Albert-Ludwigs-
Universität
Freiburg

Fakultät für
Mathematik
und Physik

Mathematisches Institut, Abteilung für Angewandte Mathematik

Higher Order Schemes for Simulation of Compressible Liquid-Vapor Flows with Phase Change

Dennis Diehl

Doctoral Dissertation

Dissertation zur Erlangung des Doktorgrades der Fakultät für Mathematik und Physik der Albert–Ludwigs–Universität Freiburg im Breisgau

Freiburg im Breisgau
2007

Dekan:	Prof. Dr. J. Flum
Gutachter:	Prof. Dr. D. Kröner
	Prof. Dr. Ch. Rohde
Datum der Promotion:	4.12.2007

Acknowledgements

I would like to thank Prof. Dr. D. Kröner and Prof. Dr. Ch. Rohde for the very interesting topic of this thesis and for their support during the last years. All my numerous questions have been answered willingly and amiable. I learned a lot.
To all my former and current colleagues of the *Abteilung für angewandte Mathematik*: Thanks for all the fruitful discussions, providing useful hints for improvement, reading draft versions of this manuscript and for any other help to complete this thesis. I have had a great time here.
To my family and friends: For your support, understanding and bearing I thank you. Primarily, this holds for Simona.

This work has been supported by the DFG-CNRS research unit *Micro-Macro Modelling and Simulation of Liquid-Vapor Flows*.

Contents

Chapter 1

Introduction

1.1 Motivation

When steam ships became available in the 19th century, engineers observed a strange damage on the blades of ship propellers caused by an unknown force. In 1917 Lord Rayleigh (Lord John William Strutt) explains in his article *On the pressure developed in a liquid during the collapse of a spherical cavity* [93] that small vapor bubbles that condense at the surface of the propeller blades are responsible for this effect. He gives an equation for the collapse of a bubble, the so called Rayleigh-Plesset equation, see Section 4.4.1 and [16]. Figure 1.3 shows this kind of cavitation damage on the surface of a modern ship propeller.

Small vapor bubbles in a liquid arise when the pressure of the surrounding liquid drops below a certain value, for example caused by operating ship propellers, fast flows or strong sound fields. This effect is called *cavitation*. From the physical point of view the fluid is decomposed into liquid and vapor phases and both phases can condensate or evaporate respectively. Thus, we have a dynamical phase boundary and in general mass transfer over this interface. Cavitation bubbles can behave quite differently, e.g. they can disappear immediately or grow until they break up into an ensemble of smaller bubbles. Depending on the environment these kinds of bubbles can also begin to oscillate. For instance in *weak* sound fields bubbles may oscillate with the frequency of the underlying sound field. In *strong* sound fields the amplitude of the oscillations can become large enough such that the bubble collapses to a tiny volume in a periodic cycle. Each time a bubble collapses due to the compression very high temperatures and pressures can be observed in the interior of the bubble. During the collapse a shock wave and also a light flash can be emitted. The latter phenomenon is called *sonoluminescence* and was first discovered by H. Frenzel and H. Schultes [45] in 1934.

The upper sequence of pictures in Figure 1.1 shows a collapsing bubble in a physical experiment with a strong sound field. The maximal radius of the cavitation bubble (1st picture) is 55 micrometers. The lower part of the figure shows the sent out shock wave which propagates about 800 micrometers in 0.38 microseconds.

The process of the bubble collapse and the emission of shock waves and light flashes

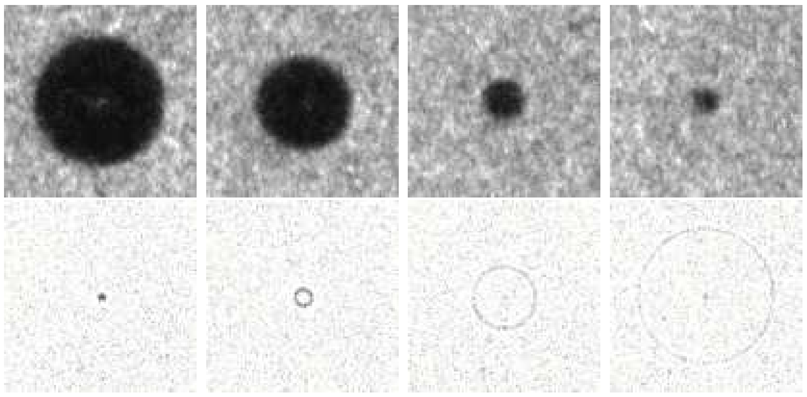

Figure 1.1: Collapsing bubble (upper sequence of pictures) and sent out shock wave (lower sequence). The pictures are taken from [48].

are far from being completely understood. Research and investigation of single bubbles and bubble ensembles are of high interest because of the following reasons:

- Industrial Interest. Turbines, pumps, ships propellers and nozzles get damaged by the resulting shock waves and suffer a loss of efficiency when the effect of cavitation occurs.
- Medical Interest. The destructive behavior of cavitation can also be of a beneficial use. For example kidney stones can be destroyed by application of focussed ultrasound which causes cavitation, see [61].
- Chemical Interest. The chemical effects of ultrasound are a result of cavitation and are investigated in the field of *sonochemistry*.

Most of the above information, including the pictures of the collapsing bubble, are taken from R. Geislers homepage [48], see also [47].

At the time of this writing an intensive work on modelling (on the micro and macro scale), numerical simulation and validation of the above mentioned processes is being carried out within the projects of the DFG-CNRS research group *Micro-Macro Modelling and Simulation of Liquid-Vapor Flows*. This work is also supported by this research group.

The presence of shock waves in the physical experiments indicate that compressibility of the fluid may have an important influence on the cavitation process. Thus, the underlying mathematical model should take the effect of compressibility into account. The main difference between the existing microscopic models for phase transition phenomena consists in the representation of the interface between the liquid and vapor phases. The first group of models use a sharp resolution of the interface. This means the interface has no spatial dilatation and the thermodynamic quantities are in general discontinuous over the interface. This is the class of *sharp interface* models. On the other hand we have the class of *diffuse interface* models. Here the interface has a small positive size and the thermodynamic quantities vary rapidly but smoothly within this interfacial region between vapor and liquid states.

The model that we consider belongs to the class of diffuse interface models and is an extension of the compressible Navier-Stokes equations that goes back to Korteweg [72] (1901). Here, for simplicity, we state this Navier-Stokes-Korteweg model (abbreviated as NSK model) only in the isothermal case

$$\begin{aligned} \rho_t + \nabla \cdot (\rho \boldsymbol{u}) &= 0, \\ (\rho \boldsymbol{u})_t + \nabla \cdot \left(\rho \boldsymbol{u}\boldsymbol{u}^T\right) + \nabla p(\rho) &= \nabla \cdot (\boldsymbol{\tau} + \boldsymbol{K}), \end{aligned} \tag{1.1}$$

with suitable initial and boundary conditions. In the equation above ρ denotes the density and $\boldsymbol{u}$ the velocity of the fluid. For the pressure p an appropriate equation of state must be chosen that has the capability of describing the pressure in the vapor as well as in the liquid phase. The simplest equation of state that can accomplish this is the *van der Waals* equation of state. Note that there is no additional order parameter in this model that distinguishes between the phases. In this model the density itself is the order parameter. Low density states characterize the vapor phase and high values of the density the liquid phase with an unphysical set of density states in between. In the equation above $\boldsymbol{\tau}$ denotes the usual viscous part of the stress tensor and the difference to the classical Navier-Stokes equations is the contribution of the Korteweg part to the stress tensor which is given by

$$\boldsymbol{K} = \lambda \left[\left(\rho \Delta \rho + \frac{1}{2} |\nabla \rho|^2 \right) \boldsymbol{I} - \nabla \rho \nabla \rho^T \right].$$

This contribution is responsible for the finite, nonzero size of the interface and acts as a penalty term for phase transitions. Thus, the interface is minimized in some sense when the flow approaches an equilibrium state with vanishing velocity. The idea of using density gradients to penalize phase transition goes back to van der Waals [115] (1894).

The diffuse Navier-Stokes-Korteweg model has several advantages over existing sharp interface models. For instance, sharp interface models need an additional jump condition because of the discontinuity over the interface (kinetic relation), see for example [85]. This kind of jump condition is not necessary for diffuse interface models because there is no jump across the interface. The NSK model implicitly includes the physical effect of surface tension, sharp interface models need an extra contribution to the stress tensor to include this effect. Topological changes in the solution are possible without special treatment and it is not necessary to track the interface by a Level Set or Volume of Fluid method as for sharp interface methods, see [29]. But there are still some disadvantages. Due to the resolution of a small diffuse interface and the presence of the higher order derivatives the time step in fully discrete numerical schemes must be chosen extremely small to guarantee the stability of the method. Moreover, most standard schemes cannot be applied because of the presence of the unphysical (elliptic) region in the state space.

Figure 1.2 shows a sketch of the basic physical experiment for the numerical simulations considered in this work. It shows a container filled with liquid and a few vapor bubbles in the surrounding liquid. At the (solid) container wall a fixed constant temperature is imposed and depending on the experiment the container wall may or may not move. In cases where the container wall moves, the boundary condition $\boldsymbol{u} = \boldsymbol{0}$ has to be replaced by $\boldsymbol{u} = \boldsymbol{u}_w$, where $\boldsymbol{u}_w$ denotes the prescribed velocity of the moving wall. We are

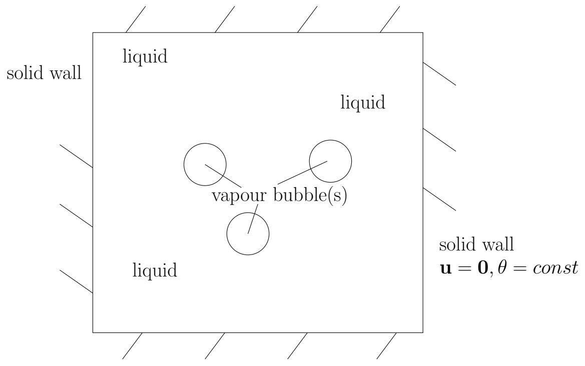

Figure 1.2: Sketch of the underlying physical experiment.

interested in the dynamics and time evolution of the configuration starting with this data.

The main goal of this work is the development and implementation of a software package for the discretization in multiple space dimensions of general evolution equations including conservative terms, nonconservative terms, sources and higher order derivatives. The resulting method should be based on modern numerical techniques such as adaptively refined meshes, load balancing, parallelization, higher order space and time discretization. The Navier-Stokes-Korteweg model should be discretized using this package. Local adaptivity and MPI based parallelization are absolutely necessary for the resolution of the interface and the processing of the high numerical cost even in two space dimensions. The resulting C++ software package should have an easy to understand modular design such that it can easily be applied to similar equations. A basic description of the package can be found in the appendix.

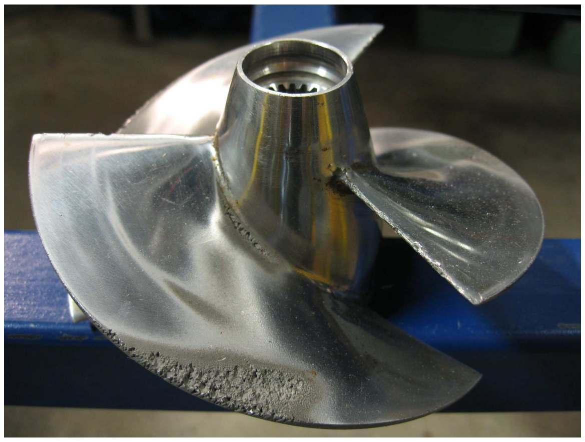

Figure 1.3: Blades of a ship propeller damaged by cavitation bubbles. This picture is published under the ShareAlike License v. 2.5.

1.2 Results and new Contributions

This section summarizes the main results and (at the time of this writing) new contributions of this thesis.

The main focus of this work is the reliable discretization of the isothermal version of the Navier-Stokes-Korteweg system and the construction of (quasi-)exact solutions that serve as benchmarks. A discretization of the full temperature dependent model has also been developed but not tested for reliability as much as it has been done for the isothermal version.

- The existence of traveling wave solution is only proven for a modified system. The first step of this proof is adapted to the original system (1.1) in Section 3.2.2. The second step can possibly also be adapted to the original system but it is technically and lengthy and does not fit properly in this work.
- Computation of static equilibrium configurations: For rotational symmetric solutions it is clear that the NSK system reduces to an ordinary differential equation. The crucial part is the appropriate choice of boundary condition in order to compute this kind of solutions successfully. This is done in Section 4.1. Using these kinds of solutions, the physical parameters such as surface tension can be identified. This verifies the formula for surface tension, given in [75], numerically.
- Computation of traveling wave solutions: The method is based on the approach given in [43] but not straightforward to generalize to compute traveling wave solutions of the NSK equations. We give this generalization of the method as well as numerical results in Section 4.2.
- In Section 5.2 we construct a new well balanced first order scheme for the discretization of the isothermal version of the Navier-Stokes-Korteweg system in multiple space dimensions. Numerical results indicate that this is a reliable discretization of the system.
- Also in Section 5.3 we demonstrate that the relaxation scheme given in [29], [30] does not produce the correct results (except for static equilibrium configurations).
- In Section 6.2.3 we give a new higher order discretization for nonconservative equations based on the Discontinuous Galerkin approach and the definition of nonconservative products given in [36]. This kind of nonconservative discretization is not limited to the NSK system. It is very well suited for the discretization of general nonconservative equations, for example equations arising from a homogenization process are usually nonconservative.
- We prove a cell entropy inequality and a resulting L^2 stability estimate for a semi-discrete Local Discontinuous Galerkin discretization of a model problem, similar to the result given in [130].
- We give the complete higher order well balanced discretization for the NSK system based on the Discontinuous Galerkin approach for conservative, nonconservative and higher order terms in Section 6.9. The numerical results are summarized in the next section.

- The main result of this work is the developed C++ software package for the discretization of general time dependent (convection dominated) partial differential equations. The package provides
  - local adaptive (stable) grid refinement of one, two and three dimensional simplicial meshes.
  - load balancing (ParMETIS based) in a parallel MPI based environment.
  - higher order Local Discontinuous Galerkin discretization including nonconservative discretization.
  - higher order time discretization based on explicit, implicit and semi-implicit Runge-Kutta methods as well as explicit and implicit Extrapolation methods.

  The discretization of the Navier-Stokes-Korteweg system is done by application of this package.

1.3 Outline of this Thesis

In Chapter 2 we provide the thermodynamic background and discuss the Navier-Stokes-Korteweg model together with appropriate boundary conditions in detail for the isothermal as well as for the temperature dependent case. We provide the dimensionless form of the complete model as well as the quantitative relations to the corresponding physical quantities. The physical effect of surface tension that is implicitly included in the model, in contrast to sharp interface models where surface tension is usually included by means of an additional *boundary condition* at the interface, is also discussed in this chapter.

In Chapter 3 we summarize some of the known theoretical results concerning the Navier-Stokes-Korteweg equations. These include the results about special kinds of solutions, such as static equilibrium and traveling wave solutions, as well as the existence of general local or global in time solutions of the corresponding Cauchy-Problem.

The system has a very complicated structure such that the construction of analytical solutions seems to be out of scope. However, for the validation of numerical schemes it is important to have exact solutions available. Some of the special solutions discussed in Chapter 3 satisfy ordinary differential equations. These kinds of solutions can be computed via reliable numerical methods very accurately. This is the main purpose of Chapter 4.

Chapter 5 is dedicated to the construction of basic first order schemes. We present three different schemes:

- a scheme in conservative form that produces the correct solutions in the tested cases,

- a well balanced scheme in non-conservative form that does a much better job than the first scheme,
- and a relaxation scheme that turned out to give the correct solution only in special cases. Therefore this scheme is of very limited use.

The second scheme, the non-conservative well balanced scheme in is then generalized to higher order schemes using the Local Discontinuous Galerkin approach in Chapter 6. We present the method in a general framework of time dependent partial differential equations including conservative terms, higher order derivatives, source terms and nonconservative products. We discuss the Local Discontinuous Galerkin discretization of simple examples such as the one dimensional scalar convection-diffusion equation and a scalar model equation for the Navier-Stokes-Korteweg system. Finally we give the complete discretization for the isothermal NSK system in one, two and three space dimensions and the extension to the temperature dependent model in two space dimensions (the extension to 3d is straightforward).

The higher order time discretization via explicit, implicit and semi-implicit Runge-Kutta methods is discussed in Chapter 7.

In order to construct efficient numerical schemes modern numerical techniques such as local mesh adaption, parallelization and load balancing are extremely important. These techniques are discussed in Chapter 8. Without these techniques it is not possible to resolve diffuse interfaces completely and solve the equation in appropriate time due to the high computational cost.

In Chapter 9 we present the numerical results using the higher order well balanced schemes. Here we summarize the results as follows:

- The approximate solutions converge to the exact solutions in the test cases where a (quasi-)exact solution is known.
- The expected order of the numerical schemes is reached in practical applications. This is observed using the test cases constructed in Chapter 4. Improving the order of the schemes really leads to more efficient schemes.
- Local mesh adaption is necessary for the resolution of the diffuse interfaces and leads to more efficient schemes. Simple heuristic indicators (based on density gradients) are sufficient to track the interfaces.
- Implicit time stepping avoids a complicated time step restriction control for the NSK system and leads to more efficient schemes.
- Parallelization of the code is necessary because of the high computational cost and high memory consumption, even in two space dimensions. It leads to more efficiency in the sense that the computation runs faster when more machines are available.

- Solutions of the Navier-Stokes-Korteweg model seem to have quantitatively the correct physical behavior. However, physical experiments on the scale of the numerical experiments are not known and therefore existing physical data is not directly comparable to the data produced by the numerical simulations.

A description of the software package (including example implementations), physical data of some fluids, notational conventions and some definitions that did not fit in the above mentioned chapters can be found in the appendix.

Chapter 2

Derivation of the Model

The aim of this chapter is to derive a system of partial differential equations with appropriate equations of state for the simulation of a liquid-vapor flow including the effect of phase transition. However, it is not really a derivation of a mathematical model, it is a derivation of sufficient conditions for a model to be thermodynamically consistent under the assumption that the Helmholtz free energy does not only depend on the state of the fluid (as in the classical case) but also on its environment, modeled by the gradient of the density.

In order to close the system we choose a van der Waals equation of state because it is one of the simplest equations of state that is capable to describe liquid and vapor phases and it is in quite good agreement with many fluids when the temperature of the fluid is close to its *critical temperature*.

The resulting governing equations, the Navier-Stokes-Korteweg system, belongs to the class of diffuse interface models and can be seen as a Cahn-Hilliard type model for the equations of gas dynamics. The model contains some nonclassical contributions of terms that guarantee that smooth solutions satisfy the second law of thermodynamics, see Theorem 2.2.2. There is no additional order parameter in the Navier-Stokes-Korteweg equations that distinguishes between the liquid and vapor phases as in other diffuse interface models, see for example [15]. Liquid and vapor phases are determined by the value of the density only. An overview of the theory of diffuse interfaces and the Navier-Stokes-Korteweg system can be found in [1], see also [41].

For the numerical treatment of the system of partial differential equations it is useful to have the thermodynamic and kinematic quantities in dimensionless form available. We provide dimensionless quantities in terms of critical values since the chosen equation of state is a good approximation to realistic values near the critical point of the fluid. The relation of all dimensionless values given throughout this chapter to the corresponding physical quantities is summarized in Section B.1. In Section B.2 the necessary physical values are provided for three different fluids. These measured values are taken from the NIST database [125].

2.1 Thermodynamic Relations

Thermodynamics is a funny subject. The first time you go through it, you don't understand it at all. The second time you go through it, you think you understand it, except for one or two small points. The third time you go through it, you know you don't understand it, but by that time you are so used to it, it doesn't bother you any more.

Arnold Sommerfeld

In the first section of this chapter we provide the necessary thermodynamic background. Most of the information given below can be found in standard textbooks such as [87] and [78]. Based on a Helmholtz free energy function of a fluid we can define all thermodynamic quantities we need in this work in terms of this free energy and its derivatives. We provide the equation of state we use for numerical simulations. This is the so called van der Waals equation of state in the most general form. With the van der Waals equation of state the coexistence of liquid and vapor phases in the fluid are possible. This equation of state is in good agreement with many fluids when the temperature of the fluid is close to the critical temperature, see Section B.2. Therefore it is appropriate to express this equation of state in dimensionless form in terms of the critical values: critical temperature, critical density and critical pressure. Later in this section we provide a dimensionless van der Waals equation of state with all unnecessary parameters scaled out. This results in a general equation with only one parameter (heat capacity at constant volume) left that has to be determined for different fluids. In Section B.2 we provide the missing data for different fluids.

Given a Helmholtz free energy function $f = f(\theta, \rho)$ that may depend on the temperature θ and the density ρ of the fluid, all other important (with respect to this work) thermodynamic quantities, namely the internal energy e, the entropy s, the pressure p and the chemical potential μ, can be expressed in terms of θ, ρ, f and derivatives of f. In general all thermodynamic quantities are functions of θ and ρ.

Definition 2.1.1 (Classical Thermodynamic Relations)
Given a Helmholtz free energy $f(\theta, \rho)$ the thermodynamic quantities are defined by the relations

$$e(\theta,\rho) = f(\theta,\rho) - \theta f_\theta(\theta,\rho), \qquad \textit{internal energy}, \tag{2.1}$$

$$s(\theta,\rho) = -f_\theta(\theta,\rho), \qquad \textit{specific entropy}, \tag{2.2}$$

$$p(\theta,\rho) = \rho^2 f_\rho(\theta,\rho), \qquad \textit{pressure}, \tag{2.3}$$

$$\mu(\theta,\rho) = (\rho f(\theta,\rho))_\rho, \qquad \textit{chemical potential.} \tag{2.4}$$

Note: In a single component fluid (fluids of the type considered in this work) the chemical potential is the same as the so called Gibbs free energy. They are not the same in multi component fluids (not considered here).

The simplest (and one of the most important) example of a compressible fluid is that of a *perfect gas*. The free energy of a perfect gas and the resulting thermodynamic quantities are given below.

Example 2.1.2 (Perfect Gas)
The Helmholtz free energy for a perfect gas and the resulting thermodynamic quantities (according to (2.1) - (2.3)) are given by

$$\begin{aligned} f(\theta,\rho) &= R\theta\log\left(\frac{\rho}{\rho_0}\right) - c\theta\log\left(\frac{\theta}{\theta_0}\right) + c\theta + cst. \\ e(\theta,\rho) &= c\theta + cst, \\ s(\theta,\rho) &= -R\log\left(\frac{\rho}{\rho_0}\right) + c\log\left(\frac{\theta}{\theta_0}\right), \\ p(\theta,\rho) &= R\rho\theta. \end{aligned}$$

$\rho_0, \theta_0 > 0$ *are reference values for the density and temperature respectively and* R, c, cst *are real constants with* $R, c > 0$.

Another important example of a compressible fluid is the *van der Waals* fluid. The advantage of a van der Waals equation of state is its capability to describe liquid-vapor phase transitions below a *critical temperature*. The free energy of a van der Waals fluid and the remaining quantities are given in Example 2.1.3.

Example 2.1.3 (van der Waals Fluid)
The Helmholtz free energy for a van der Waals fluid and the resulting thermodynamic quantities (according to (2.1) - (2.4)) are given by

$$\begin{aligned} f(\theta,\rho) &= -a\rho + k\theta\log\left(\frac{\rho}{b-\rho}\right) - c\theta\log\left(\frac{\theta}{\theta_0}\right) - d\theta + cst. \qquad (2.5) \\ e(\theta,\rho) &= -a\rho + c\theta + cst, \\ s(\theta,\rho) &= -k\log\left(\frac{\rho}{b-\rho}\right) + c\log\left(\frac{\theta}{\theta_0}\right) + c + d, \\ p(\theta,\rho) &= kb\frac{\rho\theta}{b-\rho} - a\rho^2, \\ \mu(\theta,\rho) &= k\theta\left(\frac{b}{b-\rho} + log\left(\frac{\rho}{b-\rho}\right)\right) - 2a\rho. \end{aligned}$$

Here a, b, c, d, k, cst *are real constants with* $a, b, c, k > 0$ *and* $\theta_0 > 0$ *is a reference temperature. The above quantities are defined for states* $(\theta,\rho) \in (0,\infty) \times (0,b)$ *but the state space is partially meaningless from the physical point of view. For example the pressure can become negative in parts of the state space.*

In the following we will always consider a van der Waals fluid. The free energy of a van der Waals equation of state in this general form can be found in [7], see also [78]. The constant c in the equation of state is known as the heat capacity at constant volume. Some of the constants can be omitted for our purposes because they will drop out of the equations we are interested in, but they might be important when effects like chemical reactions are taken into account.

The *critical temperature* (the smallest temperature for that the fluid can consist of only one *phase*) of a van der Waals fluid is defined (using the coefficients from Example 2.1.3) by

$$\theta_{crit} = \frac{8ab}{27k}.$$

Figure 2.1 shows the pressure p and the chemical potential μ as a function of the density ρ for a constant fixed temperature θ below, at, and above the critical temperature. The critical temperature is the smallest temperature such that the graphs of p and μ are monotonically increasing.

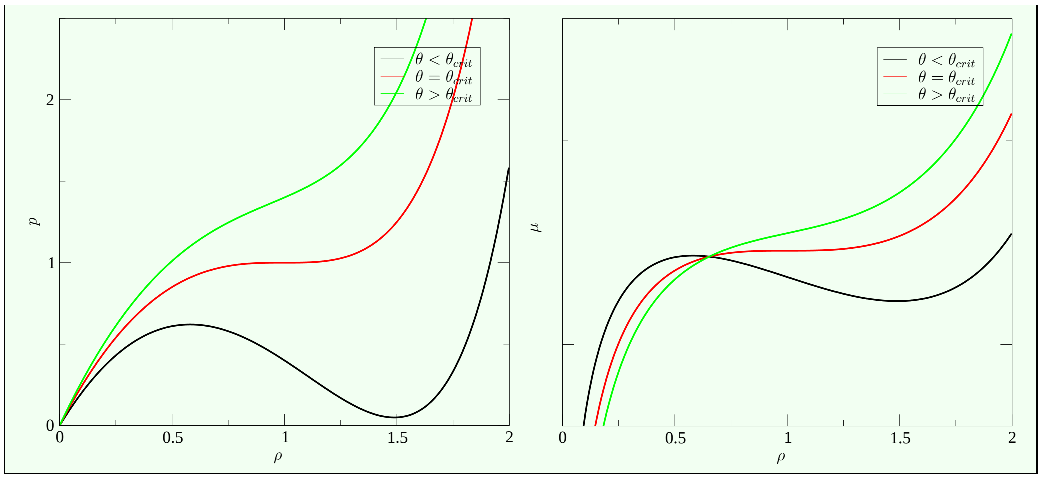

Figure 2.1: Graphs of pressure (left) and chemical potential (right) for temperatures below, at, above the critical temperature.

Below the critical temperature the graph of the pressure and the graph of the chemical potential consist of two monotone increasing branches separated by a monotone decreasing branch, the so called *elliptic region*. This behavior makes it possible to describe vapor and liquid phases in a van der Waals fluid. The first monotone increasing branch of p and μ defines the vapor phase, the second one the liquid phase. These two branches are connected smoothly by the elliptic region, which is a set of unphysical states. Above the critical temperature only one phase exists, in this case the fluid is called *supercritical*. A graph of the pressure with a comparison to measured real world data for different fluids can be found in section B.2, see Figure B.5.

Associated with the critical temperature the critical density ρ_{crit} and critical pressure p_{crit} are defined by the inflection point of the p-graph at the critical temperature. Using the coefficients from above these quantities are given by

$$\begin{aligned}\rho_{crit} &= \frac{b}{3},\\ p_{crit} &= \frac{1}{27}ab^2.\end{aligned}$$

Using the critical values θ_{crit}, ρ_{crit} and p_{crit} we can introduce a dimensionless equation of state for the pressure

$$\tilde{p}(\tilde{\theta},\tilde{\rho}) = \frac{1}{p_{crit}}p(\theta_{crit}\,\tilde{\theta},\rho_{crit}\,\tilde{\rho}) = \frac{8\tilde{\theta}\tilde{\rho}}{3-\tilde{\rho}} - 3\tilde{\rho}^2 \tag{2.6}$$

that does not depend on coefficients k, a and b anymore. Additionally we introduce a dimensionless equation of state for the internal energy. We choose a reference internal energy

$$e_{ref} = \frac{p_{crit}}{\rho_{crit}} \tag{2.7}$$

that is in general not the internal energy at the critical values θ_{crit} and ρ_{crit}. The dimensionless internal energy is defined by

$$\begin{aligned}\tilde{e}(\tilde{\theta},\tilde{\rho}) &= \frac{1}{e_{ref}}e(\theta_{crit}\,\tilde{\theta},\rho_{crit}\,\tilde{\rho})\\ &= \frac{1}{e_{ref}}\left(c\theta_{crit}\,\tilde{\theta} - a\rho_{crit}\,\tilde{\rho}\right)\\ &= \frac{\rho_{crit}}{p_{crit}}\left(c\theta_{crit}\,\tilde{\theta} - 3\frac{p_{crit}}{\rho_{crit}}\tilde{\rho}\right)\\ &= \tilde{c}\tilde{\theta} - 3\tilde{\rho} \qquad (2.8)\end{aligned}$$

with a dimensionless parameter

$$\tilde{c} = \frac{\theta_{crit}\,\rho_{crit}}{p_{crit}}c. \tag{2.9}$$

Further we define the dimensionless free energy $\tilde{f}$, entropy $\tilde{s}$ and chemical potential $\tilde{\mu}$ by

$$\begin{aligned}\tilde{f}(\tilde{\theta},\tilde{\rho}) &= \frac{1}{e_{ref}}f(\rho_{crit}\,\tilde{\rho},\theta_{crit}\,\tilde{\theta}),\\ \tilde{s}(\tilde{\theta},\tilde{\rho}) &= \frac{\theta_{crit}}{e_{ref}}s(\rho_{crit}\,\tilde{\rho},\theta_{crit}\,\tilde{\theta}),\\ \tilde{\mu}(\tilde{\theta},\tilde{\rho}) &= \frac{1}{e_{ref}}\mu(\rho_{crit}\,\tilde{\rho},\theta_{crit}\,\tilde{\theta}).\end{aligned}$$

Some of the constants in the free energy (2.5) of a van der Waals fluid are not important as long as we neglect chemical reactions, i.e., these constants will drop out of all equations we consider. Thus, we can choose $\theta_0 = \theta_{crit}$, $d = -c$, $cst = 0$. We summarize the above results and define the equations of state for a dimensionless van der Waals fluid. For simplicity we omit the tilde symbols that characterized the dimensionless quantities.

Example 2.1.4 (Dimensionless van der Waals Fluid)

$$\begin{aligned} f(\theta,\rho) &= -3\rho + \frac{8}{3}\theta \log\left(\frac{\rho}{3-\rho}\right) + c\theta(1-\log(\theta)), && (2.10)\\ e(\theta,\rho) &= -3\rho + c\theta, && (2.11)\\ s(\theta,\rho) &= -\frac{8}{3}\log\left(\frac{\rho}{3-\rho}\right) + c\log(\theta), && (2.12)\\ p(\theta,\rho) &= \frac{8\theta\rho}{3-\rho} - 3\rho^2, && (2.13)\\ \mu(\theta,\rho) &= \frac{8}{3}\theta\left(\frac{3}{3-\rho} + \log\frac{\rho}{3-\rho}\right) - 6\rho, && (2.14) \end{aligned}$$

where the dimensionless heat capacity at constant volume c is related to the physical quantity by equation (2.9).

Note: The dimensionless quantities can be obtained from the dimensionless free energy by the relations (2.1) - (2.4).

Definition 2.1.5 (Liquid and Vapor Phases in a van der Waals Fluid)
For a fixed temperature $\theta < \theta_{crit}$ let $\overline{\rho}_v \in (0,b)$ denote the state where p and μ have their local maximum, and $\underline{\rho}_l \in (0,b)$ the state of their local minimum. Then the phases of a van der Waals fluid are defined by

$$\begin{aligned} (0,\overline{\rho}_v) &: \textit{vapor phase},\\ (\overline{\rho}_v,\underline{\rho}_l) &: \textit{elliptic or spinodal region},\\ (\underline{\rho}_l,b) &: \textit{liquid phase}. \end{aligned}$$

Here b is equal to 3 in the dimensionless case.

Definition 2.1.6 (Maxwell States in a van der Waals Fluid)
Let $\theta < \theta_{crit}$ be a fixed temperature. Then the Maxwell states $\rho_v^M \in (0,\overline{\rho}_v)$ and $\rho_l^M \in (\underline{\rho}_l,b)$ are uniquely defined by the relations

$$\begin{aligned} p(\theta,\rho_v^M) &= p(\theta,\rho_l^M), && (2.15)\\ \mu(\theta,\rho_v^M) &= \mu(\theta,\rho_l^M). && (2.16) \end{aligned}$$

For equivalent definitions of the Maxwell states see section A.3.

Figure 2.2 shows the phases and Maxwell states of a van der Waals fluid below the critical temperature. The Maxwell values can be seen as equilibrium values at constant temperature, see Section 2.8. The set $\{\rho_v^M(\theta) \mid \theta \in (0,\theta_{crit})\} \cup \{\rho_l^M(\theta) \mid \theta \in (0,\theta_{crit})\}$ is also called saturation curve, see the phase diagram 2.3.

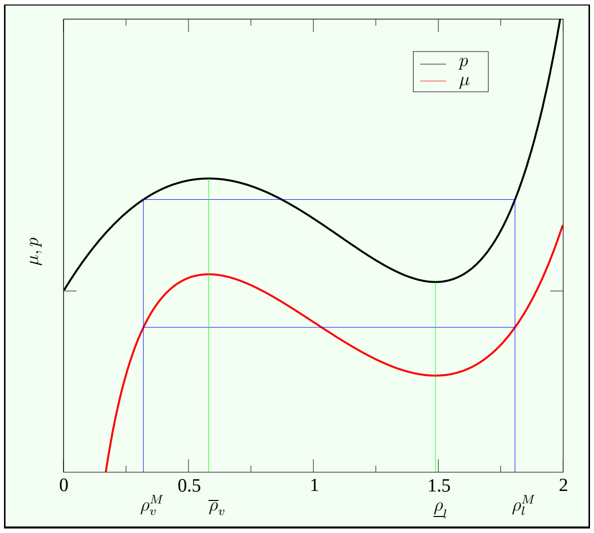

Figure 2.2: Graphs of pressure, chemical potential for a temperature below the critical temperature, Maxwell states and boundary of the elliptic region.

The Maxwell states $\rho_v^M(\theta)$ and $\rho_l^M(\theta)$ of a dimensionless van der Waals fluid can be approximated by the formulas

$$\begin{aligned} \rho_v^M(\theta) &\approx 1.0 - \sqrt{\theta}\,(2.0 - 1.5(1.0 - \theta)), \\ \rho_l^M(\theta) &\approx 1.0 + \sqrt{\theta}\,(2.0 + 0.5(1.0 - \theta)). \end{aligned}$$

Note: These formulas are obtained by curve fitting and can be used as a starting guess for a Newton iteration to compute the exact Maxwell states. The above formulas give quite accurate results in the dimensionless temperature range $\theta \in [0.6, 1.0]$.

Antanovskii [2] gives a generalization of the free energy and the thermodynamic quantities for the case when the free energy is not only a function of the states θ and ρ but also depends on the norm of the density gradient $\alpha = \frac{1}{2}|\nabla\rho|^2$. This dependence on the density gradient models a dependence on the environment of the material and allows a liquid-vapor interface to be of finite, nonzero thickness. For his definition of the free energy Antanovskii uses the fact that a fluid at static equilibrium maximizes its entropy which is assumed to depend on the density gradient. The idea of using gradients of the density to model diffuse interfaces goes back to van der Waals [115] who gave a theory based on thermodynamical principles.

Definition 2.1.7 (Extended Thermodynamic Relations)
Let an extended free energy $f = f(\theta, \rho, \alpha)$ be given. Then the extended internal energy and entropy are defined by the relations

$$e(\theta, \rho, \alpha) = f(\theta, \rho, \alpha) - \theta f_\theta(\theta, \rho, \alpha), \qquad (2.17)$$

$$s(\theta, \rho, \alpha) = -f_\theta(\theta, \rho, \alpha). \qquad (2.18)$$

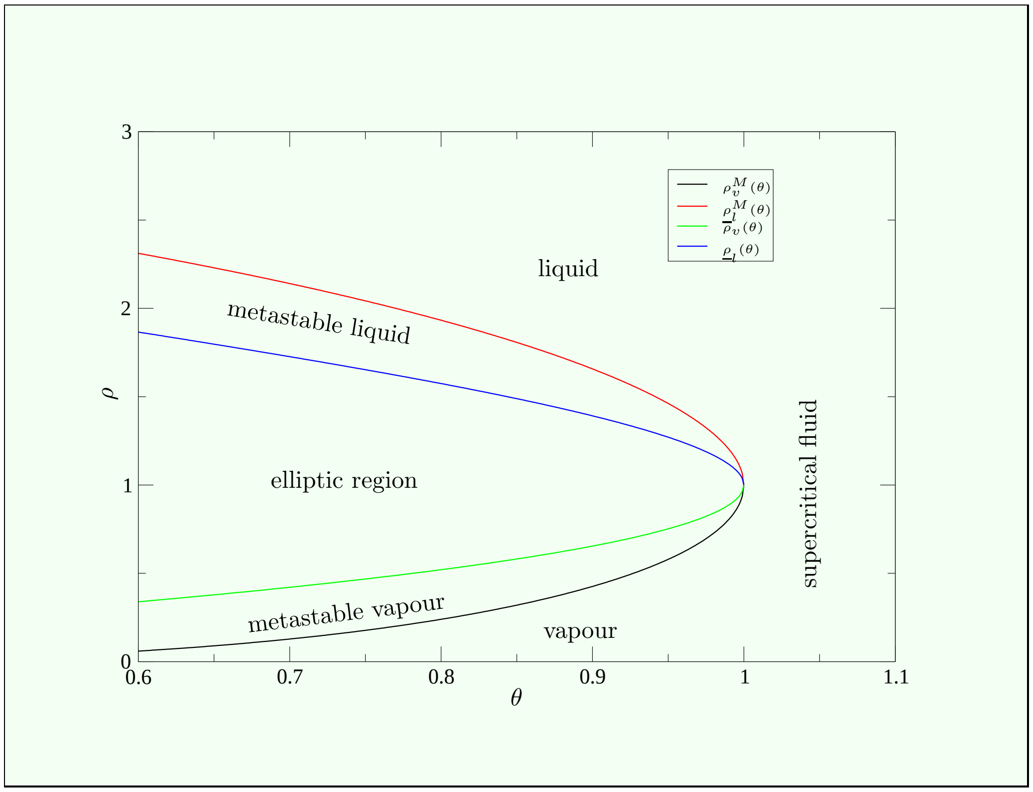

Figure 2.3: Phase diagram of the dimensionless van der Waals fluid.

For a function $\varphi = \varphi(\theta, \rho, \alpha)$ where α stands for $\frac{1}{2}|\nabla\rho|^2$ we use in the following the notation

$$[\varphi]_\rho = \varphi_\rho - \nabla \cdot (\varphi_\alpha \nabla\rho) \tag{2.19}$$

for the variational derivative as it is used in standard textbooks such as [32].

Antanovskii [2] gives also a definition of an extended pressure and an extended chemical potential. We do not use these quantities explicitly but for the sake of completeness we list these definitions at this point:

$$p = \rho^2[f]_\rho, \quad \mu = [\rho f]_\rho,$$

where we have used the definition of the variational derivative given in (2.19).

2.2 Equations of Motion

This section is dedicated to the description of the motion of a fluid in some domain $\Omega \in \mathbb{R}^3$ as a continuous medium. The motion of the fluid is governed by the fundamental physical laws of conservation of mass, conservation of momentum (Newtons second law), conservation of energy (first law of thermodynamics) and entropy production (second law of thermodynamics). Smooth solutions of the resulting governing equations will satisfy all of the above mentioned physical principles, see Theorem 2.2.2.

Based on the Reynolds transport theorem, see [42], we describe the evolution in time of a fluid in a domain $\Omega \in \mathbb{R}^3$. In the following we assume that all appearing functions are sufficiently smooth.

Let denote $\omega(t) \subset \Omega$ an arbitrary control volume that evolves in time. Then the density, momentum and total energy of the fluid have to satisfy to following balance equations and additionally the entropy production equation:

Conservation of mass

$$\frac{d}{dt} \int_{\omega(t)} \rho \, d\boldsymbol{x} = 0, \tag{2.20}$$

the momentum balance equation (Newtons second law)

$$\frac{d}{dt} \int_{\omega(t)} \rho \boldsymbol{u} \, d\boldsymbol{x} = \int_{\partial\omega(t)} \boldsymbol{P}\boldsymbol{n} \, d\sigma, \tag{2.21}$$

the energy balance equation (first law of thermodynamics)

$$\frac{d}{dt} \int_{\omega(t)} \rho \left(e + \frac{1}{2}|\boldsymbol{u}|^2 \right) \, d\boldsymbol{x} = \int_{\partial\omega(t)} \boldsymbol{P}\boldsymbol{u} \cdot \boldsymbol{n} - \boldsymbol{q}_E \cdot \boldsymbol{n} \, d\sigma, \tag{2.22}$$

and additionally the entropy production equation (second law of thermodynamics)

$$\frac{d}{dt} \int_{\omega(t)} \rho s \, d\boldsymbol{x} = \int_{\omega(t)} s_{prod} \, d\boldsymbol{x} - \int_{\partial\omega(t)} \boldsymbol{q}_S \cdot \boldsymbol{n} \, d\sigma. \tag{2.23}$$

In the above relations $\rho = \rho(\boldsymbol{x}, t) > 0$ denotes the density of the fluid, $\boldsymbol{u} = \boldsymbol{u}(\boldsymbol{x}, t) \in \mathbb{R}^3$ the velocity, $\theta = \theta(\boldsymbol{x}, t) > 0$ the temperature of the fluid. Further, $e = e(\theta, \rho, \alpha)$ is the generalized specific internal energy and $s = s(\theta, \rho, \alpha)$ is the generalized specific entropy of the fluid. Here and in the following α stands always for $\frac{1}{2}|\nabla\rho|^2$. The generalized internal energy and entropy are related to the generalized free energy by the relations (2.17), (2.18). $\boldsymbol{P} \in \mathbb{R}^{3\times 3}$ denotes a general symmetric stress tensor, $\boldsymbol{q}_E \in \mathbb{R}^3$ a general heat flux, $\boldsymbol{q}_S \in \mathbb{R}^3$ a general entropy flux and $s_{prod} > 0$ a general entropy production. They will depend on the variables ρ, $\boldsymbol{u}$, θ and on derivatives (possibly higher order derivatives) of these variables.

Note: The symmetry of the general stress tensor $\boldsymbol{P}$ implies the conservation of angular momentum, see for example [42].

For thermodynamic consistency it is important (otherwise the behavior of the fluid would be unphysical) that the entropy production s_{prod} is nonnegative. Using the thermodynamic relations (2.17) and (2.18) we derive sufficient conditions on the stress tensor $\boldsymbol{P}$ and the heat flux $\boldsymbol{q}_E$ that ensure the entropy production to be nonnegative.

Using Reynolds transport theorem, the Gauss theorem and the fact that $\omega(t)$ can be chosen arbitrarily (for details see [42]) we derive from the integral equations (2.20) - (2.23) the equations of motion in differential form

$$\begin{aligned}
\frac{D}{Dt}\rho &= -\rho\nabla\cdot\boldsymbol{u}, && (2.24)\\
\rho\frac{D}{Dt}\boldsymbol{u} &= \nabla\cdot\boldsymbol{P}, && (2.25)\\
\rho\frac{D}{Dt}e &= -\nabla\cdot\boldsymbol{q}_E + \boldsymbol{P}:\nabla\boldsymbol{u}, && (2.26)\\
\rho\frac{D}{Dt}s &= -\nabla\cdot\boldsymbol{q}_S + s_{prod}. && (2.27)
\end{aligned}$$

where $\frac{D}{Dt} = \partial_t + \boldsymbol{u}\cdot\nabla$ denotes the material derivative.

using the continuity equation (2.24) and the *chain rule*

$$\frac{D}{Dt}e = \left(e_\theta\frac{D}{Dt}\theta + e_\rho\frac{D}{Dt}\rho + e_\alpha\nabla\rho\cdot(-\nabla(\rho\nabla\cdot\boldsymbol{u}) - \nabla\boldsymbol{u}^T\nabla\rho)\right) \qquad (2.28)$$

we derive the relation

$$\begin{aligned}
\boldsymbol{P}:\nabla\boldsymbol{u} - \nabla\cdot\boldsymbol{q}_E &= \rho\frac{D}{Dt}e\\
&= \rho\left(e_\theta\frac{D}{Dt}\theta + e_\rho\frac{D}{Dt}\rho + e_\alpha\nabla\rho\cdot(-\nabla(\rho\nabla\cdot\boldsymbol{u}) - \nabla\boldsymbol{u}^T\nabla\rho)\right)\\
&= \rho e_\theta\frac{D}{Dt}\theta - \nabla\cdot(\rho^2\nabla\cdot\boldsymbol{u}e_\alpha\nabla\rho)\\
&\quad - \left(\rho^2 e_\rho\boldsymbol{I} - \rho^2\nabla\cdot(e_\alpha\nabla\rho)\boldsymbol{I} - \rho e_\alpha|\nabla\rho|^2\boldsymbol{I} + \rho e_\alpha\nabla\rho\nabla\rho^T\right):\nabla\boldsymbol{u}.
\end{aligned}$$

In the above equations the colored terms from one equation to the next correspond to each other (by multiplying the colored terms with the remaining terms). Using the notation (2.19), this gives

$$\begin{aligned}
\rho e_\theta\frac{D}{Dt}\theta &= \left(\boldsymbol{P} + \rho^2[e]_\rho\boldsymbol{I} + \rho e_\alpha(\nabla\rho\nabla\rho^T - |\nabla\rho|^2\boldsymbol{I})\right):\nabla\boldsymbol{u}\\
&\quad -\nabla\cdot\left(\boldsymbol{q}_E - \rho^2\nabla\cdot\boldsymbol{u}e_\alpha\nabla\rho\right).
\end{aligned}$$

Now, using $e_\theta = \theta s_\theta$ and $f = e - \theta s$, see (2.17) and (2.18), we get

$$\begin{aligned}
\rho s_\theta \frac{D}{Dt}\theta &= \frac{1}{\theta}\left(\boldsymbol{P} + \rho^2[f]_\rho + \rho f_\alpha\left(\nabla\rho\nabla\rho^T - |\nabla\rho|^2\boldsymbol{I}\right)\right) : \nabla\boldsymbol{u} \\
&\quad + \frac{1}{\theta}\left(\rho^2[\theta s]_\rho + \rho\theta s_\alpha\left(\nabla\rho\nabla\rho^T - |\nabla\rho|^2\boldsymbol{I}\right)\right) : \nabla\boldsymbol{u} \\
&\quad - \frac{1}{\theta}\nabla\cdot\left(\boldsymbol{q}_E - \rho^2\nabla\cdot\boldsymbol{u}f_\alpha\nabla\rho\right) + \frac{1}{\theta}\nabla\cdot\left(\rho^2\nabla\cdot\boldsymbol{u}\theta s_\alpha\nabla\rho\right) \\
&= \frac{1}{\theta}\left(\boldsymbol{P} + \rho^2[f]_\rho + \rho f_\alpha\left(\nabla\rho\nabla\rho^T - |\nabla\rho|^2\boldsymbol{I}\right)\right) : \nabla\boldsymbol{u} \\
&\quad - \nabla\cdot\left(\frac{1}{\theta}(\boldsymbol{q}_E - \rho^2\nabla\cdot\boldsymbol{u}f_\alpha\nabla\rho)\right) - \frac{1}{\theta^2}(\boldsymbol{q}_E - \rho^2\nabla\cdot\boldsymbol{u}f_\alpha\nabla\rho)\cdot\nabla\theta \\
&\quad + \frac{1}{\theta}\left(\rho^2(\theta[s]_\rho - s_\alpha\nabla\rho\cdot\nabla\theta) + \rho\theta s_\alpha\left(\nabla\rho\nabla\rho^T - |\nabla\rho|^2\boldsymbol{I}\right)\right) : \nabla\boldsymbol{u} \\
&\quad + \frac{1}{\theta}\nabla\cdot\left(\rho^2\nabla\cdot\boldsymbol{u}\theta s_\alpha\nabla\rho\right).
\end{aligned}$$

A further manipulation shows that

$$\begin{aligned}
&-\rho s_\rho\frac{D}{Dt}\rho - \rho s_\alpha\nabla\rho\cdot(-\nabla(\rho\nabla\cdot\boldsymbol{u}) - \nabla\boldsymbol{u}^T\nabla\rho) \\
&\quad = \frac{1}{\theta}\left(\rho^2(\theta[s]_\rho - s_\alpha\nabla\rho\cdot\nabla\theta) + \rho\theta s_\alpha\left(\nabla\rho\nabla\rho^T - |\nabla\rho|^2\boldsymbol{I}\right)\right) : \nabla\boldsymbol{u} \\
&\qquad + \frac{1}{\theta}\nabla\cdot\left(\rho^2\nabla\cdot\boldsymbol{u}\theta s_\alpha\nabla\rho\right).
\end{aligned}$$

Using the chain rule (2.28) for s instead of e we finally arrive at

$$\begin{aligned}
\rho\frac{D}{Dt}s &= \frac{1}{\theta}\left(\boldsymbol{P} + \rho^2[f]_\rho + \rho f_\alpha\left(\nabla\rho\nabla\rho^T - |\nabla\rho|^2\boldsymbol{I}\right)\right) : \nabla\boldsymbol{u} \\
&\quad - \nabla\cdot\left(\frac{1}{\theta}\left(\boldsymbol{q}_E - \rho^2\nabla\cdot\boldsymbol{u}f_\alpha\nabla\rho\right)\right) \\
&\quad - \frac{1}{\theta^2}\left(\boldsymbol{q}_E - \rho^2\nabla\cdot\boldsymbol{u}f_\alpha\nabla\rho\right)\cdot\nabla\theta.
\end{aligned}$$

We must ensure that the entropy production is a nonnegative function. This gives rise to the definition of a *material of Korteweg type.*

Definition 2.2.1 (Korteweg type material)
We call a material (a fluid) to be of Korteweg type if the stress tensor, the heat flux and

the entropy flux are given by the relations

$$\begin{aligned}
\boldsymbol{P} &= -\rho^2[f]_\rho \boldsymbol{I} - \rho f_\alpha \left(\nabla\rho\nabla\rho^T - |\nabla\rho|^2\boldsymbol{I}\right) + \boldsymbol{\tau}, && (2.29)\\
\boldsymbol{q}_E &= \rho^2 \nabla\cdot\boldsymbol{u} f_\alpha \nabla\rho - \kappa\nabla\theta, && (2.30)\\
\boldsymbol{q}_S &= -\frac{\kappa}{\theta}\nabla\theta, && (2.31)\\
s_{prod} &= \frac{1}{\theta}\boldsymbol{\tau} : \nabla\boldsymbol{u} + \frac{1}{\theta^2}\kappa|\nabla\theta|^2. && (2.32)
\end{aligned}$$

$\boldsymbol{\tau} = \mu\left(\nabla\boldsymbol{u} + \nabla\boldsymbol{u}^T\right) + \nu\nabla\cdot\boldsymbol{u}\boldsymbol{I}$ *denotes the usual Navier-Stokes Tensor,* μ *and* ν *with* $\mu > 0,\ 2\mu + 3\nu \geq 0$ *the coefficients of viscosity and* $\kappa > 0$ *the coefficient of heat conductivity. The coefficients* μ, ν *and* κ *may depend on temperature and density.*

Note that the coefficient ν might be negative but the condition $2\mu+3\nu \geq 0$ ensures that the entropy production is nonnegative, see for example [42]. A typical choice for the coefficients of viscosity is $\mu > 0$ and $\nu = -\frac{2}{3}\mu$ which is physically correct for one-atomic gases. The expressions for the stress tensor and the heat flux contain the classical contributions of the Stokes and Fourier laws as well as nonclassical contributions in terms of $\nabla\rho$, whereas the entropy flux and entropy production contain only the classical contributions.

