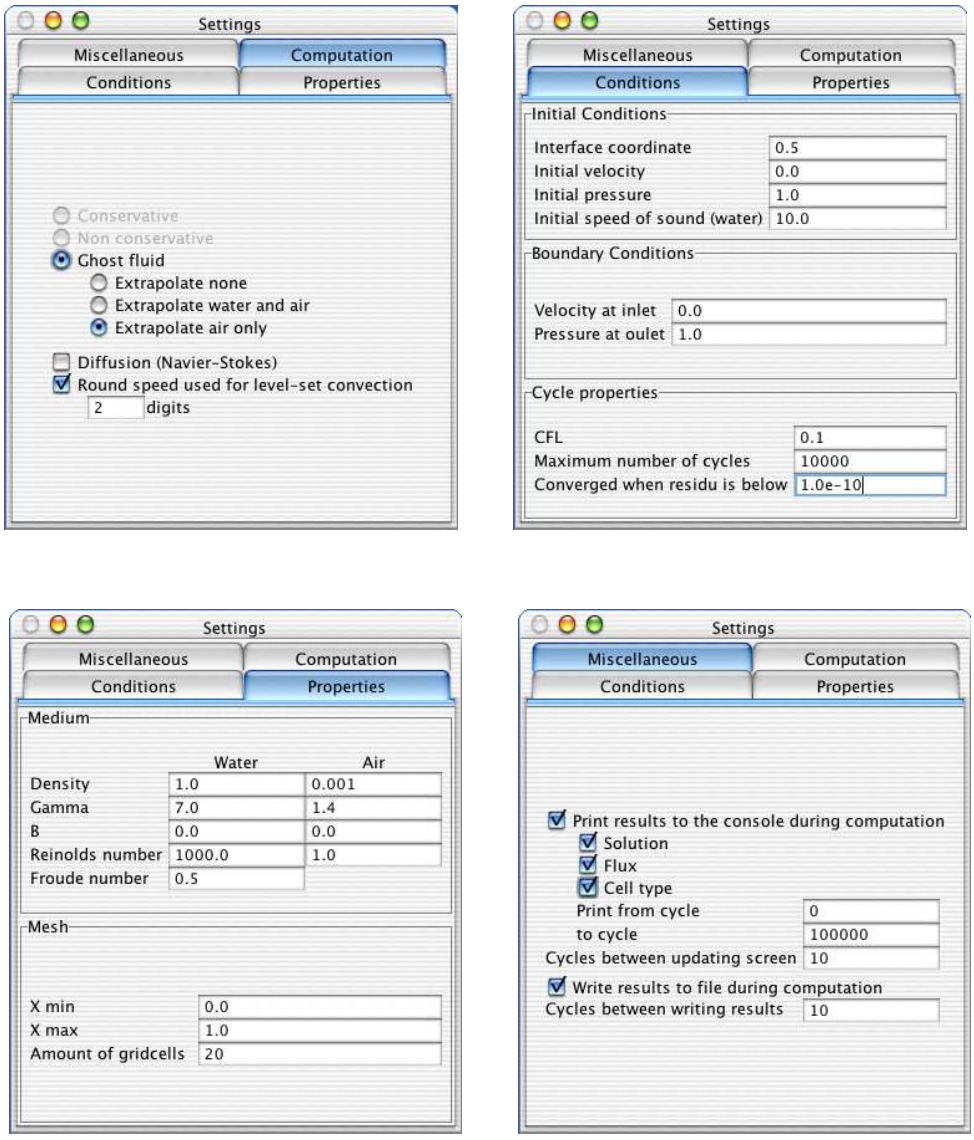

Figure III.3: Available tab pages of the Settings window

- **B:** Tait's constant. This value is automatically computed by the application. Altered values are ignored.
- **Reynolds number:** Reynolds number Re, ratio of inertial to viscous forces. This value is only used for viscous computations, the option "diffusion" in tab pane "Computation" has to be turned on. For all non-viscous computations, the Reynolds number is ignored.
- **Froude number:** Froude number Fr, ratio of inertial to buoyancy forces. This value is used for the source term for modelling gravity.
- **X min:** Minimum value of the x coordinate of the mesh. The default value is 0.
- **X max:** Maximum value of the x coordinate of the mesh.
- **Amount of gridcells:** Amount of gridcells N. The cell width h is given by $\frac{x_{max}-x_{min}}{N}$.

III.3.4 Miscellaneous
This tab pane enables the user to control the amount of output during computation.

Appendix IV
Visual Shock Tube Solver

IV.1. Introduction

The program Visual Shock Tube Solver enables the user to investigate all kinds of Riemann problems, like blast waves, colliding shock waves, near-vacuum expansions and even two-fluid problems, without putting one's life or health in danger. All shock tube problems can be investigated without leaving the safe environment of the office desk and armed wheelchair. Not only a bunch of numbers will be shown, but all aspects with respect to the Riemann solution can be watched in real time animations and crystal clear graphs. The program offers a user friendly Graphical User Interface, so the solutions to almost all Riemann problems are just a mouse click away. These aspects make the program suitable for both educational purposes and as daily reference calculator. It's a "must have" for all scientists, who happen to do anything related to gasdynamics.

IV.2. Users guide

Start the application Visual Shock Tube Solver by clicking on the appropriate icon or by typing **java -jar libGasdynamics.jar** in a console. The application is launched and shows the main window of the application. Let's have a quick look at the screen. At the top of the window, the menu is visible and has the following menu items:

- **File → Save UP Plot...**: When selected, a file location and file name is asked by a dialog window. After clicking Ok, the (u, p)-plot will be saved in plain text format. The data file can be used immediately by gnuplot. For example, the gnuplot commando for showing the plot on screen is: *plot "filename" using 1:2 w l, "filename" using 3:4 w l.*
- **File → Reference State...**: This menu item or the key combination [ctrl]-[r] pops up the reference state dialog. The present computations of two-fluid Riemann problems rely on Tait's equation of state. Pressure and density from either state 1 or state 4 are coupled to a reference state at infinity. Note that for single-fluid computations, the reference state is not used.
- **File → Print Preview...**: This menu item or the key combination [ctrl]-[shift]-[p] shows a print preview of the solution. The print preview shows the result to be expected when selecting Print.
- **File → Print**: This menu item or the key combination [ctrl]-[p] sends the (u, p)-plot, the (x, t)-plot and the numerical values to the default printer. Depending on the underlying operating system, a printer dialog may appear before printing.

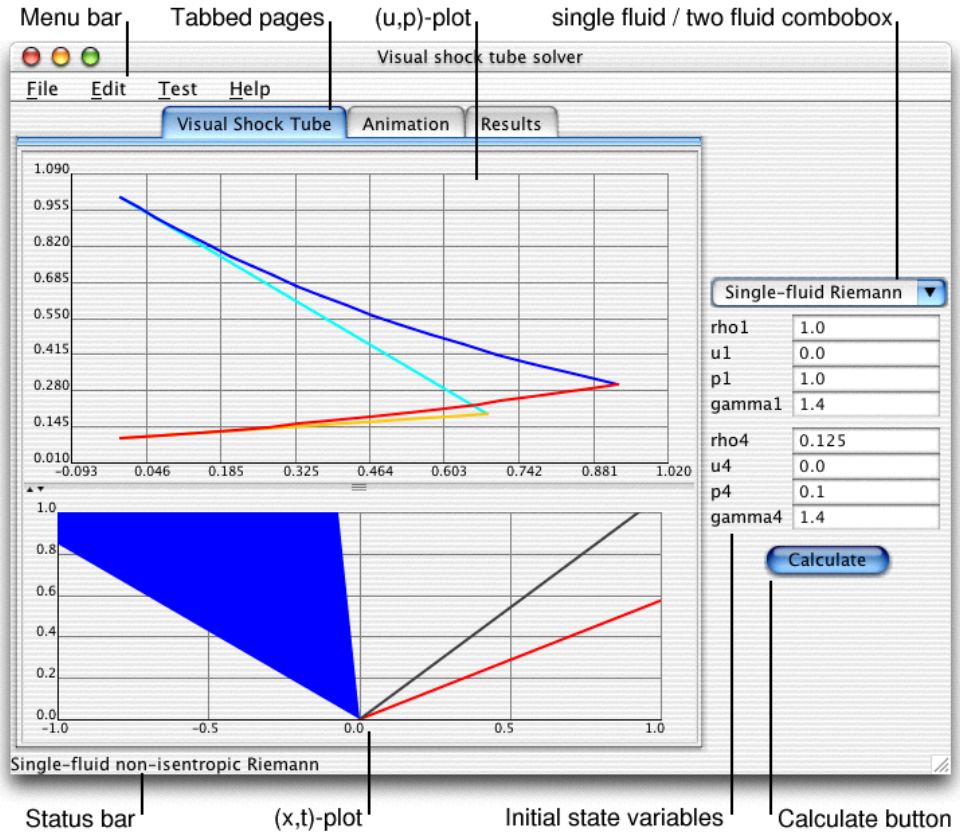

Figure IV.1: Visual Shock Tube Solver: $(u.p)$-plot and (x,t)-plot .

- **File → Exit**: This menu item or the key combination [ctrl]-[x] terminates the application. Note that no data are saved automatically. If you want to store the results, they have to be printed or saved before closing the application.

- **Edit**: The results shown in the tab pane Results, can be easily used in other programs, like word processors or spreadsheets by simply selecting the desired text with the mouse and copying the selected text to the clipboard of the operating system by selecting **Edit → Copy** from the menu or by pressing [ctrl]-[c] at the same time. From within the destination application, select **Edit → Paste** or press [ctrl]-[v]. (The paste action can be application specific. Please refer to the user manual of the application if applicable.)

- **Test**: This menu contains some single-fluid and two-fluid test cases, as described in chapter 4.

- **Help → About...**: This menu item shows an information window, showing the name of the program, authors, version number, copyright information, some nice logo's and an Ok button to close the message box.

The right hand side of the main window shows a combobox and an array of state variables for the initial states 1 and 4. State 1 represents the left state and state 4 represents the right state of the Riemann problem. For a singe-fluid computation, pressure and density values can be chosen freely. For a two-fluid computation, pressure and density are coupled to each other via a reference state. When changing pressure, the corresponding density is automatically updated and vice versa. Checking the reference state, before doing a two-fluid computation is highly recommended to avoid unexpected results. For a two-fluid computation, the extra variable B, a material constant, used by Tait's equation of state, will be visible.

When all initial values are given and the "calculate" button is pressed, the tab pane "Visual Shock Tube" draws the (u, p)-plot and the (x, t)-plot (figure IV.1). The (u, p)-plot gives two solutions: the linear solution and the exact, non-linear solution. The linear solution shows compressions as orange coloured, straight lines and expansions as cyan coloured, straight lines. The non-linear solution shows compressions (known as Hugoniot curves) as red coloured, curved lines and expansions (known as Poisson curves) as blue coloured, curved lines. The (x, t)-plot shows the wave movements in time. Again, shocks are red coloured, expansion waves are blue coloured and contact discontinuities are grey coloured.

The tab pane "Animation" (figure IV.2) contains 3 kinds of graphs. The above-left section contains three coloured bars, representing the state values density ρ, velocity u and pressure p of the tube at the time of the time slider. Red denotes a high value, blue denotes a low value. Of course, all values in between look something like reddish-blue or blueish-red. The lower-left section contains the same (x, t)-plot as the first tab pane does. However, a horizontal line is added, denoting the time corresponding to the indication of the time slider. The right half contains three plots: (x, ρ), (x, u) and (x, p). The three plots give a representation of the state variables of the tube at the chosen time. When pressing the "Play" button, all plots animate simultaneously what happens in the Riemann shock tube. The "Back" button sets the scene to the initial state, the "Forward" button sets all values instantaneously to $t = 1$. The time slider enables the user to see the state of the whole shock tube for every time value between $t = 0$ and $t = 1$.

The tab pane "Results" (figure IV.3) shows all numerical values. Those values can be exchanged to other programs with copy and paste actions from the edit menu.