We summarize the statements above as a theorem.

Theorem 2.2.2 *Let a material of Korteweg type be given and let* $(\rho, \boldsymbol{u}, \theta)$ *be a sufficiently smooth solution of (2.24), (2.25), (2.26). Then the solution satisfies the entropy equation (2.27) with the entropy flux given by (2.31) and a* **positive** *entropy production given by (2.32), i.e., the solution makes sense from the physical point of view.*

Note: The above given definition of a material of Korteweg type is not the only known way to ensure the positivity of the entropy production. It is possible to add a nonclassical contribution to the entropy flux in favor of the contribution to the heat flux. This results in the classical Fourier law for the heat flux, see the appendix in [75].

2.3 The Navier-Stokes-Korteweg System

For the special choice of the extended free energy

$$f(\theta, \rho, \alpha) = f^{vdW}(\theta, \rho) + \frac{\lambda}{\rho}\alpha \qquad (2.33)$$

where f^{vdW} denotes the van der Waals free energy (2.5) and $\lambda > 0$ is a constant equations (2.24) - (2.26) in conservative form read

$$\begin{aligned}
\rho_t + \nabla\cdot(\rho\boldsymbol{u}) &= 0, && (2.34)\\
(\rho\boldsymbol{u})_t + \nabla\cdot(\rho\boldsymbol{u}\boldsymbol{u}^T) + \nabla p &= \nabla\cdot(\boldsymbol{\tau} + \boldsymbol{K}), && (2.35)\\
\mathcal{E}_t + \nabla\cdot((\mathcal{E} + p)\boldsymbol{u}) &= \nabla\cdot((\boldsymbol{\tau} + \boldsymbol{K})\boldsymbol{u}) - \nabla\cdot\boldsymbol{q}_E. && (2.36)
\end{aligned}$$

Here $\boldsymbol{K} = \lambda\left[\left(\rho\Delta\rho + \frac{1}{2}|\nabla\rho|^2\right)\boldsymbol{I} - \nabla\rho\nabla\rho^T\right]$ denotes the Korteweg part of the stress tensor, $p = p(\theta, \rho)$ the pressure with respect to the van der Waals free energy, $\mathcal{E} = \rho\left(e(\theta,\rho) + \frac{1}{2}|\boldsymbol{u}|^2\right) + \frac{\lambda}{2}|\nabla\rho|^2$ the total energy of the fluid and $\boldsymbol{q}_E$ the heat flux from (2.30).

The contribution $\frac{\lambda}{\rho}\alpha$ in (2.33) is chosen such that we arrive at the classical Navier-Stokes-Korteweg system given in the literature (e.g. [1]). Antanovskii uses the contribution $\lambda\theta\alpha$ instead, see [2]. Hattori and Li state that the choice $\lambda\alpha$ might be more physical but complicated to handle from the mathematical point of view, see [57].

For the Korteweg part of the stress tensor we have the useful identity

$$\nabla\cdot\boldsymbol{K} = \lambda\rho\nabla\Delta\rho. \tag{2.37}$$

The *first order part* of system (2.34) - (2.36) is not hyperbolic in the complete state space because of the shape of the pressure p below the critical temperature. This results in an unstable behavior of solutions in parts of the state space on the one hand and causes problems for the numerical treatment of the system on the other hand, i.e., numerical schemes that are based on Riemann-Solvers and Flux-Vector-Splitting schemes cannot be applied to this system (at least not in the parts of the state space where the sound speed is imaginary).

The Korteweg part of the stress tensor $\boldsymbol{K}$ was first given by Korteweg [72] in 1901. There, the density gradients modeled a nonlocal interaction of molecules within the liquid vapor interface. The system given by equations (2.34) - (2.36) can be found in this form in [1]. In Chapter 3 we give some references to theoretical results associated with the Navier-Stokes-Korteweg system.

2.4 Dimensionless Form of the NSK-System

We provide a dimensionless scaling of all thermodynamic and kinematic quantities we have seen up to now in this section. The result is the dimensionless Navier-Stokes-Korteweg system that has exactly the same structure and the same number of coefficients as system (2.34) - (2.36). The reference values for the thermodynamic quantities are the critical values of the fluid.

Working with dimensionless values can be extremely useful for the numerical treatment of the system since density, velocity, temperature, pressure and other values are always close to the value 1. Using dimensionless values makes it is easy to decide when an interface is small or large or when a viscosity is too small to be resolved numerically. However, the use of dimensionless quantities does not improve the efficiency of the numerical method. It just gives a clearer sight of the situation on the one hand and on the other hand expressions like a total L^2-error of a numerical solution may not make sense for physical values when a vector valued solution has different units for different components.

All relations between dimensionless and physical quantities given throughout this chapter are summarized in Section B.1. In the following, values with a tilde denote dimensionless quantities, whereas the corresponding values without the tilde symbol denote the associated physical one.

In order to derive a dimensionless system with exactly the same structure we choose new scaled variables (denoted by tilde symbols) as follows:

$$\boldsymbol{x} = L\tilde{\boldsymbol{x}}, \qquad L > 0 \text{ reference length}, \tag{2.38}$$

$$t = T\tilde{t}, \qquad T > 0 \text{ reference time}, \tag{2.39}$$

$$\rho = \frac{m}{L^3}\tilde{\rho}, \qquad m > 0 \text{ reference mass in a cube } L^3, \tag{2.40}$$

$$\boldsymbol{u} = \frac{L}{T}\tilde{\boldsymbol{u}}, \qquad \frac{L}{T} \text{ reference velocity}, \tag{2.41}$$

$$\theta = \theta_{crit}\tilde{\theta}, \qquad \theta_{crit} \text{ critical temperature}, \tag{2.42}$$

$$(\mu, \nu) = \frac{m}{LT}(\tilde{\mu}, \tilde{\nu}), \qquad \text{viscosity}, \tag{2.43}$$

$$\lambda = \frac{L^7}{mT^2}\tilde{\lambda}, \qquad \text{capillarity}, \tag{2.44}$$

$$\kappa = \frac{mL}{\theta_{crit}T^3}\tilde{\kappa}, \qquad \text{heat conduction.} \tag{2.45}$$

The only non-standard scaling is the relation (2.44) for the capillarity coefficient λ. The scaling between the *physical* and the dimensionless capillarity coefficient is chosen such that the final system has exactly the same structure as the original system. The reference length L is usually related to the domain Ω, for example the side length of a cube that contains Ω. When we have chosen L we identify the reference mass m and reference time T by the relations

$$\rho_{crit} = \frac{m}{L^3}, \tag{2.46}$$

$$\frac{p_{crit}}{\rho_{crit}} = \frac{L^2}{T^2}. \tag{2.47}$$

Further we choose the reference internal energy to be of size

$$e_{ref} = \frac{L^2}{T^2}. \tag{2.48}$$

Using the scaling (2.38) - (2.45) the mass balance equation reads

$$\frac{\rho_{crit}}{T}\tilde{\rho}_{\tilde{t}} + \frac{\rho_{crit}L}{LT}\tilde{\nabla} \cdot (\tilde{\rho}\tilde{\boldsymbol{u}}) = 0,$$

where $(\cdot)_{\tilde{t}}$ denotes the derivative with respect to the dimensionless time variable $\tilde{t}$ and $\tilde{\nabla}$ the derivatives with respect to the dimensionless space variable $\tilde{\boldsymbol{x}}$. With these scalings and the dimensionless van der Waals equation of state (2.6) the momentum balance equation can be written as

$$\begin{aligned}
&\frac{\rho_{crit}L}{T^2}(\tilde{\rho}\tilde{\boldsymbol{u}})_{\tilde{t}} + \frac{\rho_{crit}L^2}{LT^2}\tilde{\nabla}\cdot(\tilde{\rho}\tilde{\boldsymbol{u}}\tilde{\boldsymbol{u}}^T) + \frac{p_{crit}}{L}\tilde{\nabla}\tilde{p}(\tilde{\theta},\tilde{\rho}) \\
&= \frac{mL}{L^3T^2}\tilde{\nabla}\cdot\left(\tilde{\mu}(\tilde{\nabla}\tilde{\boldsymbol{u}}+\tilde{\nabla}\tilde{\boldsymbol{u}}^T)+\tilde{\nu}\tilde{\nabla}\cdot\tilde{\boldsymbol{u}}\boldsymbol{I}\right) \\
&\quad + \frac{\rho_{crit}^2L^7}{L^3mT^2}\tilde{\nabla}\cdot\left[\tilde{\lambda}\left(\tilde{\rho}\tilde{\Delta}\tilde{\rho}+\frac{1}{2}|\tilde{\nabla}\tilde{\rho}|^2\right)\boldsymbol{I}-\tilde{\lambda}\tilde{\nabla}\tilde{\rho}\tilde{\nabla}\tilde{\rho}^T\right].
\end{aligned}$$

The total energy density can be expressed in terms of the dimensionless values in the following way

$$\begin{aligned}
\mathcal{E} &= \rho_{crit}\,\tilde{\rho}\left(e_{ref}\;\tilde{e}(\tilde{\theta},\tilde{\rho})+\frac{L^2}{T^2}\frac{1}{2}|\tilde{\boldsymbol{u}}|^2\right)+\frac{L^7\rho_{crit}^2}{L^2mT^2}\frac{\tilde{\lambda}}{2}|\tilde{\nabla}\tilde{\rho}|^2 \\
&= \frac{\rho_{crit}\,L^2}{T^2}\left(\tilde{\rho}\tilde{e}(\tilde{\theta},\tilde{\rho})+\frac{1}{2}\tilde{\rho}|\tilde{\boldsymbol{u}}|^2+\frac{\tilde{\lambda}}{2}|\tilde{\nabla}\tilde{\rho}|^2\right) \\
&= \frac{\rho_{crit}\,L^2}{T^2}\tilde{\mathcal{E}}.
\end{aligned}$$

Here we have used the dimensionless equation of state for the internal energy (2.8) and the relations (2.38) - (2.45). Thus, finally the energy balance equation becomes

$$\begin{aligned}
&\frac{\rho_{crit}\,L^2}{T^3}\tilde{\mathcal{E}}_{\tilde{t}} + \frac{\rho_{crit}\,L^3}{LT^3}\tilde{\nabla}\left((\tilde{\mathcal{E}}+\tilde{p}(\tilde{\theta},\tilde{\rho}))\tilde{\boldsymbol{u}}\right) \\
&= \frac{mL^2}{L^3T^3}\tilde{\nabla}\cdot(\tilde{\boldsymbol{\tau}}\tilde{\boldsymbol{u}}) + \frac{\rho_{crit}^2\,L^8}{L^3mT^3}\tilde{\nabla}\cdot\left(\tilde{\boldsymbol{K}}\tilde{\boldsymbol{u}}\right) \\
&\quad + \frac{mL\theta_{crit}}{L^2\theta_{crit}\,T^3}\tilde{\nabla}\cdot\left(\tilde{\kappa}\tilde{\nabla}\tilde{\theta}\right) - \frac{\rho_{crit}^2\,L^8}{L^3mT^3}\tilde{\nabla}\cdot\left(\tilde{\lambda}\tilde{\rho}\tilde{\nabla}\cdot\tilde{\boldsymbol{u}}\tilde{\nabla}\tilde{\rho}\right).
\end{aligned}$$

$\tilde{\boldsymbol{\tau}}$ and $\tilde{\boldsymbol{K}}$ denote the dimensionless Navier-Stokes and Korteweg part of the stress tensor. Multiplying the mass balance equation by $\frac{T}{\rho_{crit}}$, the momentum balance equation by $\frac{T^2}{L\rho_{crit}}$, the energy balance equation by $\frac{T^3}{L^2\rho_{crit}}$ and using the relations (2.46), (2.47) gives the dimensionless Navier-Stokes-Korteweg system

$$\begin{aligned}
\tilde{\rho}_{\tilde{t}} + \tilde{\nabla}\cdot(\tilde{\rho}\tilde{\boldsymbol{u}}) &= 0, \\
(\tilde{\rho}\tilde{\boldsymbol{u}})_{\tilde{t}} + \tilde{\nabla}\cdot(\tilde{\rho}\tilde{\boldsymbol{u}}\tilde{\boldsymbol{u}}^T) + \tilde{\nabla}\tilde{p}(\tilde{\theta},\tilde{\rho}) &= \tilde{\nabla}\cdot(\tilde{\boldsymbol{\tau}}+\tilde{\boldsymbol{K}}), \\
\tilde{\mathcal{E}}_{\tilde{t}} + \tilde{\nabla}\cdot\left((\tilde{\mathcal{E}}+\tilde{p}(\tilde{\theta},\tilde{\rho}))\tilde{\boldsymbol{u}}\right) &= \tilde{\nabla}\cdot\left((\tilde{\boldsymbol{\tau}}+\tilde{\boldsymbol{K}})\tilde{\boldsymbol{u}}\right) \\
&\quad + \tilde{\nabla}\cdot\left(\tilde{\kappa}\tilde{\nabla}\tilde{\theta}-\tilde{\lambda}\tilde{\rho}\tilde{\nabla}\cdot\tilde{\boldsymbol{u}}\tilde{\nabla}\tilde{\rho}\right).
\end{aligned}$$

The dimensionless equations of state are given by (2.6), (2.8) and the dimensionless quantities are related to the physical ones by the scaling (2.38) - (2.45).

2.5 Including Gravity

Up to now we have neglected external (volumetric) forces, such as gravity, and heat sources. Taking these effects into account the Navier-Stokes-Korteweg system must be modified as

$$\begin{aligned}\rho_t + \nabla \cdot (\rho \boldsymbol{u}) &= 0,\\ (\rho \boldsymbol{u})_t + \nabla \cdot (\rho \boldsymbol{u}\boldsymbol{u}^T) + \nabla p &= \nabla \cdot (\boldsymbol{\tau} + \boldsymbol{K}) + \rho \boldsymbol{g},\\ \mathcal{E}_t + \nabla \cdot ((\mathcal{E} + p)\boldsymbol{u}) &= \nabla \cdot ((\boldsymbol{\tau} + \boldsymbol{K})\boldsymbol{u}) - \nabla \cdot \boldsymbol{q}_E + \rho \boldsymbol{g} \cdot \boldsymbol{u} + Q.\end{aligned}$$

where the volumetric force $\boldsymbol{g} \in R^n$ and the heat source $Q \in \mathbb{R}$ may depend on space and time variables in general. In the case of gravity, $\boldsymbol{g}$ is simply a constant vector.

Using the notation of the previous section the physical and dimensionless quantities are related to each other by the relations

$$\boldsymbol{g} = \frac{L}{T^2}\tilde{\boldsymbol{g}}, \tag{2.49}$$

$$Q = \rho_{crit}\frac{L^2}{T^3}\tilde{Q}. \tag{2.50}$$

2.6 Boundary Conditions

Typical boundary conditions on $\partial\Omega$ for the Navier-Stokes equations are homogeneous Dirichlet data for the velocity field (no-slip) and Dirichlet data for the temperature.

$$\boldsymbol{u} = \boldsymbol{0}, \tag{2.51}$$

$$\theta = \theta_b, \tag{2.52}$$

where θ_b is a given function on $\partial\Omega$. Because of the presence of the higher order terms in the Navier-Stokes-Korteweg equations an additional boundary equation is required. The additional boundary conditions we use have the effect that they control the contact angle of a diffuse interface at the boundary. The simplest choice is

$$\nabla\rho \cdot \boldsymbol{n} = 0, \tag{2.53}$$

which is a special form of

$$-\frac{\nabla\rho}{|\nabla\rho|} \cdot \boldsymbol{n} = \cos\varphi, \tag{2.54}$$

where φ is the contact angle between interface and boundary, i.e., φ depends on the material of the fluid as well as on the material of the boundary.

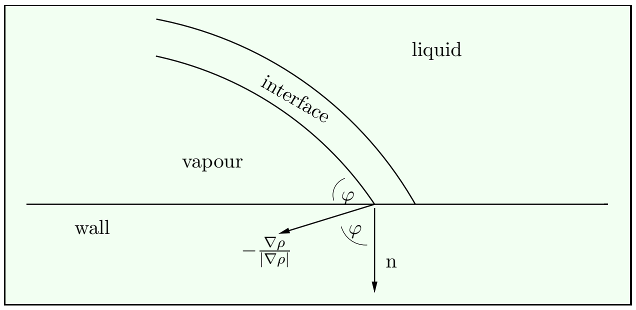

Figure 2.4: Contact angle of a diffuse interface.

2.7 The Isothermal Case

The main focus of this work is the development and verification of reliable numerical methods for the isothermal version of the Navier-Stokes-Korteweg system. In the isothermal case, i.e., we neglect the energy balance equation and assume that the temperature stays at a constant state, the Navier-Stokes-Korteweg system reduces to

$$\rho_t + \nabla\cdot(\rho\boldsymbol{u}) = 0, \tag{2.55}$$

$$(\rho\boldsymbol{u})_t + \nabla\cdot(\rho\boldsymbol{u}\boldsymbol{u}^T) + \nabla p(\rho) = \nabla\cdot(\boldsymbol{\tau} + \boldsymbol{K}), \tag{2.56}$$

with the no-slip boundary condition (2.51) and (either (2.53) or (2.54)).

Here the free energy f^{vdW} depends only on the density ρ and therefore the pressure is given by $p(\rho) = \rho^2 f_\rho^{vdW}(\rho)$. The temperature is only a parameter which is kept at a constant state below the critical temperature such that phase transitions are allowed.

In the case of boundary condition (2.53) we have additionally an energy decay equation.

Lemma 2.7.1 *Let $(\rho, \rho\boldsymbol{u})$ be a (sufficiently smooth) solution of the isothermal Navier-Stokes-Korteweg equations with boundary conditions (2.51) and (2.53). Then the energy decay equation*

$$\frac{d}{dt}\int_\Omega \mathcal{E}(\rho, \rho\boldsymbol{u}, \alpha)\,d\boldsymbol{x} = -\int_\Omega \tau : \nabla\boldsymbol{u}\,d\boldsymbol{x} \quad \leq 0 \tag{2.57}$$

is satisfied, where $\mathcal{E} = \rho\left(f(\rho) + \frac{|\boldsymbol{u}|^2}{2}\right) + \lambda\alpha$ denotes the total physical energy density and $\alpha = \frac{1}{2}|\nabla\rho|^2$.

Proof. We set

$$W(\rho) = \rho f^{vdW}(\rho). \tag{2.58}$$

Then, because of the definition of the pressure (2.3), we have the relation

$$p(\rho) = \rho W'(\rho) - W(\rho) \tag{2.59}$$

for the pressure. We multiply the continuity equation (2.55) with $(W'(\rho) - \frac{|\boldsymbol{u}|^2}{2})$ and the momentum equation by $\boldsymbol{u}$. Summation of both parts and integration over the domain

Ω gives

$$\int_\Omega \left(W'(\rho) - \frac{|\boldsymbol{u}|^2}{2}\right)(\rho_t + \nabla\cdot(\rho\boldsymbol{u}))$$
$$+\boldsymbol{u}\cdot\big((\rho\boldsymbol{u})_t + \nabla\cdot(\rho\boldsymbol{u}\boldsymbol{u}^T) + \nabla p(\rho) - \nabla\cdot\boldsymbol{\tau} - \nabla\cdot\boldsymbol{K}\big)\ d\boldsymbol{x} = 0.$$

We replace the term $\nabla\cdot\boldsymbol{K}$ using the relation (2.37), by reordering the terms we get

$$\begin{aligned}\int_\Omega W'(\rho)\rho_t - \frac{|\boldsymbol{u}|^2}{2}\rho_t + \boldsymbol{u}\cdot(\rho\boldsymbol{u})_t\, d\boldsymbol{x} - \int_\Omega \lambda\rho\boldsymbol{u}\cdot\nabla\Delta\rho\, d\boldsymbol{x}\\
= \int_\Omega \boldsymbol{u}\cdot\nabla\cdot\boldsymbol{\tau}\, d\boldsymbol{x} - \int_\Omega \boldsymbol{u}\cdot(W'(\rho)\nabla\rho + \nabla p(\rho)) + W'(\rho)\rho\nabla\cdot\boldsymbol{u}\, d\boldsymbol{x}\\
-\int_\Omega \boldsymbol{u}\cdot\nabla\cdot(\rho\boldsymbol{u}\boldsymbol{u}^T) - \frac{|\boldsymbol{u}|^2}{2}\nabla\cdot(\rho\boldsymbol{u})\, d\boldsymbol{x}.\end{aligned}$$

We apply integration by parts to the colored terms. All boundary integrals vanish due to the boundary condition $\boldsymbol{u} = \boldsymbol{0}$ on $\partial\Omega$.

$$\begin{aligned}\int_\Omega \frac{d}{dt}W(\rho) + \frac{d}{dt}\left(\rho\frac{|\boldsymbol{u}|^2}{2}\right)\ d\boldsymbol{x} + \int_\Omega \lambda\nabla\cdot(\rho\boldsymbol{u})\ \Delta\rho\, d\boldsymbol{x}\\
= \ -\int_\Omega \boldsymbol{\tau} : \nabla\boldsymbol{u}\, d\boldsymbol{x} - \int_\Omega \boldsymbol{u}\cdot\nabla(W(\rho) + p(\rho) - W'(\rho)\rho)\, d\boldsymbol{x}\\
-\int_\Omega \boldsymbol{u}\cdot\nabla\cdot(\rho\boldsymbol{u}\boldsymbol{u}^T) + \rho\boldsymbol{u}\cdot\nabla\left(\frac{|\boldsymbol{u}|^2}{2}\right)\ d\boldsymbol{x}.\end{aligned}$$

Now we use the continuity equation and replace the term $\nabla\cdot(\rho\boldsymbol{u})$ by $-\rho_t$ in the second integral on the left hand side and again we perform integration by parts on this term. The resulting boundary integral vanishes because of the boundary condition $\nabla\rho\cdot\boldsymbol{n} = 0$ on $\partial\Omega$. The second integral on the right hand side of the equation vanishes because of the identity (2.59). The integrand in the last integral can be written in divergence form.

$$\begin{aligned}\int_\Omega \frac{d}{dt}W(\rho) + \frac{d}{dt}\left(\rho\frac{|\boldsymbol{u}|^2}{2}\right)\ d\boldsymbol{x} + \int_\Omega \lambda\nabla\rho_t\cdot\nabla\rho\, d\boldsymbol{x}\\
= -\int_\Omega \boldsymbol{\tau} : \nabla\boldsymbol{u}\, d\boldsymbol{x} - \int_\Omega \nabla\cdot\left(\rho|\boldsymbol{u}|^2\boldsymbol{u}\right)\ d\boldsymbol{x}.\end{aligned}$$

The last integral vanishes due to the Gauss theorem and $\boldsymbol{u} = \boldsymbol{0}$ on $\partial\Omega$. Finally we get

$$\frac{d}{dt}\int_\Omega W(\rho) + \rho\frac{|\boldsymbol{u}|^2}{2} + \frac{\lambda}{2}|\nabla\rho|^2\ d\boldsymbol{x} = -\int_\Omega \boldsymbol{\tau} : \nabla\boldsymbol{u}\, d\boldsymbol{x} \quad \leq 0.$$

An important class of solutions are static equilibrium solutions, i.e., steady state solutions where the velocity field vanishes completely in the domain Ω. This kind of solutions satisfies a nonlinear elliptic equation. We state this in the following lemma.

Lemma 2.7.2 *Let Ω be connected and let $(\rho, \rho\boldsymbol{u})$ be a smooth static equilibrium solution of the isothermal Navier-Stokes-Korteweg equations, i.e., a solution that satisfies $\rho_t = 0$ and $\boldsymbol{u} = \boldsymbol{0}$ in $\Omega \times (0, \infty)$. Then the density satisfies the nonlinear elliptic equation*

$$\mu(\rho) - \lambda\Delta\rho = cst, \tag{2.60}$$

where the constant on the right hand side is in general not known and μ denotes the chemical potential (2.4) that does not depend on temperature in the isothermal case.

Proof. All terms including $\boldsymbol{u}$ and the gradient of $\boldsymbol{u}$, this means τ, drop out and only the pressure and Korteweg term in the momentum equation remain

$$\nabla p(\rho) = \nabla \cdot \boldsymbol{K}.$$

Using the identity $p_\rho(\rho) = \rho\mu_\rho(\rho)$ (2.3), (2.4) and the identity (2.37) we obtain

$$\nabla\mu(\rho) = \lambda\nabla\Delta\rho$$

and therefore we have for some constant

$$\mu(\rho) - \lambda\Delta\rho = cst,$$

which completes the proof.

From another point of view equation (2.60) is the Euler-Lagrange equation (and the constant on the right hand side of this equation the Lagrange multiplier) for the minimization problem

$$\int_\Omega W(\rho) + \frac{\lambda}{2}|\nabla\rho|^2 \, d\boldsymbol{x} \;\rightarrow\; \min, \tag{2.61}$$

with the constraint that the total mass is conserved

$$\int_\Omega \rho \, d\boldsymbol{x} \;=\; m, \tag{2.62}$$

where the function W is defined as in (2.58) and m is some positive constant. The energy functional in (2.61) consists of the (isothermal) internal energy and the Korteweg part of the energy. A fluid of a given fixed mass at static equilibrium should minimize this energy functional. From (2.61) one can clearly see that the Korteweg term acts like a panelization term for phase transitions (at least at static equilibrium) because where a phase transition occurs the gradient of the density is large.

The minimization problem (2.61), (2.62) and its Euler-Lagrange equation (2.60) play an important role in giving a physical meaning to the parameter λ in the Korteweg term. This parameter is related to surface tension (at least at static equilibrium) in some sense. The next section is dedicated to this relation, the theoretical background is summarized in Section 3.1.

2.8 Surface Tension at Static Equilibrium

The goal of this section is to relate the coefficient λ in the Korteweg term to the physical effect of surface tension. For the theoretical background see Section 3.1.

In sharp interface models, i.e., models where the change from one phase to another is discontinuous and the interface itself is a set of measure zero, the effect of surface tension is usually modeled by an additional contribution to the stress tensor that acts only on the interface, see for example [79] and Section 4.4.1.

In a sharp interface model for a liquid-vapor flow we can decompose the domain Ω into two distinct subsets Ω_v, Ω_l, the vapor and liquid parts respectively, and an interface of measure zero. At a static equilibrium configuration the density in the vapor and liquid part are constant values denoted by ρ_v, ρ_l and they satisfy a *mechanical equilibrium condition* (the Young-Laplace law) and a *phase equilibrium condition* namely

$$\begin{aligned} p(\rho_l) - p(\rho_v) &= (n-1)\sigma k_m, && (2.63)\\ \mu(\rho_l) - \mu(\rho_v) &= 0, && (2.64) \end{aligned}$$

see [75] and the references therein, see also [79]. Up to now we have always considered the three dimensional space, in order to be more general we consider the n-dimensional space for $n \geq 1$. k_m denotes the mean curvature of the interface and the constant coefficient σ denotes the *surface tension* of the fluid.
Note: If surface tension is neglected ($\sigma = 0$) and in the planar case the values ρ_v and ρ_l are equal to the Maxwell values (2.15), (2.16).

Characteristic for the class of diffuse interface models is the smooth and continuous change from one phase to another. Thus, we cannot simply decompose the domain in liquid parts, vapor parts and an interface of measure zero as we could in the case of sharp interface models. But in the case of our model, the Navier-Stokes-Korteweg system, we can find distinct sets $\Omega_v \subset \Omega$, $\Omega_l \subset \Omega$, $\Omega_i \subset \Omega$ with $\Omega_v \cup \Omega_l \cup \Omega_i = \Omega$ such that there exist constant density states ρ_v and ρ_l with $|\rho - \rho_v|$ is *small* in Ω_v, $|\rho - \rho_l|$ is *small* in Ω_l and the measure of the *interface* Ω_i is *small*. See Section 3.1 for a rigorous treatment of the above and the following statements.

Kraus and Dreyer showed [75] that the mechanical equilibrium condition and the phase equilibrium condition can be recovered (up to an error of higher order in the coefficient λ) in the case of the Navier-Stokes-Korteweg model at static equilibrium. They showed that

$$\begin{aligned} p(\rho_l) - p(\rho_v) &= (n-1)c_0\sqrt{\lambda}k_m + o(\sqrt{\lambda}), && (2.65)\\ \mu(\rho_l) - \mu(\rho_v) &= o(\sqrt{\lambda}). && (2.66) \end{aligned}$$

Where

$$c_0 = \sqrt{2}\int_{\rho_v^M}^{\rho_l^M} \sqrt{\rho f(\rho) - \rho\mu(\rho_v^M) + p(\rho_v^M)}\, d\rho. \tag{2.67}$$

Thus, comparing (2.63) and (2.65), we can identify the effect of surface tension that is *implicitly included* in the Navier-Stokes-Korteweg model by the relation

$$\sigma = c_0 \sqrt{\lambda}. \tag{2.68}$$

Note that the error term $o(\sqrt{\lambda})$ is neglected.

Note that equation (2.65) is an asymptotic formula for $\lambda \to 0$. Therefore the identification with surface tension (2.68) may make sense only in a regime where λ is sufficiently small. In Section 4.1.2 we approve by numerical computations of static equilibrium solutions that the error in the asymptotic is negligible in the regime of our interest and hence, the identification with surface tension makes sense in the case of our numerical simulations.

In general, the free energy f and therefore the Maxwell values and the coefficient c_0 depend on the temperature. Using dimensionless variables in terms of the critical values, i.e., $\rho_{crit}\,\tilde{\rho} = \rho$ and $\theta_{crit}\,\tilde{\theta} = \theta$, we have

$$\begin{aligned} c_0(\theta) &= \sqrt{2} \int\limits_{\rho_v^M(\theta)}^{\rho_l^M(\theta)} \sqrt{\rho f(\theta,\rho) - \rho\mu(\theta,\rho_v^M(\theta)) + p(\theta,\rho_v^M(\theta))}\, d\rho \\ &= \rho_{crit}\,\sqrt{p_{crit}}\,\tilde{c}_0(\tilde{\theta}), \end{aligned}$$

where the dimensionless quantity $\tilde{c}_0$ is defined by

$$\tilde{c}_0(\tilde{\theta}) = \sqrt{2} \int\limits_{\tilde{\rho}_v^M(\tilde{\theta})}^{\tilde{\rho}_l^M(\tilde{\theta})} \sqrt{\tilde{\rho}\tilde{f}(\tilde{\theta},\tilde{\rho}) - \tilde{\rho}\tilde{\mu}(\tilde{\theta},\tilde{\rho}_v^M(\tilde{\theta}) + \tilde{p}(\tilde{\theta},\tilde{\rho}_v^M(\tilde{\theta}))}\, d\tilde{\rho}.$$

The coefficient $\tilde{c}_0$ can be approximated by the formula

$$\tilde{c}_0(\tilde{\theta}) \approx \sqrt{2} \cdot \sqrt{1.0 - \tilde{\theta}} \cdot \left(6.4 \cdot (1.0 - \tilde{\theta}) - 0.7 \cdot (1.0 - \tilde{\theta})^2\right). \tag{2.69}$$

This formula is obtained by curve fitting and gives quite accurate results in the dimensionless temperature range $\tilde{\theta} \in [0.6, 1.0]$.

Using the relation (2.68) and the scaling (2.44) this gives for the dimensionless coefficient $\tilde{\lambda}$ the identity

$$\tilde{\lambda} = \left(\frac{\sigma(\theta_{crit}\,\tilde{\theta})}{L\, p_{crit}\,\tilde{c}_0(\tilde{\theta})} \right)^2. \tag{2.70}$$

Note: The coefficient $\tilde{\lambda}$ depends on the temperature $\tilde{\theta}$ in general whereas we have assumed that it is a constant coefficient in the Navier-Stokes-Korteweg model. Thus, it has to be fixed to some *mean temperature* in the temperature dependent model and to the reference temperature in the isothermal model.

2.9 Interface Width

A scaling argument given in [95] (Proposition 2.2.7) shows that the width of the diffuse interface between the phases must be proportional to $\sqrt{\lambda}$. The proportional constant remains to be determined. However, the definition of the interface itself is arbitrary (up to some degree). We give possible definitions for the interface and the interface width in Section 4.1. Using the numerically computed profiles of static bubbles for different reference temperatures and different coefficients λ we can determine the width of the interface $w(\theta, \lambda)$.

Below the critical temperature we observe (see Section 4.1 especially (4.9)) that the dimensionless interface width $\tilde{w}(\tilde{\theta}, \tilde{\lambda})$ of a dimensionless fluid can (roughly) approximated by the formula

$$\tilde{w}(\tilde{\theta}, \tilde{\lambda}) = 5.4 \cdot \tilde{\theta}^2 \cdot \sqrt{\tilde{\lambda}}.$$

This is a very rough formula but quite useful to construct initial data that consists of liquid and vapor phases and for rough estimates of the interface size. Numerical experiments show that a *suitable* interface size of the initial data is important to guarantee the stability of solutions. Otherwise instabilities are observed. The above formula is obtained from the computations of static bubbles. The interface size may also depend on the dynamics of the phase boundary. Such effects are not taken into account but this should not make a big difference.

2.10 Realistic Length Scale

The goal of this section is to determine the maximal possible diameter L_{max} of a domain that can be chosen for realistic numerical simulations of liquid-vapor flows when all physical parameters are adjusted *correctly*. This means, chosen as described in the previous sections. The result of this section is that L_{max} is extremely small (in the micrometer regime).

We assume that the diameter of the dimensionless computational domain is equal to one. We assume that the minimal possible interface that must be resolved by the underlying computational mesh is $\tilde{w}_{min} = 1.0 \cdot 10^{-3}$. This is what is possible at time of this writing when all modern numerical techniques such as local mesh adaption, parallelization, load balancing, higher order schemes are combined and for the computation it is possible to run on many processors and for many days (possibly weeks).

Using the formula for the interface width from the previous section together with the scaling for $\tilde{\lambda}$ from (2.70) we get a formula for L_{max} in terms of $\tilde{w}_{min}$, the reference temperature and the critical values.

$$L_{max} = 5.4 \cdot \tilde{\theta}_{ref}^2 \, \frac{\sigma(\tilde{\theta})}{\tilde{w}_{min} \, p_{crit} \, \tilde{c}_0(\tilde{\theta}_{ref})} \tag{2.71}$$

As an example we choose water at different temperatures.

Example 2.10.1 *(Water at different temperatures)*

We choose the three dimensionless reference temperatures $\theta_{ref} = 0.85$, $\theta_{ref} = 0.90$ and $\theta_{ref} = 0.95$. As discussed above we choose the minimal possible interface size

$$\tilde{w}_{min} = 1.0 \cdot 10^{-3}$$

and the critical pressure of water is

$$p_{crit} = 22.064 \cdot 10^6 \frac{N}{m^2},$$

see Section B.2 and Table B.1. Using the formula (2.69) we have for the coefficient $\tilde{c}_0$

$$\tilde{c}_0(0.85) = 0.52, \quad \tilde{c}_0(0.90) = 0.29, \quad \tilde{c}_0(0.95) = 0.10$$

and for the determination of the surface tension (roughly is sufficient) we can make use of figure B.4

$$\sigma(0.85) = 2.0 \cdot 10^{-2} \frac{N}{m}, \quad \sigma(0.90) = 1.2 \cdot 10^{-2} \frac{N}{m}, \quad \sigma(0.95) = 5.0 \cdot 10^{-3} \frac{N}{m}.$$

As result using the formula above we get

$$\begin{aligned} L_{max}(0.85) &= 3.54 \cdot 10^{-6} m, \\ L_{max}(0.90) &= 2.38 \cdot 10^{-6} m, \\ L_{max}(0.95) &= 1.10 \cdot 10^{-6} m. \end{aligned}$$

This means that the largest possible domain for realistic numerical simulations must be in the micrometer regime. This is at least one or two orders of magnitude too small for realistic numerical simulations of the experiment discussed in the introduction, see Section 1.1.

2.11 Artificial Enlargement of the Interface

We have seen in the previous section that the domain in that a liquid-vapor flow can be simulated using the Navier-Stokes-Korteweg model must be extremely small (in the micrometer regime). This is because the diffuse interface must be completely resolved by the underlying computational mesh. One way to overcome this problem is to enlarge the interface by increasing the coefficient λ. The width of the interface in proportional to $\sqrt{\lambda}$. The problem is that at the same time the surface tension force is increased that is also proportional to $\sqrt{\lambda}$. In cases where other forces do not significantly dominate and the effect of surface tension can not be increased without changing the dynamics completely, this approach cannot be applied.

In [64] an approach is presented to artificially enlarge the interface without changing the effect of surface tension. But with this approach it is necessary to change the behavior of the fluid by replacing the equation of state. The idea is to replace the van der Waals equation of state by a modified equation of state such that certain thermodynamical properties are preserved for density states close to the Maxwell states.

Thus, in this approach the model is modified and therefore we do not consider this approach in the present work since we are interested in the validation and applicability of the original Navier-Stokes-Korteweg model.

Chapter 3

Summary of Theoretical Results

We give a summary of existing theoretical results concerning the Navier-Stokes-Korteweg system. In the first two sections we discuss the existence of special solutions such as static equilibrium solutions and traveling wave solutions. Under some assumptions, these types of solutions satisfy ordinary differential equations and can be solved by application of existing ordinary boundary value problem solvers. We do this in the next chapter such that we have these kinds of solution as benchmarks for numerical schemes available. Another aspect of the first section is the clarification of the effect of surface tension that the Korteweg term in the NSK system implicitly includes. The third section is dedicated to the discussion of general solutions, i.e., solutions of the Cauchy problem in multiple space dimensions for the isothermal and the temperature dependent Navier-Stokes-Korteweg model as well as solutions of the initial boundary value problem.

3.1 Static Equilibrium Solutions and Surface Tension

The Objectives of this section are the discussion of the existence of static equilibrium solutions for the isothermal Navier-Stokes-Korteweg equations on the one hand and the rigorous clarification of the role of surface tension in the model on the other hand. Throughout this section we assume that the reference temperature of the van der Waals equation of state is fixed to a value below the critical temperature such that the fluid can undergo phase transitions. According to Lemma 2.7.2 a smooth solution of the NSK equations satisfies the nonlinear elliptic equation (2.60). This equation is also the Euler-Lagrange equation for the minimization problem we state below.

Let $\Omega \subset \mathbb{R}^n$ be an open bounded domain and let f denote the free energy of an isothermal van der Waals fluid. We define $W(\rho) = \rho f(\rho)$. For a constant $m > 0$, a scaling parameter $\varepsilon > 0$ and $\rho^\varepsilon \in H^1(\Omega)$ we consider the following minimization problem with the constraint that the total mass in Ω is conserved:

$$\int\limits_\Omega W(\rho^\varepsilon(\boldsymbol{x})) + \frac{\varepsilon^2}{2}|\nabla \rho^\varepsilon(\boldsymbol{x})|^2 \, d\boldsymbol{x} \to \min, \qquad \int\limits_\Omega \rho^\varepsilon(\boldsymbol{x}) \, d\boldsymbol{x} = m. \tag{3.1}$$

In this section we characterize static equilibrium solutions by minimizers of the minimization problem (3.1). A smooth minimizer of the minimization problem (3.1) satisfies the Euler-Lagrange Equation

$$W'(\rho^\varepsilon) - \varepsilon^2 \Delta \rho^\varepsilon = c^\varepsilon \quad \text{in } \Omega, \tag{3.2}$$

where the constant $c^\varepsilon \in \mathbb{R}$ is the Lagrange multiplier, see for example [32]. By definition of W the function W' is equal to the chemical potential μ (see definition 2.1.1) and thus, equation (3.2) is the same as the equilibrium condition (2.60). Here the coefficient λ is replaced by ε^2.

Gurtin and Matano proved the existence of minimizers of the variational problem (3.1), see [56]. Gurtin and Matanos theorem, the theorem by Modica [86] and the results by Kraus and Dreyer, we will discuss in this chapter, are not restricted to a van der Waals equation of state. These results are valid for a general (double well) free energy f with certain properties, see [75] for details. These properties are satisfied by a van der Waals equation of state if the reference temperature is fixed to some value below the critical temperature.

We summarize the results by Gurtin and Matano in the following theorem.

Theorem 3.1.1 *(Gurtin, Matano [56])*
Let $\varepsilon > 0$.

(i) There exists a global minimizer ρ^ε of the minimization problem (3.1).

(ii) A local minimizer ρ^ε is contained in $C^3(\Omega)$, satisfies the Euler-Lagrange equation (3.2) and has natural boundary conditions

$$\nabla \rho^\varepsilon \cdot \boldsymbol{n} = 0 \quad \textit{on } \partial\Omega.$$

Note: This is boundary condition (2.53).

Modica [86] considered a family of global minimizers $(\rho^\varepsilon)_{\varepsilon>0}$ of the minimization problem (3.1). He proved that for $\varepsilon \to 0$ a subsequence converges in $L^1(\Omega)$ to some limit function ρ^0 that assumes only two values (the Maxwell states) almost everywhere and the (sharp) interface between the liquid and the vapor phase is minimized in some sense. For the statement of Modicas results we need some definitions given below. For some function $u \in L^1(\Omega)$ we define

$$\int\limits_\Omega |Du(\boldsymbol{x})| \, d\boldsymbol{x} = \sup \left\{ \int_\Omega u(\boldsymbol{x}) \nabla \psi(\boldsymbol{x}) \, d\boldsymbol{x} \;\middle|\; \psi \in C^\infty(\Omega), |\psi| \le 1 \right\}.$$

For some measurable set $E \in \mathbb{R}^n$ we define the perimeter of E in Ω by

$$P_\Omega[E] = \int\limits_\Omega |D\chi_E(\boldsymbol{x})| \, d\boldsymbol{x}.$$

In the above definition χ_E denotes the characteristic function of the set E. The perimeter is a generalization of the $(n-1)$-dimensional Hausdorff measure, i.e., if $\partial E \cap \Omega$ is a Lipschitz continuous hypersurface then $\mathcal{H}_{n-1}(\partial E \cap \Omega)$ equals $P_\Omega[E]$.

Theorem 3.1.2 *(Modica [86])*
For $\varepsilon > 0$ let ρ^ε denote a global minimizer of the minimization problem (3.1) and let $m \in [\rho_v^M|\Omega|,\ \rho_l^M|\Omega|]$.

(i) There exists a sequence $(\varepsilon_k)_{k\in\mathbb{N}}$ with $\lim\limits_{k\to\infty} \varepsilon_k = 0$, a corresponding sequence of global minimizers ρ^{ε_k} and a function $\rho^0 \in L^1(\Omega)$ such that

$$\lim_{k\to\infty} ||\rho^{\varepsilon_k} - \rho^0||_{L^1(\Omega)} = 0.$$

(ii) For the function ρ^0 we have

$$\rho^0 = \rho_v^M \quad or \quad \rho^0 = \rho_l^M$$

almost everywhere and $\rho_0 \in BV(\Omega)$.

(iii) The set $U_v = \{\boldsymbol{x} \in \Omega \mid \rho^0(\boldsymbol{x}) = \rho_v^M\}$ is a minimizer of the geometric variational problem

$$P_\Omega[U_v] = \min\left\{P_\Omega[F] \;\middle|\; F \subset \Omega, |F| = \frac{\rho_l^M|\Omega| - m}{\rho_l^M - \rho_v^M}\right\}.$$

Furthermore we define the set $U_l = \Omega\backslash U_v$.