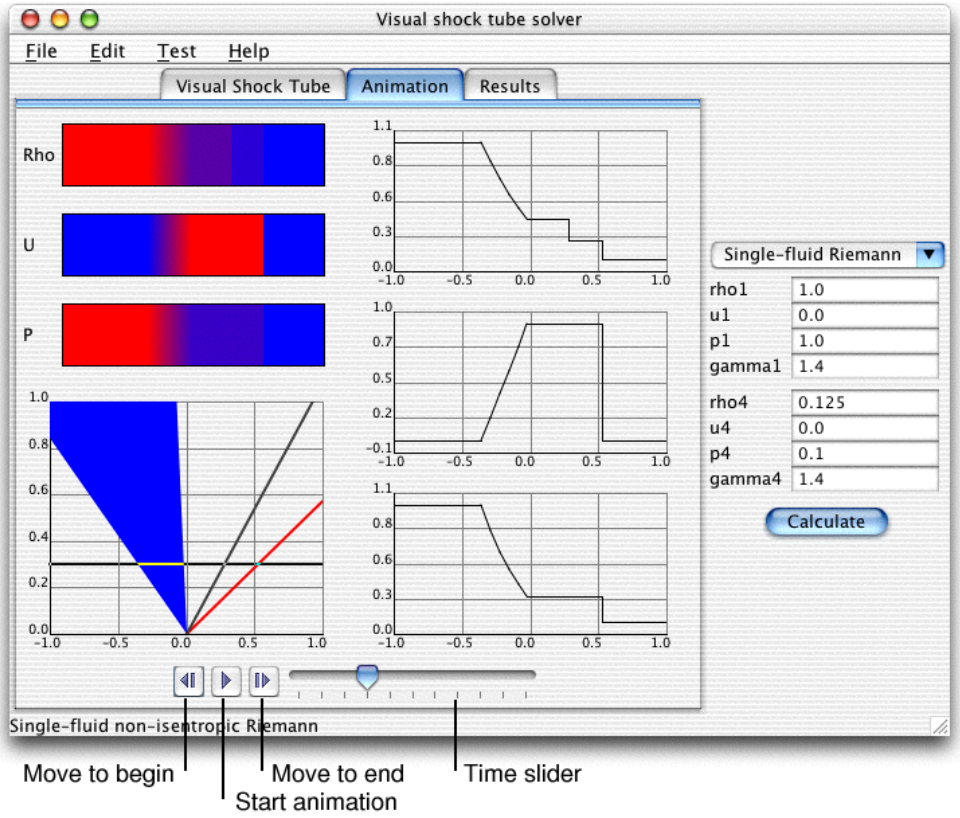

Figure IV.2: Visual Shock Tube Solver: Animations

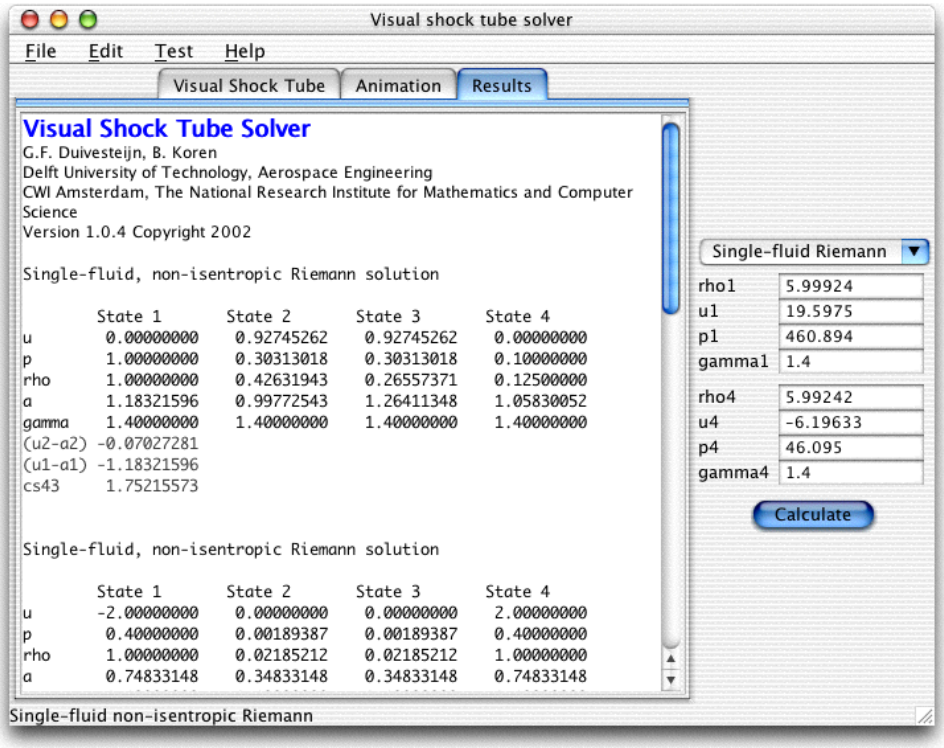

Figure IV.3: Visual Shock Tube Solver: Numerical results

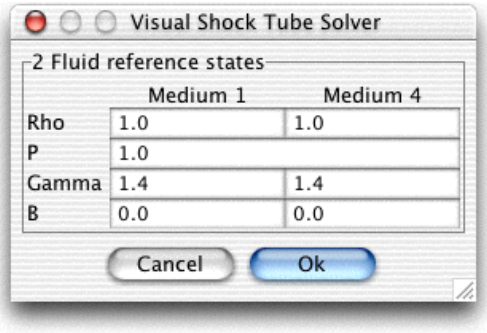

Figure IV.4: Visual Shock Tube Solver: Reference state dialog

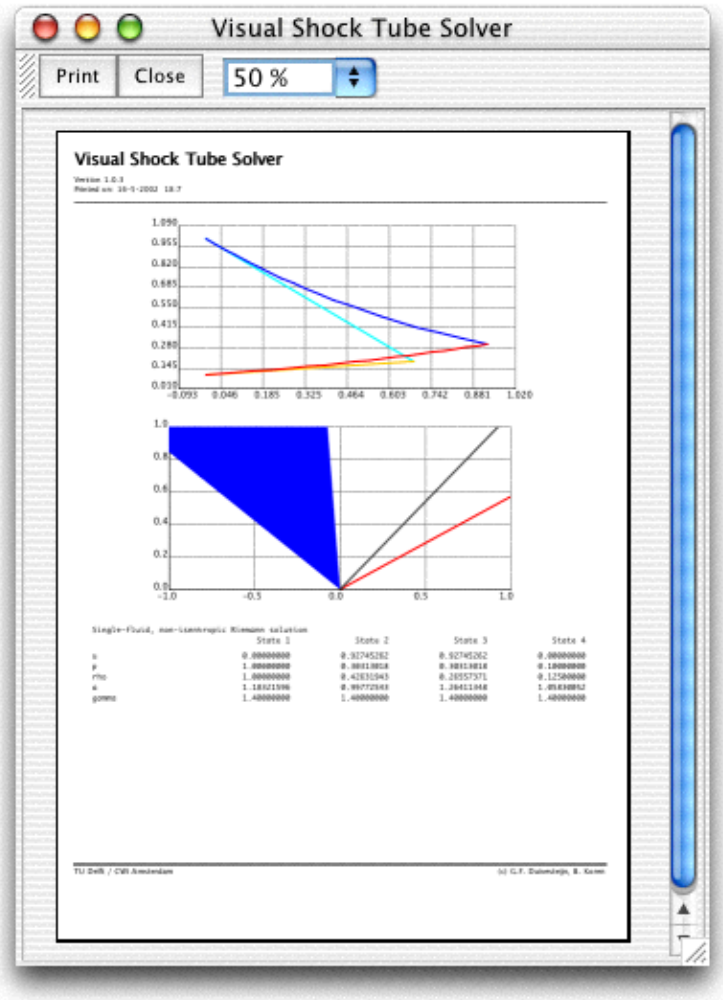

Figure IV.5: Visual Shock Tube Solver: Print Preview

IV.3. Internal structure

This section explains briefly some programming related features, used for "Visual Shock Tube Solver". The program has been developed entirely in native Java. Its byte code is platform independent, the code is object-oriented, with reusability in mind and the program itself is designed in the MVC-style.

IV.3.1 MVC design

The (simplified) structure of the program is given in table :

View objects	Description
RiemannPlotUP	Graph bean for plotting Hugoniot curves and Poisson curves. The input variables consist of 4 sets of plot data. The bean has autoscale facilities. This object is not data aware and has to be controlled by a Controller.
RiemannPlotXT	Graph bean for plotting shock- and expansion waves in physical plane. The input variables consist of 4 state objects. Like the previous object, this object is not data aware and has to be controlled by a Controller when used as static graph. However, when used for animations, the horizontal line, denoting the current time, *is* data aware.
RiemannColourBar	Graph bean for plotting a coloured state representation. The bean uses an adaptive grid with exact wave locations. The input variables consist of an array of wave locations and an array of state values for given locations. To improve animation performance, the bean is data aware. As for the input variables, not the real values are given, only the memory addresses that point to the value arrays.
RiemannPlotRUPX	Graph bean for plotting (ρ, x), (u, x) or (p, x). The bean uses an adaptive grid with exact wave locations. The input variables consists of an array of wave locations and an array of state values for given locations. To improve animation performance, the bean is data aware. As for the input variables, not the real values are given, only the memory addresses that points to the value arrays.
Swing toolkit (J2SDK)	Contains all necessary standard components to develop the graphical user interface.

Controller objects	Descripton
RiemannFrame	Main Controller object. All events, given by the user are interpreted and processed. Data from edit boxes are parsed and sent to Model objects. Returned values from the Model objects are stored in memory or sent to the appropreate View objects.
AnimationThread	Initially started by the main Controller. (Strictly, this object sits in between a Controller object and a Model object.) This object acts as core engine for the animation.

Model objects	Description
State1D	Data container, describing all flow properties for a given control volume. The data is protected and only accessible by the object's methods.
Riemann	Library of gasdynamic related functions. This object acts as intellectual core of the program.
Colt	Highly optimized math library, developed by CERN.

IV.3.2 Animation
The animation is controlled by a threaded object, called AnimationThread. The AnimationThread object is started by the RiemannFrame Controller. The thread increases time with small steps. For every time step, the new wave locations are computed and stored in memory. Instead of copying all data values to the individual graph objects, the graph objects have the pointer of the data arrays, which makes them data aware. This enables *all* graph objects to gather the data from the same resource directly from memory, without expensive internal memory copy actions. When the AnimationThread has updated the data, it sends a repaint request to the operating system. The operating system fires a repaint action to the graph objects. The graph objects repaint themselves with the data as found in memory.

IV.3.3 Double buffering
When the graphs are updated (erased and repainted with new values) onscreen, a perceptive flickering can result. Double buffering eliminates flickering by updating the graph objects in an offscreen buffer and subsequentially copying the appropriate portion of the offscreen buffer to the object's onscreen representation. Double buffering is done in three steps (figure IV.7). First, all paint actions are done in an offscreen buffer, called buffer 1. When painting actions are finished, the whole buffer is copied to buffer 2. In buffer 2, the freshly painted component waits, until a repaint request is sent from the operating system. The reason for using two offscreen buffers instead of one, is that the moment of the repaint action by the operating system is unpredictable. If using a single offscreen buffer, the operating system can request the buffer in the middle of the painting process, resulting in showing an unfinished painted component. By using 2 offscreen buffers, the repaint action always gets a fully painted graph.

IV.3.4 UML
The Unified Modelling Language (UML) is designed as as standard for building Object-Oriented software. The definition of UML, as stated by the OMG, is:

"The Unified Modeling Language (UML) is a graphical language for visualizing, specifying, constructing, and documenting the artefacts of a software-intensive system. The UML offers a standard way to write a system's blueprints, including conceptual things such as business pro-

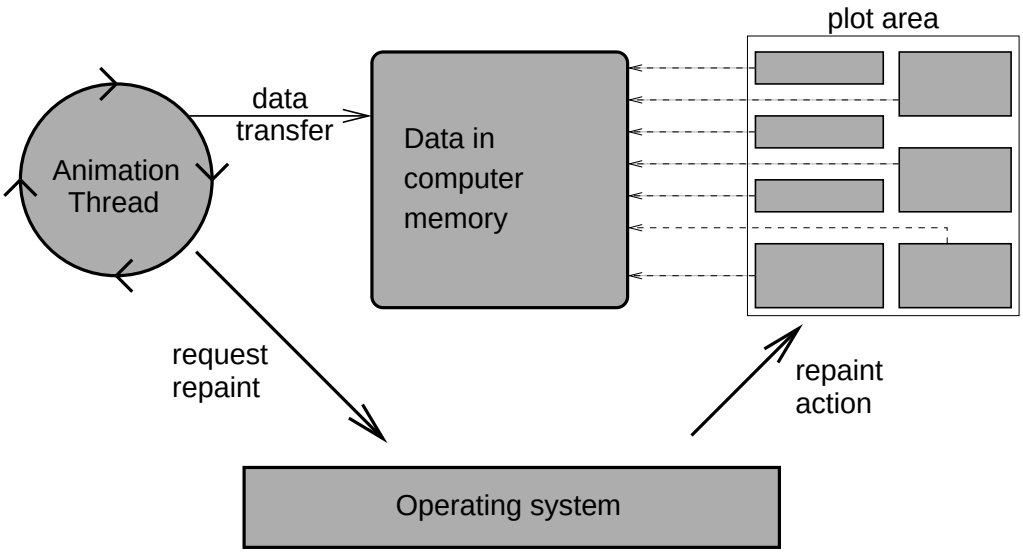

Figure IV.6: AnimationThread

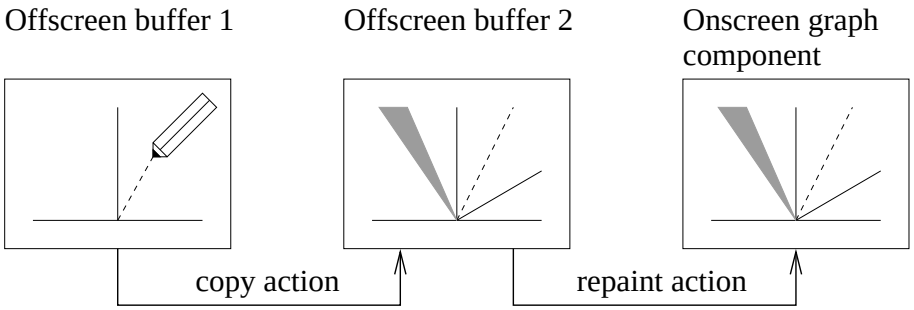

Figure IV.7: The tree steps of the double buffering sequence.

cesses and system functions as well as concrete things such as programming language statements, database schemes, and reusable software components."

The important point to note here is that UML is a "language" for specifying and not a method or procedure. The UML is used to define a software system; to detail the artefacts in the system, to document and construct - it is the language that the blueprint is written in. The UML may be used in a variety of ways to support a software development methodology, but in itself it does not specify that methodology or process.

UML defines the notation and semantics for the following domains:

- The User Interaction or Use Case Model - describes the boundary and interaction between the system and users. Corresponds in some respects to a requirements model.
- The Interaction or Collaboration Model - describes how objects in the system will interact with each other to get work done.
- The Dynamic Model - State charts describe the states or conditions that classes assume over time. Activity graphs describe the workflows the system will implement.
- The Logical or Class Model - describes the classes and objects that will make up the system.
- The Physical Component Model - describes the software (and sometimes hardware components) that make up the system.
- The Physical Deployment Model - describes the physical architecture and the deployment of components on that hardware architecture.