In the case where the mean density has a value between the two Maxwell states Modica proved that in the limit $\varepsilon \to 0$ a subsequence of global minimizers of problem (3.1) converges to function ρ^0 in $L^1(\Omega)$, where the function ρ^0 assumes only the Maxwell states and the interface between the liquid and vapor phases is minimized. From the physical point of view a minimal interface is the correct behavior but this also means that the pressure in the vapor phase equals the pressure in the liquid phase. We have for the liquid and vapor states ρ_l and ρ_v

$$p(\rho_l) - p(\rho_v) \quad = \quad 0,$$

in contrast to the Young-Laplace law (2.63), see also [79], that must be satisfied by the physical relevant solution

$$p(\rho_l) - p(\rho_v) \quad = \quad (n-1)\sigma k_m.$$

According to the Young-Laplace law this means that either the mean curvature k_m of the interface is equal to zero (e.g. a flat interface) or the surface tension is equal to zero (surface tension neglected). However, surface tension is a very important physical property and cannot be neglected in most cases. Therefore the limit function ρ^0 is obviously not the correct solution from the physical point of view and in the (sharp interface)-limit $\varepsilon \to 0$ there is no surface tension left. Thus, the above contradiction cannot be solved by the sharp interface limit.
Now the idea is not choose the limit function but some function ρ^ε from the limit process for some small value $\varepsilon > 0$ as the relevant solution. In this case we have a diffuse interface and inside the interface the function ρ^ε changes rapidly from one nearly constant state to another nearly constant state. We cannot decompose the domain into vapor and

liquid sets U_v, U_l and an interface I of measure zero but we can decompose it into sets $\hat{U}_v$, $\hat{U}_l$ where the function ρ^ε nearly assumes constant vapor and liquid states. Finally the diffuse interface $\hat{I}$ does not have measure zero but has a *small* measure.

Thus, the challenge is to determine the right small parameter $\varepsilon > 0$ such that the Young-Laplace law and the phase equilibrium condition (2.64) are satisfied in some sense. Kraus and Dreyer [75] showed that the parameter $\varepsilon > 0$ can be identified with surface tension and they give an asymptotic formula that relates the parameter ε to surface tension. We summarize the main statements of this work in the following theorem.

Theorem 3.1.3 *(Kraus, Dreyer [75])*
Let $(\varepsilon_k)_{k\in\mathbb{N}}$ with $\lim_{k\to\infty} \varepsilon_k = 0$, ρ^{ε_k} a sequence of global minimizers of the variational problem (3.1) that converges to ρ^0 as in theorem 3.1.2. Further let $\hat{U}_v \subset\subset U_v$ and $\hat{U}_l \subset\subset U_l$. Then

(i) $$\rho^\varepsilon(\boldsymbol{x}) = \begin{cases} \rho_v^M + \varepsilon^k \hat{\rho}_v + o(\varepsilon^k) & \boldsymbol{x} \in \hat{U}_v, \\ \rho_l^M + \varepsilon^k \hat{\rho}_l + o(\varepsilon^k) & \boldsymbol{x} \in \hat{U}_l. \end{cases}$$

(ii) $p\left(\rho^{\varepsilon_k}(x_l)\right) - p\left(\rho^{\varepsilon_k}(x_v)\right) = (n-1)c_0 k_m \varepsilon_k + o(\varepsilon_k)$ *for almost all $x_v \in \hat{U}_v$ and $x_l \in \hat{U}_l$. Here k_m is the constant mean curvature of the* reduced boundary *of U_v.*

(iii) $\mu\left(\rho^{\varepsilon_k}(x_l)\right) - \mu\left(\rho^{\varepsilon_k}(x_v)\right) = o(\varepsilon_k)$ *for almost all $x_v \in \hat{U}_v$ and $x_l \in \hat{U}_l$.*

The constant c_0 is given by relation (2.67).

Note: The reduced boundary $\partial^* U_v$ of U_v is a dense subset of ∂U_v which consists of countable union of smooth hypersurfaces, see [75] and the references therein.

As discussed before the physically relevant solution ρ has to satisfy the Young-Laplace law and the phase equilibrium condition namely

$$\begin{aligned} p(\rho(x_l)) - p(\rho(x_v)) &= (n-1)\sigma k_m, \\ \mu(\rho(x_l)) - \mu(\rho(x_v)) &= 0, \end{aligned} \qquad \text{for } x_v \in U_v \text{ and } x_l \in U_l$$

and, of course, ρ assumes only two values in the case where two phases are present. We compare these requirements to the formulas given in item (ii) and (iii) of the above theorem

$$\begin{aligned} p(\rho(x_l)) - p(\rho^\varepsilon(x_v)) &= (n-1)c_0\varepsilon k_m + o(\varepsilon), \\ \mu(\rho(x_l)) - \mu(\rho^\varepsilon(x_v)) &= o(\varepsilon), \end{aligned} \qquad \text{for } x_v \in \hat{U}_v \text{ and } x_l \in \hat{U}_l.$$

As a result we can associate the parameter ε with surface tension, required that ε is sufficiently small, by the relation

$$\sigma = c_0\varepsilon. \tag{3.3}$$

The question that remains is where the asymptotic regime begins (where ε is small enough) such that the above formula is applicable. The numerical justification of this formula performed in Section 4.1 shows that the error term $o(\varepsilon)$ is negligible even for quite large interfaces (large values of ε).

3.2 Traveling Wave Solutions

Benzoni-Gavage proved in [11], [12] based on [106] the existence of traveling wave solutions for the isothermal Navier-Stokes-Korteweg system with a modified third order capillarity term. The proof is split into two parts. The first part shows that traveling wave solutions exist when the viscosity in the model is neglected, the second part generalizes this to the case with small viscosity. We summarize these results and give a proof for the first part for the unmodified Navier-Stokes-Korteweg equations.

We consider another kind of special solutions to the isothermal Navier-Stokes-Korteweg equations in this section. We investigate the existence of propagating planar phase boundaries and therefore we can restrict ourself to the one dimensional system which reduces to

$$\begin{aligned} \rho_t + (\rho u)_x &= 0, \\ (\rho u)_t + (\rho u^2 + p(\rho))_x &= \varepsilon u_{xx} + \lambda\left(\rho\rho_{xx} - \tfrac{1}{2}\rho_x^2\right), \end{aligned} \quad \text{in } \mathbb{R}\times\mathbb{R}_{>0}. \tag{3.4}$$

We are interested in traveling wave solutions of system (3.4), i.e. smooth solutions of the form

$$\begin{aligned} \rho(x,t) &= \tilde{\rho}(x-st), \\ u(x,t) &= \tilde{u}(x-st), \end{aligned} \tag{3.5}$$

that connect left states (ρ^-, u^-) and right states (ρ^+, u^+) in different phases and propagate with a constant speed $s \in \mathbb{R}$ ($'$ denotes the derivative with respect to $x - st$)

$$\tilde{\rho}(\pm\infty) = \rho^{\pm}, \quad \tilde{u}(\pm\infty) = u^{\pm}, \quad \tilde{\rho}'(\pm\infty) = 0. \tag{3.6}$$

The left and right states must satisfy, see for example [100] or [34], the Rankine-Hugoniot relation

$$\rho^-(u^- - s) = \rho^+(u^+ - s) \quad =: \quad m, \tag{3.7}$$

$$\rho^- u^-(u^- - s) + p(\rho^-) = \rho^+ u^+(u^+ - s) + p(\rho^+) \quad =: \quad \pi. \tag{3.8}$$

This ansatz leads to an algebraic relation between the velocity $\tilde{u}$ and the density $\tilde{\rho}$ and results in a second order ordinary differential equation for $\tilde{\rho}$. For notational simplicity we omit the tilde symbol

$$\lambda\left(\rho\rho'' - \frac{1}{2}(\rho')^2\right) = \frac{\varepsilon m}{\rho^2}\rho' + \frac{m^2}{\rho} + p(\rho) + ms - \pi(\rho^-), \tag{3.9}$$

where $\pi(\rho^-)$ is defined by the relation

$$\pi(\rho^-) = \frac{m^2}{\rho^-} + ms + p(\rho^-). \tag{3.10}$$

Here we have used definitions (3.7) and (3.8).

3.2.1 Existence of Traveling Wave Solutions for a modified System

In [12], [11] the existence of propagating planar phase boundaries was proven for the modified Navier-Stokes-Korteweg system

$$\begin{aligned} \rho_t + (\rho u)_x &= 0, \\ (\rho u)_t + (\rho u^2 + p(\rho))_x &= \varepsilon u_{xx} - \lambda v_{xxx}, \end{aligned} \tag{3.11}$$

where $v = \frac{1}{\rho}$ denotes the specific volume. The difference to the unmodified system (3.4) is the Korteweg part of the stress tensor. The Korteweg term in (3.11) is the one that usually appears in Lagrangian coordinates, see [106]. It is not clear if this term has any physical relevance in Eulerian coordinates.

The traveling wave solution ansatz (3.5), (3.6) leads to the profile equation

$$\lambda v'' = \varepsilon m v' - m^2 v - \tilde{p}(v) + \tilde{\pi}(v^-), \tag{3.12}$$

$$v(\pm\infty) = v^{\pm}, \quad v'(\pm\infty) = 0, \tag{3.13}$$

with the definitions

$$\tilde{p}(v) = p\left(\frac{1}{v}\right), \quad \tilde{\pi}(v^-) = \tilde{p}(v^-) + m^2 v^-.$$

The proof for the existence of solutions including phase transitions of (3.12), (3.13) in [12], [11] is split into two parts. In the first part the existence of profiles is shown in the case where the viscosity ε is equal to zero. The second part extends this to small viscosity $\varepsilon > 0$. We summarize these results below.

Lemma 3.2.1 *(Benzoni-Gavage)*
Let $\varepsilon = 0$. Then there exists a constant $m_0 > 0$ such that for all $m \in (-m_0, m_0)$ there exist left (vapor) and right (liquid) states $v^-(m)$ and $v^+(m)$ in neighborhoods of the Maxwell states and a solution of (3.12), (3.13) that connects $v^- = v^-(m)$ with $v^+ = v^+(m)$. This profile is unique up to translation.

Theorem 3.2.2 *(Benzoni-Gavage)*
There exists a $m_0 > 0$ and an $\varepsilon_0 > 0$ such that for $(m, \varepsilon) \in (-m_0, m_0) \times (0, \varepsilon_0)$ there exist left (vapor) and right (liquid) states $v^-(m, \varepsilon)$ and $v^+(m, \varepsilon)$ in neighborhoods of the Maxwell states and a solution of (3.12), (3.13) that connects $v^- = v^-(m, \varepsilon)$ with $v^+ = v^+(m, \varepsilon)$. This profile is unique up to translation.

Besides the existence of solutions for equation (3.12) in [12] it is also proven that traveling phase boundaries of the modified system (3.11) have certain stability properties in the case $\varepsilon > 0$.

3.2.2 Possible Extension to the unmodified System

The next lemma shows that the unmodified version of the NSK system has traveling wave solutions in the case where the viscosity is equal to zero, i.e., (3.9) has heteroclinic orbits.

Lemma 3.2.3
Let $\varepsilon = 0$. Then there exists a constant $m_0 > 0$ such that for all $m \in (-m_0, m_0)$ exist unique left (vapor) and right (liquid) states $\rho^-(m) \in (0, \overline{\rho}_v)$ and $\rho^+(m) \in (\underline{\rho}_l, b)$ and a heteroclinic orbit of (3.9) that connects $\rho^-(m)$ with $\rho^+(m)$ and is unique up to translation.
For $m \to 0$ the left and right states converge to the Maxwell states, i.e.,

$$\lim_{m\to 0} \rho^-(m) = \rho_v^M, \quad \lim_{m \to 0} \rho^+(m) = \rho_l^M.$$

Note: It is also possible to choose the left state ρ^- in the liquid phase and the right state ρ^+ in the vapor phase and Lemma 3.2.3 is also valid in this case.

Proof. We multiply equation (3.9) by $\rho'\rho^{-2}$. Since viscosity coefficient ε is equal to zero this results in

$$\frac{\lambda}{2}\left((\rho')^2\rho^{-1}\right)' = \left[m^2\left(\frac{1}{\rho^3} - \frac{1}{\rho^2\rho^-}\right) + \frac{p(\rho) - p(\rho^-)}{\rho^2}\right]\rho'$$

and integrating this equation from $-\infty$ to t, using $\rho(-\infty) = \rho^-$ and the transformation formula we get

$$\frac{\lambda}{2}\frac{\rho'(t)^2}{\rho(t)} = \phi(m, \rho^-, \rho(t)),$$

where the function ϕ is defined by the relation

$$\phi(m, \rho^-, \rho) = \int_{\rho^-}^{\rho} m^2\left(\frac{1}{s^3} - \frac{1}{s^2\rho^-}\right) + \frac{p(s) - p(\rho^-)}{s^2}\, ds.$$

We show that for sufficiently small m there exist unique states $\rho^-(m)$ and $\rho^+(m)$ close to the Maxwell states such that $\phi(m, \rho^-(m), \rho^+(m)) = 0$ is satisfied.

We define a function $F : \mathbb{R} \times \mathbb{R}_{>0} \times \mathbb{R}_{>0} \to \mathbb{R}^2$ by

$$F(m, \rho^-, \rho^+) = \begin{pmatrix} m^2\left(\frac{1}{\rho^+} - \frac{1}{\rho^-}\right) + p(\rho^+) - p(\rho^-) \\ \phi(m, \rho^-, \rho^+) \end{pmatrix}.$$

For $m = 0$, $\rho^- = \rho_v^M$ and $\rho^+ = \rho_l^M$ we have

$$F(0, \rho_v^M, \rho_l^M) = 0$$

by lemma A.3.2 and for the derivative with respect to (ρ^-, ρ^+)

$$D_{(\rho^-, \rho^+)}F(0, \rho_v^M, \rho_l^M) = \begin{pmatrix} -p'(\rho_v^M) & p'(\rho_l^M) \\ p'(\rho_v^M)\left(\frac{1}{\rho_l^M} - \frac{1}{\rho_v^M}\right) & 0 \end{pmatrix}$$

and

$$\det\left(D_{(\rho^-, \rho^+)}F(0, \rho_v^M, \rho_l^M)\right) = p'(\rho_v^M)p'(\rho_l^M)\left(\frac{1}{\rho_v^M} - \frac{1}{\rho_l^M}\right) > 0.$$

Note that the van der Waals pressure function is always a monotonically increasing function in the vicinity of the Maxwell states. Thus, by the implicit function theorem there exists a constant $m_0 > 0$ such that for $|m| < m_0$ we have unique states $\rho^-(m)$ and $\rho^+(m)$ in the neighborhoods of the Maxwell states with

$$\phi(m, \rho^-(m), \rho^+(m)) = 0.$$

By similar arguments as in lemma A.3.3 we can show that for sufficient small m we have

$$\phi(m, \rho^-(m), \rho) > 0 \ \text{ for all } \ \rho \in (\rho^-(m), \rho^+(m)).$$

For small m we set

$$\Phi(\rho) = \sqrt{\frac{2}{\lambda}\rho\phi(m, \rho^-(m), \rho)}.$$

Φ is a strictly positive smooth function on the interval $(\rho^-(m), \rho^+(m))$ and a continuous function on $[\rho^-(m), \rho^+(m)]$. We have $\Phi(\rho^-(m)) = 0$ and $\Phi(\rho^+(m)) = 0$. Hence, for the scalar equation

$$\rho'(t) = \Phi(\rho(t))$$

there exists a heteroclinic profile that connects the states $\rho^-(m)$ and $\rho^+(m)$. This profile is unique up to s shift and monotonically increasing.

Hence, we have the existence and uniqueness (up to s shift) of a heteroclinic profile for equation (3.9) in the case where ε is equal to zero. This completes the proof.

The existence of heteroclinic profiles for equation (3.4) is not proven up to now. The first step for this existence is proven in Lemma 3.2.3. For the second step one could try to apply the Centermanifold theorem as for equation (3.12), see [11]. The numerics in Section 4.2 indicate that profiles exist for this equation. We formulate these speculations as conjecture.

Conjecture 3.2.4
There exists a $m_0 > 0$ and an $\varepsilon_0 > 0$ such that for $(m, \varepsilon) \in (-m_0, m_0) \times (0, \varepsilon_0)$ there exist unique left (vapor) and right (liquid) states $\rho^-(m, \varepsilon) \in (0, \overline{\rho}_v)$ and $\rho^+(m, \varepsilon) \in (\underline{\rho}_l, b)$ and a heteroclinic profile of (3.9) that connects $\rho^-(m, \varepsilon)$ with $\rho^+(m, \varepsilon)$ and is unique up to translation.
For $m \to 0$ the left and right states converge to the Maxwell states, i.e.,

$$\lim_{m \to 0} \rho^-(m, \varepsilon) = \rho_v^M, \quad \lim_{m \to 0} \rho^+(m, \varepsilon) = \rho_l^M.$$

Provided that this conjecture is true we can reformulate it in a form we use in Section 4.2.

Corollary 3.2.5
Let left and right states (ρ^-, u^-) and (ρ^+, u^+) that satisfy the Rankine-Hugoniot relation with density states close to the Maxwell states and small *velocity states be given. Then there exists an $\varepsilon > 0$ such that a profile of (3.9) exists. This profile is unique up to translation.*

3.3 General Solutions

We give a summary of results concerning existence and uniqueness of solutions for the isothermal and full temperature dependent Navier-Stokes-Korteweg model in multiple space dimensions. The results are concerning the Cauchy problem as well as the initial boundary value problem.

Local Existence for the Cauchy Problem

Hattori and Li [57] showed that for sufficiently smooth initial data the Cauchy-Problem ($\Omega = \mathbb{R}^n$, here with $n = 2$) for the isothermal Navier-Stokes-Korteweg system has a (short time) solution.

For the existence result the monotonicity of the pressure p is not required as in other existence results. The main result is the following theorem.

Theorem 3.3.1
For any initial data $(\rho_0, \boldsymbol{u}_0)$ such that the condition $\rho_0 \geq \delta > 0$ is satisfied and $(\rho_0 - \bar{\rho}_0, \boldsymbol{u}_0) \in H^k(\mathbb{R}^2)^3$ for $k \geq 4$, where $\bar{\rho}_0 > 0$ is a constant, there exists a time $T > 0$ such that in $[0, T]$ the Cauchy-Problem for the isothermal Navier-Stokes-Korteweg system (2.55), (2.56) has a unique solution $(\rho, \boldsymbol{u})$ such that $\rho - \bar{\rho}_0 \in L^\infty\left([0,T]; H^{k+1}(\mathbb{R}^2)\right)$ and $\boldsymbol{u} \in L^\infty\left([0,T]; H^k(\mathbb{R}^2)^2\right)$.

Additionally the solution can be estimated by the initial values in some norm, see [57].

Note: The authors state that the same result can be obtained in the three dimensional case.

Global Existence for the Cauchy Problem for the full System

In [58] Hattori and Li give a local in time existence theorem as well as a global existence theorem for *small* initial data for the full temperature dependent Navier-Stokes-Korteweg model (2.34) - (2.36) in three space dimensions with $\Omega = \mathbb{R}^3$, i.e., the Cauchy problem.

For these results some restrictions on the thermodynamic quantities are necessary such as the monotonicity of the pressure in the density. This means that only one phase can exist. The requirements are the following

$$\begin{aligned} p_\rho(\theta, \rho) &> 0, \\ e_\theta(\theta, \rho) &> 0, \\ f_{\theta\theta}(\theta, \rho) &< 0, \end{aligned}$$

for all density values ρ and temperature values θ in the state space. In the above equation p denotes the pressure, e the specific internal energy and f the Helmholtz free energy.

Before we state the main results of [58] we introduce some definitions for notational simplicity. For $T > 0$ we define the Banach space $Y_k(T)$ by

$$Y_k(T) = C\left([0,T),\ H^k(\mathbb{R}^3)\right) \cap L^2\left((0,T),\ H^{k+1}(\mathbb{R}^3)\right)$$

and for functions $\tilde{\rho} \in Y^{k+1}(T)$, $u^i \in Y^k(T)$ for $i = 1,2,3$ and $\tilde{\theta} \in Y^k(T)$ we define

$$\begin{aligned} E_k\left[\tilde{\rho}, \boldsymbol{u}, \tilde{\theta}\right](t) &= \sup_{s\in[0,t]} \left(||\tilde{\rho}(s)||^2_{H^{k+1}(\mathbb{R}^3)} + \sum_{i=1}^{3} ||u^i(s)||^2_{H^k(\mathbb{R}^3)} + ||\tilde{\theta}(s)||^2_{H^k(\mathbb{R}^3)} \right), \\ F_k\left[\tilde{\rho}, \boldsymbol{u}, \tilde{\theta}\right](t) &= \int_0^t \left(||\tilde{\rho}(s)||^2_{H^{k+2}(\mathbb{R}^3)} + \sum_{i=1}^{3} ||u^i(s)||^2_{H^{k+1}(\mathbb{R}^3)} + ||\tilde{\theta}(s)||^2_{H^{k+1}(\mathbb{R}^3)} \right) ds. \end{aligned}$$

For the existence of local in time solutions a smallness assumption on the initial data is not necessary (at least not explicitly stated in [58]). However, the initial density and temperature should be at least positive to be meaningful from the physical point of view.

Theorem 3.3.2 *(Local Existence)*
Let the initial data $(\tilde{\rho}_0 + \bar{\rho},\ \boldsymbol{u}_0,\ \tilde{\theta}_0 + \bar{\theta})$ satisfy for some $k \in \mathbb{N}$ with $k \geq 3$

$$\begin{aligned} \tilde{\rho}_0 &\in H^{k+1}(\mathbb{R}^3), \\ u_0^i &\in H^k(\mathbb{R}^3) \quad \text{for } i = 1,2,3, \\ \tilde{\theta}_0 &\in H^k(\mathbb{R}^3), \end{aligned} \tag{3.14}$$

where $\bar{\rho}$ and $\bar{\theta}$ are some positive constants. Then there exists a time $T > 0$ such that we have a unique solution $(\tilde{\rho} + \bar{\rho}, u^1, u^2, u^3, \tilde{\theta} + \bar{\theta})$ of the temperature dependent Navier-Stokes-Korteweg system (2.34) - (2.36) with $\tilde{\rho} \in Y^{k+1}(T)$, $u^i \in Y^k(T)$ for $i = 1,2,3$ and $\tilde{\theta} \in Y^k(T)$.

In contrast to the theorem above for the global existence result a smallness assumption on the initial data is necessary.

Theorem 3.3.3 *(Global Existence)*
Let the initial data satisfy (3.14) for some $k \in \mathbb{N}$ with $k \geq 3$. Then there exist positive constants ε_0 and C_0 such that for $E_k\left[\tilde{\rho}_0, \boldsymbol{u}_0, \tilde{\theta}_0\right](0) \leq \varepsilon_0$ we have a unique global solution $(\tilde{\rho} + \bar{\rho}, u^1, u^2, u^3, \tilde{\theta} + \bar{\theta})$ of the temperature dependent Navier-Stokes-Korteweg system (2.34) - (2.36) with $\tilde{\rho} \in Y^{k+1}(\infty)$, $u^i \in Y^k(\infty)$ for $i = 1,2,3$ and $\tilde{\theta} \in Y^k(\infty)$ and the solution satisfies the estimate

$$E_k\left[\tilde{\rho}, \boldsymbol{u}, \tilde{\theta}\right](t) + F_k\left[\tilde{\rho}, \boldsymbol{u}, \tilde{\theta}\right](t) \leq C_0 E_k\left[\tilde{\rho}_0, \boldsymbol{u}_0, \tilde{\theta}_0\right](0) \quad \textit{for } \ t \geq 0.$$

Global Existence of Weak Solutions

In [17] the isothermal Navier-Stokes-Korteweg system in a slightly modified form is considered.

$$\begin{array}{rcrcl} \rho_t & + & \nabla\cdot(\rho\boldsymbol{u}) & = & 0, \\ (\rho\boldsymbol{u})_t & + & \nabla\cdot(\rho\boldsymbol{u}\boldsymbol{u}^T)+\nabla p(\rho) & = & \nabla\cdot(\hat{\boldsymbol{\tau}}+\boldsymbol{K})\,. \end{array} \tag{3.15}$$

Here the viscous part of the stress tensor $\hat{\boldsymbol{\tau}}$ differs from the previous definition of the viscous stress. $\hat{\boldsymbol{\tau}}$ is given by the relation

$$\hat{\boldsymbol{\tau}} = \nu\rho\left(\nabla\boldsymbol{u}+\nabla\boldsymbol{u}^T\right).$$

The modification is done mainly for technical reasons and not for physical motivation. Bresch, Desjardins and Lin [17] proved the global existence of *weak solutions* in a periodic domain without the restriction of smallness of initial data. However, the proof requires that we have for the pressure

$$p_\rho(\rho) \geq 0$$

for all density values ρ, i.e., the fluid does not undergo phase transition. The main result is the following theorem.

Theorem 3.3.4
Let the space dimension be $n=2$ or $n=3$. Then there exists a global weak solution *$(\rho,\boldsymbol{u})$ of equation (3.15).*

For the definition of weak solutions of equation (3.15) see [17].

Existence for the Initial Boundary Value Problem

Kotschote proved in [74] the local existence and uniqueness of a solution of the initial boundary value problem (2.55), (2.56), (2.51), (2.53) for the isothermal Navier-Stokes-Korteweg system. The monotonicity of the pressure is not required. We summarize the main result in the following theorem.

Theorem 3.3.5
Let $\Omega\subset\mathbb{R}^n$ be an open bounded domain with C^3-boundary and $n+2<p<\infty$. Let the initial data satisfy the following regularity and compatibility conditions

- $\boldsymbol{u}_0 \in B_{pp}^{2-\frac{2}{p}}(\Omega;\mathbb{R}^n)$, $\quad \rho_0\in B_{pp}^{3-\frac{2}{p}}(\Omega)$, *$\rho_0>0$ in $\overline{\Omega}$ (regularity),*
- $\boldsymbol{u}_0=\boldsymbol{0}$ *in* $B_{pp}^{2-\frac{2}{p}}(\partial\Omega;\mathbb{R}^n)$, $\quad \nabla\rho_0\cdot\boldsymbol{n}=0$ *in* $B_{pp}^{2-\frac{3}{p}}(\partial\Omega)$ *(compatibility of the initial data with the boundary conditions).*

Then it exists a $T>0$ such that the initial boundary value problem (2.55), (2.56), (2.51), (2.53) has a unique solution $(\rho,\boldsymbol{u})$ with

$$\begin{array}{rcl} \rho & \in & C^{\frac{3}{2}}\left((0,T);\;C^1(\Omega)\right)\cap C\left((0,T);\;C^3(\Omega)\right) \\ \boldsymbol{u} & \in & C^1\left((0,T);\;C(\Omega;\mathbb{R}^n)\right)\cap C\left((0,T),\;C^3(\Omega)\right). \end{array}$$

In [74] the theorem is formulated more general. The capillarity and viscosity coefficients may depend on time and p has not necessarily to be given by the van der Waals equation of state.

In [73] the author proved a similar result for a temperature dependent model. But the model that is considered is not exactly the same the temperature dependent Navier-Stokes-Korteweg system (2.34), (2.35) and (2.36) because some of the terms are missing there.

Chapter 4

Construction of Solutions and Benchmarks

The purpose of this chapter is the construction of solutions of the NSK system and other benchmark tests for three different reasons. These are

- the validation of the numerical schemes. Therefore, we construct initial configurations such as static equilibrium and traveling wave solutions.
- the identification of physical parameters such as surface tension and interface width.
- the validation of the model.

Due to the complexity of the model it seems to be out of scope to give analytical solutions. Thus, we seek for solutions of special form such that the resulting equation reduces to an ordinary differential equation equipped with suitable boundary conditions. These kind of problems can be solved very accurately.

These solutions are used to identify physical relevant parameters such as surface tension and the size of the diffuse interface.

For the validation of the model we choose the physical experiment of an oscillating bubble in a liquid. When mass transfer over the interface is neglected and the liquid is *nearly incompressible* an equation (the classical Rayleigh-Plesset equation) for the radius of the bubble can be derived from the incompressible Navier-Stokes equations equipped with suitable boundary conditions. The behavior of an oscillating bubble as a solution of the Navier-Stokes-Korteweg system can then be compared to solutions of the Rayleigh-Plesset equation. However, at this point it is not clear if this is really comparable because of the assumption of incompressibility and the assumption of absence of mass transfer. These effects have to be *small* to be comparable.

4.1 Static Equilibrium

In this section we will construct radial symmetric static equilibrium solutions to the isothermal Navier-Stokes-Korteweg system by means of solving an ordinary boundary value problem numerically. These kind of solutions serve as benchmarks for numerical algorithms on the one hand and for determination of the coefficient λ, that is related to surface tension, on the other hand. Theorem 3.1.3 provides an asymptotic formula for surface tension that is implicitly included in the Navier-Stokes-Korteweg model by the third order term. With the numerical computations in this section we approve that this formula is correct and the error term in negligible for our numerical simulations. Since the solutions we construct in this section do not touch the boundary they are only local minimizers of the energy functional investigated in theorem 3.1.3 whereas this theorem makes a statement about global minimizers. However, this should not make a difference.

A static equilibrium solution of the isothermal Navier-Stokes-Korteweg equations, i.e., a solution with zero-velocity field and density independent of time satisfies the elliptic equation

$$\mu(\rho) - \lambda\Delta\rho = cst \text{ in } \Omega, \tag{4.1}$$

where cst is a constant which is unknown in general (Lagrange multiplier). Now let $\Omega \subset \mathbb{R}^n$ be a ball of radius L with the origin as center. A radial symmetric solution (with respect to the origin) $\rho = \rho(r)$ of (4.1) fulfills the equation

$$\mu(\rho) - \lambda\left(\rho_{rr} + \frac{n-1}{r}\rho_r\right) = cst.$$

In order to get rid of the unknown constant cst we differentiate this equation with respect to r. This gives the third order ODE

$$\rho_{rrr} = \left(\frac{\mu'(\rho)}{\lambda} + \frac{n-1}{r^2}\right)\rho_r - \frac{n-1}{r}\rho_{rr} \text{ in } (0, L). \tag{4.2}$$

Thus, we require three boundary conditions. The first one is the boundary condition (2.53)

$$\rho_r(L) = 0. \tag{4.3}$$

The second boundary condition ensures smoothness at the origin

$$\rho_r(0) = 0. \tag{4.4}$$

The third one ensures that the Young-Laplace law is satisfied

$$p(\rho(L)) - p(\rho(0)) = \xi, \tag{4.5}$$

where $\xi > 0$ is some *suitable chosen* constant. Apriori, we do not know the radius R of the bubble (or drop) we compute. After the computation of such a profile ρ we have the radius available but we have to define what the radius of a bubble or drop with a diffuse interface is.

Note, there is some arbitrariness in the definition of the radius because of the diffuse interface. We distinguish between bubbles and drops and define their radiuses R_B and R_D by

$$\begin{aligned} R_B &= \sup\{r \in (0,L) \mid \rho(r) \leq \hat{\rho}\}, \\ R_D &= \inf\{r \in (0,L) \mid \hat{\rho} \leq \rho(r)\}, \end{aligned}$$

where $\hat{\rho}$ can be chosen as the arithmetic average of the *phase boundaries* $\overline{\rho}_v$, $\underline{\rho}_l$ or as the uniquely defined inflection point of the pressure function.

Now, using some definition of the radius R, we can calculate the surface tension σ that is associated with the parameter λ by the Young-Laplace relation

$$\xi = (n-1)\frac{\sigma}{R}.$$

Further we define the diffuse interface to consist of the density values of the elliptic region and possibly a little bit more. Then the interface width w for bubbles and drops is defined by

$$\begin{aligned} I &= \{r \in (0,L) \mid \hat{\rho}_v \leq \rho(r) \leq \hat{\rho}_l\}, \\ w &= \sup I - \inf I. \end{aligned}$$

The definition of the interface I depends on the definition of the density states $\hat{\rho}_v$ and $\hat{\rho}_l$. These states can be defined in terms of the phase boundary states $\overline{\rho}_v$, $\underline{\rho}_l$ or as a fraction of the Maxwell states. The latter seems to be the better choice.

The boundary value problem (4.2), (4.3) - (4.5) can be solved with every solver for nonlinear ordinary boundary value problems. We use the COLNEW solver [6]. The existence and uniqueness of solutions of the nonlinear ordinary boundary value problem is not discussed in this work. However, the numerics indicate that unique solutions exist for *suitable* chosen parameters.

4.1.1 Computation of Static Bubbles and Drops

We define the bubble radius by the inflection point of the pressure function. Then the radius of the bubble and the drop are equal to each other and the radius can be defined by the intersection point of both profiles. The parameters are chosen as follows

$$\begin{aligned} n &= 3, \\ L &= 1, \\ \lambda &= 0.001, \\ \xi &= 0.115. \end{aligned}$$

All quantities are dimensionless, as equation of state we choose the dimensionless van der Waals equations of state (2.13), (2.14) for the pressure and the chemical potential with reference temperature $\theta_{ref} = 0.85$. The result (bubble or drop) depends on the initial guess we have to provide for the BVP-solver. Figure 4.1 shows both results, i.e., the density profiles for the bubble and the drop. The computations give as results for the radius R, surface tension σ and interface width w

$$\begin{aligned} R &= 0.284, \\ \sigma &= 0.016, \\ w &= 0.121. \end{aligned}$$

For the definition of the radius we choose the state $\hat{\rho}$ to be equal to the inflection point of the pressure function. Further we have to define the states $\hat{\rho}_v$ and $\hat{\rho}_l$ for the definition of the interface width w. We define these states by a fraction of the Maxwell states

$$\hat{\rho}_v = 1.1 \cdot \rho_v^M, \quad \hat{\rho}_l = 0.9 \cdot \rho_l^M.$$

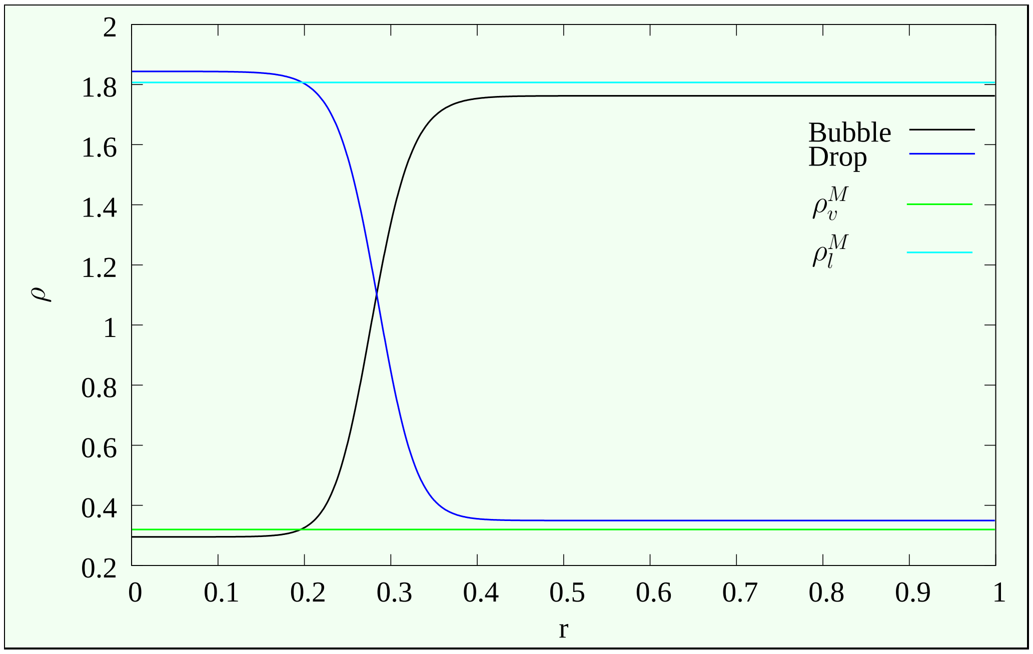

Figure 4.1: Profiles of a bubble and a drop and the Maxwell states.

4.1.2 Computation of Surface Tension and Interface Width

We define the radius and the interface of a bubble as in the previous section. For different reference temperatures and different values of the coefficient λ we compute profiles of bubbles for $n = 3$ and radius of the domain $L = 1$. As we have the profile we can determine the surface tension from this computation denoted by σ_{comp} by the formula

$$p(\rho(L)) - p(\rho(0)) = \frac{n-1}{R}\sigma_{comp}. \tag{4.6}$$

Theorem 3.1.3 and equation (3.3) give an asymptotic formula for the surface tension that includes an error term $e(\lambda)$ that we want to determine in this section.

$$p(\rho(L)) - p(\rho(0)) = \frac{n-1}{R}\sigma_{form} + e(\lambda), \tag{4.7}$$

where σ_{form} is given by $\sigma_{form} = c_0\sqrt{\lambda}$, see Section 3.1. Now we can use equation (4.6) and equation (4.7) to determine the error term since we have

$$|e(\lambda)| = \frac{n-1}{R}|\sigma_{comp} - \sigma_{form}|. \tag{4.8}$$

The left part of Figure 4.2 shows the dependence of σ_{form} (solid line) and σ_{comp} (discrete points) on the parameter λ for different values of the temperature θ. The error $|e(\lambda)|$ is shown in Table 4.1 for some of this parameters. It can clearly be seen that the error converges to zero as λ tends to zero and the error is almost negligible even for relatively large interfaces as the one in Figure 4.1. Thus, Formula (3.3) is applicable for our numerical simulations.

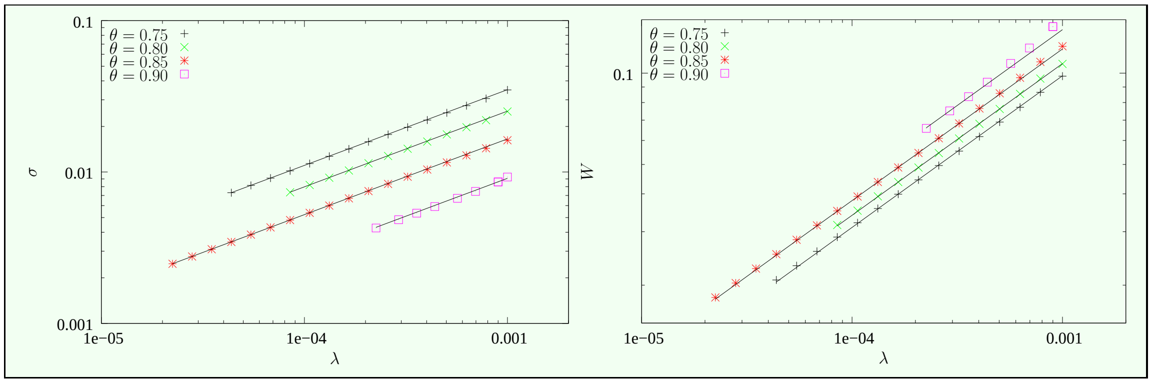

Figure 4.2: Coefficient λ versus surface tension (left) and λ versus interface width (right).

The right part of Figure 4.2 shows the dependence of the interface width on the coefficient λ and the reference temperature. For a temperature θ below the critical temperature we can construct a (rough) formula to approximate the interface width of a dimensionless fluid modeled by the Navier-Stokes-Korteweg equations.

$$w(\theta, \lambda) = 5.4 \cdot \theta^2 \cdot \sqrt{\lambda}. \tag{4.9}$$

This formula is simply obtained by curve fitting using the computed values shown in the right part of Figure 4.2. As noted before, this is a very rough formula but can be useful to construct initial data.

4.2 Traveling Wave Solutions

We compute traveling wave solutions of the isothermal Navier-Stokes-Korteweg system that are supposed to exist. The existence of such solutions is only completely proven

	$\theta = 0.75$		$\theta = 0.85$	
λ	error	EOC	error	EOC
7.8472e-04	3.9974e-03		3.0457e-03	
6.2835e-04	2.8642e-03	1.500	2.2464e-03	1.370
5.0314e-04	2.0633e-03	1.476	1.6292e-03	1.445
4.0288e-04	1.4888e-03	1.469	1.1750e-03	1.471
3.2260e-04	1.0723e-03	1.477	8.7254e-04	1.339
2.5831e-04	7.7166e-04	1.480	6.2802e-04	1.480
2.0684e-04	5.5450e-04	1.487	4.5193e-04	1.481
1.6562e-04	3.9157e-04	1.565	3.1612e-04	1.608
1.3262e-04	2.7951e-04	1.517	2.2839e-04	1.463
1.0619e-04	2.0384e-04	1.421	1.6636e-04	1.426
8.5032e-05	1.4618e-04	1.496	1.1965e-04	1.483
6.8088e-05	1.0768e-04	1.376	8.7439e-05	1.411
5.4520e-05	7.5755e-05	1.582	6.2637e-05	1.501
4.3656e-05	5.6876e-05	1.290	4.5604e-05	1.428

Table 4.1: Error and EOC.

for a modified system. For the unmodified system we have proven only the first step (without viscosity) in Section 3.2. However, without viscosity these profiles suffer a lack of stability and are therefore useless for quantitative benchmark tests. The numerical computations below indicate that even with viscosity these kinds of solutions exist but the existence is not proven theoretically, see Section 3.2 especially Conjecture 3.2.4 and corollary 3.2.5.

We consider the Navier-Stokes-Korteweg system in one space dimension

$$\begin{aligned} \rho_t + (\rho u)_x &= 0, \\ (\rho u)_t + (\rho u^2 + p(\rho))_x &= \varepsilon u_{xx} + \lambda(\rho\rho_{xx} - \tfrac{1}{2}\rho_x^2)_x, \end{aligned} \tag{4.10}$$

and we are interested in traveling wave solutions of (4.10), i.e. smooth solutions of the form

$$\begin{aligned} \rho(x,t) &= \tilde{\rho}(x - st), \\ u(x,t) &= \tilde{u}(x - st), \end{aligned}$$

that connects left states (ρ^-, u^-) and right states (ρ^+, u^+) in different phases that satisfy the Rankine-Hugoniot relation and propagate with a constant speed s ($'$ denotes the derivative with respect to $x - st$)

$$\tilde{\rho}(\pm\infty) = \rho^\pm, \quad \tilde{u}(\pm\infty) = u^\pm, \quad \tilde{\rho}'(\pm\infty) = 0. \tag{4.11}$$

This ansatz leads to a second order ODE for $\tilde{\rho}$ that we write as a system of first order equations

$$\begin{pmatrix} \tilde{\rho} \\ \tilde{\rho}' \end{pmatrix}' = F(\tilde{\rho}, \tilde{\rho}') := \begin{pmatrix} \tilde{\rho}' \\ \frac{1}{\lambda\tilde{\rho}}\left(\frac{\lambda}{2}(\tilde{\rho}')^2 + \varepsilon\frac{m}{\tilde{\rho}^2}\tilde{\rho}' + \frac{m^2}{\tilde{\rho}} + p(\tilde{\rho}) + ms - j\right) \end{pmatrix}, \tag{4.12}$$

with some known constants m and j coming from the Rankine-Hugoniot relation. For ρ^- and ρ^+ chosen close to the Maxwell states traveling wave solutions may exist but the parameters λ and ε have to satisfy a special ratio depending on the left and right states. see Section 3.2 especially Conjecture 3.2.4 and corollary 3.2.5. This means if we fix left and right hand states and the parameter λ we have to compute the parameter ε such that a traveling wave solution can exist. For this purpose we add the equation

$$\varepsilon' = 0. \tag{4.13}$$

For the numerical computation we have to truncate the interval $(-\infty, \infty)$ to some finite interval (τ^-, τ^+) and introduce suitable boundary conditions, we apply the method introduced in [43] and successfully applied in [44], [39]. An exact solution of (4.12), (4.11) has to satisfy

$$\begin{pmatrix} \tilde{\rho} \\ \tilde{\rho}' \end{pmatrix}(\tau^-) \in W_u(\rho^-, 0) \quad \text{and} \quad \begin{pmatrix} \tilde{\rho} \\ \tilde{\rho}' \end{pmatrix}(\tau^+) \in W_s(\rho^+, 0),$$

where W_u and W_s denote the local unstable and stable manifolds of F that are one dimensional manifolds when ρ^- and ρ^+ are close to the Maxwell states. The computation of the unstable and stable manifolds is as difficult as the computation of the traveling wave solution itself but they can be approximated by their tangent spaces and the tangent spaces can be determined by the eigenspaces of the Jacobian of F. Hence, we introduce the boundary conditions

$$\begin{pmatrix} \tilde{\rho} \\ \tilde{\rho}' \end{pmatrix}(\tau^-) \in T_{(\rho^-,0)} W_u(\rho^-, 0) \quad \text{and} \quad \begin{pmatrix} \tilde{\rho} \\ \tilde{\rho}' \end{pmatrix}(\tau^+) \in T_{(\rho^+,0)} W_s(\rho^+, 0). \tag{4.14}$$

If $\tilde{\rho}$ is a solution of (4.12), (4.11) then $\tilde{\rho}(\cdot + \xi)$ is also one for all $\xi \in \mathbb{R}$. We single out one of these solutions by the relation

$$\int_{\tau^-}^{\tau^+} \tilde{\rho}(\tau) - \rho^*(\tau) d\tau = \xi, \tag{4.15}$$

where ρ^* is a reference object, for example the jump from ρ^- to ρ^+. Now we have three equations (4.12), (4.13) and three boundary conditions (4.14), (4.15). This nonlinear boundary value problem can be solved with every BVP-solver but the crucial part is to find a good initial guess. A smeared out jump is usually a good candidate. For the computations below we have applied the COLNEW BVP-solver [6].

We have computed two different profiles. One belongs to a compressive wave and the other to an undercompressive wave. For the definition of compressive and undercompressive waves see standard textbook such as [34]. The undercompressive wave is considered to be typical for a propagating phase boundary whereas the compressive wave is less typical since phase boundaries usually propagate with a subsonic speed. Figure 4.3 shows both profiles. For the computation the parameter λ is fixed to a constant and the viscosity parameter ε is computed such that a traveling wave solution exist according to the additional equation (4.13), i.e., it is different for every profile. For the two profiles we have the following parameters

$$\begin{aligned} \lambda &= 0.001, \\ \varepsilon_c &= 0.0056977, \quad s_c = -1.25273, \\ \varepsilon_u &= 0.0136644, \quad s_u = -0.32141, \end{aligned}$$

where ε_c, ε_u, s_c and s_u denote the viscosity parameter and the speed of propagation for the compressive and undercompressive profile respectively.

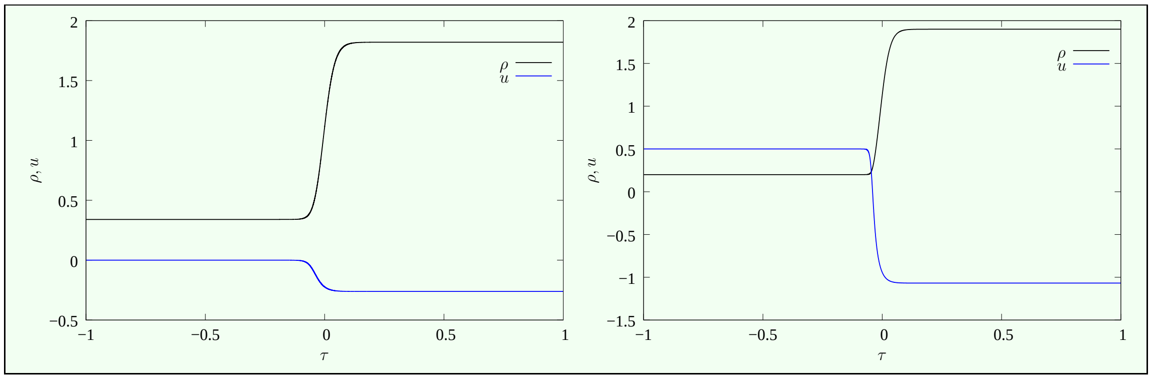

Figure 4.3: Profiles of an undercompressive wave (left) and a compressive wave (right).

4.3 Towards Static Equilibrium

In the two previous sections we have provided exact static equilibrium solutions and planar dynamical solutions. In this section we provide an initial configuration such that the corresponding solution of the NSK system includes multidimensional dynamics, changes in topology and converges to some nontrivial static equilibrium as time tends to infinity. However, it is not possible to give an exact solution that shows such a complicated behavior, but we can construct a configuration consisting of three bubbles of different sizes such that the smaller bubbles vanish and the larger bubble grows and finally converges to a static bubble.