For the program Visual Shock Tube Solver, the Class Model is given in this section. The User Interaction Model, Physical Component Model and Physical Deployment Model is not too complicated and regarded as trivial for this program.

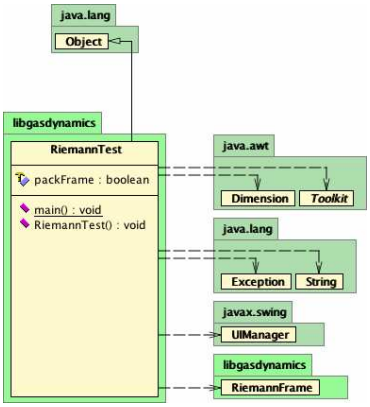

Figure IV.8: RiemannTest.class

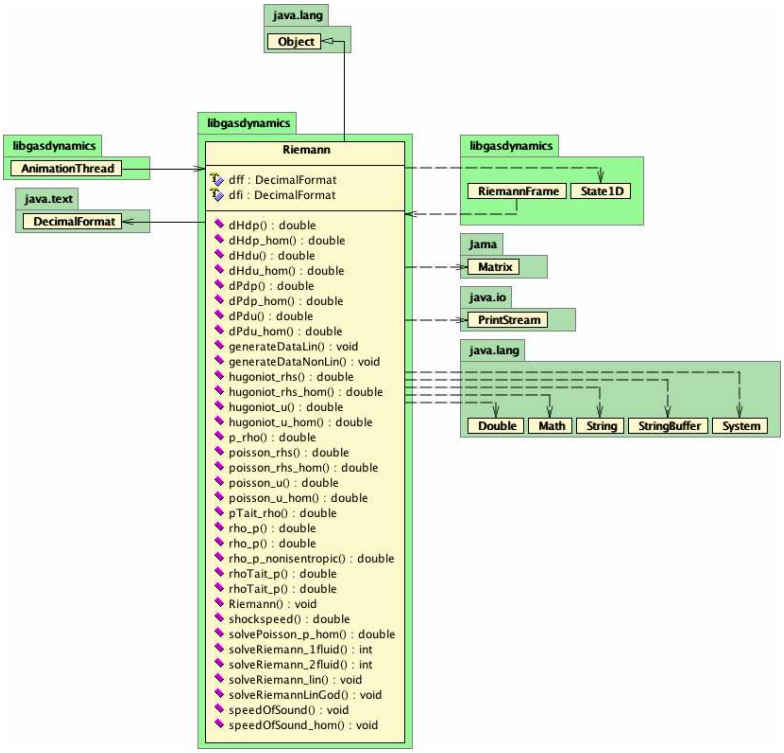

Figure IV.9: Riemann.class

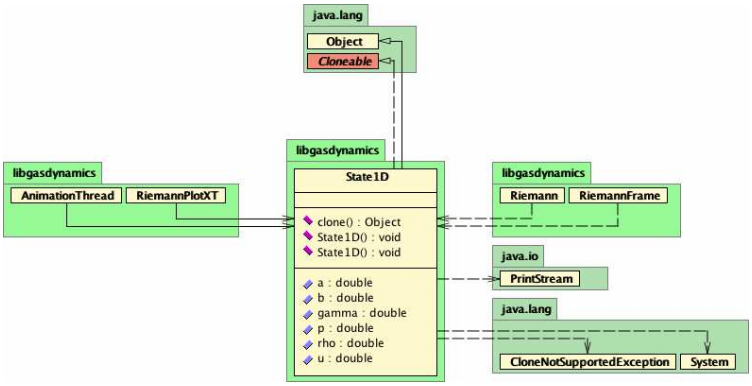

Figure IV.10: State1D.class

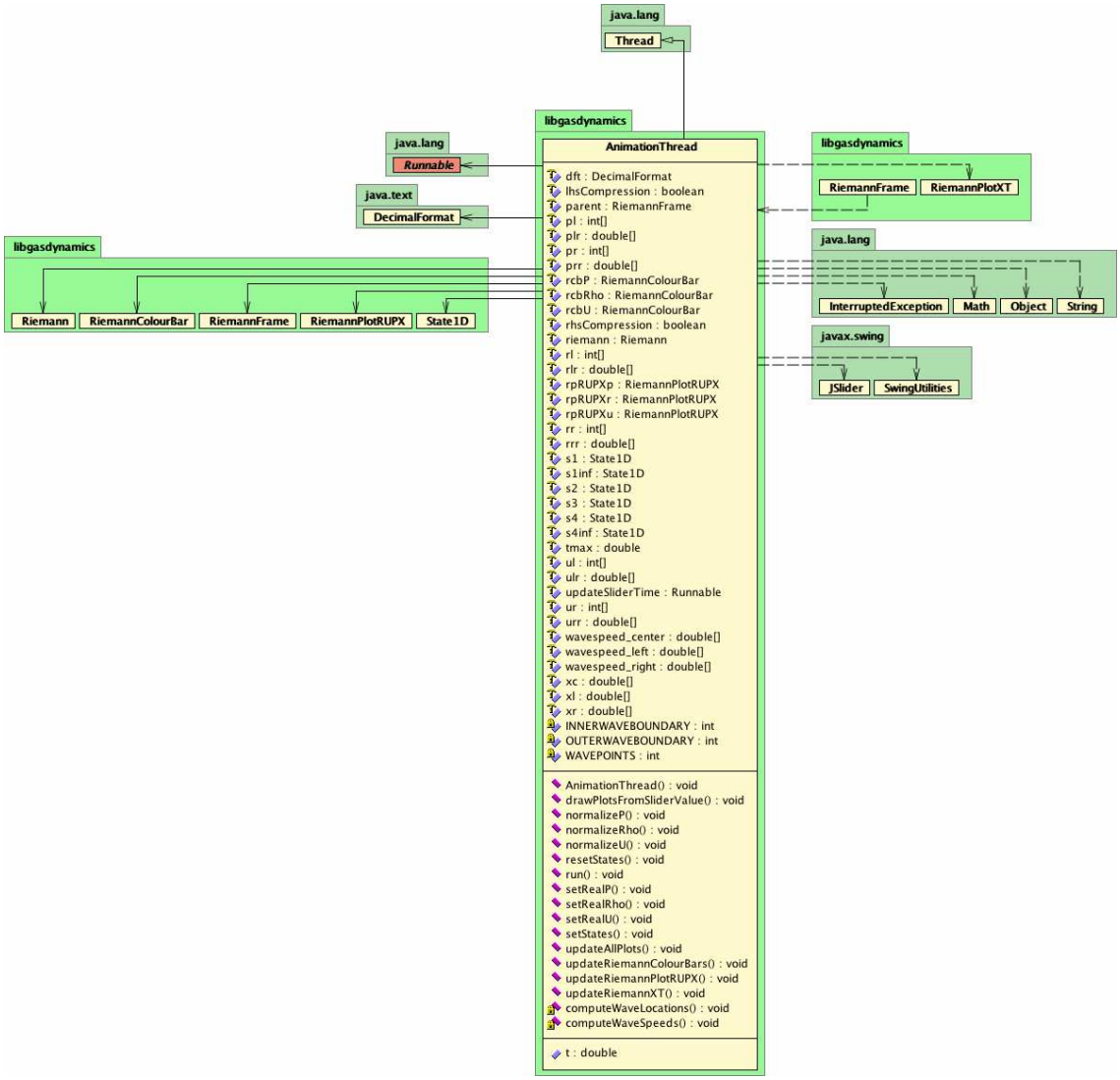

Figure IV.11: AnimationThread.class

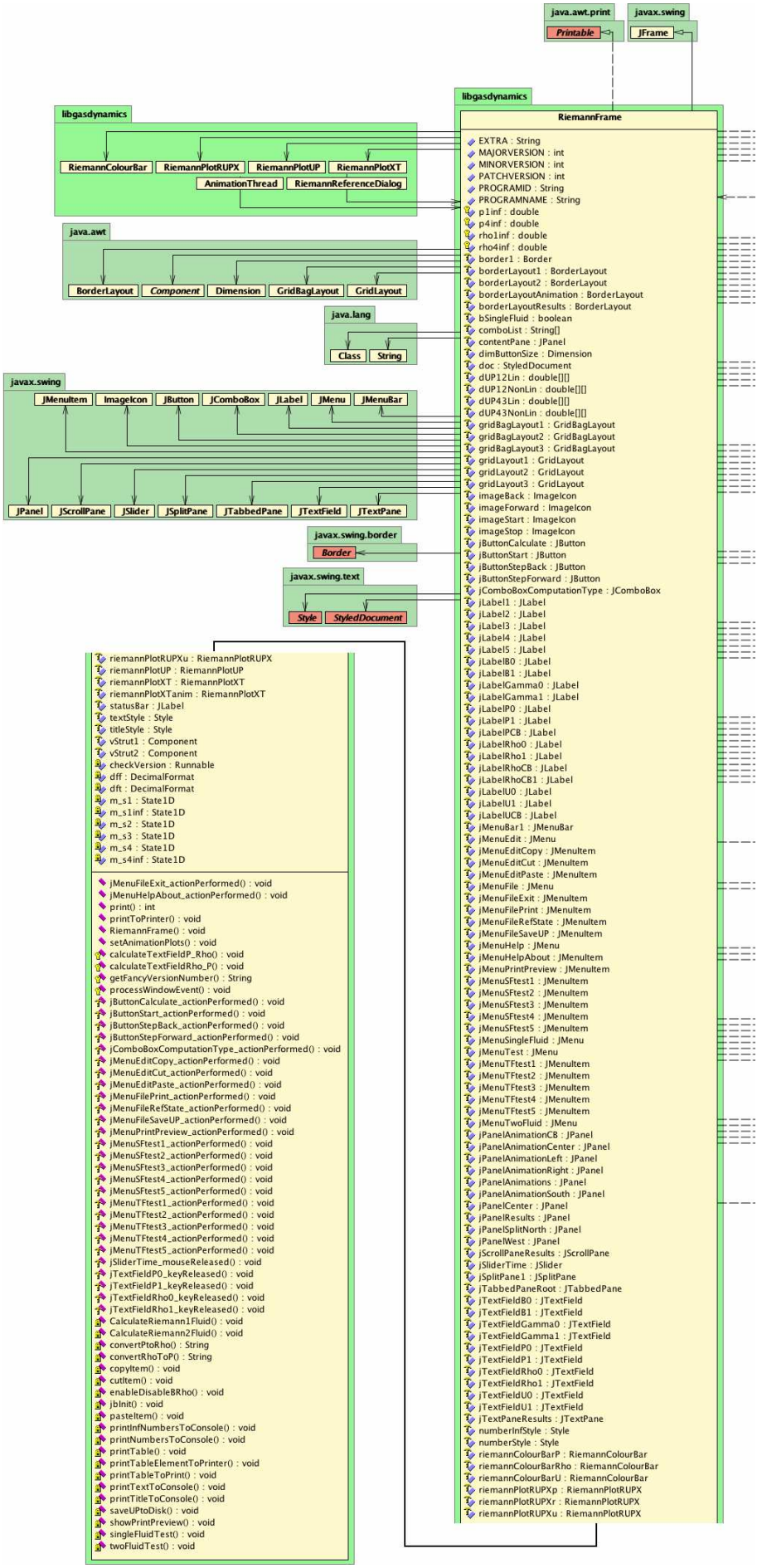

Figure IV.12: RiemannFrame.class

Figure IV.13: RiemannFrame.class (continued)

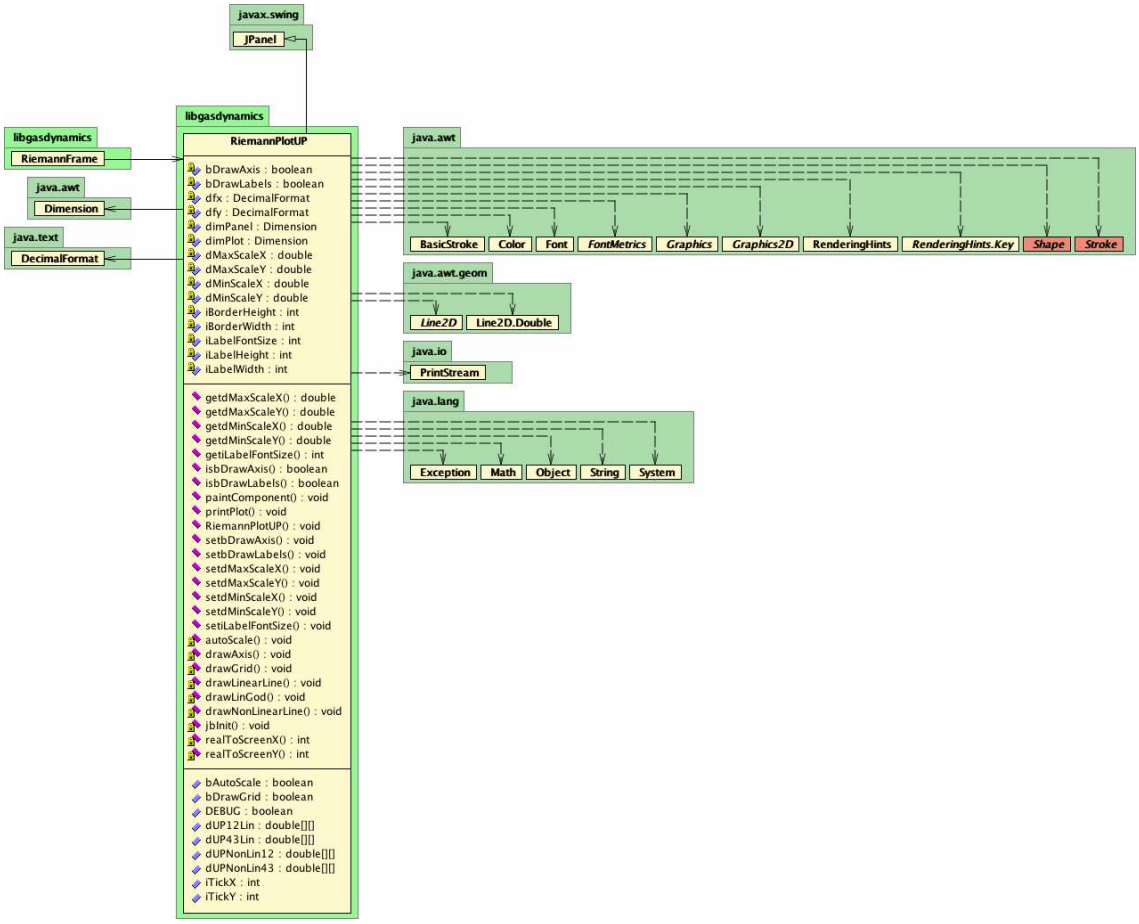

Figure IV.14: RiemannPlotUP.class

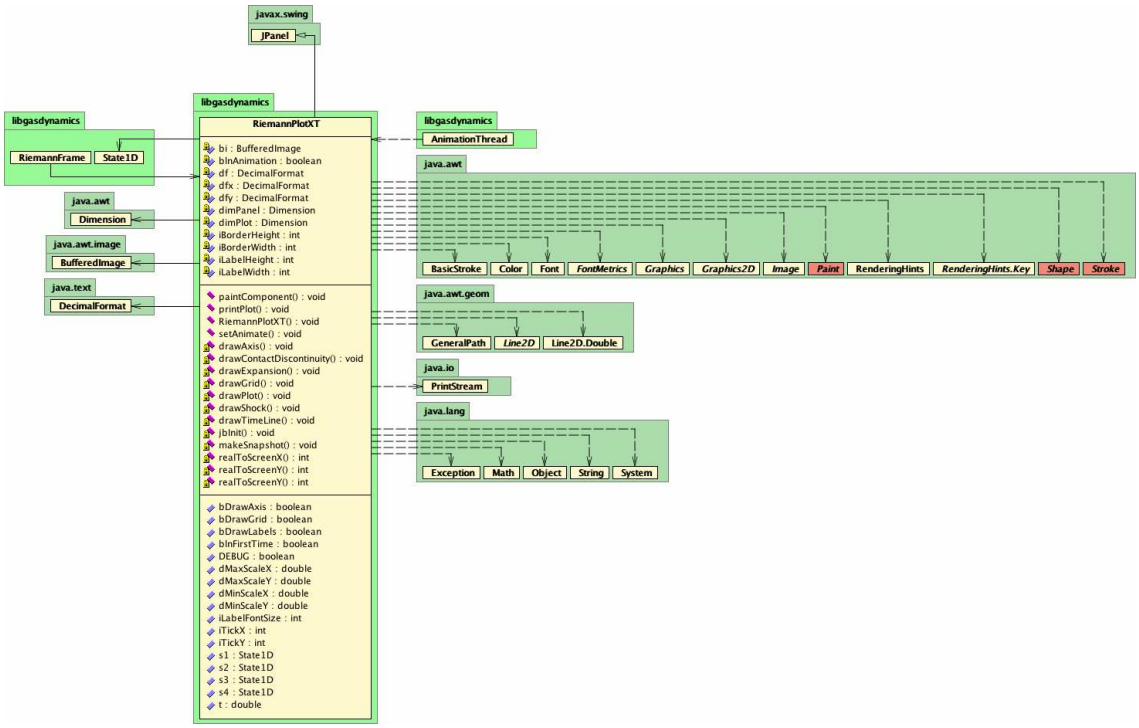

Figure IV.15: RiemannPlotXT.class

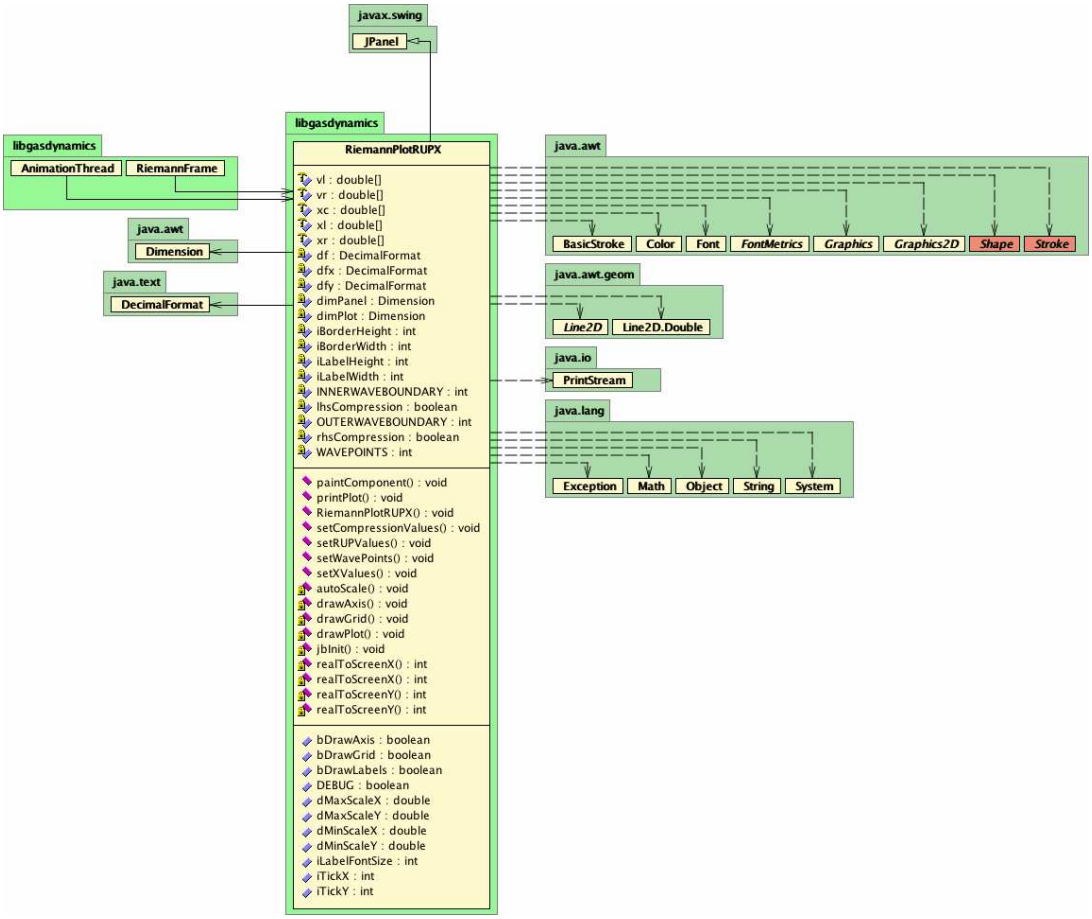

Figure IV.16: RiemannRUPX.class