Figure 4.4 illustrates this behavior. The first picture shows the initial data at $t = 0$ consisting of three bubbles (blue) in the liquid (red) with a zero velocity field. This configuration is not a steady state. Hence, we have some dynamical changes shown in the second picture with a nonzero velocity field (not shown). Finally the third picture shows the static equilibrium at $t = \infty$.

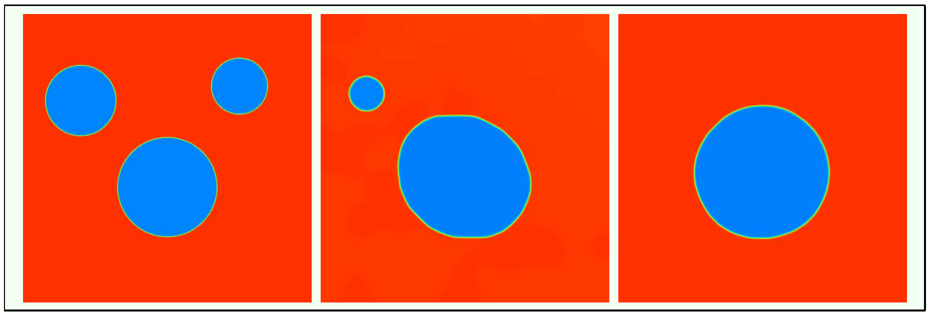

Figure 4.4: Initial configuration of three bubbles of different size, intermediate state with two bubbles and final static equilibrium solution consisting of one large bubble.

We cannot compare a numerical approximation with an exact solution but we can test the approximate solution for

(i) Energy decay on the discrete level as it is satisfied on the continuous level, see Lemma 2.7.1.

(ii) Vanishing of kinetic energy as time tends to infinity.

(iii) The equilibrium condition (4.1) as time approaches infinity.

For the latter test we can monitor the function

$$t \mapsto ||\nabla \left(\mu(\rho(\cdot,t)) - \lambda\Delta\rho(\cdot,t)\right)||_{L^2(\Omega)}$$

which should converge to zero as $t \to \infty$.

Note: The energy decay equation is also satisfied in the case where the computational domain Ω is an n-dimensional cube and the Navier-Stokes-Korteweg system is equipped with periodic boundary conditions.

For the construction of the initial configuration at a given fixed temperature below the critical temperature of the fluid we use the Maxwell values as liquid and vapor states and Formula (4.9) for the width of the interface. Liquid and vapor states are smoothly connected by a smeared out interface using the tanh function.

4.4 Formulas for the Bubble Radius

In contrast to the first section of this chapter we consider spherical symmetric gas (not necessary the vapor of the liquid) bubbles that oscillate in a liquid (instead of staying in equilibrium). The oscillations of the bubble can be caused by pressure perturbations in the liquid or by prescribing the velocity of the liquid at certain points. The former corresponds to the physical application of a sound field, the latter to a variation of the container wall that holds the liquid. In order to derive a simple formula for the time-dependent radius of an oscillating gas bubble we assume the liquid to be incompressible and neglect mass transfer over the interface, i.e., no phase transformation takes place. Therefore the formulas apply only to vapor bubbles if the amount of mass transfer over the interface is small. The goal is, provided that one of the formulas is *applicable*, to (roughly) predict the behavior of an oscillating and/or collapsing bubble in a simple way. For the applicability of these formulas see the numerical experiments and discussion in Section 9.12.

4.4.1 Rayleigh-Plesset Equation

The derivation of the classical Rayleigh-Plesset formula follows that in [16]. We start from the incompressible Navier-Stokes equations and a free boundary condition at the bubble interface, (see standard textbooks, e.g. [79]). The aim is to derive an expression

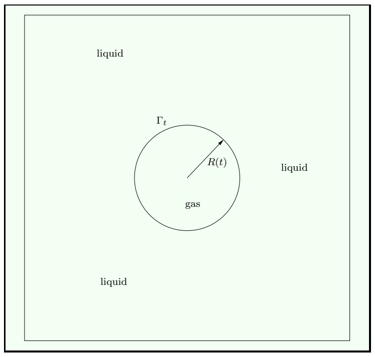

Figure 4.5: Gas bubble in incompressible liquid of radius $R(t)$.

for bubble radius $R(t)$ which will depend on time.

We assume that the motion of the liquid in the domain $\Omega_t = \mathbb{R}^n \backslash \overline{B_{R(t)}(0)}$, $n \geq 2$ obeys the incompressible Navier-Stokes Equations

$$\boldsymbol{u}_t + (\nabla \boldsymbol{u})\boldsymbol{u} + \frac{1}{\rho_l}\nabla p = \frac{\mu_l}{\rho_l}\Delta \boldsymbol{u}, \tag{4.16}$$

$$\nabla \cdot \boldsymbol{u} = 0, \tag{4.17}$$

where $\rho_l > 0, \mu_l > 0$ are the constant density and constant viscosity of the liquid. Further we assume that the Young-Laplace law is satisfied at the free boundary $\Gamma_t = \partial B_{R(t)}(0)$.

$$(\boldsymbol{P}_l - \boldsymbol{P}_g)\boldsymbol{n} = (n-1)\sigma k_m \boldsymbol{n} \quad \text{on } \Gamma_t. \tag{4.18}$$

$\boldsymbol{P}_g = -p_g \boldsymbol{I}$ and $\boldsymbol{P}_l = -p\boldsymbol{I} + \mu\left(\nabla \boldsymbol{u} + \nabla \boldsymbol{u}^T\right)$ are the stress tensors of the gas and liquid phase respectively, $k_m = \frac{1}{R(t)}$ the mean curvature of the free boundary, $\sigma > 0$ the surface tension and p_g the pressure of the gas which is assumed to be rotationally symmetric. We provide rotational symmetric (with respect to the origin) initial values for the velocity and we assume that the solution of the problem stays rotational symmetric for all times $t > 0$. Thus, we seek for rotational symmetric solutions of the incompressible Navier-Stokes equations of the form

$$\boldsymbol{u}(\boldsymbol{x}, t) = v(r,t)\frac{\boldsymbol{x}}{|\boldsymbol{x}|}, \quad p(\boldsymbol{x}, t) = \tilde{p}(r,t), \quad r = |\boldsymbol{x}|,$$

with scalar functions v and $\tilde{p}$. Using this structure of the velocity and the divergence constraint (4.17) we get

$$0 = r^{n-1}\nabla \cdot \boldsymbol{u}(\boldsymbol{x}, t) = r^{n-1}\left(v_r(r,t) + \frac{n-1}{r}v(r,t)\right) = (r^{n-1}v(r,t))_r.$$

This means

$$v(r,t) = \frac{\tilde{v}(t)}{r^{n-1}}. \tag{4.19}$$

for some function $\tilde{v}$ that does not depend on the spatial variable. The momentum equation (4.16) for rotational symmetric solutions reads

$$\left(v_t + vv_r + \frac{1}{\rho_l}\tilde{p}_r - \frac{\mu}{\rho_l}\left[v_{rr} + (n-1)\left(\frac{v_r}{r} - \frac{v}{r^2}\right)\right]\right)\frac{\boldsymbol{x}}{|\boldsymbol{x}|} = \boldsymbol{0},$$

and together with equation (4.19)

$$\frac{\tilde{v}'(t)}{r^{n-1}} - (n-1)\frac{\tilde{v}(t)^2}{r^{2n-1}} + \frac{1}{\rho_l}\tilde{p}_r(r,t) = 0.$$

Note that the viscous term vanishes. For the velocity at the interface we have the relation

$$R'(t) = v(R(t),t) = \frac{\tilde{v}(t)}{R(t)^{n-1}}. \tag{4.20}$$

With this identity and the equation above we get

$$\frac{R(t)^{n-1}R''(t) + (n-1)R(t)^{n-2}R'(t)^2}{r^{n-1}} - (n-1)\frac{\left(R(t)^{n-1}R'(t)\right)^2}{r^{2n-1}} + \frac{1}{\rho_l}\tilde{p}_r(r,t) = 0.$$

Integrating this equation from $R(t)$ to $L > R(t)$ with respect to r gives

$$\begin{aligned}&\left(R(t)^{n-1}R''(t) + (n-1)R(t)^{n-2}R'(t)^2\right)\int_{R(t)}^{L}\tfrac{dr}{r^{n-1}}\\ &\quad -\tfrac{1}{2}(R'(t))^2\left(1 - \left(\tfrac{R(t)}{L}\right)^{2n-2}\right) + \tfrac{1}{\rho_l}\left(\tilde{p}(L,t) - \tilde{p}(R(t),t)\right) = 0.\end{aligned} \tag{4.21}$$

We will replace the term $p(R(t),t)$ using the boundary condition at the interface. With $k_m = \frac{1}{R(t)}$ and equation (4.19) boundary condition (4.18) reduces to

$$\tilde{p}(R(t),t) = \tilde{p}_g(R(t),t) - \sigma(n-1)\frac{1}{R(t)} - 2\mu(n-1)\frac{R'(t)}{R(t)}.$$

Plugging this relation into equation (4.21) we get for space dimension $n = 2$

$$\begin{aligned}&\left(R(t)R''(t) + R'(t)^2\right)\log\left(\frac{L}{R(t)}\right) - \frac{1}{2}R'(t)^2\left(1 - \frac{R(t)^2}{L^2}\right)\\ &\quad = \frac{1}{\rho_l}\left(\tilde{p}_g(R(t),t) - \tilde{p}(L,t) - \sigma\frac{1}{R(t)} - 2\mu\frac{R'(t)}{R(t)}\right),\end{aligned} \tag{4.22}$$

and for $n = 3$

$$\begin{aligned}&\left(R(t)R''(t) + 2R'(t)^2\right)\left(1 - \frac{R(t)}{L}\right) - \frac{1}{2}R'(t)^2\left(1 - \frac{R(t)^4}{L^4}\right)\\ &\quad = \frac{1}{\rho_l}\left(\tilde{p}_g(R(t),t) - \tilde{p}(L,t) - 2\sigma\frac{1}{R(t)} - 4\mu\frac{R'(t)}{R(t)}\right).\end{aligned} \tag{4.23}$$

In the case $n = 3$ and $L >> R(t)$ we can neglect terms in $\left(\frac{R(t)}{L}\right)^k$, $k \geq 1$ and the above equation becomes

$$R(t)R''(t) + \frac{3}{2}R'(t)^2 = \frac{1}{\rho_l}\left(\tilde{p}_g(R(t),t) - \tilde{p}(L,t) - 2\sigma\frac{1}{R(t)} - 4\mu\frac{R'(t)}{R(t)}\right), \qquad (4.24)$$

which is the classical Rayleigh-Plesset equation, see for example [16]. The initial value problem for (4.22), (4.23) and (4.24) can be solved with every ODE solver. It remains to prescribe the pressure of the gas at the interface $\tilde{p}_g$, for example by a barotropic or isothermal equation of state, and the pressure in the liquid, which is the *input* for this equation.

4.4.2 Vibrating Container Wall

We consider a spherical container of radius L that holds the liquid and we assume that the container wall Γ_t^w vibrates symmetrically, i.e., $\Gamma_t^w = \partial B_{L+x(t)}(0)$, where the function x models the movement of the boundary.

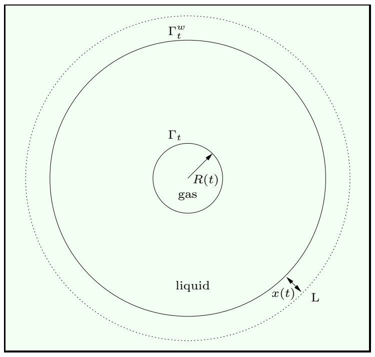

Figure 4.6: Gas bubble inside a vibrating container.

In addition to (4.16), (4.17) and (4.18) we introduce a boundary condition for the velocity at the container wall Γ_t^w

$$\begin{aligned} \boldsymbol{u}\cdot\boldsymbol{n} &= x'(t), \\ \boldsymbol{u}\cdot\boldsymbol{\tau}_i &= 0, \quad i = 1,\dots,n-1, \end{aligned}$$

here $\boldsymbol{\tau}_i$ denote $n-1$ linear independent tangential vectors.

This boundary condition and expression (4.19), which is a consequence of the incompressibility constraint (4.17) give the relation

$$x'(t) = v(L + x(t), t) = \frac{\tilde{v}(t)}{(L + x(t))^{n-1}}.$$

This and equation (4.20) result in the formula

$$R'(t) = \left(\frac{L + x(t)}{R(t)}\right)^{n-1} x'(t). \qquad (4.25)$$

This formula is simply given by the incompressibility constraint and the radius of the bubble does not depend on the state of the gas as in the Rayleigh-Plesset equation. But the force that is necessary to achieve the variation $x(t)$ does. It also depends on the mass of the liquid.

Chapter 5

First Order Accurate Schemes

In this chapter we will construct basic first order schemes for the numerical approximation of solutions of the isothermal Navier-Stokes-Korteweg system. The system itself is a system in divergence form. Therefore one would naturally discretize it in a conservative form. We will see that the discretization in conservative form leads to several problems. On the one hand the appearance of strange velocities inside the interface between the liquid and vapor phases. Similar problems were observed in [64] and solved in [65] by discretizing certain terms in a nonconservative fashion. On the other hand an energy decay with time is not satisfied on the discrete level as it is on the continuous level, see Lemma 2.7.1.

In order to get rid of these problems we will discretize the pressure and the Korteweg term in the system in a nonconservative form. This results in a *well balanced* scheme, i.e., a scheme that is able to preserve a static equilibrium solution on the discrete level. We will see that the approximate solutions generated by this scheme will converge to the correct solution in our test cases and the total energy decays with time on the discrete level as it does on the continuous level. This is the scheme we will generalize to higher order schemes by application of the *Local Discontinuous Galerkin* method in the next chapter.

As a third scheme we present a *relaxation* scheme given in [29], [30]. This scheme is designed to preserve the static equilibrium but the test case with the traveling wave solution shows that it fails to produce the dynamics correctly. This scheme can only be used to construct nontrivial static equilibrium solutions.

Throughout this chapter we make the following assumptions:

- The viscous part in the isothermal Navier-Stokes-Korteweg system $\nabla \cdot \boldsymbol{\tau}$ is equal to $\varepsilon\Delta\boldsymbol{u}$ for simplicity. In fact, the viscous term reduces to $\varepsilon\Delta\boldsymbol{u}$ for a special choice of the viscosity parameters μ and ν, but this choice may not make sense from the physical point of view.

- For notational simplicity we consider only uniform Cartesian meshes. Each coordinate direction is subdivided into N parts. Therefore the mesh consists of N^d cells in total (where d denotes the space dimension). The width of a cell is $h > 0$

in each coordinate direction.

- Unless otherwise noted, we consider periodic boundary conditions in every coordinate direction. The treatment of other boundary conditions is done in the next chapter.

Thus, the system to solve is

$$\begin{array}{rclcl} \rho_t & + & \nabla\cdot(\rho\boldsymbol{u}) & = & 0, \\ (\rho\boldsymbol{u})_t & + & \nabla\cdot(\rho\boldsymbol{u}\boldsymbol{u}^T)+\nabla p(\rho) & = & \nabla\cdot\boldsymbol{K}+\varepsilon\Delta\boldsymbol{u}, \end{array} \quad \text{in } \Omega\times(0,T) \qquad (5.1)$$

and Ω must be an d-dimensional cube because of the restriction on the underlying mesh. For simplicity we always choose the unit cube $\Omega = [-1,1]^d$.

The three schemes we discuss in the next sections belong to the class of Finite Volume schemes. Finite Volume schemes are characterized by their specific numerical fluxes. However, on uniform Cartesian meshes a Finite Volume scheme has an equivalent Finite Difference scheme. In the case of the first two schemes we will use the Finite Difference formulation for simplicity and omit the definition of numerical fluxes. The general Finite Volume formulation of the well balanced scheme (the second scheme) on arbitrary nonconform meshes can be found in the next chapter. For more information on Finite Difference and Finite Volume schemes see standard textbooks such as [51], [52], [76], [81].

5.1 A Scheme in Conservative Form

In this section we construct a basic first order scheme to solve the Navier-Stokes-Korteweg (5.1) system numerically. The resulting conservative scheme is based on the Lax-Friedrichs flux for the first order part of the equation. We have chosen the Lax-Friedrichs flux because it does not require hyperbolicity of the first order part of the equation in the whole state space explicitly as schemes based on Riemann-Solvers or Flux-Vector-Splitting schemes do. The viscous and Korteweg terms in the equation are discretized by central differences in conservative form.
The test cases with the traveling wave solution and the static equilibrium solution indicate that discrete solution converges to the correct solution. However, we observe the appearance of strange velocity fields close to the interface and on the discrete level we do not have an energy decay as for a smooth analytical solution. Similar problems were observed in [64]. In [64] these velocities are called parasitic currents.

The idea of the Lax-Friedrichs scheme is to stabilize the scheme by adding an artificial viscosity that tends to zero with the mesh size h, see standard textbooks such as [52], [76], [81]. Thus, it performs the vanishing viscosity method on the discrete level.

$$\begin{array}{rclcl} \rho_t & + & \nabla\cdot(\rho\boldsymbol{u}) & = & \frac{\alpha\, h}{2}\,\Delta\rho, \\ (\rho\boldsymbol{u})_t & + & \nabla\cdot(\rho\boldsymbol{u}\boldsymbol{u}^T)+\nabla p(\rho) & = & \frac{\alpha\, h}{2}\,\Delta(\rho\boldsymbol{u})+\nabla\cdot\boldsymbol{K}+\varepsilon\Delta\boldsymbol{u}, \end{array} \qquad (5.2)$$

where the parameter α is chosen to be equal to the *fastest wave speed*.

Note: The artificial viscosity in the momentum equation can be omitted when the underlying mesh is fine enough such that the natural viscosity dominates. The artificial viscosity in the continuity equation is important for the convergence to the correct solution (at least in combination with the Discontinuous Galerkin approach). In Section 9.4 we will see that without this viscosity the approximate solution generated by the higher order well balanced scheme (described in the next chapter) does not converge to the correct solution. The same is true when the Lax-Friedrichs type scheme is generalized to higher order schemes.

We present the complete numerical algorithm in one space dimension for simplicity, the extension to two space dimensions is then straightforward.

The Numerical Algorithm in 1d

In the following we consider a uniform mesh of N cells defined by the $N+1$ points

$$x_{-\frac{1}{2}} < x_{\frac{1}{2}} < \ldots < x_{N-\frac{1}{2}}$$

and the (uniform) diameter of a cell is denoted by h. In one space dimension the Korteweg tensor $\boldsymbol{K}$ reduces to the scalar quantity

$$K = \lambda \left(\rho\rho_{xx} - \frac{1}{2}\rho_x^2 \right).$$

We provide discrete initial data by projection

$$\begin{aligned} \rho_i^0 &= \frac{1}{h}\int_{x_{i-\frac{1}{2}}}^{x_{i+\frac{1}{2}}} \rho_0(x)\,dx, \\ (\rho u)_i^0 &= \frac{1}{h}\int_{x_{i-\frac{1}{2}}}^{x_{i+\frac{1}{2}}} (\rho_0 u_0)(x)\,dx, \end{aligned}$$

for $i = 0, \ldots, N-1$. The numerical scheme is then defined by the update procedure from the n-th to the $(n+1)$-th time step.

$$
\begin{aligned}
K_i^n &= \frac{\lambda}{h^2}\left(\rho_i^n(\rho_{i+1}^n - 2\rho_i^n + \rho_{i-1}^n) - \frac{1}{8}(\rho_{i+1}^n - \rho_{i-1}^n)^2\right), \\
u_i^n &= \frac{(\rho u)_i^n}{\rho_i^n}, \\
\rho_i^{n+1} &= \rho_i^n - \frac{\Delta t}{2h}\left((\rho u)_{i+1}^n - (\rho u)_{i-1}^n - \alpha(\rho_{i+1}^n - 2\rho_i^n + \rho_{i-1}^n)\right), \\
(\rho u)_i^{n+1} &= (\rho u)_i^n - \frac{\Delta t}{2h}\left((\rho u)_{i+1}^n u_{i+1}^n - (\rho u)_{i-1}^n u_{i-1}^n + p(\rho_{i+1}^n) - p(\rho_{i-1}^n)\right) \\
&\quad + \frac{\Delta t}{2h}\left(K_{i+1}^n - K_{i-1}^n\right) \\
&\quad + \frac{\Delta t}{2h}\alpha\left((\rho u)_{i+1}^n - 2(\rho u)_i^n + (\rho u)_{i-1}^n\right) \\
&\quad + \frac{\Delta t}{h^2}\varepsilon\left(u_{i+1}^n - 2u_i^n + u_{i-1}^n\right).
\end{aligned}
$$

In the scheme given above we choose α to be equal to the fastest wave speed in the liquid and vapor phases

$$\alpha = \max_i \left\{ |u_i^n \pm \sqrt{p'(\rho_i^n)}| \right\},$$

where the maximum is built only over the values in the liquid and vapor phases since the sound speed is imaginary in the elliptic region and the above statement does not make sense there. The time step size Δt has to be small enough to guarantee stability of the scheme. It is not clear how to choose it exactly but we observed that it is of order $O(h^2)$, similar to the time step size of the relaxation scheme, see Section 5.3 and (5.22), (5.23). The parameter α and the time step size Δt may vary between the time steps. For notational simplicity this dependence is omitted.

Numerical Results

This paragraph is dedicated to numerical tests with the conservative scheme presented above. We apply the first three test cases proposed in Chapter 4. The test case with the undercompressive *Traveling Wave* solution is performed in one space dimension with fixed constant boundary states and the test cases *Static Equilibrium* and *Towards Static Equilibrium* are performed in two space dimensions with periodic boundary conditions. Throughout this chapter the NSK system is equipped with a dimensionless van der Waals equation of state (2.13) where the reference temperature is fixed to $\theta_{ref} = 0.85$. The computational domain in one space dimension is the interval $[-1,1]$ and in two space dimensions the square $[-1,1]^2$.

Test Case: Traveling Wave Solution
For this one dimensional test we have chosen the undercompressive traveling wave solution we computed in Section 4.2. The corresponding parameters are

$$\begin{aligned}\lambda &= 0.001,\\ \varepsilon &= 0.01366,\\ s &= -0.3214,\end{aligned}$$

where s denotes the speed the wave travels with to the left. We compare the values of the approximate density and the momentum with the values of the exact solution at time $T = 0.5$ (which is the profile shifted to the left by $s \cdot T$). For this test we cannot use periodic boundary conditions. Hence, we use the values that come from the exact solution as boundary values. The underlying equidistant meshes vary between $n = 200$ and $n = 1800$ cells.

The approximate solution on the finest grid $n = 1800$ is plotted in Figure 5.1. Differences between exact and numerical solution seem to be small for this mesh size and cannot be seen from the plot.

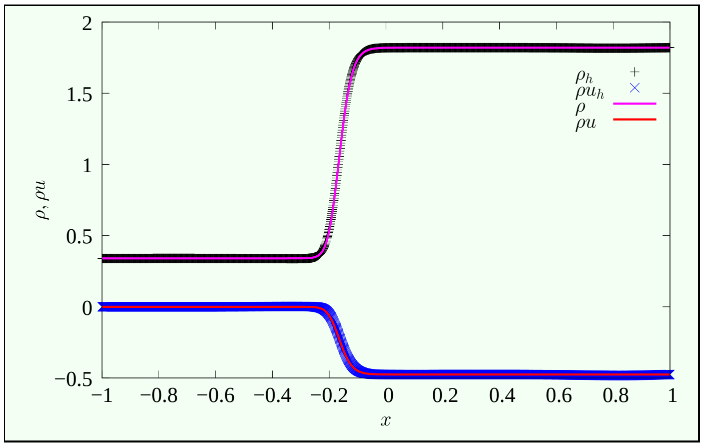

Figure 5.1: Exact and approximate traveling wave solution generated by the conservative scheme for $n = 1800$.

Table 5.1 shows the convergence characteristic of the conservative scheme. Errors in density and momentum are shown separately for different mesh sizes. The EOC (experimental order of convergence) clearly demonstrates first order convergence.

Test Case: Static Equilibrium
For the test with a static equilibrium initial configuration we choose a density profile computed in Section 4.1. For this test the correct computational domain is a ball of radius one with boundary conditions (2.51) and (2.53). Nevertheless we use the square $\Omega = [-1, 1]^2$ as computational domain and apply periodic boundary conditions for simplicity. This should not make a difference since the density values in the liquid near the boundary are equal to some constant (up to the roundoff error). The parameter λ is already chosen by the choice of the density profile. The viscosity parameter is arbitrary. We choose it according to the parameters in the test with the undercompressive wave

	ρ		ρu	
h	L^2-error	EOC	L^2-error	EOC
1.0000e-02	2.4799e-02		4.4079e-02	
5.0000e-03	1.6488e-02	0.589	2.2129e-02	0.994
3.3333e-03	1.2221e-02	0.739	1.4832e-02	0.987
2.5000e-03	9.6790e-03	0.811	1.1173e-02	0.985
2.0000e-03	8.0036e-03	0.852	8.9692e-03	0.984
1.6667e-03	6.8193e-03	0.878	7.4953e-03	0.985
1.4286e-03	5.9388e-03	0.897	6.4395e-03	0.985
1.2500e-03	5.2590e-03	0.911	5.6456e-03	0.985
1.1111e-03	4.7183e-03	0.921	5.0267e-03	0.986

Table 5.1: L^2-errors and EOC for the approximate traveling wave solution generated by the conservative scheme.

above.

$$\begin{aligned} \lambda &= 0.001, \\ \varepsilon &= 0.01366. \end{aligned}$$

The resolution of the $n \times n$ Cartesian meshes varies between $n = 100$ and $n = 800$. The (computational) time at the end of the computation is $T = 20.0$.

Table 5.2 shows the convergence characteristic of the conservative scheme at a static equilibrium configuration. The error seems to be not in the asymptotic regime at these mesh sizes. The EOC should approach the value 1 as h tends to zero.

	density and momentum	
h	total L^2-error	EOC
2.0000e-02	3.8279e-02	
1.0000e-02	1.4162e-02	1.435
6.6667e-03	1.2013e-02	0.406
5.0000e-03	1.0672e-02	0.411
4.0000e-03	9.5017e-03	0.521
3.3333e-03	8.5101e-03	0.605
2.8571e-03	7.6804e-03	0.665
2.5000e-03	6.9851e-03	0.711

Table 5.2: Test Case: Static Equilibrium. Total L^2-error and EOC for the approximate solution generated by the conservative scheme.

In Figure 5.2 the density distribution at computational time $T = 20.0$ is shown. The density values vary approximately between 0.3 (blue) and 1.8 (red). These values are close to the Maxwell values for the chosen equation of state. The velocity field (which is equal to zero for all times in the exact solution) is represented by the black arrows. This display style is used throughout this chapter. The approximate solution is very close to a discrete equilibrium, i.e., there are almost no changes in time, but we can see a velocity field inside the interface that is of order $O(h)$. The scaling of the velocity field is the same in all sub-figures and the sequence of computations with $n = 100,\ 200,\ 400$ shows that this velocity field converges to zero with h.

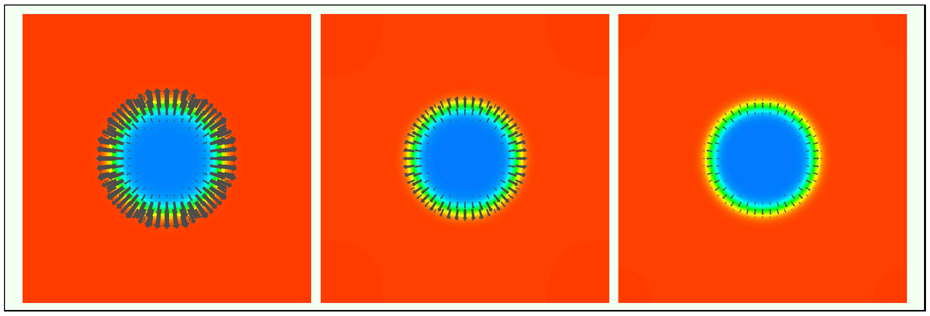
Figure 5.2: Test Case: Static Equilibrium. Density and velocity field produced by the conservative scheme at $T = 20.0$ for $n = 100,\ 200,\ 400$.

The question that arises is how can this be a discrete steady state configuration. So we have to ask why density and momentum are independent of time. For the density this can be seen by rewriting the mass balance equation from equation (5.2) in the form

$$\rho_t + \nabla \cdot \left(\rho \boldsymbol{u} - \frac{\alpha\, h}{2}\, \nabla \rho \right) = 0.$$

The gradient of the density points from the vapor bubble to the liquid phase. Hence, with a velocity field shown in Figure 5.2 the blue term in the above equation cancels out the red term (artificial viscosity) and therefore the density does not depend on time. From the above equation it can clearly be seen that the velocity field inside the interface must be of order $O(h)$. For the momentum this is more complicated. It is essentially due to the structure of the pressure and Korteweg term.

Such a velocity field inside the interface can cause problems especially in the case when an interface is in contact with a solid wall and boundary condition (2.51) is imposed such that the velocity must vanish at the boundary. The velocity field inside the interface can then cause instabilities in the numerical solution since the approximate solution is not consistent with the prescribed boundary condition. Similar velocity fields (so called parasitic currents) were observed in [64]. In [65] it was shown that these parasitic currents can be eliminated when the pressure and Korteweg term are discretized in a nonconservative fashion. The nonconservative discretization is also one of the basic ideas of the well balanced scheme presented in the next section.

Test Case: Towards Static Equilibrium

This is the test case for testing the qualitative behavior of the approximate solutions produced by the numerical schemes such as decay of the total energy, vanishing velocity field and the equilibrium condition at the discrete level. The computational domain is again the periodic square $\Omega = [-1, 1]^2$ and the capillarity and viscosity parameters are chosen as in the last test case $\lambda = 0.001$ and $\varepsilon = 0.01366$. The rest of the setting is as proposed in Section 4.3. The Cartesian $n \times n$ meshes have a resolution of $n =$

100, 200, 400 and the approximate solutions are computed up to computational time $T = 20.0$. For this test case at this time there is still a little bit movement but changes in topology are completed and the solution is not too far from a static equilibrium state.

Figure 5.3 shows the initial data and the approximate solution at times $t = 1.12$ and $t = 20.0$. The two smaller bubbles vanish and the larger on grows as time evolves. Finally the solution approaches an equilibrium state on the discrete level. Again, the rising velocity field inside the interface can clearly be seen.

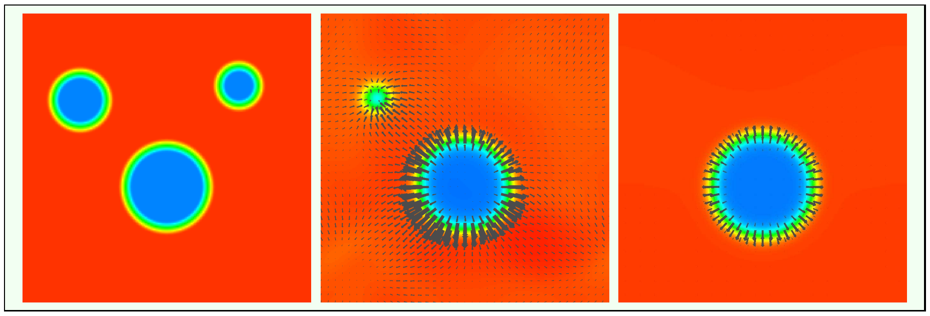

Figure 5.3: Towards static equilibrium test. Density and velocity field produced by the conservative scheme at $t = 0.0,\ 1.12,\ 20.0$ for $n = 200$.

The time dependent behavior of the total energy and the kinetic energy is presented in Figure 5.4 for three computations with different mesh sizes ($n = 100,\ 200,\ 400$). On the discrete level the total energy is not a monotonically decreasing function in time as on the continuous level. But the oscillations are smaller on finer grids such that we can hope for convergence to the exact solution (for $h \to 0$), for which total energy is a decreasing function of time. The right part of the figure shows that the kinetic energy does not converge to zero as time tends to infinity. This is due to the velocity field inside the interface.

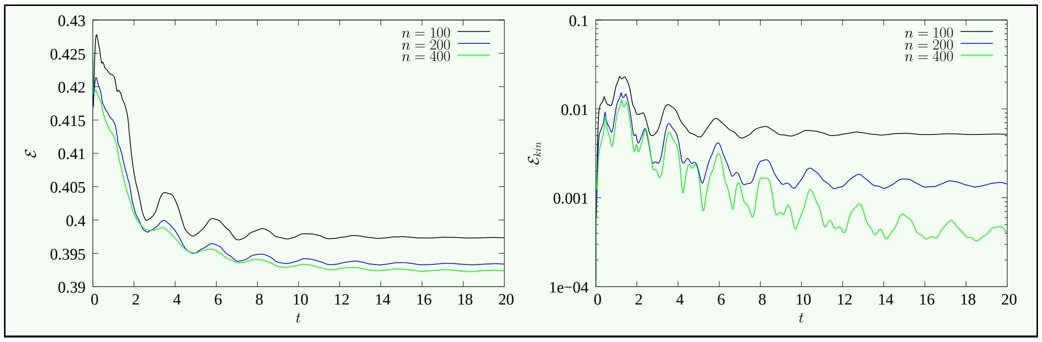

Figure 5.4: Total energy and kinetic energy for the conservative scheme. $n = 100,\ 200,\ 400$.

Finally, the value κ does not approach a constant state as time evolves as it does on the continuous level when the solution approaches a static equilibrium state. This can be seen in figure 5.5 because the gradient of κ does not converge to zero.

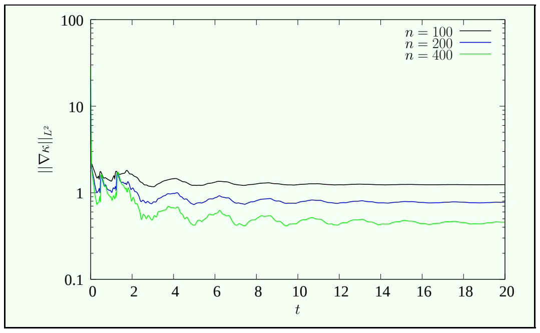

Figure 5.5: Gradient of κ for the conservative scheme. $n = 100,\ 200,\ 400$.

5.2 A Well Balanced Scheme

The problems at static equilibrium configurations we have seen in the previous section were caused by the artificial viscosity (that is necessary to stabilize the numerical solution) on the one hand and on the other hand by the structure of the pressure and Korteweg term in the momentum equation. In this section we discretize these both terms together in nonconservative form by application of the theory of nonconservative products, see [36] and Section A.4. This approach leads in a natural way to a *well balanced* scheme, i.e., a scheme that is able to preserve a static equilibrium solution on the discrete level. In general the application of nonconservative discretizations can cause problems. It is well known that nonconservative schemes can converge to wrong solutions when discontinuities are present [60]. This is not a problem in our case since solutions are supposed to be sufficiently smooth (at least not discontinuous). The test cases show that the numerical solutions converge to the exact solutions and the energy decays on the discrete level. This scheme seems to be the most promising scheme to construct approximate solutions of the NSK-system and therefore we will generalize this scheme to higher order schemes by application of the Discontinuous Galerkin approach in Chapter 6.

The NSK System in Nonconservative Form

The scheme is based on the equivalent nonconservative reformulation of the NSK system

$$\left.\begin{array}{rclcl} \rho_t & + & \nabla\cdot(\rho\boldsymbol{u}) & = & 0, \\ (\rho\boldsymbol{u})_t & + & \nabla\cdot(\rho\boldsymbol{u}\boldsymbol{u}^T) + \rho\nabla\kappa(\rho,\Delta\rho) & = & \varepsilon\Delta\boldsymbol{u}, \end{array}\right\} \text{ in } \Omega\times(0,T), \tag{5.3}$$

where the variable κ is defined by the relation

$$\kappa = \kappa(\rho,\Delta\rho) = \mu(\rho) - \lambda\Delta\rho \tag{5.4}$$

and μ denotes the chemical potential. In order to see that this is an equivalent formulation we refer to Lemma 2.7.2. The idea of the numerical scheme is to add a linear viscosity term scaled with the mesh size to the momentum equation (the same as in the

Lax-Friedrichs type scheme in the previous section) and a nonlinear viscosity combined with a fourth order term

$$\frac{\alpha_1\,h}{2}\,\Delta\kappa = \frac{\alpha_1\,h}{2}\left[\nabla\cdot\left(\frac{p'(\rho)}{\rho}\nabla\rho\right) - \lambda\Delta\Delta\rho\right]$$

to the continuity equation. From the above equation we can see that the nonlinear viscosity has a positive sign in the vapor and liquid phases and the fourth order term has also the correct sign to stabilize the scheme. The resulting system including the artificial viscosity then becomes

$$\begin{aligned} \rho_t &+ \nabla\cdot(\rho\boldsymbol{u}) &= \tfrac{\alpha_1\,h}{2}\,\Delta\kappa, \\ (\rho\boldsymbol{u})_t &+ \nabla\cdot(\rho\boldsymbol{u}\boldsymbol{u}^T) + \rho\nabla\kappa(\rho,\Delta\rho) &= \tfrac{\alpha_2\,h}{2}\,\Delta(\rho\boldsymbol{u}) + \varepsilon\Delta\boldsymbol{u}. \end{aligned} \tag{5.5}$$

The advantage of the nonlinear viscosity in combination with the fourth order term is that it vanishes at static equilibrium because κ is a constant at the static equilibrium, see Lemma 2.7.2. Thus, discretizing the above equation by central differences results in a scheme that preserves the static equilibrium on the discrete level, i.e., it is a well balanced scheme. The parameter α_2 should be chosen to be equal to the *fastest wave speed* (as the parameter α in the previous section) and then parameter α_1 should be chosen such that $\alpha_1\frac{p'(\rho)}{\rho}$ is of the *size of* α_2.

Note: Again, the artificial viscosity in the momentum equation can be omitted when the underlying mesh is fine enough such that the natural viscosity dominates.

Below we give the complete numerical algorithm in one space dimension for simplicity. The scheme is based on space discretization by central differences and application of the explicit Euler scheme for time integration. The extension to two or more space dimensions is straightforward.

The Numerical Algorithm in 1d

As in the previous section we provide discrete initial data by projection. Then the update from one time step to another defines the complete algorithm.

$$
\begin{aligned}
\kappa_i^n &= \mu(\rho_i^n) - \frac{\lambda}{h^2}(\rho_{i+1}^n - 2\rho_i^n + \rho_{i-1}^n), \\
u_i^n &= \frac{(\rho u)_i^n}{\rho_i^n}, \\
\rho_i^{n+1} &= \rho_i^n - \frac{\Delta t}{2h}\left((\rho u)_{i+1}^n - (\rho u)_{i-1}^n - \alpha_1(\kappa_{i+1}^n - 2\kappa_i^n + \kappa_{i-1}^n)\right), \\
(\rho u)_i^{n+1} &= (\rho u)_i^n - \frac{\Delta t}{2h}\left((\rho u)_{i+1}^n u_{i+1}^n - (\rho u)_{i-1}^n u_{i-1}^n\right) \\
&\quad -\frac{\Delta t}{4h}\left((\rho_{i+1}^n + \rho_i^n)(\kappa_{i+1}^n - \kappa_i^n) + (\rho_i^n + \rho_{i-1}^n)(\kappa_i^n - \kappa_{i-1}^n)\right) \\
&\quad +\frac{\Delta t}{2h}\alpha_2\left((\rho u)_{i+1}^n - 2(\rho u)_i^n + (\rho u)_{i-1}^n\right) \\
&\quad +\frac{\Delta t}{h^2}\varepsilon\left(u_{i+1}^n - 2u_i^n + u_{i-1}^n\right),
\end{aligned}
$$

for $i = 0, \ldots, N-1$. Due to the artificial fourth order term scaled by h in the mass balance equation the time step size must be chosen extremely small. It is not clear how small exactly but we observed that it is of order $O(h^3)$. The time step size was determined by successively lowering the time step size until the method was not longer unstable. To overcome this restriction we apply implicit time stepping to the generalized higher order schemes in the following chapters.

Numerical Results

The setting for the numerical tests with the well balanced scheme presented above is exactly the same as for the tests with the conservative scheme.

Test Case: Traveling Wave Solution
The test with the traveling wave solution in one space dimension demonstrates the superiority of the nonconservative well balanced scheme over the conservative scheme. Table 5.3 shows the L^2-errors of density and momentum at time $T = 0.5$. Compared to the errors produced by the conservative scheme the errors associated with the well balanced scheme are an order of magnitude smaller. See also Section 5.4 for a comparison of the schemes. A plot of the numerical solution on the finest grid $n = 1800$ is presented in Figure 5.6.

Test Case: Static Equilibrium
Table 5.4 shows a second order convergence rate but the projection of the initial values itself produces an error of order $O(h)$. The second order rate is due to the use of the midpoint integration formula for initial projection and the computation of the error. At

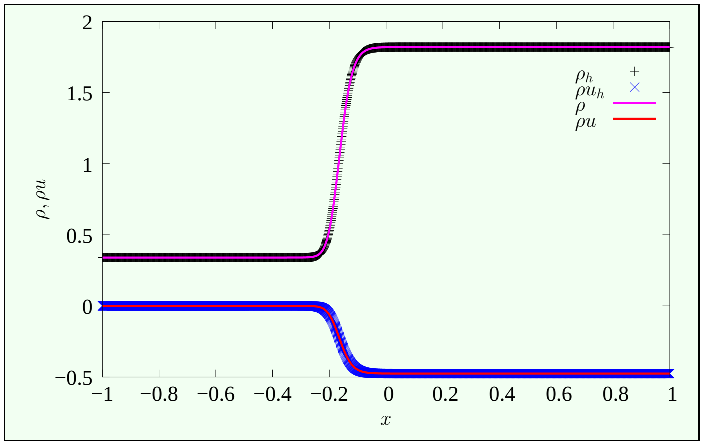

Figure 5.6: Exact and approximate traveling wave solution generated by the nonconservative scheme for $n = 1800$.

	ρ		ρu	
h	L^2-error	EOC	L^2-error	EOC
1.0000e-02	3.2879e-03		1.4845e-03	
5.0000e-03	2.1207e-03	0.633	8.5980e-04	0.788
3.3333e-03	1.5396e-03	0.790	6.1558e-04	0.824
2.5000e-03	1.2063e-03	0.848	4.8070e-04	0.860
2.0000e-03	9.9120e-04	0.880	3.9467e-04	0.884
1.6667e-03	8.4113e-04	0.900	3.3492e-04	0.900
1.4286e-03	7.3050e-04	0.915	2.9095e-04	0.913
1.2500e-03	6.4559e-04	0.925	2.5723e-04	0.923
1.1111e-03	5.7837e-04	0.934	2.3054e-04	0.930

Table 5.3: Test Case: Traveling Wave Solution. L^2-errors and EOC for the approximate solution generated by the well balanced scheme.

the midpoints the scheme produces an error of second order (pointwise). The scheme is designed to preserve the static equilibrium initial values. The projected values are actually not in equilibrium on the discrete level but they are very close to an discrete static equilibrium configuration. Thus, the errors the scheme produces are neglegible. Finally, the time step is very small such that the forward Euler time stepping does not destroy the convergence rate. The use of a higher degree integration formula would show only first order convergence due to the initial projection. This is what we will see in Section 9.1 using higher order schemes on unstructured meshes.

In contrast to the conservative scheme the well balanced scheme does not produce a strange velocity field inside the liquid-vapor interface. This is because the scheme is designed to preserve a static equilibrium configuration on the discrete level. In fact, there is a small velocity field but several orders of magnitude smaller than the velocity field produced by the conservative scheme. A very small velocity arises because the projected initial values are not a discrete equilibrium but very close to one. Thus, some dynamics develop but the velocity converges (up to roundoff error) completely to zero

	density and momentum	
h	total L^2-error	EOC
2.0000e-02	3.7490e-03	
1.0000e-02	9.0510e-04	2.050
6.6667e-03	3.9981e-04	2.015
5.0000e-03	2.2442e-04	2.007
4.0000e-03	1.4349e-04	2.004
3.3333e-03	9.9590e-05	2.003
2.8571e-03	7.3145e-05	2.002
2.5000e-03	5.5990e-05	2.002

Table 5.4: Test Case: Static Equilibrium. Total L^2-error and EOC for the approximate solution generated by the well balanced scheme.

as time tends to infinity. Figure 5.7 shows the density distribution at $T = 20.0$ for the three different mesh sizes ($n = 100,\ 200,\ 400$). The velocity field is also shown but scaled in the same way as for the conservative scheme and therefore it cannot be seen in the figure.

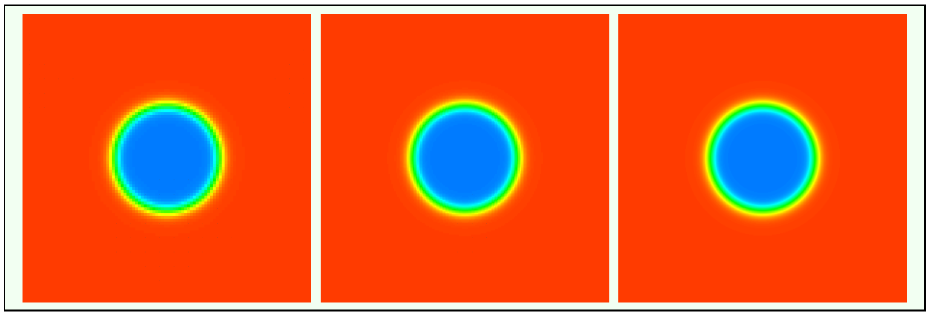

Figure 5.7: Test Case: Static Equilibrium. Density and velocity field produced by the well balanced scheme at $T = 20.0$ for $n = 100,\ 200,\ 400$.

Test Case: Towards Static Equilibrium
This is the test case proposed in Section 4.3. The setting is the same as for the conservative scheme.

Figure 5.8 shows the initial data with zero velocity field and three bubbles at time $t = 0$. The mesh size is the same as in the corresponding test with the conservative scheme ($n = 200$). At time $t = 1.12$ there are only two bubbles left and the smaller one will disappear soon. The velocity field is represented by the black arrows. The scaling of the velocity field is exactly the same as for the conservative scheme in all sub-figures. Finally the solution approaches a static equilibrium. At time $T = 20.0$ (third picture) there is still movement but starting from this point the density distribution will not change essentially as time tends to infinity. In contrast to the conservative scheme there are no nonphysical velocities inside the liquid-vapor interface.

The behavior of the total energy and kinetic energy can be seen in Figure 5.9. The values of the three computations are close to each other such that one graph may hide

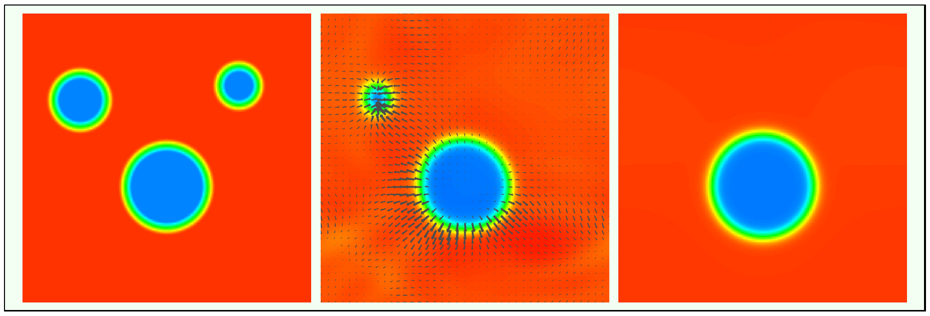

Figure 5.8: Towards static equilibrium test. Density and velocity field produced by the well balanced scheme at $t = 0.0,\ 1.12,\ 20.0$ for $n = 200$.

another graph. The total energy of the discrete solutions are monotonically decreasing functions in time. This is the correct behavior as in the continuous case. The right part of the figure shows an exponential decay of the *mean* kinetic energy. At time $T = 20.0$ there is still a little bit movement in the approximate solution. But as time evolves further, the kinetic energy converges completely to zero up to a roundoff error (this is not shown).

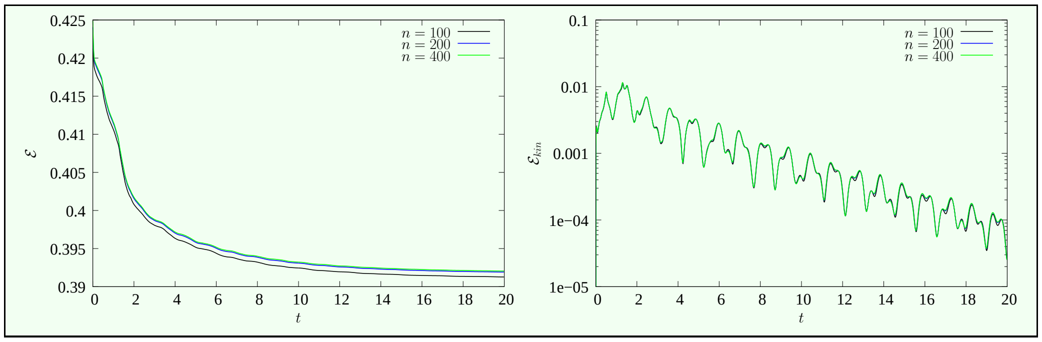

Figure 5.9: Total energy and kinetic energy for the well balanced scheme. $n = 100,\ 200,\ 400$.

In contrast to the conservative scheme the *mean value* of $||\nabla\kappa||_{L^2(\Omega)}$ does not converge to a constant other than zero, see Figure 5.10. Up to time $T = 20.0$ the mean of this value decays exponentially. As time evolves further this value converges to zero (not shown). This means κ converges to a constant as time tends to infinity as in the continuous case when a static equilibrium state is approached.

5.3 A Relaxation Scheme

The goal of this section is to provide an additional numerical scheme in nonconservative form. As noted before it is not possible to apply Riemann-Solver based schemes directly to the NSK system due to the lack of hyperbolicity of the first order part of the equation in the elliptic region. Here we present an approach (given in [29], [30]) that is based on the reformulation of the system as a *relaxation system*. This kind of reformulation

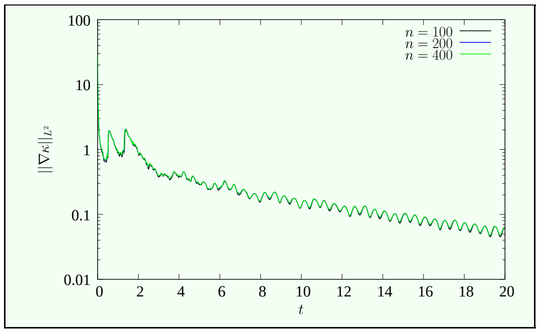

Figure 5.10: Gradient of κ for the well balanced scheme. $n = 100,\ 200,\ 400$.

was first proposed by Suliciu [110] and applied to the equation of gas dynamics in Lagrangian coordinates. The relaxation approach can be very useful for the treatment of complicated pressure laws, see [31]. The idea of the relaxation approach is to add an additional evolution equation for the variable κ that already appeared in the previous section and treat this variable as an *independent* variable. The additional equation is chosen such that the resulting system is hyperbolic and the corresponding Riemann problem can be solved very efficiently. By construction, the scheme is designed to preserve static equilibrium solutions on the discrete level. But the drawback of this scheme is that in general the generated approximative solution does not converge to the correct solution. The test case with the traveling wave solution shows this behavior. However, the scheme can be used to construct solutions towards a static equilibrium configuration.

5.3.1 The Relaxation System

Since the discretization of the term $\varepsilon\Delta\boldsymbol{u}$ is not the source of the difficulties described in Section 5.1 we omit this term for a moment, i.e., we set $\varepsilon = 0$. With the definition of κ in (5.4) we can rewrite the isothermal NSK system (5.1) as

$$\left.\begin{array}{rcrcl} \rho_t & + & \nabla\cdot(\rho\boldsymbol{u}) & = & 0, \\ (\rho\boldsymbol{u})_t & + & \nabla\cdot(\rho\boldsymbol{u}\boldsymbol{u}^T) + \rho\nabla\kappa(\rho,\Delta\rho) & = & 0, \end{array}\right. \quad \text{in } \Omega\times(0,T). \tag{5.6}$$

In the next step we understand $\kappa = \kappa(\boldsymbol{x}, t) \in \mathbb{R}$ as a *new independent unknown* and consider the following relaxation approximation for (5.6). We search for $(\rho, \boldsymbol{u}, \kappa)^T : \mathbb{R}^2 \times (0,T) \to (0,\infty)\times\mathbb{R}^3$ such that

$$\begin{array}{rcrcl} \rho_t & + & \nabla\cdot(\rho\boldsymbol{u}) & = & 0, \\ (\rho\boldsymbol{u})_t & + & \nabla\cdot(\rho\boldsymbol{u}\boldsymbol{u}^T) + \rho\nabla\kappa & = & 0, \\ \kappa_t & + & \boldsymbol{u}\cdot\nabla\kappa + \frac{a^2}{\rho^2}\nabla\boldsymbol{u} & = & \frac{\tilde{\mu}(\rho,\Delta\rho)-\kappa}{d} \end{array} \tag{5.7}$$

holds in $\Omega\times(0,T)$. The parameter $d > 0$ is the (small) relaxation parameter and

$$\tilde{\mu}(\rho,\Delta\rho) := \mu(\rho) - \lambda\Delta\rho.$$

The constant a is chosen according to a generalized Whitham condition:

$$a^2 > \rho^2 c^2, \quad c := \sqrt{p'(\rho) + \frac{\lambda\rho}{h^2}} \tag{5.8}$$

Note: This approach can be considered as Sulicius relaxation method in Eulerian coordinates ([110]).

Before we discuss the discretization let us note some basic facts on system (5.7). Since system (5.7) is rotationally invariant it suffices for all our analytical issues to consider the one-dimensional version. The one-dimensional system is (of course) also a nonconservative system. Omitting the right hand side in (5.7) we get in primitive variables the first-order system

$$\begin{array}{rclcl} \rho_t &+& (\rho u)_x &=& 0, \\ u_t &+& u u_x + \kappa_x &=& 0, \\ \kappa_t &+& u\kappa_x + \frac{a^2}{\rho^2} u_x &=& 0. \end{array} \tag{5.9}$$

Let us summarize the primitive unknowns ρ, u, κ of (5.9) into the vector

$$\boldsymbol{w} = (\rho, u, \kappa)^T.$$

The Jacobian of the nonconservative flux in (5.9) is given by

$$\boldsymbol{D} := \begin{pmatrix} u & \rho & 0 \\ 0 & u & 1 \\ 0 & a^2/\rho^2 & u \end{pmatrix}.$$

Straightforward calculus leads us to

Lemma 5.3.1 (Hyperbolicity and characteristic fields)

(i) *The system (5.9) is hyperbolic in $\mathcal{U} := (0,\infty) \times \mathbb{R}^2$. The eigenvalues of $D \in \mathbb{R}^{3\times 3}$ are given by*

$$\lambda_1(\boldsymbol{w}) = u - \frac{a}{\rho}, \quad \lambda_2(\boldsymbol{w}) = u, \quad \lambda_3(\boldsymbol{w}) = u + \frac{a}{\rho} \qquad (\boldsymbol{w} \in \mathcal{U}).$$

and the corresponding eigenvectors are

$$\boldsymbol{r}_1(\boldsymbol{w}) = \begin{pmatrix} \rho^3/a^2 \\ -\rho/a \\ 1 \end{pmatrix}, \quad \boldsymbol{r}_2(\boldsymbol{w}) = \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix}, \quad \boldsymbol{r}_3(\boldsymbol{w}) = \begin{pmatrix} \rho^3/a^2 \\ \rho/a \\ 1 \end{pmatrix}.$$

(ii) *All characteristic fields are linear degenerate, i.e., we have for $i = 1, 2, 3$ and all $\boldsymbol{w} \in \mathcal{U}$*

$$\nabla\lambda(\boldsymbol{w}) \cdot \boldsymbol{r}_i(\boldsymbol{w}) = 0.$$

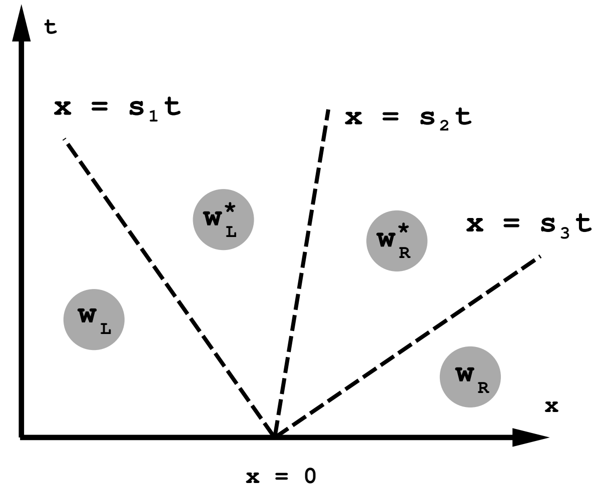

Figure 5.11: The structure of the self-similar solution of the Riemann problem in the (x, t)-halfspace.

5.3.2 The Riemann Problem for the Relaxation System

In this section we solve the Riemann problem for (5.9) globally, i.e., we consider for each $\boldsymbol{w}_L, \boldsymbol{w}_R \in \mathcal{U}$ the initial datum

$$\boldsymbol{w}_0(x) = \begin{cases} \boldsymbol{w}_L: & x < 0, \\ \boldsymbol{w}_R: & x > 0. \end{cases}$$

This Riemann problem cannot be treated by routine methods since system (5.9) is in nonconservative form. However, due to the linear degeneracy of (5.9), it is possible to give meaning to the nonconservative products.
We suppose that the solution of the Riemann problem is self-similar and consists of (at most) three elementary waves of contact discontinuity type. For $i = 1, 2, 3$ we call the corresponding elementary wave i-wave. An i-wave travels with the speed $s_i \in \mathbb{R}$ given by

$$s_i = s_i(\boldsymbol{w}) = \lambda_i(\boldsymbol{w}) \quad (\boldsymbol{w} \in \mathcal{U}).$$

Let us denote the (unknown) middle states by $\boldsymbol{w}_L^*$ and $\boldsymbol{w}_R^*$. so that the solution of the Riemann problem has the structure as in Fig. 5.11. We apply the theory for nonconservative systems as developed in [36]. To obtain a wave connecting states $\boldsymbol{w}_-, \boldsymbol{w}_+ \in \mathcal{U}$ with speed s there must be constants $\tilde{\rho}, \tilde{\tau} \in \mathbb{R}$ such that the generalized Rankine-Hugoniot conditions

$$\begin{aligned} -s[\rho] + [\rho u] &= 0, \\ -s[\rho u] + [\rho u^2] + \tilde{\rho}[\kappa] &= 0, \\ -s[\rho\kappa] + [\rho u \kappa] + a^2\tilde{\tau}[u] &= 0 \end{aligned} \tag{5.10}$$

hold. Here we denote by $[\varphi]$ the jump $\varphi_- - \varphi_+$ for some function $\varphi = \varphi(\boldsymbol{w})$, $\boldsymbol{w} \in \mathcal{U}$.

Lemma 5.3.2
Let $\boldsymbol{w}_-, \boldsymbol{w}_+ \in \mathcal{U}$ be states such that (5.10) *are satisfied with $s = s_i$ for some $i \in \{1, 2, 3\}$.*

Then we have either

$$[u] = [\kappa] = 0 \tag{5.11}$$

or

$$[u], [\kappa] \neq 0, \quad m^2 = a^2 \tilde{\tau}\tilde{\rho}. \tag{5.12}$$

Thereby we defined $m = \rho_+(u_+ - s) = \rho_-(u_- - s)$.

Proof. We observe that the second and the third equation in (5.10) can be rewritten in the form

$$\begin{aligned} m[u] + \tilde{\rho}[\kappa] &= 0, \\ m[\kappa] + a^2\tilde{\tau}[u] &= 0. \end{aligned}$$

This is a linear system for the jumps and the statement follows.

From Lemma 5.3.2 and $[\lambda_2(\boldsymbol{w})] = 0$ we deduce that for $i = 2$ the condition (5.11) must hold since $m_2 = 0$. Thus for a 2-wave we can choose $\tilde{\rho}, \tilde{\tau}$ arbitrarily and have

$$[v] = [\kappa] = 0. \tag{5.13}$$

For an 1/3-wave we have $m_{1/3}^2 = a^2 \neq 0$ by $[\lambda_{1/3}(\boldsymbol{w})] = 0$. Thus (5.12) applies and leads to the relation

$$\tilde{\rho} = \frac{1}{\tilde{\tau}}. \tag{5.14}$$

Now, let the factors $\tilde{\rho}$ of the 1/3-wave depend on the left hand and right hand density states:

$$\tilde{\rho}_1 = \tilde{\rho}_1(\rho_L, \rho_L^*), \quad \tilde{\rho}_3 = \tilde{\rho}_3(\rho_R^*, \rho_R).$$

We then have from the Rankine-Hugoniot conditions (5.10) and (5.13), (5.14) the equations

$$\begin{aligned} a(u_L^* - u_L) + \tilde{\rho}_1(\rho_L, \rho_L^*)(\kappa_L^* - \kappa_L) &= 0, \\ u_L^* &= u_R^*, \\ \kappa_L^* &= \kappa_R^*, \\ a(u_R^* - u_L^*) + \tilde{\rho}_3(\rho_R^*, \rho_R)(\kappa_R - \kappa_R^*) &= 0. \end{aligned} \tag{5.15}$$

To avoid solving a system of nonlinear equations we define now

$$\tilde{\rho}_1(\rho_L, \rho_L^*) := \rho_L, \quad \tilde{\rho}_3(\rho_R^*, \rho_R) = \rho_R.$$

From the first and the third equation of (5.15) we find with the second equation (and $\tau = 1/\rho$)

$$\begin{aligned} u_L^* = u_R^* &= \frac{\tau_L u_L + \tau_R u_R - \frac{1}{a}(\kappa_R - \kappa_L)}{\tau_L + \tau_R}, \\ \kappa_L^* = \kappa_R^* &= \kappa_L + a\tau_L(u_L - u_L^*). \end{aligned} \tag{5.16}$$

Here it is important that κ and v do not jump via the 2-wave. Finally we define according to the linear degeneracy of the characteristic fields

$$\rho_L^* = \frac{a}{u_L^* - s_1}, \; \rho_R^* = -\frac{a}{u_R^* - s_3}. \tag{5.17}$$

Now we have defined all states in the postulated solution of the Riemann problem. It is straightforward to check for all three waves that all definitions of the middle states, in particular (5.17), are consistent with the original conditions (5.10).
We summarize the results in a theorem.

Theorem 5.3.3 (Solution of the Riemann Problem)
Let the states $u_L, u_R \in \mathcal{U}$ be given. Then there exists a generalized solution $u : \mathbb{R} \times [0,T] \to \mathcal{U}$ of the corresponding Riemann problem (in the sense of [36]).
The solution u consists of the four states $u_L, u_L^*, u_R^*, u_R \in \mathcal{U}$ which are separated by three contact discontinuities which travel with speeds $s_1, s_2, s_3 \in \mathbb{R}$ given by

$$\begin{aligned} s_1 &= u_L - \frac{a}{\rho_L} &&= u_L^* - \frac{a}{\rho_L^*}, \\ s_2 &= u_L^* &&= u_R^*, \\ s_3 &= u_R + \frac{a}{\rho_R} &&= u_R^* + \frac{a}{\rho_R}. \end{aligned} \tag{5.18}$$

The states $u_L^*, u_R^* \in \mathcal{U}$ are defined by (5.16) and (5.17).

5.3.3 The Complete Numerical Algorithm

In this section we present the complete numerical algorithm for solving the initial value problem for the Navier-Stokes-Korteweg system (5.1) in one and two space dimensions. The discretization relies on the relaxation system (5.7) rather than on (5.1) directly.

The Scheme in 1d

First we provide the discretized initial data

$$\begin{aligned} \rho_j^0 &= \frac{1}{h}\int_{x_{j-\frac{1}{2}}}^{x_{j+\frac{1}{2}}} \rho_0(x)\,dx, \\ (\rho u)_j^0 &= \frac{1}{h}\int_{x_{j-\frac{1}{2}}}^{x_{j+\frac{1}{2}}} (\rho_0 u_0)(x)\,dx, \\ \kappa_j^0 &= \mu(\rho_j^0) - \frac{\lambda}{h^2}(\rho_{j+1}^0 - 2\rho_j^0 + \rho_{j-1}^0). \end{aligned}$$

The most important step of the update procedure from one timestep to another consists of two parts. First we neglect the sources in (5.7) and consider the first-order system

$$\begin{aligned} \rho_t + (\rho u)_x &= 0, \\ (\rho u)_t + (\rho u^2)_x + \rho\kappa_x &= 0, \\ \kappa_t + u\kappa_x + \tfrac{a^2}{\rho^2}u_x &= 0. \end{aligned} \tag{5.19}$$

The second step is the relaxation step. We solve the ordinary differential equations

$$\begin{aligned} \rho_t &= 0, \\ (\rho u)_t &= 0, \\ \kappa_t &= \frac{\tilde{\mu}(\rho, \Delta\rho) - \kappa}{d} \end{aligned}$$

as the relaxation parameter d tends to zero. As initial data we take the data from the first step. This means κ is projected back to the *equilibrium manifold.* Because d tends to zero we simply get

$$\kappa = \tilde{\mu}(\tilde{\rho}, \Delta\tilde{\rho}),$$

where $\tilde{\rho}$ is the data that comes from the first step. In the following we include the viscous term again. We summarize the update procedure from time step n to $n+1$ as follows:

1) Choose the parameter a in (5.19) locally at the cell interfaces according to the generalized Whitham condition (5.8)

$$a^2_{j+\frac{1}{2}} = \max_{i=j,j+1} \left\{ (\rho_i^n)^2 \left(p'(\rho_i^n) + \frac{\lambda \rho_i^n}{h^2} \right) \right\}.$$

2) Solve the Riemann Problem at each cell interface $x_{j+\frac{1}{2}}$ with initial data $(\rho_j^n, u_j^n, \kappa_j^n)$, $(\rho_{j+1}^n, u_{j+1}^n, \kappa_{j+1}^n)$. Let $(\tilde{\rho}_{j+\frac{1}{2}}, \tilde{u}_{j+\frac{1}{2}}, \tilde{\kappa}_{j+\frac{1}{2}})$ denote the solution of the corresponding Riemann Problem.

$$\begin{aligned} \rho_j^{n+1} &= \rho_j^n - \frac{\Delta t}{h} \left(\tilde{\rho}_{j+\frac{1}{2}}(0)\tilde{u}_{j+\frac{1}{2}}(0) - \tilde{\rho}_{j-\frac{1}{2}}(0)\tilde{u}_{j-\frac{1}{2}}(0) \right), \\ (\rho v)_j^{n+1} &= (\rho v)_j^n - \frac{\Delta t}{h} \left(\tilde{\rho}_{j+\frac{1}{2}}(0)\tilde{u}_{j+\frac{1}{2}}(0)^2 - \tilde{\rho}_{j-\frac{1}{2}}(0)\tilde{u}_{j-\frac{1}{2}}(0)^2 \right) \\ &\quad - \frac{\Delta t}{h}(\nu^R_{j-\frac{1}{2}} + \nu^L_{j+\frac{1}{2}}) \\ &\quad + \varepsilon \frac{\Delta t}{h^2}(u^n_{j+1} - 2u^n_j + u^n_{j-1}), \end{aligned}$$

with

$$\begin{aligned} \nu^R_{j-\frac{1}{2}} &= \int_{x_{j-\frac{1}{2}}}^{x_j} \tilde{\rho}_{j-\frac{1}{2}}(x)\partial_x \tilde{\kappa}_{j-\frac{1}{2}}(x)dx, \\ \nu^L_{j+\frac{1}{2}} &= \int_{x_j}^{x_{j+\frac{1}{2}}} \tilde{\rho}_{j+\frac{1}{2}}(x)\partial_x \tilde{\kappa}_{j+\frac{1}{2}}(x)dx. \end{aligned}$$

3) Perform the relaxation step

$$\kappa_j^{n+1} = \mu(\rho_j^{n+1}) - \frac{\lambda}{h^2}(\rho^{n+1}_{j+1} - 2\rho^{n+1}_j + \rho^{n+1}_{j-1}).$$

Note: Let $a_1 < a_2 < a_3$ and

$$\rho(x) = \begin{cases} \rho_l, & x \in (a_1, a_2) \\ \rho_r, & x \in (a_2, a_3) \end{cases}, \quad \kappa(x) = \begin{cases} \kappa_l, & x \in (a_1, a_2) \\ \kappa_r, & x \in (a_2, a_3) \end{cases}.$$

Then we set $\int_{a_1}^{a_2} \rho(x)\kappa_x(x)dx = \frac{1}{2}(\rho_l + \rho_r)(\kappa_r - \kappa_l)$. Note that the solution of the Riemann Problem can have zero, one or two jumps in the intervals $(x_{j-\frac{1}{2}}, x_j)$ and $(x_j, x_{j+\frac{1}{2}})$.

2d Extension of the Scheme

In order to describe the scheme in two space dimensions on a Cartesian mesh we have to consider planar waves solving the system (5.7). Due to rotational invariance it is sufficient to consider planar waves that propagate in x-direction only. These waves satisfy the equation

$$\begin{array}{ccccc} \rho_t & + & (\rho u_1)_x & = & 0, \\ (\rho u_1)_t & + & (\rho u_1^2)_x + \rho\kappa_x & = & 0, \\ (\rho u_2)_t & + & (\rho u_1 u_2)_x & = & 0, \\ \kappa_t & + & u_1\kappa_x + \dfrac{a^2}{\rho^2}u_{1,x} & = & \dfrac{\tilde{\mu}(\rho, \Delta\rho) - \kappa}{d}. \end{array} \tag{5.20}$$

We can easily verify the following

Lemma 5.3.4 (Hyperbolicity and characteristic fields)

(i) The system (5.20) is hyperbolic (but not strictly hyperbolic) in $\mathcal{U} := (0, \infty) \times \mathbb{R}^3$. The eigenvalues of the corresponding Jacobian are given by

$$\lambda_1(\boldsymbol{w}) = u_1 - \frac{a}{\rho}, \quad \lambda_2(\boldsymbol{w}) = \lambda_3(\boldsymbol{w}) = u_1, \quad \lambda_4(\boldsymbol{w}) = u_1 + \frac{a}{\rho} \qquad (\boldsymbol{w} \in \mathcal{U}).$$

and the corresponding eigenvalues are

$$\boldsymbol{r}_1(\boldsymbol{w}) = \begin{pmatrix} \rho^3/a^2 \\ -\rho/a \\ 0 \\ 1 \end{pmatrix}, \ \boldsymbol{r}_2(\boldsymbol{w}) = \begin{pmatrix} 1 \\ 0 \\ 0 \\ 0 \end{pmatrix}, \ \boldsymbol{r}_3(\boldsymbol{w}) = \begin{pmatrix} 0 \\ 0 \\ 1 \\ 0 \end{pmatrix}, \ \boldsymbol{r}_4(\boldsymbol{w}) = \begin{pmatrix} \rho^3/a^2 \\ \rho/a \\ 0 \\ 1 \end{pmatrix}.$$

(ii) All characteristic fields are linear degenerate, i.e., we have for $i = 1, \ldots, 4$ and all $\boldsymbol{w} \in \mathcal{U}$

$$\nabla\lambda(\boldsymbol{w}) \cdot \boldsymbol{r}_i(\boldsymbol{w}) = 0.$$

Now the solution of the Riemann Problem of the hyperbolic part of equation (5.20) has almost the same structure as the Riemann Problem for the 1-D equation. We generalize Theorem 5.3.3.

Theorem 5.3.5 (Solution of the Riemann Problem)
Let the states $\boldsymbol{w}_L, \boldsymbol{w}_R \in \mathcal{U}$ be given. Then there exists a generalized solution $\boldsymbol{w} : \mathbb{R}^2 \times [0,T] \to \mathcal{U}$ of the corresponding planar Riemann problem (in the sense of [36]). The solution u consists of the four states $\boldsymbol{w}_L, \boldsymbol{w}_L^*, u_R^*, u_R \in \mathcal{U}$ which are separated by four contact discontinuities which travel with speeds $s_1, s_2 = s_2', s_3 \in \mathbb{R}$ given by (5.18). The states $\boldsymbol{w}_L^*, \boldsymbol{w}_R^* \in \mathcal{U}$ are defined by (5.16), (5.17) and

$$\begin{aligned} u_{2,L}^* &= u_{2,L}, \\ u_{2,R}^* &= u_{2,R}. \end{aligned} \tag{5.21}$$

Thus, the formulation of the scheme on Cartesian meshes is straightforward. We omit the details.

Restriction on the Time Step Size

Now, the critical task is to give a correct restriction on the time step size that ensures the stability of the method in some sense. The presence of second and third order terms in the Navier-Stokes-Korteweg system and the lack of hyperbolicity of the first-order part of the Navier-Stokes-Korteweg system make it difficult to give rigorous arguments on the restriction of the time step size. Nevertheless, for sake of completeness of the algorithm, we state at this point the condition we actually use.

We restrict ourselves to the 1-D situation. The extension to 2-D is straightforward. Solving the local Riemann problems gives the condition

$$\frac{\Delta t}{h} \max_j \left\{ \left| u_j - \frac{a_{j+\frac{1}{2}}}{\rho_j} \right|, \left| u_{j+1} + \frac{a_{j+\frac{1}{2}}}{\rho_{j+1}} \right| \right\} \leq \frac{1}{2}, \tag{5.22}$$

and the approximation of the viscous term gives the condition

$$\frac{\Delta t}{h^2} \max_j \left\{ \frac{\varepsilon}{\rho_j} \right\} \leq \frac{1}{2}. \tag{5.23}$$

If we take the choice of the parameters $a_{j+\frac{1}{2}}$ into account we can see that the time step size is of order $O(h^2)$.

Numerical Results

The configuration for the numerical tests with the relaxation scheme is the same as with the conservative scheme. However, we omit the third test case *Towards Static Equilibrium* because the scheme does not produce the dynamics of the solution correctly as we will see in the test case with the traveling wave solution. Therefore an additional test does not make sense.

Test Case: Static Equilibrium
Again we observe second order convergence as with the well balanced scheme. This is

only due to the use of a quadrature formula of insufficient degree. See the corresponding test case with the well balanced scheme in the previous section. We can also see that the errors produced by the relaxation scheme and the well balanced scheme are nearly identical because the errors are produced mainly by initial projection. Table 5.5 illustrates the results of the computations.

	density and momentum	
h	total L^2-error	EOC
2.0000e-02	3.7576e-03	
1.0000e-02	9.0721e-04	2.050
6.6667e-03	4.0074e-04	2.015
5.0000e-03	2.2494e-04	2.007
4.0000e-03	1.4382e-04	2.004
3.3333e-03	9.9822e-05	2.003
2.8571e-03	7.3315e-05	2.002

Table 5.5: Test Case: Static Equilibrium. Total L^2-error and EOC for the approximate solution generated by the relaxation scheme.

As with the well balanced scheme in the last section, a very small velocity field arises because the discrete initial data is not a perfect discrete equilibrium. But this velocity field converges completely to zero as time tends to infinity. A sequence of density profiles for different mesh sizes at computational end time would exactly look like these shown in Figure 5.7. Therefore we omit it.

Test Case: Traveling Wave Solution
The approximate solution generated by the relaxation scheme seems to converge to some limit function as the mesh size tends to zero. But this function is not the exact solution as shown in Figure 5.12 and Table 5.6. Only the momentum is shown in Figure 5.12 such that the difference between exact and approximate solution can be seen more clearly. The L^2-errors of the density profiles and the momentum profiles are illustrated by Table 5.6.

5.4 Comparison of the three Different Schemes

We compare the two quantitative tests applied in the previous sections to the conservative scheme, the nonconservative well balanced scheme and the nonconservative relaxation scheme.

The left part of Figure 5.13 shows the error of the density profiles of the three different schemes in the test case with the traveling wave solution. The conservative scheme and the nonconservative well balanced scheme converge with order 1 to the exact solution and the error of the well balanced scheme is an order of magnitude smaller than the error of the conservative scheme. The discrete solution generated by the relaxation scheme does not converge to the exact solution. The momentum profiles (not shown) show exactly the same behavior.

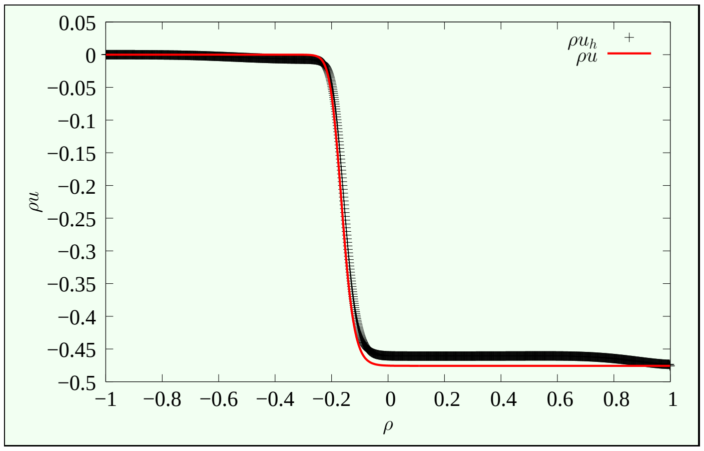

Figure 5.12: Exact and approximate traveling wave solution (momentum only) generated by the relaxation scheme for $n = 1800$.

	ρ		ρu	
h	L^2-error	EOC	L^2-error	EOC
1.0000e-02	3.3850e-02		2.3703e-02	
5.0000e-03	3.3000e-02	0.037	2.1270e-02	0.156
3.3333e-03	3.2677e-02	0.024	2.0440e-02	0.098
2.5000e-03	3.2506e-02	0.018	2.0025e-02	0.071
2.0000e-03	3.2399e-02	0.015	1.9776e-02	0.056
1.6667e-03	3.2327e-02	0.012	1.9610e-02	0.046
1.4286e-03	3.2275e-02	0.011	1.9492e-02	0.039
1.2500e-03	3.2235e-02	0.009	1.9404e-02	0.034
1.1111e-03	3.2204e-02	0.008	1.9335e-02	0.030

Table 5.6: L^2-error and EOC for the approximate traveling wave solution generated by the relaxation scheme.

The right part of Figure 5.13 compares the convergence rates in the test with the static equilibrium solution. The values for the nonconservative well balanced scheme and the relaxation scheme are almost the same because after initial projection the data does not change essentially. Therefore the values of the relaxation scheme hide the values of the well balanced scheme in the figure. As discussed in the previous sections, the well balanced scheme and the relaxation scheme are not really second order schemes as the figure suggests and the convergence rate of the conservative scheme should approach one if the mesh is further refined. From the figure we can conclude that the results at a static equilibrium computed by the well balanced scheme and the relaxation scheme are several magnitudes better than the results given by the conservative scheme.

There is clearly a difference in the qualitative behavior of the numerical solutions produced by the conservative and the well balanced scheme. For those generated by the well balanced scheme the total energy is a decreasing function of time, when the solution tends to a static equilibrium state on the discrete level the kinetic energy tends to zero and the value κ approaches a constant as time tends to infinity. This is exactly the

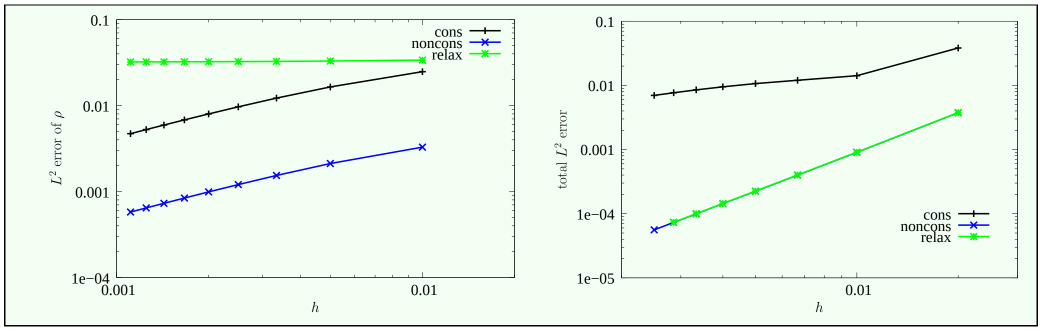

Figure 5.13: Left: L^2-errors of the density profiles for the three different schemes, traveling wave test. Right: total L^2-errors for the three schemes, static equilibrium test.

behavior of exact solutions. For the numerical solutions produced by the conservative scheme we do not have these properties for a fixed mesh size h.

Thus, the nonconservative well balanced scheme seems to be the most promising scheme. The smaller time step size in comparison to the other schemes is not an issue since this can be bypassed using implicit time stepping. This is the scheme we will generalize to higher order schemes on arbitrary nonconform meshes by application of the Discontinuous Galerkin approach in the next chapter. The numerical experiments show that the relaxation scheme is of very limited use. It can only be used to construct nontrivial, first order accurate static equilibrium solutions. Also the generalization to higher order schemes is much more involved for the relaxation scheme than for the other schemes.

Chapter 6

Higher Order Schemes: The Discontinuous Galerkin Approach

The Discontinuous Galerkin (DG) method is a class of Finite Element methods that uses completely discontinuous ansatz functions as a basis of the Finite Element space. In application to systems of conservation laws these inter element discontinuities give extra degrees of freedom that can be used to stabilize the method. At the discontinuities usually numerical fluxes are applied that are known from the Finite Volume framework, see standard textbooks such as [51], [52], [76], [81]. Thus, the Discontinuous Galerkin approach is a combination of Finite Element and Finite Volume methods and a natural generalization of Finite Volume methods to arbitrary higher order schemes.

The Discontinuous Galerkin method has several advantages over other higher order Finite Volume methods such as methods based on ENO or WENO reconstruction.

- In the framework of the DG approach it is very easy to design higher order ansatz spaces. The polynomial degree can be chosen locally which makes the schemes ideally suited for p-adaptivity.
- Arbitrary, nonconform unstructured meshes can be used, possibly with hanging nodes due to the discontinuous ansatz functions.
- The method is extremely local. It is only necessary to communicate with the direct neighbor cells. Thus, it is very well suited for parallel implementations.

But there are still some drawbacks as the need for slope limiters when the approximated solution is not sufficiently smooth. Sometimes the computational cost may be higher because at the cell boundaries in general integration formulas of twice the degree as for reconstruction based schemes have to be used. Additionally a volume integral has to be computed. Depending on the application this extra cost can be higher or lower than the reconstruction step in ENO or WENO methods. A complete numerical comparison between these methods applied to systems of interest is not available at time of this writing.

The first Discontinuous Galerkin method was proposed 1973 by Reed and Hill [94]. During the last two decades a major development of this type of numerical schemes was carried out by Cockburn, Shu and coworkers in the series of papers [22], [21], [24], [23], [26]. The method has found rapid applications in many different areas. The review paper [22] provides a good overview and many useful references concerning the DG approach.

The *Local* Discontinuous Galerkin (LDG) method is a generalization of the standard DG method for conservation laws proposed by Bassi and Rebay [8]. It is designed for the use with convection dominated conservation laws that include higher order derivatives, such as the compressible Navier-Stokes equations. Further development of this method was done by Cockburn, Shu and coworkers especially the application to equations with third or higher order derivatives, see for example [25], [130].

The LDG method has all the advantages of the standard DG method. In contrast to other DG type methods for convection dominated convection-diffusion equations, such as the Baumann and Oden method [9], the Local Discontinuous Galerkin method can be easily applied to equations with third or higher order derivatives. This property makes it ideally suited for the application to the Navier-Stokes-Korteweg system.

In this chapter we discuss the Local Discontinuous Galerkin method and its application to conservative terms, higher order term and source terms in detail. Additionally we present an approach for the DG discretization of nonconservative terms based on the definition of nonconservative products [36] and on the formulation given in [63]. We describe the method in a general framework of evolution equations and discuss the discretization of some simple examples. This general framework has also been successfully applied to many other problems, see for example [19]. For a scalar model problem for the NSK system we prove a L^2-stability result of a semi-discrete Local Discontinuous Galerkin discretization, similar to the result given in [130]. Based on the discretization of the model problem we give the complete discretization of the Navier-Stokes-Korteweg system in multiple space dimension at the end of this chapter.

For the application of the method we use unstructured triangular and tetrahedral meshes since these type of meshes are very well suited in approximation of complicated geometries and have the extra advantage that the reference mapping to the standard cell is an affine linear transformation. This has several advantages (listed in the following section) in combination with the Discontinuous Galerkin method. We allow the meshes to be nonconform in order to perform local mesh adaption efficiently in parallel and as simple as possible. However, most (but not all) of the following applies to more general meshes as well. We start with the description of the simplicial meshes we use.

6.1 Simplicial Meshes

The use of simplicial meshes in combination with the Discontinuous Galerkin method has the advantage that the mappings between the reference cell and the cells of the mesh are affine linear functions. This linearity of the reference mappings has two important consequences. On the one hand the computational cost is significantly lower and on the other hand orthogonality of local base functions is preserved as we will see in Section 6.3. The latter is also an improvement of the efficiency of the method, especially in combination with explicit time stepping, and leads to a simpler implementation of the method. We start with the description of the underlying meshes that will in general be nonconform but not arbitrary nonconform meshes. We are mainly interested in nonconform meshes that are generated by successive refinement of a conform *macro mesh*. For the ease of implementation and the numerical stability of the method it is also desirable to restrict the number of *levels of nonconformity*.

The n-dimensional reference simplex (reference cell) is defined by

$$\hat{\Delta} = \left\{ \boldsymbol{x} \in \mathbb{R}^n \mid x_i \geq 0, \ \sum_{i=1}^{n} x_i \leq 1 \right\}. \tag{6.1}$$

Let $T_j : \hat{\Delta} \to \mathbb{R}^n$ a nondegenerate, affine linear mapping for $j = 0, \dots, n_{cells} - 1$. We define

$$\begin{aligned} \Delta_j &= T_j(\hat{\Delta}), \\ \mathcal{T} &= \{\Delta_j \mid j = 0, \dots, n_{cells} - 1\}. \end{aligned}$$

In the following we denote both, the compact set Δ_j as defined above, as well its open interior set by the symbol Δ_j depending on what is more appropriate and provided that the meaning is clear. Vertices of cells are called 0-dimensional interfaces, edges 1-dimensional interfaces, faces 2-dimensional interfaces and so on.

Definition 6.1.1 (Simplicial Mesh)
$\mathcal{T}$ is called a nonconform simplicial Mesh if for all $i \neq j$, $\Delta_i, \Delta_j \in \mathcal{T}$ we have $\mathcal{H}^n(\Delta_i \cap \Delta_j) = 0$ and if $\mathcal{H}^{n-k}(\Delta_i \cap \Delta_j) \neq 0$ for $k = 1, \dots, n$ one of the following two conditions holds

(i) a $(n-k)$-dimensional interface of Δ_i is subset of a $(n-k)$-dimensional interface of Δ_j,

(ii) a $(n-k)$-dimensional interface of Δ_j is subset of a $(n-k)$-dimensional interface of Δ_i.

A nonconform Simplicial Mesh is called conform if conditions (i) and (ii) hold simultaneously, i.e., the cells Δ_i and Δ_j share a common $(n-k)$-dimensional interface.

In the above definition $\mathcal{H}^m$ denotes the m-dimensional Hausdorff measure in $\mathbb{R}^n$. A Simplicial Mesh is called a Triangulation for $n = 2$ and a Tetrahedralization for $n = 3$. If $\mathcal{H}^{n-1}(\Delta_i \cap \Delta_j) \neq 0$ for two different cells Δ_i and Δ_j of the mesh then they are called neighbors.
For a family of simplicial meshes $(\mathcal{T}_h)_{h>0}$ we will assume in the following

$$\begin{aligned} \delta(\mathcal{T}_h) &\leq h, \\ \sup_h \kappa(T_h) &< \infty, \end{aligned}$$

where δ and κ are defined by

$$\begin{aligned} \delta(\mathcal{T}) &= \sup \left\{ \operatorname{diam}(\Delta_j) \mid \Delta_j \in \mathcal{T} \right\}, \\ \kappa(\mathcal{T}) &= \sup \left\{ \frac{\delta(\mathcal{T})^n}{|\Delta_j|} \mid \Delta_j \in \mathcal{T} \right\}. \end{aligned}$$

This means that for this family the mesh size tends to zero and angles remain bounded from below. $\Omega_h = \bigcup_{\Delta_j \in \mathcal{T}_h} \Delta_j$ denotes the domain that is partitioned by $\mathcal{T}_h$ and $|\mathcal{T}_h|$ the number of cells of mesh $\mathcal{T}_h$.

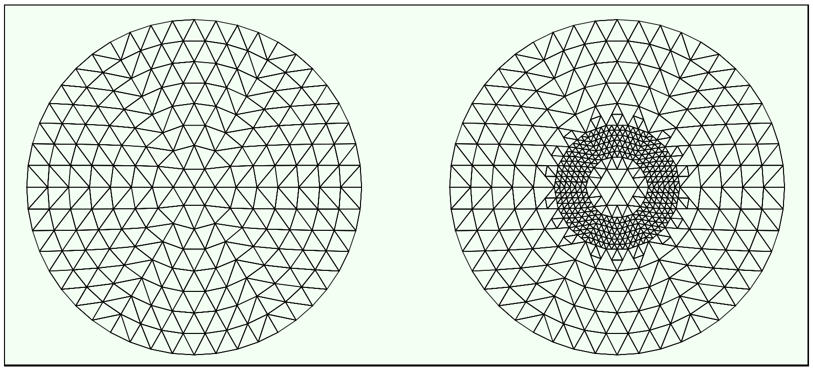

Figure 6.1: Conform mesh (left) and a nonconform mesh obtained by successive refinement of the conform mesh (right).

6.2 The Local Discontinuous Galerkin Method

Here we discuss the higher order spatial discretization of conservative systems with or without higher order derivatives, non-conservative parts and source terms. The term *local* in Local Discontiunuous Galerkin Method is used when higher order derivatives are involved. The discretization leads to a semi-discrete formulation, i.e., a ordinary differential equation. The higher order time discretization of ordinary initial value problems is discussed in Chapter 7.

6.2.1 First Order Conservative Systems

In this section we consider first order conservation laws of the form

$$\boldsymbol{u}_t + \mathcal{L}[\boldsymbol{u}] = \boldsymbol{0} \quad \text{in } \Omega \subset \mathbb{R}^n. \tag{6.2}$$

In the above equation $\mathcal{L} : C^1(\Omega,\mathcal{U}) \to C^0(\Omega,\mathcal{U})$ denotes a differential operator that is defined by

$$\mathcal{L}[\boldsymbol{u}](x) = \sum_{i=1}^{n} \frac{\partial}{\partial x_i} \boldsymbol{f}_i(\boldsymbol{u}(\boldsymbol{x}), \boldsymbol{x}),$$

where $\boldsymbol{f}_i : \mathcal{U} \times \Omega \to \mathbb{R}^d$, $i = 1, \ldots, n$ are smooth functions (*physical* fluxes) and might in general depend on further parameters such as time. The open set $\mathcal{U} \subset \mathbb{R}^d$ is called state space. For example the Euler equations of gas dynamics in multiple space dimensions and the inviscid Burgers equation are systems of conservation laws. Usually first order systems are required to be *hyperbolic* in (at least parts of) the state space $\mathcal{U}$, otherwise the conservation law usually suffers a lack of well posedness, see standard textbooks on

Hyperbolic Conservation Laws such as [34], [51], [100]. The aim of this section is the discretization of the spatial differential operator $\mathcal{L}$.

Let $(\cdot,\cdot)_\Omega$ denote the L^2 inner product with respect to Ω. Using partial integration we have (for smooth functions $\boldsymbol{u}$ and $\boldsymbol{\varphi} \in C^1(\Omega, \mathbb{R}^d)$) the expression

$$(\mathcal{L}[\boldsymbol{u}], \varphi)_\Omega = \int\limits_{\partial\Omega} \sum_{i=1}^{n} n_i \boldsymbol{f}_i(\boldsymbol{u}(\boldsymbol{x}), \boldsymbol{x}) \cdot \boldsymbol{\varphi}(\boldsymbol{x})\, d\sigma(\boldsymbol{x}) - \int\limits_{\Omega} \sum_{i=1}^{n} \boldsymbol{f}_i(\boldsymbol{u}(\boldsymbol{x}), \boldsymbol{x}) \cdot \frac{\partial}{\partial x_i}\boldsymbol{\varphi}(\boldsymbol{x})\, d\boldsymbol{x}, \tag{6.3}$$

where the n_i denote the components of the outer normal vector on $\partial\Omega$.

We introduce the scalar Discontinuous Galerkin space V_h by the definition

$$V_h \;=\; \left\{\varphi : \Omega_h \to \mathbb{R} \mid \varphi|_{\Delta_j} \in \mathbb{P}_k,\ \Delta_j \in \mathcal{T}_h\right\},$$

where the basis functions φ usually belong to the space of polynomials $\mathbb{P}_k$ of degree k locally, i.e., on each cell Δ_j of the underlying mesh $\mathcal{T}_h$, see Section 6.3. The set $\Omega_h \subset \mathbb{R}^n$ denotes an approximation (in some sense) of the domain Ω which is partitioned by $\mathcal{T}_h$. The space of polynomials could be replaced by some other space with similar approximation properties. Based on V_h we denote the space of vector valued ansatz functions with values in $\mathbb{R}^d$ by V_h^d.

Let us define a *discrete differential* operator $\mathcal{L}_h : V_h^d \to V_h^d$ by (L^2-)projecting $\mathcal{L}[\boldsymbol{u}]$ to V_h in a sense that is discussed in the following. Therefore we apply the definition of nonconservative products [36] to the expression (6.3), see Appendix A.4. For $\boldsymbol{u}_h \in V_h^d$ we define $\mathcal{L}_h[\boldsymbol{u}_h]$ by the relation

$$\begin{aligned}
&(\mathcal{L}_h[\boldsymbol{u}_h], \boldsymbol{\varphi})_{\Omega_h} \\
= \quad & -\sum_{j=0}^{|\mathcal{T}_h|-1} \int\limits_{\Delta_j} \sum_{i=1}^{n} \boldsymbol{f}_i(\boldsymbol{u}_h(\boldsymbol{x}), \boldsymbol{x}) \cdot \frac{\partial}{\partial x_i}\boldsymbol{\varphi}(\boldsymbol{x})\, d\boldsymbol{x} \qquad (6.4)\\
& +\frac{1}{2}\sum_{j=0}^{|\mathcal{T}_h|-1} \int\limits_{\partial\Delta_j \setminus \partial\Omega_h} \boldsymbol{g}(\boldsymbol{u}_h|_{\Delta_j}(\boldsymbol{x}), \boldsymbol{u}_h|_{\Delta_{j'}}(\boldsymbol{x}), \boldsymbol{x}, \boldsymbol{n}) \cdot (\boldsymbol{\varphi}|_{\Delta_j}(\boldsymbol{x}) - \boldsymbol{\varphi}|_{\Delta_{j'}}(\boldsymbol{x}))\, d\sigma(\boldsymbol{x}) \\
& +\sum_{j=0}^{|\mathcal{T}_h|-1} \int\limits_{\partial\Delta_j \cap \partial\Omega_h} \sum_{i=1}^{n} n_i \boldsymbol{f}_i(\boldsymbol{u}_h|_{\Delta_j}(\boldsymbol{x}), \boldsymbol{x}) \cdot \boldsymbol{\varphi}|_{\Delta_j}(\boldsymbol{x})\, d\sigma(\boldsymbol{x})
\end{aligned}$$

for all $\boldsymbol{\varphi} \in V_h^d$. The cells $\Delta_{j'}$ denote the corresponding neighboring cells of cell Δ_j in the *surface* integral above. The factor $\frac{1}{2}$ in front of the second term of the right hand side appears because all interfaces are counted twice. The last term in the equation above can be used to prescribe several kinds of boundary data. The Discontinuous Galerkin method is well defined when the physical fluxes $\boldsymbol{f}_i$ and numerical flux $\boldsymbol{g}$ are chosen. The physical fluxes are completely determined by the equation whereas the choice of the numerical flux is the crucial part in the method. For a reasonable method the numerical flux should satisfy at least

(i) $\boldsymbol{g}(\boldsymbol{u}, \boldsymbol{u}, \boldsymbol{x}, \boldsymbol{n}) = \sum_{i=1}^{n} n_i \boldsymbol{f}_i(\boldsymbol{u}, \boldsymbol{x})$ for all $\boldsymbol{u} \in \mathcal{U}, \boldsymbol{x} \in \Omega$ and $\boldsymbol{n} \in S^{n-1}$ (Consistency).