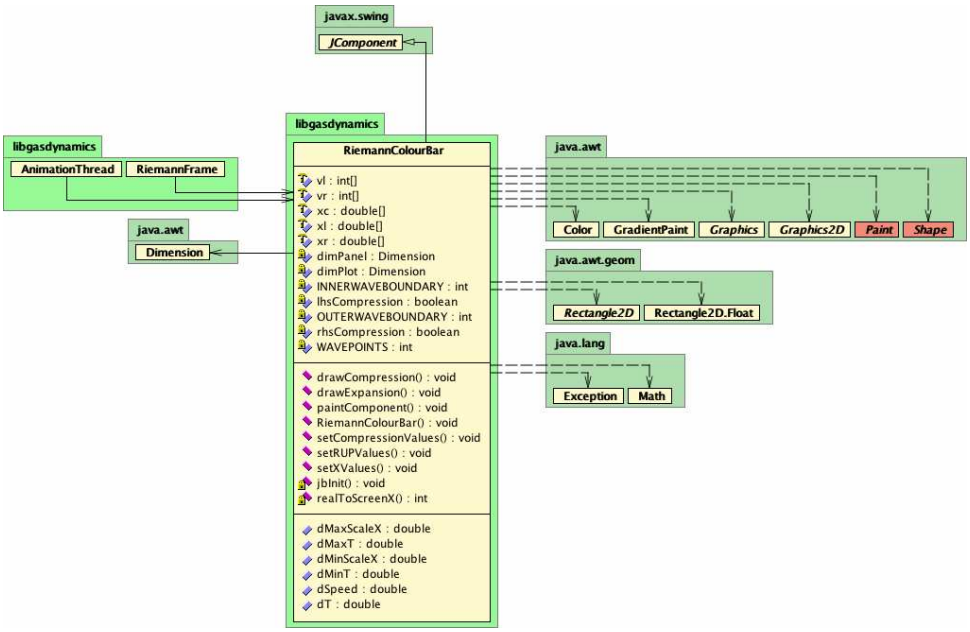

Figure IV.17: RiemannColourBar.class

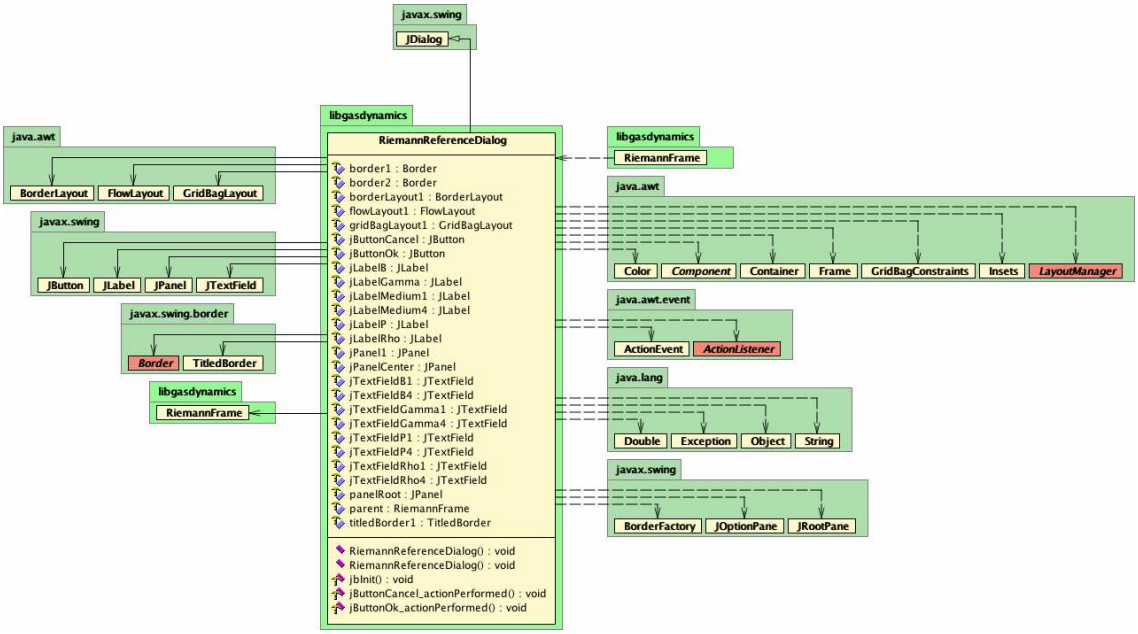

Figure IV.18: RiemannReferenceDialog.class

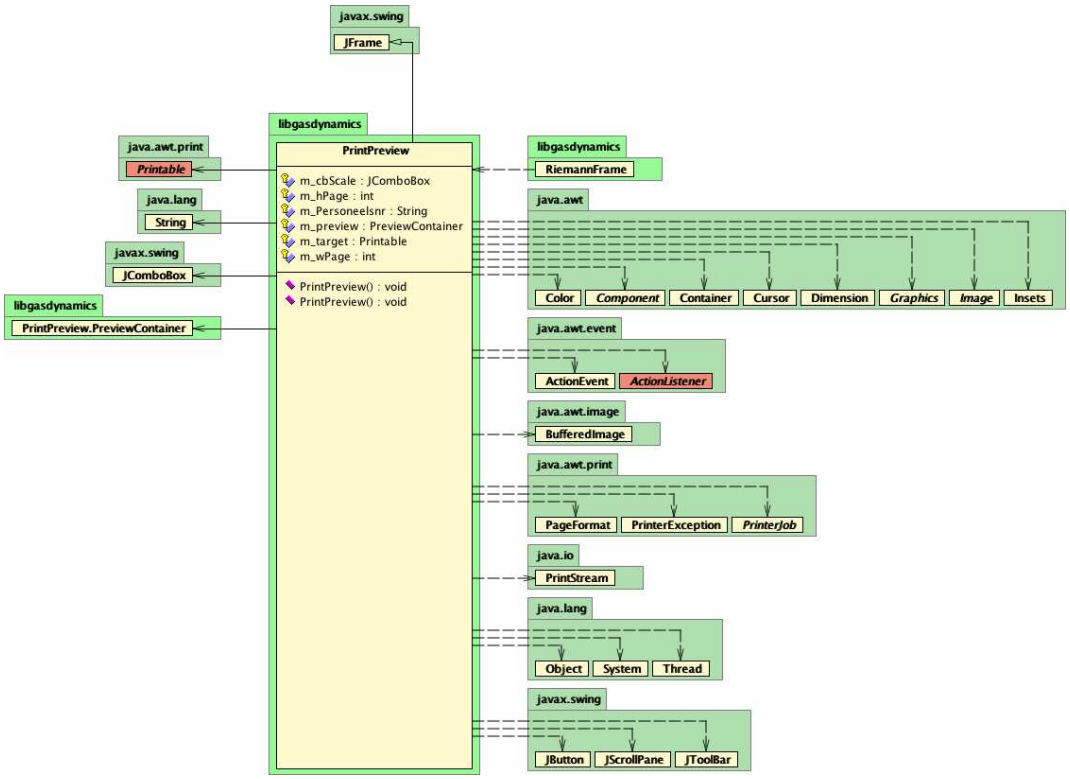

Figure IV.19: PrintPreview.class