(ii) $\boldsymbol{g}$ locally Lipschitz continuous.

(iii) $\boldsymbol{g}(\boldsymbol{u}, \boldsymbol{v}, \boldsymbol{x}, \boldsymbol{n}) = -\boldsymbol{g}(\boldsymbol{v}, \boldsymbol{u}, \boldsymbol{x}, -\boldsymbol{n})$ for all $\boldsymbol{u}, \boldsymbol{v} \in \mathcal{U}, \boldsymbol{x} \in \Omega$ and $\boldsymbol{n} \in S^{n-1}$ (Conservation Property).

Many numerical fluxes for different kinds of equations can be found in standard textbooks such as [52], [76], [81], [111].

Now let

$$\boldsymbol{u}_h(\boldsymbol{x}, t) = \sum_{l=0}^{|V_h|-1} \varphi_l(\boldsymbol{x}) \boldsymbol{\alpha}_l(t), \quad \{\varphi_0, \ldots, \varphi_{|V_h|-1}\} \text{ basis of } V_h$$

then the semi-discrete formulation of the conservation law 6.2 can be written as

$$\left(\frac{\partial}{\partial t}\boldsymbol{u}_h(\cdot, t), \boldsymbol{\varphi}\right)_{\Omega_h} + (\mathcal{L}_h[\boldsymbol{u}_h(\cdot, t)], \boldsymbol{\varphi})_{\Omega_h} = 0 \quad \text{for all } \boldsymbol{\varphi} \in V_h^d,\ t \in (0, \infty). \tag{6.5}$$

This is the DG space discretization given by Cockburn and Shu, see for example [21], [24], [23], [26]. The initial value problem (initial values have to be provided by a projection to the space V_h^d) for the ordinary differential equation (6.5) can be solved by means of Runge-Kutta methods or other schemes like multistep schemes. For the use with conservation laws Shu and Osher [103] have developed special Runge-Kutta methods (TVD or Strong Stability Preserving) that preserve certain properties of conservation laws on the discrete level (such as the TVD property of scalar conservation laws), see Chapter 7.

6.2.2 Conservative Systems with Higher Order Terms

The idea of the treatment of higher order derivatives in the framework of the Local Discontinuous Galerkin method is to reformulate higher order differential operators as combination of first order differential operators and to apply the method described in the previous section.

As an example we consider the multidimensional nonlinear convection-diffusion equation

$$\begin{aligned} u_t + \mathcal{L}^2[u] &= 0, \\ \mathcal{L}^2[u] &= \nabla \cdot \boldsymbol{F}(u) - \nabla \cdot (\varepsilon \nabla u), \end{aligned}$$

with some nonlinear flux $\boldsymbol{F}$. This equation can be reformulated by using two first order differential operators

$$\begin{aligned} u_t + \mathcal{L}_2^1[(u, \mathcal{L}_1^1[u])] &= 0, \\ \mathcal{L}_1^1[u] &= \nabla u, \\ \mathcal{L}_2^1[(u, \boldsymbol{v})] &= \nabla \cdot \boldsymbol{F}(u) - \nabla \cdot (\varepsilon \boldsymbol{v}). \end{aligned}$$

In general, a m-th order differential operator can be constructed by means of m first order differential operators.

$$\begin{aligned}
\boldsymbol{u}^0 &= \boldsymbol{u}, \\
\boldsymbol{u}^1 &= \mathcal{L}_1^1[(\boldsymbol{u}^0)], \\
\boldsymbol{u}^2 &= \mathcal{L}_2^1[(\boldsymbol{u}^0, \boldsymbol{u}^1)], \\
&\vdots \\
\mathcal{L}^m[\boldsymbol{u}] &= \mathcal{L}_m^1[(\boldsymbol{u}^0, \boldsymbol{u}^1, \dots, \boldsymbol{u}^{m-1})].
\end{aligned}$$

Here the first order differential operators are of the form

$$\mathcal{L}_k^1[\boldsymbol{u}^0, \dots, \boldsymbol{u}^{k-1}](\boldsymbol{x}) = \sum_{i=1}^{n} \frac{\partial}{\partial x_i} \boldsymbol{f}_i^k\big(\boldsymbol{u}^0(\boldsymbol{x}), \dots, \boldsymbol{u}^{k-1}(\boldsymbol{x}), \boldsymbol{x}\big), \quad k = 1, \dots, m,$$

with $\boldsymbol{u}^k(\boldsymbol{x}) \in \mathbb{R}^{d_k}$, $\boldsymbol{x} \in \mathbb{R}^n$. These first order operators can then be discretized as in the previous section. The method is called *Local* Discontinuous Galerkin method because the temporary functions u^k can be eliminated locally without solving a large system of equations.

Now the space discretization at a time $t > 0$ of a conservation law including higher (m-th) order derivatives, represented by the spatial differential operator $\mathcal{L}^m$, of the form

$$\boldsymbol{u}_t + \mathcal{L}^m[\boldsymbol{u}] = \boldsymbol{0} \tag{6.6}$$

can be carried out by the definition of a discrete differential operator $\mathcal{L}_h^m$ of order m following the algorithm:

```
set $\boldsymbol{u}_h^0 = \boldsymbol{u}_h(\cdot, t)$;
for $k = 1, \dots, m$ {
    compute $\mathcal{L}_{h,k}^1[(\boldsymbol{u}_h^0, \dots \boldsymbol{u}_h^{k-1})]$ using the physical fluxes $\boldsymbol{f}_i^k$
    and consistent numerical fluxes $\boldsymbol{g}^k$ as in the previous section;
    set $\boldsymbol{u}_h^k = \mathcal{L}_{h,k}^1[(\boldsymbol{u}_h^0, \dots \boldsymbol{u}_h^{k-1})]$;
}
set $\mathcal{L}_h^m[\boldsymbol{u}_h(\cdot, t)] = \boldsymbol{u}_h^m$.
```

Here the discrete first order differential operators $\mathcal{L}_h^1$ are defined as in the previous section. Using the discrete spatial operator we can formulate the semi-discrete version of the higher order conservation law

$$\left(\frac{\partial}{\partial t}\boldsymbol{u}_h(\cdot, t), \boldsymbol{\varphi}\right)_{\Omega_h} + (\mathcal{L}_h^m[\boldsymbol{u}_h(\cdot, t)], \boldsymbol{\varphi})_{\Omega_h} = 0 \quad \text{for all } \boldsymbol{\varphi} \in V_h^d,\ t \in (0, \infty). \tag{6.7}$$

Example 6.2.1 (Scalar Convection-Diffusion Equation)

For the complete Discontinuous Galerkin discretization of the scalar convection diffusion equation from the beginning of this section we have to define the physical fluxes $\boldsymbol{f}_i^1$, f_i^2 and numerical fluxes $\boldsymbol{g}^1$, g^2. We use the following fluxes

$$\begin{aligned}
\boldsymbol{f}_i^1(u) &= u\boldsymbol{e}_i, \quad i = 1, \dots, n, \\
\boldsymbol{g}^1(u, \tilde{u}, \boldsymbol{n}) &= \frac{1}{2}(u + \tilde{u})\boldsymbol{n}, \\
f_i^2(u, \boldsymbol{v}) &= F_i(u) - \varepsilon v_i, \quad i = 1, \dots, n, \\
g^2(u, \boldsymbol{v}, \tilde{u}, \tilde{\boldsymbol{v}}, \boldsymbol{n}) &= G(u, \tilde{u}, \boldsymbol{n}) - \frac{\varepsilon}{2}(\boldsymbol{v} + \tilde{\boldsymbol{v}}),
\end{aligned}$$

where the vectors $\boldsymbol{e}_i$ denote the standard unit vectors in $\mathbb{R}^n$ and G is a consistent numerical flux for the fluxes F_i, for example the Lax-Friedrichs flux.

This is the original method of Bassi and Rebay introduced in [8] applied to the scalar convection-diffusion equation (Bassi and Rebay applied this method to the compressible Navier-Stokes equations). In the above discretization we have neglected the treatment of boundary conditions for simplicity, therefore the discretization of the convection diffusion is not yet complete (with the exception of a mesh with periodic boundary). However, the treatment of boundary conditions is another crucial part of the Discontinuous Galerkin method and depends, of course, on the kind of boundary condition. We will see examples for the DG-discretization of several kinds of boundary conditions at the end of this chapter in conjunction with the DG-discretization of the Navier-Stokes-Korteweg System in one, two and three space dimensions.

6.2.3 Non-Conservative Systems

In this section we consider first order systems including non-conservative terms, i.e., parts of the equation that can not be written in divergence form. Systems that arise from physics are usually in conservative form but a nonlinear transformations of coordinates or a homogenization process can lead to a non-conservative system of equations. In our case the discretization in non-conservative form simply leads to a more reliable discretization of the system. The discretization of non-conservative first order systems has to be done with care. In general the sequence of approximate solutions generated by a scheme in non-conservative form does not converge to the physical relevant solution in the case where discontinuities are present, see [60]. This is not an issue in our case because solutions of the NSK system are supposed to be sufficiently smooth. The approach we describe in this section can formally be generalized to systems with higher order terms and/or conservative terms as described in the previous sections. We consider differential operators of the form

$$\mathcal{L}[\boldsymbol{u}](x) = \sum_{i=1}^{n} \boldsymbol{A}_i(\boldsymbol{u}(\boldsymbol{x}), \boldsymbol{x}) \, \frac{\partial}{\partial x_i}\boldsymbol{u}(\boldsymbol{x}),$$

with matrix valued functions $\boldsymbol{A}_i : \mathbb{R}^d \times \mathbb{R}^n \to \mathbb{R}^{d\times d}$. At the moment the function $\boldsymbol{u}$ is supposed to be smooth. The aim of this section is to define a discrete operator that can be applied to discontinuous functions $\boldsymbol{u}_h$ of the finite element space V_h^d. Similar to Section 6.2.1 we apply the definition of nonconservative products [36], see Appendix A.4. In the following the notation of Appendix A.4 is used.

We define the discrete operator applied to $\boldsymbol{u}_h \in V_h^d$ by the relation

$$\begin{aligned}
&(\mathcal{L}_h[\boldsymbol{u}_h], \boldsymbol{\varphi})_{\Omega_h} \\
&= \int\limits_{\Omega_h} d\left[\sum_{i=1}^{n} \boldsymbol{\varphi}^T \boldsymbol{A}_i(\boldsymbol{u}_h, \cdot)\, \frac{\partial}{\partial x_i}\boldsymbol{u}_h\right]_{\phi} \\
&= \sum_{j=0}^{|\mathcal{T}_h|-1} \int\limits_{\Delta_j} \sum_{i=1}^{n} \boldsymbol{\varphi}(\boldsymbol{x})^T \boldsymbol{A}_i(\boldsymbol{u}_h(\boldsymbol{x}), \boldsymbol{x}) \cdot \frac{\partial}{\partial x_i}\boldsymbol{u}_h(\boldsymbol{x})\, d\boldsymbol{x} \qquad (6.8) \\
&\quad + \frac{1}{2}\sum_{j=0}^{|\mathcal{T}_h|-1} \int\limits_{\partial\Delta_j \backslash \partial\Omega_h} \int\limits_0^1 \sum_{i=1}^{n} n_i \boldsymbol{\phi}_{\boldsymbol{\varphi}}(t, \boldsymbol{x})^T \boldsymbol{A}_i\big(\phi_{\boldsymbol{u}}(t, \boldsymbol{x}), \boldsymbol{x}\big)\ \phi'_{\boldsymbol{u}}(t, \boldsymbol{x})\, dt\, d\sigma(\boldsymbol{x})
\end{aligned}$$

for all $\boldsymbol{\varphi} \in V_h^d$. $\phi_{\boldsymbol{u}}$ and $\phi_{\boldsymbol{\varphi}}$ in the above equation denote the $\boldsymbol{u}$- and $\boldsymbol{\varphi}$-components of the path ϕ, i.e.,

$$\begin{pmatrix} \phi_{\boldsymbol{u}} \\ \phi_{\boldsymbol{\varphi}} \end{pmatrix}(t, x) = \phi\Big(t;\ (\boldsymbol{u}_h|_{\Delta_j}(x), \boldsymbol{\varphi}|_{\Delta_j}(x)),\ (\boldsymbol{u}_h|_{\Delta_{j'}}(x), \boldsymbol{\varphi}|_{\Delta_{j'}}(x))\Big)$$

with the property that ϕ is linear in the test function arguments φ. The factor $\frac{1}{2}$ in the last term of equation (6.8) is necessary because all interfaces are counted twice. Note that there is no contribution of the boundary in equation (6.8) that can be used to impose boundary conditions as in the conservative case (6.4). The difference to boundary data can be regarded as discontinuity and therefore additional boundary terms have to be added to equation (6.8) in order to prescribe data on parts of the boundary of the domain.

The last term in equation (6.8) can be approximated by an averaging process. In practical applications we use the following variation

$$\begin{aligned}
&(\mathcal{L}_h[\boldsymbol{u}_h], \boldsymbol{\varphi})_{\Omega_h} \\
&= \sum_{j=0}^{|\mathcal{T}_h|-1} \int\limits_{\Delta_j} \sum_{i=1}^{n} \boldsymbol{\varphi}(\boldsymbol{x})^T \boldsymbol{A}_i(\boldsymbol{u}_h(\boldsymbol{x}), \boldsymbol{x}) \cdot \frac{\partial}{\partial x_i}\boldsymbol{u}_h(\boldsymbol{x})\, d\boldsymbol{x} \qquad (6.9) \\
&\quad + \frac{1}{2}\sum_{j=0}^{|\mathcal{T}_h|-1} \int\limits_{\partial\Delta_j \backslash \partial\Omega_h} \{\boldsymbol{\varphi}(\boldsymbol{x})^T\}_{\zeta} \left\{\sum_{i=1}^{n} n_i \boldsymbol{A}_i(\boldsymbol{u}_h(\boldsymbol{x}), \boldsymbol{x})\right\} [\boldsymbol{u}_h(\boldsymbol{x})]\, d\sigma(\boldsymbol{x}).
\end{aligned}$$

Here the blue term denotes the linear average in the test function

$$\{\boldsymbol{\varphi}(\boldsymbol{x})^T\}_{\zeta} = \zeta\boldsymbol{\varphi}|_{\Delta_j}(x) + (1-\zeta)\boldsymbol{\varphi}|_{\Delta_{j'}}(x)$$

for some $\zeta \in [0, 1]$. The red term denotes the average in the term $\sum n_i A_i$, not necessarily the arithmetic average. And

$$[\boldsymbol{u}_h(\boldsymbol{x})] = \Big(\boldsymbol{u}_h|_{\Delta_{j'}}(x) - \boldsymbol{u}_h|_{\Delta_j}(x)\Big)$$

denotes the jump of $\boldsymbol{u}_h$ over the cell interface in direction of the normal $\boldsymbol{n}$.

In practical applications the test functions have usually only support on one cell of the mesh. In this case the average in the above equation has the same structure as for the conservative terms, see (6.4). When non-conservative and conservative terms appear simultaneously both average procedures can be combined in one *generalized* numerical flux function. The parameter ζ can then control the average value in the test function.

6.2.4 Source Terms

Source terms in a balance law are simply projected to the the ansatz space V_h^d. This means a balance law of the form

$$\boldsymbol{u}_t + \mathcal{L}[\boldsymbol{u}] = \boldsymbol{B}(\boldsymbol{u}),$$

where $\mathcal{L}$ is a differential operator that can include higher order derivatives or non-conservative parts and $\boldsymbol{B}$ is the source term that can in general also depend on space and time variables, is discretized in the following way

$$\left(\frac{\partial}{\partial t}\boldsymbol{u}_h(\cdot,t),\boldsymbol{\varphi}\right)_{\Omega_h} + (\mathcal{L}_h[\boldsymbol{u}_h(\cdot,t)],\boldsymbol{\varphi})_{\Omega_h} = (\boldsymbol{B}(\boldsymbol{u}_h(\cdot,t)),\boldsymbol{\varphi})_{\Omega_h} \quad \text{for all } \boldsymbol{\varphi}\in V_h^d,\ t\in(0,\infty).$$

The formally simple approach does not necessarily mean that source terms are trivial to handle. The presence of source terms often results in stiff ordinary differential equations of the semi-discrete system. Therefore it is sometimes convenient to apply explicit-implicit Runge-Kutta methods (see Chapter 7), where the convective part of the equation is discretized in an explicit fashion, and the source term is treated implicitly.

6.3 Construction of Local Basis Functions

In this section we construct an orthogonal set of basis functions that spans the Finite Element space V_h. For this construction it is important that the reference mapping from the reference cell to an arbitrary cell of the mesh is an affine linear function. This is the case for simplicial meshes as well as for Cartesian meshes. For general meshes this is not the case. By this property orthogonality on the reference cell leads to orthogonality on an arbitrary cell.

We denote the L^2-inner product on the reference cell $\hat{\Delta}$ by

$$(\phi,\psi) = \int_{\hat{\Delta}} \phi(\boldsymbol{x})\psi(\boldsymbol{x})\,d\boldsymbol{x}$$

and polynomials of degree at most m and the dimension of this space by

$$\mathbb{P}_m = \operatorname{span}\left\{\boldsymbol{x}\mapsto\prod_{i=1}^n x_i^{k_i} \mid k_i\in\mathbb{N},\sum_{i=1}^n k_i\le m\right\},\quad |\mathbb{P}_m| = \frac{1}{n!}\prod_{i=1}^n(m+i).$$

We construct an orthonormal basis of $\mathbb{P}_m$ with respect to the inner product $(\cdot,\cdot)$ by application of the Gram-Schmidt procedure. It is well known that the Gram-Schmidt procedure is not stable when floating point arithmetic is used. Therefore we use rational arithmetic to overcome this problem by exploiting the fact that

$$\int_{\hat{\Delta}} \prod_{i=1}^{n} x_i^{k_i} \, d\boldsymbol{x} = \frac{\prod_{i=1}^{n} k_i!}{(n + \sum_{i=1}^{n} k_i)!},$$

which is a rational expression. This way the orthogonalization can be carried out without loss of accuracy. The normalization of the base functions is done at the end using high precision floating point arithmetic. Both, rational arithmetic and arbitrary precision floating point arithmetic, are provided by the GNU MP package [82] which provides a C++ interface including overloaded arithmetic operators.

We denote the orthonormal base polynomials of $\mathbb{P}_m$ by $p_0, \ldots, p_{|\mathbb{P}_m|-1}$. Here $|\mathbb{P}_m|$ stands for the dimension of $\mathbb{P}_m$. Using these local base functions we can define global orthogonal base functions on Ω_h

$$\varphi_l^j(\boldsymbol{x}) = \chi_{\Delta_j}(\boldsymbol{x})\, p_l(T_j(\boldsymbol{x})^{-1}), \quad j = 0, \ldots, |\mathcal{T}| - 1, \quad l = 0, \ldots, |\mathbb{P}_m| - 1. \tag{6.10}$$

Orthogonality of the local base polynomials p_l is preserved because the mapping T_j from the reference cell to the simplex Δ_j of $\mathcal{T}$ is an affine linear mapping. Now we define the $|\mathcal{T}| \cdot |\mathbb{P}_m|$ -dimensional space of base functions for the (m+1)-th order Discontinuous Galerkin Method by

$$V_h = \{\varphi_l^j \mid j = 0, \ldots, |\mathcal{T}| - 1,\ l = 0, \ldots, |\mathbb{P}_m| - 1\}.$$

Note: On Cartesian grids as well as on nonuniform one-dimensional grids Legendre-Polynomials can be used to construct orthogonal basis functions. In this case an orthogonalization procedure as above is not necessary.

A more sophisticated method to construct an orthonormal polynomial basis is presented in [62]. The resulting basis has additional symmetry properties. Using this symmetry in the base polynomials together with symmetries in quadrature formulas can be exploited to improve the performance of the Discontinuous Galerkin method (The number of floating point operations can be reduced by exploiting these symmetries). In one space dimension the Legendre-Polynomials are already symmetric and skew symmetric. See example 6.5.1 in Section 6.5 for an exploit of symmetries in this case.

6.4 Quadrature Formulas

For the general treatment of nonlinear partial differential equations by the Discontinuous Galerkin method described above we need quadrature formulas to evaluate the *volume* and *surface* integrals that appear in the Discontinuous Galerkin formulation. In certain special cases the application of quadrature formulas can be avoided by the application of a *quadrature-free* implementation which improves the efficiency the Discontinuous Galerkin method. This is possible for example for linear equations with constant coefficients, Burgers equation or Euler equations with equation of state of a perfect gas [4], [83].

A n-dimensional quadrature formula with respect to the n-dimensional reference simplex is a set of points $\boldsymbol{x}_0, \ldots, \boldsymbol{x}_{n_q-1}$ and corresponding weights, $w_0, \ldots, w_{n_q-1}$ such that the sum

$$I_h(f) = \sum_{r=0}^{n_q-1} w_r f(\boldsymbol{x}_r)$$

approximates the integral

$$I(f) = \int_{\hat{\Delta}} f(\boldsymbol{x})\, d\boldsymbol{x}$$

in some sense for a given function $f : \mathbb{R}^n \to \mathbb{R}$.
Note: It is not required (but recommended) that the points $\boldsymbol{x}_r$ lie inside the reference simplex.

Definition 6.4.1 (Order of Quadrature Formulas)
A quadrature formula $(\boldsymbol{x}_r, w_r)_{r=0,\ldots,n_q-1}$ *is of order* $m \in \mathbb{N}\backslash\{0\}$ *if equation*

$$I_h(p) = I(p)$$

holds for all polynomials $p \in \mathbb{P}_m$.

As noted above the use of quadrature formulas with points $\boldsymbol{x}_r$ outside the reference simplex is not recommended because functions may not be defined at points they are evaluated at by the use of such a formula. Some applications have problems when formulas with negative weights are used but the Discontinuous Galerkin method (at least in our applications) seems not to be sensitive to this issue. The use of quadrature rules with negative weights results in the loss of the positivity property of the numerical integral but not in the loss of accuracy in general.

For the implementation of Discontinuous Galerkin schemes the analysis carried out by Cockburn, Hou and Shu in [23] shows that for the volume integrals quadrature formulas of order $2m$ and for the interface integrals quadrature formulas of order $2m+1$ are sufficient when polynomials of degree m are used as ansatz functions. However, in the complete linear case quadrature formulas of order $2m-1$ and $2m$ are sufficient for exact integration for the volume and interface integrals respectively. For linear source terms a volume quadrature rule of order $2m$ must be chosen. Even for nonlinear equations this choice may be sufficient as we have observed in applications with the Navier-Stokes-Korteweg system.

6.4.1 1d Quadrature Formulas

Quadrature formulas in one space dimension can be constructed very easily by computing the zeroes of Legendre polynomials to obtain the quadrature points and the corresponding weights are obtained by solving a linear system of equations. In [92] the C/C++ method *gauleg* is provided that constructs Gaussian quadrature formulas of arbitrary degree that are known to be optimal in the sense that the number of points of a quadrature rule of a given order is minimized. This means with n_q points a Gaussian quadrature formula of order $2n_q - 1$ can be constructed. The weights and points of Gaussian quadrature formulas of lower order can also be found in standard textbooks, e.g. [109] or [107].

6.4.2 2d Quadrature Formulas

Table 6.1 lists some properties and references to existing 2d quadrature formulas with respect to the triangle. All of the 2d formulas are taken from [40] some of them can also be found in [109] and in other sources.

order	number of points	remark
1	1	
2	3	
3	4	has negative weights
4	6	
5	7	has negative weights
6	12	
7	13	
8	16	
9	19	
10	25	
11	27	has negative baryzentric coordinates
12	33	
13	37	

Table 6.1: References to 2d quadrature formulas.

[40] provides integration formulas up to order 20. Some of the additional formulas not listed in Table 6.1 also have negative weights or negative coordinates.

6.4.3 3d Quadrature Formulas

Table 6.2 lists some properties and references to existing 3d quadrature formulas with respect to the tetrahedron.

Many of the 2d and 3d quadrature formulas above can be obtained from the *Encyclopedia of Cubature Formulas* website [28], [88].
Note: On Cartesian meshes in arbitrary space dimensions 1d-Gaussian quadrature formulas can be used to construct formulas of optimal order.

order	number of points	remark	reference
1	1		[109]
2	4		[109], page 307
3	5	has negative weights	[109], page 308
4	11	has negative weights	[70]
5	14		[117]
6	24		[70]
7	35	has negative weights	[117]
7	31	has negative weights	[70]
8	43	has negative weights	[10]
9	53	has negative weights has negative baryzentric coordinates	[10]
11	87	has negative weights has negative baryzentric coordinates not computed very accurately	[99]

Table 6.2: References to 3d quadrature formulas.

6.5 Implementational Details

We discuss some details on the implementation of the DG discretization for scalar first order conservation laws and scalar nonconservative equations. The extension to vector valued equations and equations with higher order derivatives is then straightforward.

Scalar first order conservation laws
We consider the Discontinuous Galerkin discretization of the scalar conservation law

$$u_t + \nabla \cdot \boldsymbol{f}(u) = 0$$

in n space dimensions. Application of the discretization given in (6.4) and using the notation of the previous sections we get for the j-th cell of the mesh

$$\begin{aligned}\int\limits_{\Delta_j} \frac{\partial}{\partial t} u^j(\boldsymbol{x},t)\varphi_k^j(\boldsymbol{x})\,d\boldsymbol{x} &= \int\limits_{\Delta_j} \boldsymbol{f}(u^j(\boldsymbol{x},t)) \cdot \nabla\varphi_k^j(\boldsymbol{x})\,d\boldsymbol{x} \\ &\quad - \sum_{e\in\partial\Delta_j} \int\limits_e g(u^j(\boldsymbol{x},t), u^{j'}(\boldsymbol{x},t), \boldsymbol{n})\,\varphi_k^j(\boldsymbol{x})\,d\sigma(\boldsymbol{x}),\end{aligned}$$

where $\varphi_k^j \in V_h$ denote the basis functions that are not identical equal to zero on the cell Δ_j for $k = 0, \ldots, n_p - 1 = |\mathbb{P}_m| - 1$. The approximate solution on the cells Δ_j and the corresponding neighboring cells $\Delta_{j'}$ are defined by

$$u^j(\boldsymbol{x},t) = \sum_{k=0}^{n_p-1} \alpha_k^j(t)\varphi_k^j(\boldsymbol{x}), \quad u^{j'}(\boldsymbol{x},t) = \sum_{k=0}^{n_p-1} \alpha_k^{j'}(t)\varphi_k^{j'}(\boldsymbol{x}).$$

By application of the transformation formula and the definition of the test functions φ_k^j in (6.10) we get

$$\begin{aligned}
&(\alpha_k^j)'(t) \int\limits_{\hat{\Delta}_n} p_k(\boldsymbol{x}) p_k(\boldsymbol{x}) |\det DT_j(\boldsymbol{x})|\, d\boldsymbol{x} \\
&= \int\limits_{\hat{\Delta}_n} \left[DT_j(\boldsymbol{x})^{-1} \boldsymbol{f}(u^j(T_j(\boldsymbol{x}),t)) \right] \cdot \nabla p_k(\boldsymbol{x})\; |\det DT_j(\boldsymbol{x})|\, d\boldsymbol{x} \\
&\quad - \sum_{i=0}^{n} \int\limits_{\hat{\Delta}_{n-1}} g\left(u^j(S_j^i(\boldsymbol{x}),t), u^{j'}(S_j^i(\boldsymbol{x}),t), \boldsymbol{n} \right) \varphi_k^j(S_j^i(\boldsymbol{x})) \sqrt{\det\left((DS_j^i)^T DS_j^i \right)}\, d\boldsymbol{x}.
\end{aligned}$$

Here $T_j : \hat{\Delta}_n \to \Delta_j$ denotes the affine linear reference mapping from the n-dimensional reference cell to the cell Δ_j and $S_j^i : \hat{\Delta}_{n-1} \to e_j^i$ the reference mapping from the $(n-1)$-dimensional reference cell to the i-th interface of the cell Δ_j. We have

$$|\det DT_j(\boldsymbol{x})| = n! \cdot |\Delta_j| \quad \text{and} \quad \sqrt{\det\left(DS_j^i(\boldsymbol{x})^T DS_j^i(\boldsymbol{x}) \right)} = (n-1)! \cdot |e_j^i|.$$

Using this, the orthogonality of the test functions and application of n-dimensional and $(n-1)$-dimensional integrations formulas with q^n and q^{n-1} points respectively gives

$$\begin{aligned}
(\alpha_k^j)'(t) &= \sum_{r=0}^{q^n-1} w_r^n \left[DT_j^{-1} \boldsymbol{f} \left(\sum_{l=0}^{n_p-1} \alpha_l^j(t) p_l(\boldsymbol{x}_r^n) \right) \right] \cdot \nabla p_k(\boldsymbol{x}_r^n) \\
&\quad - \sum_{i=0}^{n} \frac{|e_j^i|}{n|\Delta_j|} \sum_{r=0}^{q^{n-1}-1} w_r^{n-1} g(\boldsymbol{x}_r^{n-1},t)\; p_k(T_j^{-1} S_j^i(\boldsymbol{x}_r^{n-1})) \\
&\quad + R_k^j(t),
\end{aligned} \tag{6.11}$$

with the abbreviation

$$g(\boldsymbol{x}_r^{n-1},t) = g\left(\sum_{l=0}^{n_p-1} \alpha_l^j(t) p_l(T_j^{-1} S_j^i(\boldsymbol{x}_r^{n-1})),\; \sum_{l=0}^{n_p-1} \alpha_l^{j'}(t) p_l(T_{j'}^{-1} S_j^i(\boldsymbol{x}_r^{n-1})),\; \boldsymbol{n} \right).$$

$R_k^j(t)$ denotes the (small) approximation error of the quadrature formulas and is neglected in the implementation. Note that an evaluation of the basis functions p_k is necessary only once at the beginning of the computation for the volume integrals. This does also hold for the surface integrals when the maximum level of nonconformity is restricted in the mesh.

Scalar nonconservative equations
We consider the nonconservative scalar equation in multiple space dimensions

$$u_t + a(u, \nabla u, \boldsymbol{x}, t) = 0.$$

Here a is of the form

$$a(u, \nabla u, \boldsymbol{x}, t) = b(u, x, t) + \sum_{i=1}^{n} a_i(u, x, t) \frac{\partial}{\partial x_i} u$$

and includes also a source term b. We apply the discretization given by equation (6.8). The computation of the surface integral is similar to the computation of the surface integral in the conservative case discussed in the previous paragraph. So we omit it and discuss only the computation of the volume integral. With the notation of the previous paragraph we have

$$\begin{aligned}
&\int_{\Delta_j} a\left(u_j(\boldsymbol{x}, t), \nabla u_j(\boldsymbol{x}, t), \boldsymbol{x}, t\right) \ \varphi_k^j(\boldsymbol{x}) \, d\boldsymbol{x} \\
&= \int_{\Delta_j} a\left(\sum_{l=0}^{n_p-1} \alpha_l^j(t) p_l(T_j^{-1}\boldsymbol{x}), \ \sum_{l=1}^{n_p-1} \alpha_l^j(t)(DT_j)^{-T} \nabla p_l(T_j^{-1}\boldsymbol{x}), \ \boldsymbol{x}, \ t\right) p_k(T_j^{-1}\boldsymbol{x}) \, d\boldsymbol{x} \\
&= n! \cdot |\Delta_j| \int_{\hat{\Delta}_n} a\left(\sum_{l=0}^{n_p-1} \alpha_l^j(t) p_l(\boldsymbol{x}), \ \sum_{l=1}^{n_p-1} \alpha_l^j(t)(DT_j)^{-T} \nabla p_l(\boldsymbol{x}), \ \boldsymbol{x}, \ t\right) p_k(\boldsymbol{x}) \, d\boldsymbol{x}
\end{aligned}$$

by application of the transformation formula. Note that p_0 is a constant and can be omitted in the computation of ∇u_j. For implementation a suitable quadrature formula as to be applied in general. This is the same as in the conservative case.

Example 6.5.1 *(Symmetry Exploit)*

In (6.11) we have seen that for the computation of the volume integral over the cell Δ_j the approximate solution u_h has to be evaluated at the integration points $T_j(x_r)$. The same can be done with less computational cost using symmetries in the basis polynomials and in the quadrature formulas. We consider the one dimensional case. In this the basis polynomials are given by scaled Lagrange polynomials. For simplicity we assume that we have an even number n_p of local basis functions. As quadrature formulas we choose a Gaussian quadrature formula with an even number of points n_q. Both, the quadrature formulas and the basis function have symmetries. We have the properties

$$\begin{aligned}
p_l(x_r) &= p_l(x_{n_q-r-1}) && \text{if } l \text{ is an even number,} \\
p_l(x_r) &= -p_l(x_{n_q-r-1}) && \text{if } l \text{ is an odd number}
\end{aligned}$$

for $l = 0, \ldots, n_p - 1$ and $r = 0, \ldots, n_q - 1$.

For $r = 0, \ldots, n_q - 1$ it is necessary to compute

$$u_h(T_j(x_r)) = \sum_{l=0}^{n_p-1} \alpha_l^j \ p_l(x_r).$$

Using the properties above this can be done also in the following way. For $r = 0, \dots, \frac{n_q}{2} - 1$ compute

$$\begin{aligned}
\beta_r &= \sum_{l=0}^{\frac{n_p}{2}-1} \alpha_{2l}\, p_{2l}(x_r), \\
\gamma_r &= \sum_{l=0}^{\frac{n_p}{2}-1} \alpha_{2l+1}\, p_{2l+1}(x_r), \\
u_h(T_j(x_r)) &= \beta_r + \gamma_r, \\
u_h(T_j(x_{n_q-r-1})) &= \beta_r - \gamma_r.
\end{aligned}$$

In this approach only half the number of multiplication and less additions are necessary to evaluate the approximate solution at the quadrature points. Note that this is not restricted to even number of basis polynomials and quadrature points. A similar approach is also possible in multiple space dimensions when the orthogonal basis functions have additional symmetry properties. In [62] such a kind of basis polynomials are constructed.

6.6 Simple Examples

This section is dedicated to the Local Discontinuous Galerkin discretization of two simple examples in one space dimensions including higher order terms. In Section 6.2 we have already discussed the discretization of the scalar convection-diffusion equation in multiple space dimensions as the method was proposed by Bassi and Rebay [8]. Here we give other possible numerical fluxes like these proposed by Cockburn and Shu in [25]. The second example is the LDG discretization of an equation that includes third order terms. This equation serves as a model problem for the Navier-Stokes-Korteweg system not because it is related to this system in special situations but the way it is discretized is similar.

6.6.1 Nonlinear Convection-Diffusion Equation in 1d

In Example 6.2.1 we already discussed the multidimensional nonlinear convection-diffusion equation and we discretized it by application of the original LDG method proposed by Bassi and Rebay [8]. Here we consider this equation again in one space dimension for simplicity and give alternative discretizations with superior properties proposed by Cockburn and Shu [25]. We consider the problem

$$\begin{aligned}
u_t + F(u)_x &= \varepsilon u_{xx} \ \text{ in } \mathbb{R} \times \mathbb{R}_{>0}, \\
u(\cdot, 0) &= u_0 \ \text{ in } \mathbb{R},
\end{aligned}$$

where $F : \mathbb{R} \to \mathbb{R}$ is some smooth (in general nonlinear) function and the constant $\varepsilon > 0$ is supposed to be *small* such that the equation is convection dominated and the Discontinuous Galerkin discretization is an appropriate choice.

We rewrite the equation using two first order differential operators $\mathcal{L}_1^1$ and $\mathcal{L}_2^1$ as in Section 6.2

$$u_t + \mathcal{L}_2^1\left[u,\ \mathcal{L}_1^1[u]\right] = 0,$$

the differential operators are defined by

$$\begin{aligned} \mathcal{L}_1^1[u] &= f_1^1(u)_x, & f_1^1(u) &= u, \\ \mathcal{L}_2^1[u,v] &= f_1^2(u,v)_x, & f_1^2(u,v) &= F(u) - \varepsilon v. \end{aligned}$$

Here f_1^1 and f_1^2 denote the physical fluxes. For the complete numerical discretization we have to define consistent numerical fluxes, We choose the Local Lax-Friedrichs flux for the convective part of the equation and a family of viscous fluxes parameterized by $\xi \in [0,1]$ for the viscous part of the equation.

$$\begin{aligned} g^1(u^-,u^+,n) &= \left(\xi u^- + (1-\xi)u^+\right)n, \\ g^2(u^-,v^-,u^+,v^+,n) &= \tfrac{1}{2}\left(F(u^-)+F(u^+)\right)n - \tfrac{\alpha}{2}(u^+-u^-) \\ &\quad -\varepsilon\left((1-\xi)v^- + \xi v^+\right)n, \end{aligned} \tag{6.12}$$

where n denotes the one dimensional normal (i.e. 1 or -1) and α is chosen to be equal to the fastest wave speed $\alpha = \max(|F'(u^-)|,|F'(u^+)|)$. For simplicity we omit the treatment of boundary conditions.

Choosing the parameter ξ equal to zero or one means that in a first order scheme the viscous part of the equation is discretized by the usual three point stencil

$$\frac{\varepsilon}{h^2}\left(u_{i-1} - 2u_i + u_{i+1}\right),$$

whereas the choice $\xi = \frac{1}{2}$ leads to a discretization by the spread out five point wide stencil

$$\frac{\varepsilon}{4h^2}\left(u_{i-2} - 2u_i + u_{i+2}\right).$$

Numerical experiments show that the latter choice can lead to a suboptimal order of convergence in the case where polynomials of odd degree are used as ansatz functions, see [102]. This is not a problem with the choice $\xi = 0$ and $\xi = 1$ but this choice can lead to problems with boundary conditions, see also Section 6.9.

For a first order scheme it is clear that the numerical (artificial) viscosity can be omitted in the case $\xi = 0$ and $\xi = 1$, the method proposed by Cockburn and Shu [25], provided that the computational mesh is sufficiently fine or the time step sufficiently small in the fully discrete scheme. A linear stability analysis shows that it is not possible to stabilize the scheme by physical viscosity when the parameter $\xi = \frac{1}{2}$, this is the original scheme proposed by Bassi and Rebay [8] applied to the scalar convection-diffusion equation, is chosen. Numerical experiments show that the same is true for the higher order schemes.

6.6.2 Nonlinear Convection-Diffusion-Dispersion Equation in 1d

As a scalar model problem for the Navier-Stokes-Korteweg system we consider the scalar equation in one space dimension

$$\begin{aligned} u_t + f(u)_x &= \varepsilon u_{xx} + \lambda u_{xxx} \text{ in } \Omega \times \mathbb{R}_{>0}, \\ u(\cdot, 0) &= u_0 \text{ in } \Omega, \end{aligned}$$

where $f : \mathbb{R} \to \mathbb{R}$ is some smooth function and ε, λ are positive constants. For simplicity we set the interval $\Omega = (0, 1)$ and consider periodic boundary conditions.

This equation serves as a model problem for the NSK system not because it is related to this system in special situations but the way it is discretized by the local Discontinuous Galerkin approach is similar. We apply the general approach from Section 6.2 to the above equation. This means we rewrite the equation as a combination of three first order differential operators. We omit the explicit definition of the differential operators, the physical, the numerical fluxes and we write down the discretization of the system as we need it in Section 6.8. We rewrite the third order equation formally as a system of first order equations by the introduction of new variables p and q.

$$\begin{aligned} u_t &+ (f(u) - p)_x &= 0, \\ p &- (\varepsilon u + q)_x &= 0, \\ q &- (\lambda u)_x &= 0. \end{aligned}$$

Application of the Local Discontinuous Galerkin discretization discussed in Section 6.2 leads to the following semi-discrete problem. Find functions $u(\cdot, t),\ p,\ q \in V_h = \{\varphi : \Omega \to \mathbb{R} \mid \varphi|_{\Delta_j} \in \mathbb{P}_k(\Delta_j)\}$ such that the equations

$$\begin{aligned} \int\limits_{\Delta_j} u_t v\, dx - \int\limits_{\Delta_j} (f(u) - p) v_x\, dx &+ (\hat{f} - \hat{p})_{j+\frac{1}{2}} v^-_{j+\frac{1}{2}} - (\hat{f} - \hat{p})_{j-\frac{1}{2}} v^+_{j-\frac{1}{2}} &= 0 \\ \int\limits_{\Delta_j} p w\, dx + \int\limits_{\Delta_j} (\varepsilon u + q) w_x\, dx &- (\varepsilon \hat{u} + \hat{q})_{j+\frac{1}{2}} w^-_{j+\frac{1}{2}} + (\varepsilon \hat{u} + \hat{q})_{j-\frac{1}{2}} w^+_{j-\frac{1}{2}} &= 0 \\ \int\limits_{\Delta_j} q z\, dx + \int\limits_{\Delta_j} (\lambda u) z_x\, dx &- (\lambda \hat{u})_{j+\frac{1}{2}} z^-_{j+\frac{1}{2}} + (\lambda \hat{u})_{j-\frac{1}{2}} z^+_{j-\frac{1}{2}} &= 0 \end{aligned} \tag{6.13}$$

are satisfied for all piecewise polynomial test functions $v, w, z \in V_h$ and all cells Δ_j. Here we denote the numerical fluxes by the hat functions and a values $\varphi^{\pm}_{j+\frac{1}{2}}$ denote the values of the function $\varphi \in V_h$ at the interface $x_{j+\frac{1}{2}}$ of the cell $\Delta_j = \left(x_{j-\frac{1}{2}}, x_{j+\frac{1}{2}}\right)$ extrapolated from the right and left respectively. Note that the function in general is discontinuous at the interface. As numerical flux for the convective part of the equation

$$\hat{f}_{j+\frac{1}{2}} = \hat{f}(u^+_{j+\frac{1}{2}}, u^-_{j+\frac{1}{2}}) \tag{6.14}$$

we choose some general monotone, Lipschitz continuous and consistent numerical flux, for example the Local Lax-Friedrichs flux as in the previous paragraph. The remaining numerical fluxes we define as follows:

$$\begin{aligned}
\hat{q}_{j+\frac{1}{2}} &= \hat{q}\left(q^-_{j+\frac{1}{2}}, q^+_{j+\frac{1}{2}}\right) &&= \tfrac{1}{2}(q^+_{j+\frac{1}{2}} + q^-_{j+\frac{1}{2}}),\\
\hat{p}_{j+\frac{1}{2}} &= \hat{p}\left(p^-_{j+\frac{1}{2}}, p^+_{j+\frac{1}{2}}\right) &&= \xi p^+_{j+\frac{1}{2}} + (1-\xi)p^-_{j+\frac{1}{2}}, \qquad (6.15)\\
\hat{u}_{j+\frac{1}{2}} &= \hat{u}\left(u^-_{j+\frac{1}{2}}, u^+_{j+\frac{1}{2}}\right) &&= (1-\xi)u^+_{j+\frac{1}{2}} + \xi u^-_{j+\frac{1}{2}},
\end{aligned}$$

for some constant $\xi \in [0,1]$. Note that the equations p and q can be eliminated locally such that it is not necessary to solve a larger system of equations. This property is responsible for the term *Local* in the Local Discontinuous Galerkin method. The discrete initial data can be constructed by L^2-projection to the Finite Element space V_h.

For the above given discretization we prove a L^2 stability result similar to that given in [130] in Section 6.8.

6.7 Summary of Theoretical Results

In this section we state some of the existing results about the Discontinuous Galerkin discretization for conservation laws and Local Discontinuous Galerkin methods for convection-diffusion equations and equations with higher order derivatives. The Local Discontinuous Galerkin discretization was proposed by Bassi and Rebay in [8] (they applied the method to the compressible Navier-Stokes equations) and further developed by Cockburn and Shu, see for example [25]. Cockburn, Shu and coworkers give several theoretical results for scalar model problems. Results concerning multidimensional systems are rare or do not exist.

Early results
Reed and Hill [94] introduced the Discontinuous Galerkin method in 1973 for the time independent, scalar, multidimensional, linear transport equation

$$\nabla \cdot (\boldsymbol{a}u) + bu = f \quad \text{in } \Omega$$

with appropriate boundary conditions. For the discretization they used numerical upwind fluxes. LeSaint and Raviart [80] proved that the L^2-error of the approximate solution is of order k when local base polynomials of degree k are used on general triangulations. Johnson and Pitkäranta [66] showed that the approximate solution converges with order $k + \frac{1}{2}$ to the exact solution of the problem.

Many references for the Discontinuous Galerkin method and its recent development can be found in the review paper [27].

Nonlinear scalar conservation laws
Zhang and Shu [133] considered the nonlinear scalar conservation law in multiple space

dimensions

$$\begin{aligned} u_t + \nabla \cdot \boldsymbol{f}(u) &= 0 \ \text{ in } \Omega \times (0,T), \\ u(\cdot,0) &= u_0 \ \text{ in } \Omega, \end{aligned}$$

where $f : \mathbb{R} \to \mathbb{R}^n$ is a sufficiently smooth vector field and $u_0 : \mathbb{R} \to \mathbb{R}$ denotes the smooth initial data. The equation is discretized in space by the Discontinuous Galerkin approach of arbitrary degree (polynomial degree $k \geq 1$) using general monotone numerical fluxes. Only Cartesian meshes are considered. In the multidimensional case tensor products of 1d base functions are used as ansatz functions. Time integration is done by application of the second order Runge-Kutta method TVD2 described in Section 7.2 (the generalization of the statements below to higher order Runge-Kutta schemes like TVD3 is a nontrivial task). In [133] Zhang and Shu obtained error estimates for smooth solutions of the scalar conservation law. They proved that the error is of order

- $O\left(h^{k+1} + \Delta t^2\right)$ in the nonlinear one dimensional case and in the linear multidimensional case. Both cases require $k = 1$ and the usual CFL-condition $\Delta t \leq C_{CFL}\, h$,
- $O\left(h^{k+\frac{1}{2}} + \Delta t^2\right)$ for $k \geq 2$ in the nonlinear multidimensional case with a more restrictive CFL-condition $\Delta t < C_{CFL}\, h^{\frac{4}{3}}$.

As usual h denotes the mesh size and Δt the time step size. The authors do not pay attention to the treatment of boundary conditions. Thus, periodic boundary conditions or compactly supported initial data is considered.

Note: Since only smooth solutions are considered a slope limiting procedure, that is usually necessary for first order conservation laws when discontinuities are present, is not necessary to maintain the stability of the method, see for example the review paper [27].

Convection-diffusion equations

Originally Bassi and Rebay proposed the Local Discontinuous Galerkin (LDG) method in application to the compressible Navier-Stokes equations [8]. A further development of this method was carried out by Cockburn and Shu within the general framework of convection-diffusion equations [25]. The method is especially well suited for convection dominated systems. In [25] the following class of equations is considered:

$$\begin{aligned} u_t + \nabla \cdot \boldsymbol{f}(u, \nabla u) &= 0 \ \text{ in } \Omega \times (0,T), \\ u(\cdot,0) &= u_0 \ \text{ in } \Omega, \end{aligned}$$

where $\boldsymbol{f} : \mathbb{R} \times \mathbb{R}^n \to \mathbb{R}^n$ is a smooth function that is linear in the second argument (∇u). Under further assumptions on the function $\boldsymbol{f}$ the authors applied a similar (in the general nonlinear case more sophisticated) discretization as we discussed it in the Section 6.6. They proved a L^2-stability result for a semi-discrete solution which leads to an error estimate in the L^2-norm for the linear case with constant coefficients. The error of the semi-discrete solution is of order Ch^k when a polynomials of degree k are used and the order of the constant C varies between 1 and h dependent on the polynomial degree and the regime of the coefficients (purely hyperbolic / purely parabolic case). For simplicity periodic domains are considered.

The Local Discontinuous Galerkin method is not the only DG type method for this type of equations. There are a lot of other method that come from the DG discretization of elliptic problems. In [3] all the available methods for elliptic equations are compactly presented and they are compared to each other numerically in [20]. Especially the method by Baumann and Oden (BO) [9] seems to be attractive for the treatment of convection-diffusion systems. A comparison between the Local Discontinuous Galerkin method and the Baumann and Oden method in the framework of convection-diffusion equations can be found in [102], we summarize the advantages and disadvantages of the LDG method over the BO method as follows. The underlying benchmark in [102] was the discretization of the one dimensional heat equation.

+ In the test with the 1d heat equation the LDG method produces much smaller errors than the BO method.

+ The LDG method converges with optimal order for polynomials of all degrees k. The BO method is not optimal for even k.

- The LDG method has a higher computational cost than the BO method. But in the test this is amortized by the smaller errors.

- In parallel implementations the LDG method has a higher communication cost than the BO method.

The main advantage of the LDG method over the Baumann and Oden method is that the LDG method can be easily generalized to systems with third or even higher order derivatives. We will see this in the next paragraph. This is not possible with the Baumann and Oden method, at least it is not straightforward to do.

KdV type equations
Yan and Shu [130] considered a general class of scalar nonlinear KdV like equations in multiple space dimensions of the form

$$\begin{aligned} u_t + \sum_{i=1}^{n} \frac{\partial}{\partial_{x_i}} \left(f_i(u) + r_i'(u) \sum_{j=1}^{n} g_{ij}(r_i(u)_{x_i})_{x_j} \right) &= 0 \text{ in } \Omega \times (0,T), \\ u(\cdot,0) &= u_0 \text{ in } \Omega, \end{aligned}$$

where f_i, r_i and g_{ij} are smooth scalar valued functions. The boundary is assumed to be periodic to avoid the complicated treatment of boundary conditions. For a semi-discrete Local Discontinuous Galerkin formulation of this equation similar to the one given for the convection-diffusion-dispersion example in Section 6.6 Yan and Shu obtained a cell entropy inequality for the *square entropy* in the multidimensional case which leads to a L^2-stability result (of course the solution of the continuous problem is L^2-stable, provided that it exists). This result holds for arbitrary nonconform simplicial meshes with possibly hanging nodes. In the one dimensional linear case this leads to an error estimate in the L^2-norm which is of order $O(h^{k+1/2})$ when polynomials of degree k are used. A slope limiting technique as for first order conservation laws is not necessary to guarantee the stability of the approximate solution.

In contrast to the convection-diffusion equation, discussed in the paragraph before, there is no alternative Discontinuous Galerkin method such as the Baumann-Oden method for these kinds of equations including third order derivatives. Thus, the application of the Local Discontinuous Galerkin method seems to be appropriate for the discretization of the Navier-Stokes-Korteweg equations. The treatment of viscous terms is missing in [130]. Therefore we give a similar stability result in one space dimension for the simple convection-diffusion-dispersion example discussed in Section 6.6 in the next section.

6.8 L^2-Stability of the LDG-Discretization for a Model Problem

We give a cell entropy inequality and a resulting L^2-stability estimate for the semi-discrete Discontinuous Galerkin discretization for the one dimensional scalar convection-diffusion-dispersion equation discussed in Section 6.6, see (6.13).

Theorem 6.8.1 *(Cell Entropy Inequality)*
Let $u \in C^1((0,\infty), V_h)$ be a solution of the semi-discrete Local Discontinuous Galerkin formulation (6.13) with numerical fluxes (6.14) and (6.15). Then there exist numerical entropy fluxes $H_{j-\frac{1}{2}}$ *such that the semi-discrete solution satisfies the entropy inequality*

$$\frac{d}{dt}\int_{\Delta_j} \frac{1}{2}u^2\,dx + H_{j+\frac{1}{2}} - H_{j-\frac{1}{2}} \leq 0 \tag{6.16}$$

for all cells Δ_j.

Proof. We denote the sum of the left hand sides of the three equations in (6.13) by $B_j(u,p,q;\ v,w,z)$ and find that this is equal to zero for all $v,w,z \in V_h$. We choose special test functions

$$v = u, \quad w = -\frac{1}{\lambda}q, \quad z = \frac{1}{\lambda}p + \frac{\varepsilon}{\lambda^2}q,$$

and with a primitive function F of f we find that

$$\begin{aligned}
0 &= B_j\left(u,p,q;\;\; u,\; -\frac{1}{\lambda}q,\; \frac{1}{\lambda}p+\frac{\varepsilon}{\lambda^2}q\right)\\
&= \frac{d}{dt}\int\limits_{\Delta_j}\frac{1}{2}u^2\,dx+\frac{\varepsilon}{\lambda^2}\int\limits_{\Delta_j}q^2\,dx+\int\limits_{\Delta_j}\frac{\partial}{\partial x}\left(pu-F(u)-\frac{1}{2\lambda}q^2\right)\,dx\\
&\quad +(\hat f-\hat p)_{j+\frac12}u^-_{j+\frac12}-(\hat f-\hat p)_{j-\frac12}u^+_{j-\frac12}\\
&\quad +\frac{1}{\lambda}(\varepsilon\hat u+\hat q)_{j+\frac12}q^-_{j+\frac12}-\frac{1}{\lambda}(\varepsilon\hat u+\hat q)_{j-\frac12}q^+_{j-\frac12}\\
&\quad -\hat u_{j+\frac12}\left(p^-_{j+\frac12}+\frac{\varepsilon}{\delta}q^-_{j+\frac12}\right)+\hat u_{j-\frac12}\left(p^+_{j-\frac12}+\frac{\varepsilon}{\delta}q^+_{j-\frac12}\right)\\
&= \frac{d}{dt}\int\limits_{\Delta_j}\frac{1}{2}u^2\,dx+\frac{\varepsilon}{\lambda^2}\int\limits_{\Delta_j}q^2\,dx+H_{j+\frac12}-H_{j-\frac12}+K_{j-\frac12}.
\end{aligned}$$

The quantities $H_{j-\frac12}$ and $K_{j-\frac12}$ are defined by the relations

$$\begin{aligned}
H_{j-\frac12} &= p^-_{j-\frac12}u^-_{j-\frac12}-F(u^-_{j-\frac12})-\frac{1}{2\lambda}(q^-_{j-\frac12})^2\\
&\quad +(\hat f-\hat p)_{j-\frac12}u^-_{j-\frac12}+\frac{1}{\lambda}(\varepsilon\hat u+\hat q)_{j-\frac12}q^-_{j-\frac12}-\hat u_{j-\frac12}(p^-_{j-\frac12}+\frac{\varepsilon}{\lambda}q^-_{j-\frac12})
\end{aligned}$$

and

$$\begin{aligned}
K_{j-\frac12} &= \int\limits_{u^-}^{u^+}\left(f(u)-\hat f(u^-,u^+)\right)\,du\\
&\quad +\frac{1}{\lambda}\left(\frac{1}{2}(q^++q^-)(q^+-q^-)-\hat q(q^-,q^+)(q^+-q^-)\right)\\
&\quad +\hat p(p^-,p^+)(u^+-u^-)+\hat u(u^-,u^+)(p^+-p^-)-p^+u^++p^-u^-.
\end{aligned}$$

The subscripts $j-\frac12$ for the values u^-,u^+,p^-,p^+,q^-,q^+ were omitted in the definition of $K_{j-\frac12}$ for notational simplicity. Using the properties of the numerical fluxes (6.14) and (6.15) we see that the quantity $K_{j-\frac12}$ is positive. The integral in the above expression is positive since $\hat f$ is a monotone flux and everything else vanishes. As a result we get the cell entropy inequality

$$\frac{d}{dt}\int\limits_{\Delta_j}\frac{1}{2}u^2\,dx+\frac{\varepsilon}{\lambda^2}\int\limits_{\Delta_j}q^2\,dx+H_{j+\frac12}-H_{j-\frac12}\le 0. \tag{6.17}$$

This completes the proof. Note that this is a sharper estimate than the one stated in the theorem above. The function q is an approximation to λu_x and therefore the second integral an approximation to $\varepsilon\int u_x^2\,dx$. The cell entropy inequality above immediately leads to a L^2-stability of the semi-discrete solution.

Corollary 6.8.2 *(L^2-stability)*
Let $u \in C^1((0,\infty), V_h)$ be a solution of the semi-discrete Local Discontinuous Galerkin formulation (6.13) with numerical fluxes (6.14) and (6.15). Then the semi-discrete solution satisfies the L^2-stability estimate

$$\frac{d}{dt}\int\limits_{\Delta_j} \frac{1}{2}u^2\,dx \le 0.$$

Proof. Using the cell entropy inequality (6.17) and summing over all cells gives

$$\frac{d}{dt}\int\limits_{\Omega} \frac{1}{2}u^2\,dx \le -\frac{\varepsilon}{\lambda^2}\int\limits_{\Omega} q^2\,dx \tag{6.18}$$

which is a sharper estimate than the one stated above. Note that periodic boundary conditions are used and therefore all numerical entropy fluxes $H_{j-\frac{1}{2}}$ have a counter part.

6.9 Navier-Stokes-Korteweg DG-Discretization

In this section we finally describe the Local Discontinuous Galerkin space discretization of the NSK equations. This is done on the basis of the background of the previous sections. The discretization of the isothermal version of the Navier-Stokes-Korteweg system in multiple space dimensions as well as the discretization of the 2d (full) NSK model is discussed in detail. The extension to three space dimensions is omitted for the full system for simplicity. It is straightforward to do.

6.9.1 1d isothermal

We consider the Local Discontinuous Galerkin space discretization of the one dimensional isothermal Navier-Stokes-Korteweg equations (2.55), (2.56) in the nonconservative formulation

$$\begin{array}{rcrcl} \rho_t & + & (\rho u)_x & = & 0, \\ (\rho u)_t & + & (\rho u^2)_x + \rho\kappa_x & = & \varepsilon u_{xx}, \end{array} \quad \text{in } \Omega \times (0,T)$$

with $\kappa = \mu(\rho) - \lambda\rho_{xx}$ and boundary conditions

$$u = 0, \quad \rho_x = 0 \ \text{ on } \partial\Omega \times (0,T)$$

and the usual initial conditions. We rewrite the third order system as a larger formally first order system.

$$\begin{array}{rcrcl} \begin{pmatrix} \rho_x \\ u_x \end{pmatrix} & - & \mathcal{L}_1^1[\rho, \rho u] & = & 0, \\ \kappa & - & \mathcal{L}_2^1[\rho, \rho_x] & = & 0, \\ \frac{\partial}{\partial t}\begin{pmatrix} \rho \\ \rho u \end{pmatrix} & + & \mathcal{L}_3^1[\rho, \rho u, u_x, \kappa] & = & 0. \end{array}$$

The first order differential operators $\mathcal{L}_1^1$, $\mathcal{L}_2^1$, $\mathcal{L}_3^1$ are defined by

$$\begin{aligned}
\mathcal{L}_1^1[\rho, \rho u] &= \frac{\partial}{\partial x}\left(\frac{\rho}{\frac{\rho u}{\rho}}\right), \\
\mathcal{L}_2^1[\rho, \rho_x] &= \mu(\rho) - \lambda \frac{\partial}{\partial x}\rho_x, \\
\mathcal{L}_3^1[\rho, \rho u, u_x, \kappa] &= \frac{\partial}{\partial x}\begin{pmatrix} \rho u \\ \rho u^2 - \varepsilon u_x \end{pmatrix} + \begin{pmatrix} 0 \\ \rho \frac{\partial}{\partial x}\kappa \end{pmatrix}.
\end{aligned} \tag{6.19}$$

Here we have all kinds of operators discussed in section 6.2. The blue terms belong to conservative parts of the differential operators. The $\mu(\rho)$ term in operator $\mathcal{L}_2^1$ has the character of a source term and the red part of $\mathcal{L}_3^1$ is a non-conservative product.

To complete the discretization we have to choose suitable numerical fluxes. At the inner cell interfaces we choose

$$\begin{aligned}
\boldsymbol{g}^1(\rho^\pm, \rho u^\pm, n) &= n\begin{pmatrix} \{\rho\} \\ \left\{\frac{\rho u}{\rho}\right\}_\xi \end{pmatrix}, \\
\boldsymbol{g}^2(\rho^\pm, \rho_x^\pm, n) &= -n\lambda\{\rho_x\}, \\
\boldsymbol{g}^3(\rho^\pm, \rho u^\pm, u_x^\pm, \kappa^\pm, n) &= \begin{pmatrix} \{\rho u\}n - \frac{\alpha_1}{2}[\kappa] \\ \{\rho u^2\}n - \varepsilon\,\{u_x\}_{1-\xi} n + \zeta\{\rho\}[\kappa]n - \frac{\alpha_2}{2}[\rho u] \end{pmatrix}.
\end{aligned} \tag{6.20}$$

Here and in the following $\{\varphi\}_\xi = \xi\varphi^- + (1-\xi)\varphi^+$ for $\xi \in [0,1]$ denotes the weighted average and $[\varphi] = (\varphi^+ - \varphi^-)$ the jump over the interface between cells for an element $\varphi \in V_h$. In the case were ξ is equal to $\frac{1}{2}$ we omit the parameter, i.e., $\{\varphi\} = \frac{1}{2}(\varphi^- + \varphi^+)$ denotes the arithmetic average. As a convention in one space dimension the $-$ values denote the values on the left side of the cell interface and $+$ the values on the right. The normal n in 1d has the values -1 or 1.

The colored terms are related to the colored terms in the differential operators. The green terms represent the numerical viscosity that we have introduced in Section 5.2. The parameters α_1 and α_2 are chosen in the same way as in Section 5.2. In $\boldsymbol{g}^3$ the averaging of conservative and nonconservative terms are combined, see (6.4) and (6.8). The parameter ζ controls the ratio in the averaging of the test function. We always choose $\zeta = \frac{1}{2}$ which leads to a central scheme. *Source terms* as $\mu(\rho)$ in $\boldsymbol{g}^2$ do not give a contribution to the numerical fluxes.

The parameter α_2 can be set to zero in the case where the momentum equation is stabilized by physical viscosity. It depends on the parameter $\xi \in [0,1]$ and the mesh size if this is the case or not. This parameter has the same meaning as in Section 6.6.1 and can be chosen differently as discussed in the following. If it is chosen globally equal to one or zero this leads to problems with the boundary condition $u = 0$ since the condition is taken into account only on one side of the interval Ω. Possible choices are listed below.

- It can be chosen locally such that all boundary conditions can be taken into account. It should vary smoothly between the cells of the grid.

- It can be chosen always equal to $\frac{1}{2}$. In this case the scheme cannot be stabilized by the physical viscosity and one has to impose artificial (numerical) viscosity to the discrete momentum equation, i.e., $\alpha_2 > 0$. A suboptimal order of convergence as for the heat equation has not been observed in the numerical tests, see the discussion in section 6.6.

- It can be switched from 1 to 0 and vice versa every time step. This is of course a dirty hack but works quite well in practical applications.

A similar approach with the combination of backward an forward differences is also possible for the second derivative $\lambda\rho_{xx}$ in the definition of κ. But in the multidimensional case on unstructured grids numerical experiments show that this can lead to problems with symmetries and thus, an unstable behavior. Therefore we build the arithmetic average twice.

For the correct treatment of the boundary conditions we choose *numerical boundary fluxes* $\boldsymbol{g}_b^i$, $i = 1, 2, 3$ at boundary interfaces. Let, without loss of generality, the $-$ values be in the interior of the domain and the $+$ values at the boundary. We set

$$\begin{aligned} u^+ &= 0, \\ \rho^+ &= \rho^-, \\ \rho u^+ &= \rho^+ u^+, \\ u_x^+ &= u_x^-, \end{aligned}$$

and with this we can define the *numerical boundary fluxes*

$$\begin{aligned} \boldsymbol{g}_b^1(\rho^\pm, \rho u^\pm, n) &= n \begin{pmatrix} \rho \\ \left\{\frac{\rho u}{\rho}\right\}_\xi \end{pmatrix}, \\ \boldsymbol{g}_b^2(\rho^\pm, \rho_x^\pm, n) &= 0, \\ \boldsymbol{g}_b^3(\rho^\pm, \rho u^\pm, u_x^\pm, \kappa^\pm, n) &= \begin{pmatrix} \{\rho u\} n \\ \{\rho u^2\} n - \varepsilon \{u_x\}_{1-\xi} n - \frac{\alpha_2}{2}[\rho u] \end{pmatrix}. \end{aligned} \tag{6.21}$$

The parameter ξ has the same meaning as before and jumps in κ do not appear at the boundary. This completes the Local Discontinuous Galerkin discretization of the one dimensional isothermal Navier-Stokes-Korteweg system.

6.9.2 2d isothermal

We consider the two dimensional Navier-Stokes-Korteweg system. In contrast to the one dimensional case we include the effect of gravity. Instead of adding the standard source term $\rho\boldsymbol{g}$ to the momentum equation we combine this term with the variable κ which leads to a well balanced scheme when gravity is present. In the following we denote the spatial coordinates by $\boldsymbol{x} = (x, y)$ and the velocity of the fluid by $\boldsymbol{u} = (u, v)$.

$$\left.\begin{aligned} \rho_t &+ \nabla\cdot(\rho\boldsymbol{u}) &= 0, \\ (\rho\boldsymbol{u})_t &+ \nabla\cdot(\rho\boldsymbol{u}\boldsymbol{u}^T) + \rho\nabla\kappa &= \nabla\cdot\boldsymbol{\tau}, \end{aligned}\right. \quad \text{in } \Omega\times(0,T)$$

with $\kappa = \mu(\rho) - \lambda\Delta\rho - (g_x x + g_y y)$ and boundary conditions

$$\boldsymbol{u} = 0 \quad \text{and} \quad -\frac{\nabla\rho}{|\nabla\rho|}\cdot\boldsymbol{n} = \cos(\varphi). \quad \text{on } \partial\Omega\times(0,T).$$

In the definition of κ the effect of gravity is included. The constants g_x and g_y denote the gravitational force in x and y direction respectively.

In two space dimensions the viscous contribution of the stress tensor can be rewritten as

$$\nabla\cdot\boldsymbol{\tau} = \begin{pmatrix} \varepsilon(u_x+v_y)_x & + & \mu(u_y-v_x)_y \\ -\mu(u_y-v_x)_x & + & \varepsilon(u_x+v_y)_y \end{pmatrix}. \tag{6.22}$$

Here we set $\varepsilon = 2\mu+\nu$ where μ and ν denote the coefficients of viscosity. With this approach we have to deal only with the two quantities u_x+v_y and u_y-v_x in the Local Discontinuous Galerkin discretization of the two dimensional NSK system instead of using the four quantities u_x, u_y, v_x and v_y. This saves computational cost and more important with this approach the treatment of boundary conditions that involve tangential and normal velocities is very easy since the terms

$$\begin{pmatrix} u \\ v \end{pmatrix}\cdot\boldsymbol{n} \quad \text{and} \quad \begin{pmatrix} -v \\ u \end{pmatrix}\cdot\boldsymbol{n}$$

appear in the corresponding numerical boundary fluxes. These two terms are the normal and tangential velocity respectively.

As in the one dimensional case we rewrite the third order system as a larger formally first order system.

$$\begin{array}{rcll} \begin{pmatrix} \rho_x \\ \rho_y \\ (u_x+v_y) \\ (u_y-v_x) \end{pmatrix} & - & \mathcal{L}_1^1[\rho,\rho\boldsymbol{u}] & = 0, \\ \kappa & - & \mathcal{L}_2^1[\rho,\rho_x,\rho_y] & = 0, \\ \frac{\partial}{\partial t}\begin{pmatrix} \rho \\ \rho u \\ \rho v \end{pmatrix} & + & \mathcal{L}_3^1[\rho,\rho\boldsymbol{u},(u_x+v_y),(u_y-v_x),\kappa] & = 0. \end{array} \tag{6.23}$$

The first order differential operators $\mathcal{L}_1^1$, $\mathcal{L}_2^1$, $\mathcal{L}_3^1$ are defined by

$$\begin{aligned} \mathcal{L}_1^1[\rho,\rho\boldsymbol{u}] &= \frac{\partial}{\partial x}\begin{pmatrix} \rho \\ 0 \\ \frac{\rho u}{\rho} \\ -\frac{\rho v}{\rho} \end{pmatrix} + \frac{\partial}{\partial y}\begin{pmatrix} 0 \\ \rho \\ \frac{\rho v}{\rho} \\ \frac{\rho u}{\rho} \end{pmatrix}, \\ \mathcal{L}_2^1[\rho,\rho_x,\rho_y] &= \mu(\rho) - \lambda\frac{\partial}{\partial x}\rho_x - \lambda\frac{\partial}{\partial y}\rho_y - (g_x x + g_y y), \\ \mathcal{L}_3^1[\rho,\rho\boldsymbol{u},(u_x+v_y),(u_y-v_x),\kappa] &= \end{aligned} \tag{6.24}$$

$$\frac{\partial}{\partial x}\begin{pmatrix} \rho u \\ \rho u^2 - \varepsilon(u_x+v_y) \\ \rho uv + \mu(u_y-v_x) \end{pmatrix} + \frac{\partial}{\partial y}\begin{pmatrix} \rho v \\ \rho uv - \mu(u_y-v_x) \\ \rho v^2 - \varepsilon(u_x+v_y) \end{pmatrix} + \begin{pmatrix} 0 \\ \rho\frac{\partial}{\partial x}\kappa \\ \rho\frac{\partial}{\partial y}\kappa \end{pmatrix}.$$

Again as in the one dimensional case the blue terms represent the conservative parts, black terms source type parts and the red term a non-conservative part of the first order differential operators.

Before we continue with the definition of the numerical fluxes we define the $+$ and $-$ sides of a cell interface. We choose a vector $\boldsymbol{\beta} \in \mathbb{R}^2$ that is not parallel to any cell interface of the mesh. Such a choice is always possible because there are only a finite number of interfaces. We take the normal of a cell interface $\boldsymbol{n} = (n_x, n_y)$ and build the product $\boldsymbol{\beta} \cdot \boldsymbol{n}$. If this product is positive then the cell the normal points to defines the $+$ side of the interface and the opposite side the $-$ side. Using this convention the numerical fluxes for the two dimensional discretization are given by

$$
\begin{aligned}
\boldsymbol{g}^1(\rho^\pm, \rho\boldsymbol{u}^\pm, \boldsymbol{n}) &= \begin{pmatrix} \{\rho\}n_x \\ \{\rho\}n_y \\ \{u\}_\xi\, n_x + \{v\}_{1-\xi}\, n_y \\ \{-v\}_{1-\xi}\, n_x + \{u\}_\xi\, n_y \end{pmatrix}, \\
\boldsymbol{g}^2(\rho^\pm, \rho_x^\pm, \rho_y^\pm, \rho_z^\pm, \boldsymbol{n}) &= -\lambda\left(\{\rho_x\}n_x + \{\rho_y\}n_y\right), \\
\boldsymbol{g}^3(\rho^\pm, \rho\boldsymbol{u}^\pm, \ldots, \kappa^\pm, \boldsymbol{n}) &= \begin{pmatrix} \{\rho u\}n_x + \{\rho v\}n_y - \frac{\alpha_1}{2}[\kappa] \\ \{\rho u^2\}n_x + \{\rho uv\}n_y - \frac{\alpha_2}{2}[\rho u] \\ \{\rho vu\}n_x + \{\rho v^2\}n_y - \frac{\alpha_2}{2}[\rho v] \end{pmatrix} \\
&\quad + \begin{pmatrix} 0 \\ \zeta\{\rho\}[\kappa]n_x - \varepsilon\,\{u_x + v_y\}_{1-\xi}\, n_x - \mu\,\{u_y - v_x\}_{1-\xi}\, n_y \\ \zeta\{\rho\}[\kappa]n_y + \mu\,\{u_y - v_x\}_\xi\, n_x - \varepsilon\,\{u_x + v_y\}_\xi\, n_y \end{pmatrix}
\end{aligned}
\tag{6.25}
$$

The treatment of the boundary conditions by definition of suitable numerical boundary fluxes is almost the same as in the one dimensional case except for the definition of $\boldsymbol{g}_b^2$ where the boundary condition $-\frac{\nabla\rho}{|\nabla\rho|} \cdot \boldsymbol{n} = \cos(\varphi)$ is taken into account. φ denotes the contact angle of the diffuse interface at a solid wall.

$$
\boldsymbol{g}_b^2(\rho_x^-, \rho_y^-, n) = \lambda\, \cos(\varphi)\, \sqrt{(\rho_x^-)^2 + (\rho_y^-)^2}. \tag{6.26}
$$

We omit the remaining numerical boundary fluxes. It is an almost straight generalization of the 1d fluxes discussed in Section 6.9.1.

6.9.3 3d isothermal

In this section we denote the spatial variable by $\boldsymbol{x} = (x, y, z)$ and the velocity by $\boldsymbol{u} = (u, v, w)$. We omit the effect of gravity we have included in the two dimensional case. It can be included in the discretization in the same way as in the two dimensional case. Thus, we consider the NSK system in 3d

$$\begin{array}{rcccl} \rho_t & + & \nabla\cdot(\rho\boldsymbol{u}) & = & 0, \\ (\rho\boldsymbol{u})_t & + & \nabla\cdot(\rho\boldsymbol{u}\boldsymbol{u}^T) + \rho\nabla\kappa & = & \nabla\cdot\boldsymbol{\tau}, \end{array} \quad \text{in } \Omega\times(0,T)$$

with $\kappa = \mu(\rho) - \lambda\Delta\rho$ and boundary conditions

$$\boldsymbol{u} = 0 \quad \text{and} \quad -\frac{\nabla\rho}{|\nabla\rho|}\cdot\boldsymbol{n} = \cos(\varphi). \quad \text{on } \partial\Omega\times(0,T).$$

In three space dimensions almost the same approach with the viscous contribution can be done as for the 2d model, see (6.22).

$$\nabla\cdot\boldsymbol{\tau} = \begin{pmatrix} \varepsilon(u_x+v_y+w_z)_x & + & \mu(u_y-v_x)_y & + & \mu(u_z-w_x)_z \\ -\mu(u_y-v_x)_x & + & \varepsilon(u_x+v_y+w_z)_y & + & \mu(v_z-w_y)_z \\ -\mu(u_z-w_x)_x & + & -\mu(v_z-w_y)_y & + & \varepsilon(u_x+v_y+w_z)_z \end{pmatrix}. \tag{6.27}$$

Again we set $\varepsilon = 2\mu + \nu$ where μ and ν denote the coefficients of viscosity. Now we can use the four quantities $u_y - v_x$, $u_z - w_x$, $v_z - w_y$ and $u_x + v_y + w_z$ instead of the nine quantities in the velocity gradient. The former three quantities correspond to the tangential velocity at the boundary and the latter to the normal velocity.

We reformulate the third order system as formally first order system as in the one and two dimensional cases.

$$\begin{array}{cccl} \begin{pmatrix} \rho_x \\ \rho_y \\ \rho_z \\ (u_x+v_y+w_z) \\ (u_y-v_x) \\ (u_z-w_x) \\ (v_z-w_y) \end{pmatrix} & - & \mathcal{L}_1^1[\rho,\rho\boldsymbol{u}] & = 0, \\ \kappa & - & \mathcal{L}_2^1[\rho,\rho_x,\rho_y,\rho_z] & = 0, \\ \frac{\partial}{\partial t}\begin{pmatrix} \rho \\ \rho u \\ \rho v \\ \rho w \end{pmatrix} & + & \mathcal{L}_3^1[\rho,\rho\boldsymbol{u},\ldots,\kappa] & = 0. \end{array} \tag{6.28}$$

The first order differential operators $\mathcal{L}_1^1$, $\mathcal{L}_2^1$, $\mathcal{L}_3^1$ are then defined by

$$\begin{aligned}
\mathcal{L}_1^1[\rho, \rho\boldsymbol{u}] &= \frac{\partial}{\partial x}\begin{pmatrix} \rho \\ 0 \\ 0 \\ u \\ -v \\ -w \\ 0 \end{pmatrix} + \frac{\partial}{\partial y}\begin{pmatrix} 0 \\ \rho \\ 0 \\ v \\ u \\ 0 \\ -w \end{pmatrix} + \frac{\partial}{\partial z}\begin{pmatrix} 0 \\ 0 \\ \rho \\ w \\ 0 \\ u \\ v \end{pmatrix}, \\
\mathcal{L}_2^1[\rho, \rho_x, \rho_y, \rho_z] &= \mu(\rho) - \lambda\frac{\partial}{\partial x}\rho_x - \lambda\frac{\partial}{\partial y}\rho_y - \lambda\frac{\partial}{\partial z}\rho_z, \\
\mathcal{L}_3^1[\rho, \rho\boldsymbol{u}, \dots, \kappa] &= \begin{pmatrix} 0 \\ \rho\frac{\partial}{\partial x}\kappa \\ \rho\frac{\partial}{\partial y}\kappa \\ \rho\frac{\partial}{\partial z}\kappa \end{pmatrix} + \frac{\partial}{\partial x}\begin{pmatrix} \rho u \\ \rho u^2 - \varepsilon(u_x + v_y + w_z) \\ \rho uv + \mu(u_y - v_x) \\ \rho uw + \mu(u_z - w_x) \end{pmatrix} \\
&\quad + \frac{\partial}{\partial y}\begin{pmatrix} \rho v \\ \rho vu - \mu(u_y - v_x) \\ \rho v^2 - \varepsilon(u_x + v_y + w_z) \\ \rho vw + \mu(v_z - w_y) \end{pmatrix} + \frac{\partial}{\partial z}\begin{pmatrix} \rho w \\ \rho wu - \mu(u_z - w_x) \\ \rho wv - \mu(v_z - w_y) \\ \rho w^2 - \varepsilon(u_x + v_y + w_z) \end{pmatrix}.
\end{aligned} \tag{6.29}$$

We define $+$ and $-$ sides of cell interfaces in the same way as discussed in the two dimensional discretization by introduction of a vector $\beta \in \mathbb{R}^3$ that is not parallel to any interface of the mesh. With this convention we define the corresponding numerical fluxes by

$$\begin{aligned}
\boldsymbol{g}^1(\rho^\pm, \rho\boldsymbol{u}^\pm, \boldsymbol{n}) &= \begin{pmatrix} \{\rho\}n_x \\ \{\rho\}n_y \\ \{\rho\}n_z \\ \{u\}_\xi n_x + \{v\}_{1-\xi} n_y + \{w\}_{1-\xi} n_z \\ \{-v\}_{1-\xi} n_x + \{u\}_\xi n_y \\ \{-w\}_{1-\xi} n_x + \{u\}_\xi n_z \\ \{-w\}_\xi n_y + \{v\}_\xi n_z \end{pmatrix}, \\
\boldsymbol{g}^2(\rho^\pm, \rho_x^\pm, \rho_y^\pm, \rho_z^\pm, \boldsymbol{n}) &= -\lambda\left(\{\rho_x\}n_x + \{\rho_y\}n_y + \{\rho_z\}n_z\right), \\
\boldsymbol{g}^3(\rho^\pm, \rho\boldsymbol{u}^\pm, \dots, \kappa^\pm, \boldsymbol{n}) &= \begin{pmatrix} \{\rho u\}n_x + \{\rho v\}n_y + \{\rho w\}n_z - \frac{\alpha_1}{2}[\kappa] \\ \{\rho u^2\}n_x + \{\rho uv\}n_y + \{\rho uw\}n_z - \frac{\alpha_2}{2}[\rho u] \\ \{\rho vu\}n_x + \{\rho v^2\}n_y + \{\rho vw\}n_z - \frac{\alpha_2}{2}[\rho v] \\ \{\rho wu\}n_x + \{\rho wv\}n_y + \{\rho w^2\}n_z - \frac{\alpha_2}{2}[\rho w] \end{pmatrix}
\end{aligned} \tag{6.30}$$

$$+ \begin{pmatrix} 0 \\ \zeta\{\rho\}[\kappa]n_x - \varepsilon\{u_x + v_y + w_z\}_{1-\xi} n_x - \mu\{u_y - v_x\}_{1-\xi} n_y - \mu\{u_z - w_x\}_{1-\xi} n_z \\ \zeta\{\rho\}[\kappa]n_y + \mu\{u_y - v_x\}_\xi n_x - \varepsilon\{u_x + v_y + w_z\}_\xi n_y - \mu\{v_z - w_y\}_{1-\xi} n_z \\ \zeta\{\rho\}[\kappa]n_z + \mu\{u_z - w_x\}_\xi n_x + \mu\{v_z - w_y\}_{1-\xi} n_y - \varepsilon\{u_x + v_y + w_z\}_\xi n_z \end{pmatrix}.$$

The definition of the numerical boundary fluxes is a straightforward generalization of the one and two dimensional fluxes. Therefore we omit it.

6.9.4 2d full model

We discuss the DG space discretization of the full temperature dependent Navier-Stokes-Korteweg model (see Section 2.3) in this section. Most of the following treatment is quite similar to the isothermal case and therefore we will omit some details. Instead of the energy equation we use the entropy equation (2.27) as additional evolution equation. The total entropy of a Korteweg type material has the advantage that it does not depend on the density gradient whereas the total energy does. The entropy equation is not in divergence form but this should not be a problem since solutions are supposed to be smooth and the momentum equation is also discretized in a nonconservative form. Since we do not discretize the energy balance equation (2.36) directly, one cannot expect that the total physical energy is exactly conserved but the loss or gain of energy should be negligible small as long the solution is smooth. We discretize the system

$$
\begin{aligned}
\rho_t + \nabla \cdot (\rho \boldsymbol{u}) &= 0, \\
(\rho \boldsymbol{u})_t + \nabla \cdot (\rho \boldsymbol{u}\boldsymbol{u}^T) + \rho \nabla \kappa + \rho s \nabla \theta &= \nabla \cdot \boldsymbol{\tau}, \\
(\rho s)_t + \nabla \cdot (\rho s \boldsymbol{u}) &= \nabla \cdot \left(\tfrac{\eta}{\theta} \nabla \theta\right) + \tfrac{1}{\theta} \boldsymbol{\tau} : \nabla \boldsymbol{u} + \tfrac{\eta}{\theta^2} |\nabla \theta|^2.
\end{aligned}
\tag{6.31}
$$

Possible boundary conditions are discussed in Section 2.6. Since the chemical potential μ depends on the temperature the value κ defined by

$$
\kappa = \kappa(\theta, \rho, \Delta\rho) = \mu(\theta, \rho) - \lambda \Delta \rho
$$

does also depend on the temperature. Here $\eta > 0$ denotes the heat conduction coefficient of the fluid that is assumed to be constant. Note that the additional nonconservative term in the momentum equation comes from the identity

$$
\nabla p(\theta, \rho) = \rho \nabla \mu(\theta, \rho) + \rho s \nabla \theta.
$$

A similar approach as for the 2d isothermal model (6.22) is also possible for the full model but the use of only the quantities $u_x + v_y$ and $u_y - v_x$ is not sufficient because of the presence of the stress tensor in the energy and entropy equation. Therefore we need additional values. This may help with the treatment of boundary conditions but does not save computational cost. Therefore we omit it and use the quantities u_x, u_y, v_x and v_y.

Similar to the previous sections we rewrite equation (6.31) by using three first order differential operators.

$$
\begin{aligned}
\begin{pmatrix} \rho_x \\ \rho_y \\ u_x \\ u_y \\ v_x \\ v_y \end{pmatrix} &- \mathcal{L}_1^1[\rho, \rho\boldsymbol{u}] &= 0, \\
\begin{pmatrix} \theta_x \\ \theta_y \\ \theta \\ \kappa \end{pmatrix} &- \mathcal{L}_2^1[\rho, \rho s, \rho_x, \rho_y] &= 0, \\
\frac{\partial}{\partial t}\begin{pmatrix} \rho \\ \rho u \\ \rho v \\ \rho s \end{pmatrix} &+ \mathcal{L}_3^1[\rho, \rho\boldsymbol{u}, \rho s, \dots, \kappa] &= 0.
\end{aligned}
\tag{6.32}
$$

In the above definitions we distribute the computational cost over the three stages. Temperature and temperature gradients are computed in the second stage because evaluation of the temperature at the quadrature points is not necessary in the first stage and in order to distribute the computational cost such that the communication in the second stage does not become a bottleneck.

The first order differential operators $\mathcal{L}_1^1$, $\mathcal{L}_2^1$, $\mathcal{L}_3^1$ are defined by

$$
\begin{aligned}
\mathcal{L}_1^1[\rho, \rho\boldsymbol{u}] &= \frac{\partial}{\partial x}\begin{pmatrix} \rho \\ 0 \\ \frac{\rho u}{\rho} \\ 0 \\ \frac{\rho v}{\rho} \\ 0 \end{pmatrix} + \frac{\partial}{\partial y}\begin{pmatrix} 0 \\ \rho \\ 0 \\ \frac{\rho u}{\rho} \\ 0 \\ \frac{\rho v}{\rho} \end{pmatrix}, \\
\mathcal{L}_2^1[\rho, \rho s, \rho_x, \rho_y] &= \frac{\partial}{\partial x}\begin{pmatrix} \tilde{\theta}(\rho, \rho s) \\ 0 \\ 0 \\ -\lambda\rho_x \end{pmatrix} + \frac{\partial}{\partial y}\begin{pmatrix} 0 \\ \tilde{\theta}(\rho, \rho s) \\ 0 \\ -\lambda\rho_y \end{pmatrix} + \begin{pmatrix} 0 \\ 0 \\ \tilde{\theta}(\rho, \rho s) \\ \mu(\tilde{\theta}(\rho, \rho s), \rho) \end{pmatrix}, \\
\mathcal{L}_3^1[\rho, \dots, \kappa] &= \frac{\partial}{\partial x}\begin{pmatrix} \rho u \\ \rho u^2 - \varepsilon(u_x + v_y) \\ \rho uv + \mu(u_y - v_x) \\ \rho s u - \frac{\eta}{\theta}\theta_x \end{pmatrix} + \frac{\partial}{\partial y}\begin{pmatrix} \rho v \\ \rho uv - \mu(u_y - v_x) \\ \rho v^2 - \varepsilon(u_x + v_y) \\ \rho s v - \frac{\eta}{\theta}\theta_y \end{pmatrix} \\
&\quad + \begin{pmatrix} 0 \\ \rho\frac{\partial}{\partial x}\kappa + \rho s\frac{\partial}{\partial x}\theta \\ \rho\frac{\partial}{\partial y}\kappa + \rho s\frac{\partial}{\partial y}\theta \\ 0 \end{pmatrix} + \begin{pmatrix} 0 \\ 0 \\ 0 \\ \frac{1}{\theta}\boldsymbol{\tau} : \nabla\boldsymbol{u} + \frac{\eta}{\theta^2}|\nabla\theta|^2 \end{pmatrix}.
\end{aligned}
\tag{6.33}
$$

Here the convention is the same as before: the blue colored terms denote the conservative terms, the red terms denote the nonconservative products and source terms are characterized by black color. The entropy production term $\frac{1}{\theta}\boldsymbol{\tau} : \nabla \boldsymbol{u} + \frac{\eta}{\theta^2}|\nabla\theta|^2$ is treated as a source term. The function $\tilde{\theta}$ computes the temperature from the density and entropy according to relation (2.12).

The definition of the numerical fluxes and numerical boundary fluxes is very similar to the isothermal case and straightforward to do. We omit the details. Besides the additional entropy equation the difference to the isothermal case is the appearance of the second nonconservative term $\rho s \nabla\theta$. As noted before the entropy production term is treated as a source term and gives therefore no contribution to the numerical flux associated with the differential operator $\mathcal{L}_3^1$.

6.10 Initial Data

The standard way to provide discrete initial data for Discontinuous Galerkin schemes is an application of L^2-projection to the underlying Finite Element space. The use of an orthogonal basis of the Finite Element space results in a very easy implementation of this kind of projection. Since we do not use slope limiters to stabilize the DG schemes it is extremely important to provide sufficiently smooth initial data on the discrete level. Discrete initial data for the Navier-Stokes-Korteweg system that consist of both phases should take the correct size of the interface, approximately given by formula (4.9), into account to avoid an unstable behavior in the approximate solution. This is especially important for the higher order schemes. Inside the interfacial region the initial configuration should vary smoothly between the phases. Here the tanh-function is very useful to construct smooth initial data.

Chapter 7

Higher Order Time Integration

The Discontinuous Galerkin space discretization of a general evolution equation, discussed in the previous chapter, results in a (in general very large) system of first order ordinary differential equations. In this chapter we discuss the time discretization of general first order ODEs by means of explicit, implicit and semi-implicit Runge-Kutta methods. In the latter two cases solving large, possibly nonlinear, systems of equations is necessary. Solving such systems is the purpose of Section 7.5.

Runge-Kutta methods have the advantage that the approximate solution at only one time step is necessary to compute an approximation on the next time level. This makes this class of methods very well suited for the use together with local mesh refinement. Contrary to Runge-Kutta methods the class of multistep methods uses more than one approximations on previous time steps. On the one hand these methods need a restriction in the variation of the time step size to guarantee the stability of the method and on the other hand these methods are complicated to implement together with local mesh refinement and coarsening which results in a change of the dimension of the ODE. Because of these disadvantages there are only a few multistep methods (for example the implicit BDF2 method) that can be applied successfully in order to construct reliable discretizations of conservation laws. In the framework of the discretization of first order conservation laws special Runge-Kutta methods have been developed that preserve certain properties (e.g. TVD) of scalar conservation laws. Initially Shu and Osher [103], [104] derived this kind of TVD methods. Later the term *Strong Stability Preserving* (SSP) was used in favor of the term TVD. Explicit or implicit extrapolation schemes could also be used but the use of these methods is not very common in the framework of Finite Volume and Discontinuous Galerkin methods. The advantage of these methods is that arbitrary high order methods can be constructed by simply modifying a parameter in the methods.

In this chapter we consider the initial value problem for first order ordinary differential equations of the form

$$u'(t) = f(t, u(t)) \quad \text{for} \quad t \in (0, T), \tag{7.1}$$

$$u(0) = u_0, \tag{7.2}$$

with $T \in (0, \infty]$, $f \in C([0,T] \times U)$ Lipschitz continuous in the variable u, $U \subset R^n$, $u_0 \in U$.

By integrating equation (7.1) from time t^m to t^{m+1} we have

$$u(t^{m+1}) - u(t^m) = \int_{t^m}^{t^{m+1}} f(t, u(t))\, dt. \tag{7.3}$$

The goal of this chapter is to compute an approximation u^{m+1} of $u(t^{m+1})$ provided that we already have an approximation u^m of $u(t^m)$.

7.1 General Runge-Kutta Methods

In the rest of this chapter we discuss the computation of the approximation u^{m+1} by different kinds of Runge-Kutta methods. In the following we use the notation given in definition A.2.2. The general s-stage Runge-Kutta scheme from time step t^m to t^{m+1} is given by

$$\begin{pmatrix} u_0 \\ \vdots \\ u_{s-1} \end{pmatrix} = \begin{pmatrix} u^m \\ \vdots \\ u^m \end{pmatrix} + \Delta t A \otimes I_n \begin{pmatrix} f(t^m + c_0 \Delta t, u_0) \\ \vdots \\ f(t^m + c_{s-1}\Delta t, u_{s-1}) \end{pmatrix}, \tag{7.4}$$

$$u^{m+1} = u^m + \Delta t b^T \otimes I_n \begin{pmatrix} f(t^m + c_0 \Delta t, u_0) \\ \vdots \\ f(t^m + c_{s-1}\Delta t, u_{s-1}) \end{pmatrix}. \tag{7.5}$$

Here the intermediate states $u_i \in U$ for $i = 0, \ldots, s-1$ are approximations of the solution at times $t_i = t^m + c_i \Delta t$. Runge-Kutta schemes defined by the Matrix $A \in \mathbb{R}^{s \times s}$ and the two vectors $b, c \in \mathbb{R}^s$ are usually represented by a so called *Butcher table* (see standard textbooks such as [108])

$$\begin{array}{c|c} c & A \\ \hline & b^T \end{array}. \tag{7.6}$$

Definition 7.1.1 *(Order of Runge-Kutta Methods)*
Let Φ denote the function that produces the approximation $u^{m+1} = \Phi(t^m, u^m, \Delta t)$ by application of a Runge-Kutta method. A Runge-Kutta method is of (consistency) order p if

$$\left| \frac{1}{\Delta t}(u(t^m + \Delta t) - u(t^m)) - \Phi(t^m, u(t^m), \Delta t) \right| = O(\Delta t^p)$$

for sufficiently smooth solutions u of the ODE (7.1).

A p-th order consistency Runge-Kutta method implies convergence of order p. Therefore it is clear that the order of a s-stage Runge-Kutta scheme cannot exceed $2s$ because in the case where the function f does not depend on the variable u a Runge-Kutta method reduces to a quadrature formula for f.

7.2 Explicit Runge-Kutta Methods

In this section we consider the class of explicit Runge-Kutta methods, i.e., methods that do not need to solve linear or nonlinear systems of equations. This class of methods is especially well suited for the use with first order conservation laws. In the framework of Discontinuous Galerkin methods the representation of the methods given by Shu and Osher in [103] and [104] is more convenient than the classical representation because *slope limiters* can efficiently be applied to the intermediate states $u_0, \dots, u_{s-1}$. Below we reformulate general explicit Runge-Kutta schemes in the representation of Shu and Osher and give some examples for this kind of methods.

A Runge-Kutta method given by the equations (7.4) and (7.5) reduces to an explicit method if the matrix $A \in R^{s\times s}$ is a strictly lower triangular matrix. By $\tilde{A} \in \mathbb{R}^{(s-1)\times(s-1)}$ we denote the sub matrix of A where the first row and the last column is omitted. The matrix $\tilde{A}$ can be decomposed into a strictly lower triangular matrix $\tilde{A}_L$ and a diagonal matrix $\tilde{A}_D$ such that we have $\tilde{A} = \tilde{A}_L + \tilde{A}_D$. In the following we assume that the matrix $\tilde{A}$ is invertible. In this case equation (7.4) reduces to

$$\begin{aligned} u_0 &= u^m, \\ \begin{pmatrix} u_1 \\ \vdots \\ u_{s-1} \end{pmatrix} &= \begin{pmatrix} u^m \\ \vdots \\ u^m \end{pmatrix} + \Delta t \tilde{A} \otimes I_n \begin{pmatrix} f(t^m + c_0 \Delta t, u_0) \\ \vdots \\ f(t^m + c_{s-2}\Delta t, u_{s-2}) \end{pmatrix}. \end{aligned} \tag{7.7}$$

By multiplication of equation (7.7) with $(\tilde{A} \otimes I_n)^{-1}$ and using properties (i) and (ii) of the Kronecker product, see lemma A.2.3, we have the identity

$$\Delta t \begin{pmatrix} f(t^m + c_0 \Delta t, u_0) \\ \vdots \\ f(t^m + c_{s-2}\Delta t, u_{s-2}) \end{pmatrix} = \tilde{A}^{-1} \otimes I_n \begin{pmatrix} u_1 - u^m \\ \vdots \\ u_{s-1} - u^m \end{pmatrix}.$$

Using this with equations (7.7) and (7.5) we get

$$u_0 = u^m, \tag{7.8}$$

$$\begin{pmatrix} u_1 \\ \vdots \\ u_{s-1} \end{pmatrix} = \begin{pmatrix} u^m \\ \vdots \\ u^m \end{pmatrix} + \tilde{A}_L \tilde{A}^{-1} \otimes I_n \begin{pmatrix} u_1 - u^m \\ \vdots \\ u_{s-1} - u^m \end{pmatrix} + \Delta t \tilde{A}_D \otimes I_n \begin{pmatrix} f_0 \\ \vdots \\ f_{s-2} \end{pmatrix} \tag{7.9}$$

$$u^{m+1} = u^m + \Delta t \tilde{b}^T \tilde{A}^{-1} \otimes I_n \begin{pmatrix} u_1 - u^m \\ \vdots \\ u_{s-1} - u^m \end{pmatrix} + b_{s-1} f_{s-1}, \tag{7.10}$$

where the vector $\tilde{b} \in R^{s-1}$ consists of the first $s-1$ components of the vector b and $f_i = f(t^m + c_i \Delta t, u_i)$. This is the representation of Runge-Kutta methods given by Shu and Osher [103]. This representation is more suitable for the time discretization of conservation laws because *slope limiters* can be applied directly to the intermediate states u_i.

Below we give some examples of explicit Runge-Kutta methods. Most of them can be found in standard textbooks such as [108]. The TVD methods developed by Shu and Osher can be found in [103].

The first order explicit Euler and the second order modified Euler schemes with one and two stages respectively.

$$
\begin{array}{c|c}
0 & 0 \\
\hline
 & 1
\end{array}
\qquad
\begin{array}{c|cc}
0 & 0 & \\
\frac{1}{2} & \frac{1}{2} & 0 \\
\hline
 & 0 & 1
\end{array}
$$

The second and third order schemed TVD2 and TVD3 given in [103]. The TVD2 scheme is also known as the Heun scheme. The TVD methods in [103] are given using the representation (7.8), (7.9) and (7.10). For consistency with all other methods in this chapter we write these methods using the classical representation.

$$
\begin{array}{c|cc}
0 & 0 & \\
1 & 1 & 0 \\
\hline
 & \frac{1}{2} & \frac{1}{2}
\end{array}
\qquad
\begin{array}{c|ccc}
0 & 0 & & \\
1 & 1 & 0 & \\
\frac{1}{2} & \frac{1}{4} & \frac{1}{4} & 0 \\
\hline
 & \frac{1}{6} & \frac{1}{6} & \frac{2}{3}
\end{array}
$$

The classical 3-stage and 4-stage order Runge-Kutta schemes of order three and four.

$$
\begin{array}{c|ccc}
0 & 0 & & \\
\frac{1}{2} & \frac{1}{2} & 0 & \\
1 & -1 & 2 & 0 \\
\hline
 & \frac{1}{6} & \frac{4}{6} & \frac{1}{6}
\end{array}
\qquad
\begin{array}{c|cccc}
0 & 0 & & & \\
\frac{1}{2} & \frac{1}{2} & 0 & & \\
\frac{1}{2} & 0 & \frac{1}{2} & 0 & \\
1 & 0 & 0 & 1 & 0 \\
\hline
 & \frac{1}{6} & \frac{1}{3} & \frac{1}{3} & \frac{1}{6}
\end{array}
$$

A sixth order scheme with seven stages.

$$
\begin{array}{c|ccccccc}
0 & 0 & & & & & & \\
\frac{1}{2} & \frac{1}{2} & 0 & & & & & \\
\frac{2}{3} & \frac{2}{9} & \frac{4}{9} & 0 & & & & \\
\frac{1}{3} & \frac{7}{36} & \frac{2}{9} & -\frac{1}{12} & 0 & & & \\
\frac{5}{6} & -\frac{35}{144} & -\frac{55}{36} & \frac{35}{48} & \frac{15}{8} & 0 & & \\
\frac{1}{6} & -\frac{1}{360} & -\frac{11}{36} & -\frac{1}{8} & \frac{1}{2} & \frac{1}{10} & 0 & \\
1 & -\frac{41}{260} & \frac{22}{13} & \frac{43}{156} & -\frac{118}{39} & \frac{32}{195} & \frac{80}{39} & 0 \\
\hline
 & \frac{13}{200} & 0 & \frac{11}{40} & \frac{11}{40} & \frac{4}{25} & \frac{4}{25} & \frac{13}{200}
\end{array}
$$

7.3 Implicit Runge-Kutta Methods

When higher order derivatives or stiff source terms are included in conservation laws the time step size restriction that guarantees the stability of the method can render a scheme inefficient. In this case an implicit time discretization may help to improve the efficiency of the method. We discuss the details on the implementation of a class of implicit methods and give some examples. It is very important to have a formulation of the method such that solving a $s \cdot n$ dimensional linear system can be avoided in favor of s times solving a n-dimensional systems, otherwise the methods are not usable for practical applications. This is possible with many implicit methods at least when the resulting Jacobians are approximated. For the class of diagonally implicit methods this is possible without approximation.

In this section we consider only diagonally implicit Runge-Kutta methods. In this class of methods the matrix $A \in \mathbb{R}^{s\times s}$ in (7.4) is a lower triangular matrix. We assume that the matrix A is invertible, otherwise some of the stages are explicit. We decompose A into a diagonal A_D and a strictly lower triangular matrix A_L with $A = A_L + A_D$. Similar to the explicit case the general Runge-Kutta method (7.4), (7.5) can be rewritten as

$$\begin{aligned}
\begin{pmatrix} u_0 \\ \vdots \\ u_{s-1} \end{pmatrix} &= \begin{pmatrix} u^m \\ \vdots \\ u^m \end{pmatrix} + A_L A^{-1} \otimes I_n \begin{pmatrix} u_0 - u^m \\ \vdots \\ u_{s-1} - u^m \end{pmatrix} + \Delta t A_D \otimes I_n \begin{pmatrix} f_0 \\ \vdots \\ f_{s-1} \end{pmatrix}, \\
u^{m+1} &= u^m + b^T A^{-1} \otimes I_n \begin{pmatrix} u_0 - u^m \\ \vdots \\ u_{s-1} - u^m \end{pmatrix}.
\end{aligned}$$

With $f_i = f(t^m + c_i \Delta t, u_i)$. Since the matrix $A_L A^{-1}$ is a strictly lower triangular matrix s n-dimensional systems for u_i have to be solved sequentially instead of one (sn)-dimensional system. For the intermediate states u_i this means

$$\begin{aligned}
u_i &= \Delta t \alpha_{ii} f(t^m + c_i \Delta t, u_i) + \gamma_i u^m + \sum_{j=0}^{i-1} \alpha_{ij} u_j \quad \text{for } i = 0, \dots, s-1, \\
u^{m+1} &= \delta + \sum_{i=0}^{s-1} \beta_i u_i,
\end{aligned}$$

where the coefficients are given by

$$\begin{aligned}
\alpha_{ij} &= (A_L A^{-1})_{ij} \text{ for } j < i, \quad \alpha_{ii} = A_{ii}, \\
\gamma_i &= 1 - \sum_{j=0}^{i-1} \alpha_{ij}, \\
\beta_i &= \sum_{j=0}^{s-1} b_j (A^{-1})_{ji}, \\
\delta &= 1 - \sum_{i=0}^{s-1} \beta_i.
\end{aligned}$$

Here C_{ij} denotes the entry (i,j) of a matrix C. A fixed point argument shows that the above nonlinear system has a unique solution in the vicinity of u^m provided that the time step Δt is sufficiently small. The s n-dimensional nonlinear systems can be solved by the Newton type method described in Section 7.5. Below we give some examples of diagonally implicit methods.

The first order implicit Euler method and the second order Crank-Nicholson method are given by

$$\begin{array}{c|c} 1 & 1 \\ \hline & 1 \end{array} \qquad\qquad \begin{array}{c|c} \frac{1}{2} & \frac{1}{2} \\ \hline & 1 \end{array}$$

which are both 1-stage methods. A 2-stage third order method is given by

$$\begin{array}{c|cc} \alpha & \alpha & \\ 1-\alpha & 1-2\alpha & \alpha \\ \hline & \frac{1-2\alpha}{2-4\alpha} & \frac{1-2\alpha}{2-4\alpha} \end{array}$$

where $\alpha = \frac{1}{2} + \frac{\sqrt{3}}{6}$. Besides other methods the above methods can be found in [108]. Additional methods of this type can also be found in the next section.

7.4 Semi-Implicit Runge-Kutta Methods

The class of semi-implicit (or implicit-explicit) Runge-Kutta methods combines the efficiency of explicit methods with the stability properties of implicit Runge-Kutta methods. They are useful for the time discretization of convection-diffusion equations or convection dominated equations with stiff source terms. The class of methods we consider in this section discretizes one part of the equation by an explicit TVD (or SSP) method and another part by a *L-stable* diagonally implicit scheme. Below we discuss the details on the implementation of these methods and we give a couple of examples collected from [131], [132] and [91].

We split the function f in (7.1) into a part that is discretized explicitly and a second part that is discretized by an implicit scheme.

$$u'(t) = f^{ex}(t,u(t)) + f^{im}(t,u(t)), \quad \text{for} \quad t \in (0,T), \tag{7.11}$$

$$u(0) = u_0. \tag{7.12}$$

This splitting is useful when the spectrum of the Jacobian of f^{ex} is some orders of magnitudes smaller than the spectrum of the Jacobian of f^{im}, i.e.,

$$\rho\left(D_u f^{ex}(t,u)\right) \quad << \quad \rho\left(D_u f^{im}(t,u)\right),$$

where ρ denotes the spectral radius.

The class of semi-implicit Runge-Kutta schemes that uses TVD/Strong-Stability-Preserving explicit schemes together with L-stable diagonally implicit schemes, considered in [131], [132] and [91], can be written as

$$\begin{aligned} k_i &= f^{ex}\left(t^m + c_i^{ex}\Delta t, u^m + \Delta t \sum_{j=0}^{i-1} a_{ij}^{ex} k_j\right) \\ &\quad + f^{im}\left(t^m + c_i^{im}\Delta t, u^m + \Delta t \sum_{j=0}^{i} a_{ij}^{ex} k_j\right) \quad \text{for } i = 0, \dots, s-1, \quad (7.13) \\ u^{m+1} &= u^m + \Delta t \sum_{i=0}^{s-1} b_i k_i. \quad (7.14) \end{aligned}$$

This class of Runge-Kutta methods can be represented by a pair of Butcher tables with a common vector b.

$$\begin{array}{c|c} c^{ex} & A^{ex} \\ \hline & b^T \end{array} \qquad \begin{array}{c|c} c^{im} & A^{im} \\ \hline & b^T \end{array}$$

The matrix $A^{ex} \in \mathbb{R}^{s\times s}$ is a strictly lower triangular matrix and $A^{im} \in \mathbb{R}^{s\times s}$ is an invertible lower triangular matrix. The coefficients a_{ij}^{ex} and a_{ij}^{im} denote the entries of the matrices A^{ex} and A^{im} respectively.

For the explicit and implicit intermediate states u_i^{ex} and u_i^{im} for $i = 0, \dots, s-1$ this can be written as

$$\begin{aligned} u_i^{ex} &= \gamma_i^{ex} u^m + \sum_{j=0}^{i-1} \alpha_{ij}^{ex} u_j^{im}, \\ u_i^{im} &= \alpha_{ii}^{im} \Delta t \left(f^{ex}(t^m + c_i^{ex}\Delta t, u_i^{ex}) + f^{im}(t^m + c_i^{im}\Delta t, u_i^{im})\right) \\ &\quad + \gamma_i^{im} u^m + \sum_{j=0}^{i-1} \alpha_{ij}^{im} u_j^{im}, \\ u^{m+1} &= \delta u^m + \sum_{i=0}^{s-1} \beta_i u_i^{im}. \end{aligned}$$

The coefficients appearing in the above equations are defined by

$$\begin{aligned}
\alpha_{ij}^{im} &= \left(A_L^{im}(A^{im})^{-1}\right)_{ij}, \quad \alpha_{ii}^{im} = A_{ii}^{im}, \\
\alpha_{ij}^{ex} &= \left(A_L^{ex}(A^{im})^{-1}\right)_{ij}, \\
\gamma_i^{im} &= 1 - \sum_{j=0}^{i-1} \alpha_{ij}^{im}, \\
\gamma_i^{ex} &= 1 - \sum_{j=0}^{i-1} \alpha_{ij}^{ex}, \\
\beta_i &= \sum_{j=0}^{s-1} b_j \left((A^{im})^{-1}\right)_{ji}, \\
\delta &= 1 - \sum_{i=0}^{s-1} \beta_i,
\end{aligned}$$

where C_{ij} denotes the entry (i,j) of a matrix C and A_L^{im} is the strictly lower triangular part (without the diagonal) of the matrix A^{im}.

Below we give several examples for semi-implicit Runge-Kutta methods taken from [131], [132] and [91].

The second order SIRK23 scheme (3 stages, L-stable).

$$\begin{array}{c|ccc}
0 & 0 & & \\
1 & 1 & 0 & \\
\frac{1}{2} & 0 & \frac{1}{2} & 0 \\
\hline
 & \frac{1}{4} & \frac{1}{4} & \frac{1}{2}
\end{array}
\qquad
\begin{array}{c|ccc}
\frac{1}{2} & \frac{1}{2} & & \\
-\frac{1}{2} & -1 & \frac{1}{2} & \\
1 & \frac{1}{4} & \frac{1}{4} & \frac{1}{2} \\
\hline
 & \frac{1}{4} & \frac{1}{4} & \frac{1}{2}
\end{array}$$

The second order SIRK23G scheme (3 stages, L-stable). We set $\alpha = \frac{1}{\sqrt{2}}$.

$$\begin{array}{c|ccc}
0 & 0 & & \\
1 & 1 & 0 & \\
1+\alpha & \alpha & 1 & 0 \\
\hline
 & \alpha & 0 & 1-\alpha
\end{array}
\qquad
\begin{array}{c|ccc}
1-\alpha & 1-\alpha & & \\
1-\alpha & 0 & 1-\alpha & \\
1 & \alpha & 0 & 1-\alpha \\
\hline
 & \alpha & 0 & 1-\alpha
\end{array}$$

The structure of the implicit scheme allows for a *low storage implementation.*

The third order YZ33 scheme (3 stages, L-stable).

$$\begin{array}{c|ccc} 0 & 0 & & \\ \frac{8}{7} & \frac{8}{7} & 0 & \\ \frac{120}{252} & \frac{71}{252} & \frac{49}{252} & 0 \\ \hline & \frac{1}{8} & \frac{1}{8} & \frac{3}{4} \end{array} \qquad \begin{array}{c|ccc} \frac{3}{4} & \frac{3}{4} & & \\ \alpha & \frac{5589}{6524} & \frac{75}{233} & \\ \beta & \frac{7691}{26096} & -\frac{26335}{78288} & \frac{65}{168} \\ \hline & \frac{1}{8} & \frac{1}{8} & \frac{3}{4} \end{array}$$

where the constants α and β are defined by

$$\alpha = \frac{5589}{6524} + \frac{75}{233}, \quad \beta = \frac{7691}{26096} - \frac{26335}{78288} + \frac{65}{168}.$$

The methods considered in [91] allow the explicit and implicit schemes to have a different number of stages. Therefore the naming convention of the schemes is IMEX-SPP(implicit stages, explicit stages, order). Here IMEX stands for implicit-explicit and SSP for strong stability preserving which is the same as TVD.

The second order IMEX-SSP(2,2,2) scheme. We set $\alpha = 1 - \frac{1}{\sqrt{2}}$.

$$\begin{array}{c|cc} 0 & 0 & \\ 1 & 1 & 0 \\ \hline & \frac{1}{2} & \frac{1}{2} \end{array} \qquad \begin{array}{c|cc} \alpha & \alpha & \\ 1-\alpha & 1-2\alpha & \alpha \\ \hline & \frac{1}{2} & \frac{1}{2} \end{array}$$

The explicit scheme is the TVD2 scheme from Section 7.2.

The second order IMEX-SPP(3,3,2) scheme.

$$\begin{array}{c|ccc} 0 & 0 & & \\ \frac{1}{2} & \frac{1}{2} & 0 & \\ 1 & \frac{1}{2} & \frac{1}{2} & 0 \\ \hline & \frac{1}{3} & \frac{1}{3} & \frac{1}{3} \end{array} \qquad \begin{array}{c|ccc} \frac{1}{4} & \frac{1}{4} & & \\ \frac{1}{4} & 0 & \frac{1}{4} & \\ 1 & \frac{1}{3} & \frac{1}{3} & \frac{1}{3} \\ \hline & \frac{1}{3} & \frac{1}{3} & \frac{1}{3} \end{array}$$

The second order IMEX-SPP(3,2,2) scheme.

$$\begin{array}{c|ccc} 0 & 0 & & \\ 0 & 0 & 0 & \\ 1 & 0 & 1 & 0 \\ \hline & 0 & \frac{1}{2} & \frac{1}{2} \end{array} \qquad \begin{array}{c|ccc} \frac{1}{2} & \frac{1}{2} & & \\ 0 & -\frac{1}{2} & \frac{1}{2} & \\ 1 & 0 & \frac{1}{2} & \frac{1}{2} \\ \hline & 0 & \frac{1}{2} & \frac{1}{2} \end{array}$$

The explicit scheme is the TVD2 scheme from Section 7.2.

The third order IMEX-SPP(4,3,3) scheme.

$$\begin{array}{c|cccc}
0 & 0 & & & \\
0 & 0 & 0 & & \\
1 & 0 & 1 & 0 & \\
\frac{1}{2} & 0 & \frac{1}{4} & \frac{1}{4} & 0 \\
\hline
 & 0 & \frac{1}{6} & \frac{1}{6} & \frac{2}{3}
\end{array}
\qquad
\begin{array}{c|cccc}
\alpha & \alpha & & & \\
0 & -\alpha & \alpha & & \\
1 & 0 & 1-\alpha & \alpha & \\
\frac{1}{2} & \beta & \eta & \frac{1}{2}-\alpha-\beta-\eta & \alpha \\
\hline
 & 0 & \frac{1}{6} & \frac{1}{6} & \frac{2}{3}
\end{array}$$

The parameters α, β and η are computed numerically

$$\alpha = 0.24169426078821, \quad \beta = 0.06042356519705, \quad \eta = 0.12915286960590.$$

The explicit scheme is the TVD3 scheme from Section 7.2.

7.5 Solving Nonlinear Equations

The application of implicit Runge-Kutta methods to ordinary differential equations methods results in a large (in general nonlinear) system of equations that has to be solved by a Newton type method. To avoid the computation of the Jacobian matrix in the Newton method, which is rather complicated for the fully discretized Navier-Stokes-Korteweg system especially in a parallel environment, we apply a Jacobian free Newton-Krylov method, see [71].

We consider the nonlinearity given by $F : U \to \mathbb{R}^n$, $U \subset \mathbb{R}^n$ and we seek for a solution $u \in U$ of the nonlinear equation

$$F(u) = 0.$$

Provided that the function F is sufficiently smooth, the Jacobian of F is nondegenerate in a vicinity U of the solution u and an initial guess $u_0 \in U$ sufficiently close to the solution is known, the solution can be computed by application of Newtons method

$$u_{n+1} = u_n - DF(u_n)^{-1}F(u_n), \quad n \geq 0. \tag{7.15}$$

For application of the Newton method it is necessary to solve a linear system for a vector p of the form

$$DF(u)p = F(u). \tag{7.16}$$

In many cases the explicit calculation of the Jacobian DF is much to expensive with respect to the computational cost, or with respect to the memory requirements or both. Sometimes it is simply to complicated to compute the Jacobian explicitly for example due to larger stencils in the discretization of underlying partial differential equations. In the higher order space discretization together with higher order implicit time discretization of the Navier-Stokes-Korteweg system all of the above mentioned issues occur.

Nevertheless, the application of the Newton method is still possible by means of matrix free methods. In combination with *Krylov space* solvers, like CG [59], BiCGSTAB [113], GMRES [98], etc., see also [97], for the linear system the Newton method does not need the Jacobian explicitly. These kind of solvers only need the matrix vector product $DF(u)p$ in (7.16) which is nothing else than the derivative of F at u in the direction of p and can be approximated by the difference quotient

$$DF(u)p \approx \frac{1}{\varepsilon}\left(F(u+\varepsilon p) - F(u)\right).$$

Here the crucial part is the choice of the parameter $\varepsilon > 0$. There are several approaches to choose this parameter, we use

$$\varepsilon = \begin{cases} \frac{\sqrt{(1+||u||)\varepsilon_{mach}}}{||p||} & \text{if } ||p||^2 > \varepsilon_{mach}, \\ \sqrt{\varepsilon_{mach}} & \text{else.} \end{cases} \tag{7.17}$$

Here ε_{mach} denotes the machine precision which is for double precision arithmetic approximately $\varepsilon_{mach} \approx 10^{-15}$. For the above choice and other possible choices see [71]. A matrix free method is, compared to other methods, rather simple to code but comes at the expense that in every iteration step of the linear solver the nonlinearity F has to be evaluated. Depending on the problem this can be a serious performance penalty. A comparison of standard and matrix free Newton methods in the framework of the discretization of the incompressible Navier-Stokes equations can be found in [84].

Another problem of this method is that standard preconditioning techniques cannot be applied since the matrix itself is not available. A matrix free preconditioning technique was proposed in [33] but not tested in this work.

7.6 Application to the Navier-Stokes-Korteweg system

The application of explicit Runge-Kutta methods to the higher order Discontinuous Galerkin space discretization of the Navier-Stokes-Korteweg system discussed in chapter 6, especially Section 6.9, leads to a time step time restriction that is difficult to control. No explicit formula is available that guarantees the stability of the method on the one hand and is sufficiently sharp on the other hand.

To avoid this problem with the time step size restriction we use the class of diagonally implicit schemes together with a matrix free nonlinear solver discussed in Section 7.3 and 7.5 for the time discretization of the Navier-Stokes-Korteweg system. Even without a preconditioner the resulting fully discrete scheme is more efficient than an explicit time discretization, see the numerical experiments in 9.6. However, also in the case of implicit discretization the time step size has to be sufficiently small such that the nonlinear system is solvable. The time step should always be chosen such that the quotient

$$\rho_m(\Delta t) = \frac{\text{cpu}(t^m, \Delta t)}{\Delta t}$$

is minimized. Here cpu$(t^m, \Delta t)$ stands for the cpu-time the computation needs to compute an approximate solution $u(t^m + \Delta t)$ starting from an approximate solution $u(t^m)$ at time t^m. If we assume that cpu$(t^m, \Delta t)$ does not depend on t^m this choice of the time step is the most efficient time step size. In general this assumption is not correct but a time step size that always minimizes $\rho_m(\Delta t)$ should be close to the optimal time step.

Finding the optimal time step in the above sense is a hard task itself. Another, simpler and sometimes more robust, approach is to control the number of iterations of the underlying linear solver. This approach heavily depends on the problem, the data and the used linear solver. For the Discontinuous Galerkin discretization in two space dimensions and second order implicit Runge-Kutta discretization and iteration count of about 20 is a good choice. In three space dimensions and for schemes of different order other choices are necessary and have to figured out manually.

The application of semi-implicit Runge-Kutta schemes is an appropriate choice for the time discretization of the compressible Navier-Stokes equations or Euler equations with stiff source terms. It is not clear how to apply this class to the NSK system since the viscous part of the equation is not the only source for the resulting small time steps. The third order Korteweg term that is discretized together with the pressure term and the artificial viscosity in the continuity equation also lead to small time steps.

Chapter 8

Mesh Adaption and Parallelization

In this chapter we discuss the local refinement and coarsening of computational meshes as well as the parallelization of the numerical algorithms. These techniques are very important in order to be able to resolve very small liquid-vapor interfaces and to satisfy memory requirements as well as computational power requirements of complex problems.

Here we discuss the local refinement and coarsening of computational meshes which is also called h-adaption. Another adaption strategy is p-adaption which makes use of the local choice of the polynomial degree in the Discontinuous Galerkin method. The latter adaption strategy is not taken into account in this work since the control of the local polynomial degree can be quite complicated. Since the Discontinuous Galerkin method does not need conformity, mesh refinement is done in a nonconform fashion by dividing a n-dimensional simplex into 2^n children. This is straightforward in one and two space dimensions but in three space dimensions there is an ambiguity. A reasonable strategy is necessary to avoid degenerating meshes.

For the parallelization of the code a domain decomposition approach is chosen since this is the most suitable approach in the Finite Volume and Discontinuous Galerkin framework where stencils are usually relatively small and thus, the mesh is only *weakly coupled*. With this approach the implementation of the code in a distributed memory environment is almost straightforward.

8.1 Refinement of Simplices

In this section we provide the basics for the application of an (h-)adaptive algorithm: The refinement of a single simplex in one, two and three space dimensions. Here we always divide a *parent* simplex into 2^n *child* simplices, where n denotes the space dimension. Therefore this leads to a straightforward method in one and two space dimensions, where simplices can be subdivided into 2^n congruent sub-simplices. In one and two space dimensions this refinement method leads to refined meshes of the *same quality* as the original meshes. However, in three space dimensions it is not possible to divide a tetrahedron into eight congruent children and therefore it is important to choose a criterion for the refinement such that a sequence of refined meshes cannot degenerate. In example 8.1.2 we give a criterion that seems to avoid the degeneration of meshes (at least in our test cases) and results in refined meshes of *quite good quality.*

In this section we describe a simplex $\Delta \subset \mathbb{R}^n$ by its n vertices $[p_0, \ldots, p_n]$. The simplex Δ is then defined by the convex hull of these vertices. Now let the parent simplex be given by

$$\Delta^p = [p_0, \ldots, p_n].$$

We define the 2^n children of the parent simplex by

$$\Delta_i^c = \left[\frac{1}{2}\left(p_{\alpha(i,0,0)} + p_{\alpha(i,0,1)}\right), \ldots, \frac{1}{2}\left(p_{\alpha(i,n,0)} + p_{\alpha(i,n,1)}\right)\right], \quad i = 0, \ldots 2^n - 1,$$

where the function $\alpha : \{0, \ldots, 2^n - 1\} \times \{0, \ldots, n\} \times \{0, 1\} \to \{0, \ldots, n\}$ must be chosen such that the 2^n children form a *regular* subdivision parent cell Δ^p. For $n = 1$ and $n = 2$ this is straightforward. In the following the point p_{kl} denotes the midpoint of point p_k and p_l, i.e., $p_{kl} = \frac{1}{2}(p_k + p_l)$. In one space dimension the children can then be defined by

$$\Delta_0^c = [p_0,\ p_{01}], \qquad \Delta_1^c = [p_{01},\ p_1], \tag{8.1}$$

and in the two space dimensions

$$\begin{aligned} \Delta_0^c &= [p_0,\ p_{01},\ p_{02}], & \Delta_2^c &= [p_{02},\ p_{12},\ p_2], \\ \Delta_1^c &= [p_{01},\ p_1,\ p_{12}], & \Delta_3^c &= [p_{12},\ p_{02},\ p_{01}]. \end{aligned} \tag{8.2}$$

Note that in one and two space dimensions the subdivision into 2^n congruent cells is unique up to renumbering of the children.

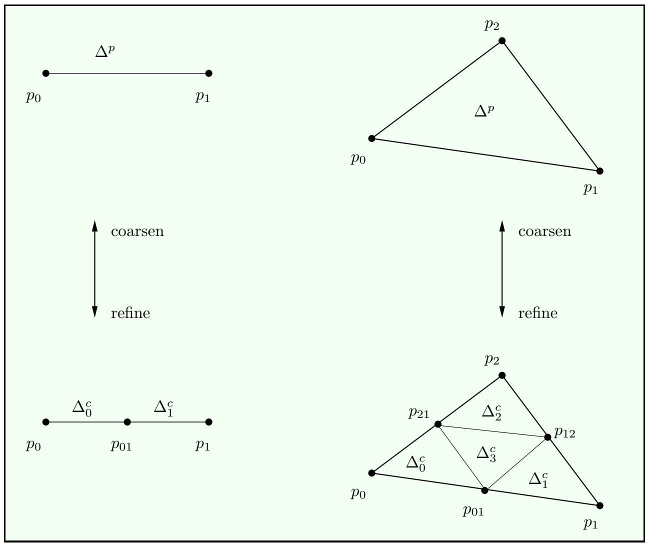

Figure 8.1: Refinement of one and two dimensional simplices.

Now, in three space dimensions the situation is more complicated because the partition of a parent cell into eight children is not unique and therefore one has to decide how to partition and this decision should ensure that successively refined meshes cannot degenerate. The eight children of a three dimensional parent cell can be defined by

$$\begin{aligned}
\Delta_0^c &= [p_0,\ p_{01},\ p_{02},\ p_{03}], & \Delta_4^c &= [q_4,\ p_{23},\ p_{13},\ p_{12}],\\
\Delta_1^c &= [p_{01},\ p_1,\ p_{12},\ p_{13}], & \Delta_5^c &= [p_{23},\ q_5,\ p_{03},\ p_{02}],\\
\Delta_2^c &= [p_{02},\ p_{12},\ p_2,\ p_{23}], & \Delta_6^c &= [p_{13},\ p_{03},\ q_6,\ p_{01}],\\
\Delta_3^c &= [p_{03},\ p_{13},\ p_{23},\ p_3], & \Delta_7^c &= [p_{12},\ p_{02},\ p_{01},\ q_7],
\end{aligned} \tag{8.3}$$

where the vertices q_4, q_5, q_6, q_7 must be defined such that the eight children form a valid partition of the parent cell. This results in three possibilities for the choice of these vertices.

$$\begin{aligned}
\text{(i)}\ \ & q_4 = q_5 = p_{01} && \text{and} && q_6 = q_7 = p_{23},\\
\text{(ii)}\ \ & q_4 = q_6 = p_{02} && \text{and} && q_5 = q_7 = p_{13},\\
\text{(iii)}\ \ & q_4 = q_7 = p_{03} && \text{and} && q_5 = q_6 = p_{12}.
\end{aligned}$$

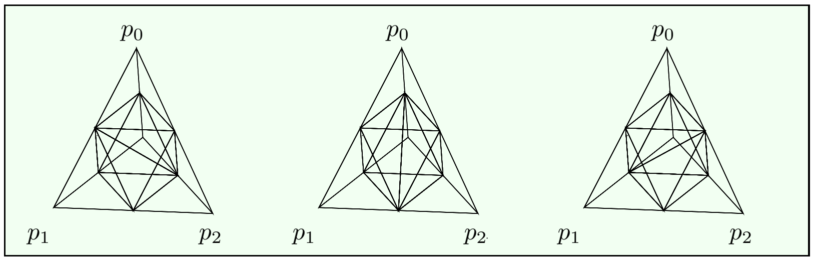

Figure 8.2: Three alternatives (i), (ii), (iii), from left to right, for the refinement of a three dimensional simplex.

At this point we have to decide which one of the three possibilities to choose. One possible choice is to guarantee that one of the points q_4, q_5, q_6, q_7 always lies on the longest edge of the tetrahedron. We call this the *longest edge criterion*.

Example 8.1.1 (Longest Edge Criterion)

We define the η by the length of the longest edge of the tetrahedron, i.e.,

$$\eta = \max\{|p_k - p_l| \mid k, l \in \{0,1,2,3\}\ \},$$

and we decide in the following way

if ($|p_0 - p_1| = \eta$ **or** $|p_2 - p_3| = \eta$) **then** set $q_4 = q_5 = p_{01},\quad q_6 = q_7 = p_{23}$,

else if ($|p_0 - p_2| = \eta$ **or** $|p_1 - p_3| = \eta$) **then** set $q_4 = q_6 = p_{02},\quad q_5 = q_7 = p_{13}$,

else set $q_4 = q_7 = p_{03},\quad q_5 = q_6 = p_{12}$.

In [134] is reported that applying this criterion on successively refined meshes can lead to a degenerate sequence of meshes, i.e., the smallest angle in a sequence of meshes is not bounded from below. Therefore the longest edge criterion seems to be not the optimal choice. Numerical examples show that using another criterion which we call the *longest two edges criterion*, that takes both edges associated with the vertices q_4, q_5, q_6, q_7 into account, gives much better results (at least in the configurations that have been tested).

Example 8.1.2 (Longest two Edges Criterion)

We define

$$\begin{aligned}\eta_1 &= |p_0 - p_1| + |p_2 - p_3|,\\ \eta_2 &= |p_0 - p_2| + |p_1 - p_3|,\\ \eta_3 &= |p_0 - p_3| + |p_1 - p_2|,\\ \eta &= \max\{\eta_l \mid l = 1,2,3\},\end{aligned}$$

and we decide in the following way

if $(\eta_1 = \eta)$ **then** set $q_4 = q_5 = p_{01}, \quad q_6 = q_7 = p_{23}$,

else if $(\eta_2 = \eta)$ **then** set $q_4 = q_6 = p_{02}, \quad q_5 = q_7 = p_{13}$,

else set $q_4 = q_7 = p_{03}, \quad q_5 = q_6 = p_{12}$.

Numerical examples show that the smallest angle in a sequence of successively refined meshes stays bounded from below using this criterion. The inverse of this criterion, the *shortest two edges criterion*, seems always to produce a sequence of degenerating meshes.

8.2 L^2 Projection of Data in the Adaption Process

We assume that a n-dimensional simplex $\Delta = T(\hat{\Delta})$ has been refined into 2^n children $\Delta_i = T_i(\hat{\Delta})$, $i = 0, \ldots, 2^n - 1$ as discussed in the previous section, where $\hat{\Delta}$ denotes the reference cell and T, T_i the reference mappings from the reference cell to the cells Δ and Δ_i respectively, see (6.1). This section is not restricted to three space dimensions as long as we assume that we already have constructed 2^n children of a cell. Due to the linearity of the reference mappings the projection of data in the refinement and coarsening process is just a matrix-matrix multiplication and no further geometry information of the mesh cells is necessary. The matrix-matrix multiplication for refinement and coarsening is given in equations (8.7) and (8.8).

Let denote $n_p = |\mathbb{P}_m|$ the dimension of the space of polynomials of degree at most m in n space dimensions as defined in section 6.3. At the moment, for simplicity, we consider only scalar data on the cells Δ and Δ_i of the form

$$u(x) = \sum_{l=0}^{n_p-1} \alpha_l \varphi_l, \quad u_i(x) = \sum_{l=0}^{n_p-1} \beta_l^i \varphi_l^i,$$

where the functions φ_l and φ_l^i denote the local basis functions as defined in Section 6.3, see (6.10). In the refinement process we have to compute the coefficients β_l^i from the coefficients α_l in some way. We do this by means of a L^2-projection. This means for all i we provide the data u_i on the sub-cells Δ_i by L^2-projection of u.

$$\int_{\Delta_i} u(x)\varphi_k^i(x)\,dx = \int_{\Delta_i} u_i(x)\varphi_k^i(x)\,dx, \quad k = 0, \ldots, n_p - 1.$$

This means

$$\sum_{l=0}^{n_p-1} \alpha_l \int_{\Delta_i} \varphi_l(x)\varphi_k^i(x)\,dx = \sum_{l=0}^{n_p-1} \beta_l^i \int_{\Delta_i} \varphi_l^i(x)\varphi_k^i(x)\,dx, \quad \forall k.$$

And by transformation of the cell Δ_i to the reference cell $\hat{\Delta}$ using the reference mapping T_i gives

$$\sum_{l=0}^{n_p-1} \alpha_l \int_{\hat{\Delta}} \varphi_l(T_i\hat{x})\varphi_k^i(T_i\hat{x})\,d\hat{x} = \sum_{l=0}^{n_p-1} \beta_l^i \int_{\hat{\Delta}} \varphi_l^i(T_i\hat{x})\varphi_k^i(T_i\hat{x})\,d\hat{x}, \quad \forall k.$$

Note that the transformation is affine linear and the factor $|det(DT_i(x))|$ from the transformation has been eliminated on both sides of the equation. Using the definition of the local basis functions φ_l and φ_l^i and the orthonormality of the functions p_l on the reference cell we get

$$\sum_{l=0}^{n_p-1} \alpha_l \int_{\hat{\Delta}} p_l(T^{-1}T_i\hat{x})\,p_k(\hat{x})\,d\hat{x} = \beta_k^i, \quad k = 0,\dots,n_p-1.$$

The combination of the mappings $T^{-1}T_i$ does not depend on the cell Δ. In fact we have $T^{-1}T_i = \hat{T}_i$, where $\hat{T}_i$ denotes the affine linear mapping from the reference cell to the i-th child of the reference cell. This finally gives the expression

$$\beta_k^i = \sum_{l=0}^{n_p-1} \alpha_l \int_{\hat{\Delta}} p_l(\hat{T}_i\hat{x})\,p_k(\hat{x})\,d\hat{x}, \quad k = 0,\dots,n_p-1,$$

which only depends on the number of the child and not on the geometry of the cell Δ. Now let us define for all children the matrices $\boldsymbol{A}^i \in \mathbb{R}^{n_p\times n_p}$ by

$$\boldsymbol{A}^i = \left(\int_{\hat{\Delta}} p_l(\hat{T}_i\hat{x})\,p_k(\hat{x})\,d\hat{x}\right)_{k,l} \tag{8.4}$$

and we compose them to a single *adaption* matrix $\boldsymbol{A} \in \mathbb{R}^{2^n n_p\times n_p}$ by

$$\boldsymbol{A} = \begin{pmatrix} \boldsymbol{A}^0 \\ \boldsymbol{A}^1 \\ \vdots \\ \boldsymbol{A}^{2^n-1} \end{pmatrix}. \tag{8.5}$$

Now, to be more general, we consider vector valued data of dimension $d \in \mathbb{N}$

$$\boldsymbol{u}(x) = \sum_{l=0}^{n_p-1} \varphi_l \boldsymbol{\alpha}_l, \quad \boldsymbol{u}_i(x) = \sum_{l=0}^{n_p-1} \varphi_l^i \boldsymbol{\beta}_l^i, \tag{8.6}$$

associated with the cells Δ and Δ_i where α_l and β_l^i are vectors in $\mathbb{R}^d$. We define matrices $\boldsymbol{\alpha}, \boldsymbol{\beta}^i \in \mathbb{R}^{n_p \times d}$ by

$$\boldsymbol{\alpha} = (\boldsymbol{\alpha}_0, \boldsymbol{\alpha}_1, \ldots, \boldsymbol{\alpha}_{n_p-1})^T, \quad \boldsymbol{\beta}^i = (\boldsymbol{\beta}_0^i, \boldsymbol{\beta}_1^i, \ldots, \boldsymbol{\beta}_{n_p-1}^i)^T,$$

and we compose the matrices $\boldsymbol{\beta}^i$ to a single matrix $\boldsymbol{\beta} \in \mathbb{R}^{2^n n_p \times d}$ by

$$\boldsymbol{\beta} = \begin{pmatrix} \boldsymbol{\beta}^0 \\ \boldsymbol{\beta}^1 \\ \vdots \\ \boldsymbol{\beta}^{2^n-1} \end{pmatrix}.$$

Using the above notation the projection of the data in the **Refinement** process can be carried out by the single matrix-matrix multiplication

$$\boldsymbol{\beta} = \boldsymbol{A}\boldsymbol{\alpha}. \tag{8.7}$$

The reverse process of the refinement process is the coarsening process. Here the problem is to compute the coefficients $\boldsymbol{\alpha}$ from given coefficients $\boldsymbol{\beta}$. Using the same notation as above the data projection in the **Coarsening** process is done by the single matrix-matrix multiplication

$$\boldsymbol{\alpha} = \frac{1}{2^n}\boldsymbol{A}^T\boldsymbol{\beta}. \tag{8.8}$$

To see that this is correct one has to start similar to the refinement process by a L^2-projection of the functions u_i to the cell Δ. Constructing matrices in a similar way to the refinement process results in the formula above.

In one and two space dimensions there is only one adaption matrix $\boldsymbol{A}$ because the refinement of a simplex into 2^n sub-cells as discussed in the previous section is unique. In three space dimensions there are three different adaption matrices, one for each of the three different refinement patterns. According to the choice of the refinement pattern the correct matrix associated with this pattern has to be chosen for the projection of the data.

There are a lot of zeros in the matrices $\boldsymbol{A}^i$, especially below the diagonal. when the projection of data becomes a bottleneck in the computation this could be exploited for a more efficient implementation of the projection. However, this is not the case in our applications.

8.3 Refinement and Coarsening Indicator

The goal of this section is to provide a criterion to decide when a cell of the mesh is to large, such that we have to refine it, and when a cell is to small and should be coarsened (if possible).

In the framework of Finite Volume and Discontinuous Galerkin discretization of conservation laws there are essentially two different kinds of strategies for this decision.

The first strategy is to use *error estimators* based on rigorous theoretical results. Usually these are only available for special cases of conservation laws. The second kind of strategies is based on *heuristic indicators* that are easy to compute, applicable to a large class of problems and give usually good results in practical applications. However, there is no theoretical justification as for the error estimators.

- *Error estimators* based on rigorous analysis, i.e., a aposteriori error control of the form

  $$||u - u_h||_K \leq \eta_h(u_h),$$

  where u is the exact solution of some conservation law, u_h an approximation of u generated by a numerical scheme, $K \subset \Omega$ is some compact set in the computational domain and $|| \cdot ||$ denotes some norm. The strategy consists (roughly) of the following procedure: If the right hand side $\eta_h(u_h)$ is too large then we refine the mesh-cells associated with the set K, in the case the right hand side is too small, the parts of the mesh associated with K should be coarsened to reduce the computational cost to a minimum. Usually it is not guaranteed that the right hand side converges to zero as the mesh size h tends to zero. Therefore this strategy does not always guarantee convergence of the algorithm. Aposteriori error estimates of the above form are available for first order Finite Volume schemes for multi dimensional scalar conservation laws (Cauchy problem and initial boundary value problem), e.g. [77], [89], [90], for nonlinear hyperbolic systems of balance laws with classical solutions see [67]. For higher order Runge-Kutta Discontinuous Galerkin approximations of multidimensional nonlinear scalar conservation laws an aposteriori error estimate can be found in [38].

- *Heuristic indicators* which (in the framework of fluid dynamics) usually depend on the local gradients (with respect to space or time-space) of thermodynamical variables such as density, energy, entropy and others. In this case the criterion is quite simple: a large gradient (measured in some norm) leads to refinement of the cell, a small gradient might lead to coarsening of a group of cells.
  The advantages of this kind of simple heuristic indicators are: They are (usually) easy to compute, they are available for complicated systems in contrast to real error estimators, and they have been successfully applied to many different problems, see for example [13], [37], [116], [50], [112].
  One disadvantage of heuristic indicators is that they may indicate for refinement even in cases where the error between exact and approximate solution is small (possibly equal to zero).

For our application, the higher order Discontinuous Galerkin discretization of the complicated Navier-Stokes-Korteweg system, an error estimator based on rigorous analysis seems to be out of scope. However, the most important challenge in the framework of the Navier-Stokes-Korteweg system is the complete resolution of the interface by the underlying computational mesh rather than the error control. As an heuristic indicator we can use the density gradient $|\nabla \rho|$. At the interface the density changes rapidly and therefore the density gradient is large. This can be used to provide a very fine mesh close to the interface.

We define a quantity η_i that is associated with the gradient of the density on the i-th cell and its direct and indirect neighbors in the mesh

$$\eta_i = \max\left\{ \frac{\mathrm{diam}(\Delta_j)}{|\Delta_j|} ||\nabla\rho||_{L^2(\Delta_j)} \mid \Delta_j \text{ is neighbor of } \Delta_i \text{ of deg. at most } m \right\}. \quad (8.9)$$

Here we call Δ_j a neighbor of Δ_i of degree at most m if there exist $m+1$ cells in the mesh $\Delta^0, \dots, \Delta^m$ with

$$\begin{aligned} &\Delta_i = \Delta^0, \\ &\Delta_j = \Delta^m, \\ &\Delta^k \text{ is a neighbor of } \Delta^{k+1}, \quad k = 0, \dots, m. \end{aligned}$$

Note: for $m = 0$ we have only a contribution of the i-th cell and for $m = 1$ we have contributions of the gradients of the i-th cell and its direct neighbors.

We choose some problem dependent upper and lower values

$$\eta_{upp} > 0 \text{ and } \eta_{low} > 0 \quad \text{with} \quad \eta_{low} < \eta_{upp} \quad (8.10)$$

and we decide if the i-th cell is a candidate for refinement or coarsening according to the following criterion

Example 8.3.1 (Mesh Adaption Criterion)

if $\eta_i > \eta_{upp}$ **then** we mark the i-th cell for refinement,

else if $\eta_i < \eta_{low}$ **then** we mark the i-th cell for coarsening.

Note: Here we only set a mark (by setting a flag) that means the cell is a candidate for refinement or coarsening. The final decision whether the refinement or coarsening is performed is discussed in the next section.

The building of the maximum in the definition of η_i in (8.9) is rather expensive especially for large values m and parallel computation, but it is important to have a layer of fine cells around an interface. Thus, it is important to choose a value $m > 1$. Otherwise local equilibrium configurations will be destroyed in the refinement and coarsening process and this will slow down the underlying iterative linear solvers.

8.4 Refinement and Coarsening of Simplicial Meshes

In this section we discuss the final refinement and coarsening of a mesh. Here we assume that the cells of the mesh are already marked for refinement and coarsening, i.e., refinement or coarsening flags of the cells are set due to the decision discussed in the previous section. Since the Discontinuous Galerkin method does not need conform meshes, refinement and coarsening could be very local without affecting neighboring cells. However, in order to improve the stability of the method it is convenient to discard the locality to some degree by restricting the *level of nonconformity* to one, i.e., the absolute difference between refinement levels of neighbor cells. In the following we assume that the mesh consists of a set of *macro cells* that cannot be further coarsened. This set is called *macro mesh*. Due to the refinement and coarsening procedure a hierarchy of cells with a parent-children relation is constructed. The set of cells that do not have children is called *leaf mesh*. Only all children of a parent cell together can be coarsened to the parent cell.

Note that initially, when the mesh adaption criterion 8.3.1 is applied, it is not possible that cells are marked with the refinement and coarsening flag simultaneously but this can happen during the refinement and coarsening process since the nonconformity level is restricted. In the following we assume that when the refinement flag of a cell is set then the coarsening flag is unset. Further we assume that a macro cell cannot have the coarsening flag set. The mesh adaption follows a *coarsening can, refinement must* policy. This means every cell that has the refinement flag set has definitely to be refined whereas the execution of coarsening of a cell (with a set coarsening flag) depends on the neighborhood of the cell.

First the refinement algorithm is carried out. The following has to be done until there is no cell left with a set refinement flag.

Algorithm 8.4.1 (Refinement)

```
while there is a cell $\Delta$ with refinement flag set {
  refine $\Delta$ into subcells $\Delta_0, \dots, \Delta_{n-1}$;

  for $i = 0, \dots, n-1$ {
    for all neighbors $\tilde{\Delta}$ of $\Delta_i$ {
      if refinement_level($\Delta_i$) $-$ refinement_level($\tilde{\Delta}$) $> 1$ then
      set the refinement flag of $\tilde{\Delta}$;
    }
  }
}
```

When the refinement has finished the coarsening algorithm has to be executed until all cells have been processed and no flags are set.

Algorithm 8.4.2 (Coarsening)

```
while there is a cell $\Delta$ with coarsening flag set {
  set $\Delta_p$ = parent cell of $\Delta$;

  if all children of $\Delta_p$ have the coarsening flag set then {
    if for all neighbors $\tilde{\Delta}$ of $\Delta_p$ we have
    refinement_level($\tilde{\Delta}$) - refinement_level($\Delta_p$) $\leq 1$ then
    coarsen all children of $\Delta_p$ to $\Delta_p$;

    else if for all neighbors $\tilde{\Delta}$ of $\Delta_p$ we have
    refinement_level($\tilde{\Delta}$) - refinement_level($\Delta_p$) $\leq 2$ and all neighbors
    with "$= 2$" have the coarsening flag set then
    requeue cell $\Delta$;
```

```
        **else** unset the coarsening flag of $\Delta$;
      }
      **else** unset the coarsening flag of $\Delta$;
    }
```

In the coarsening algorithm above we note that in the **else if** statement we cannot decide whether the cell can be coarsened or not since some of the neighbor cells have to be coarsened (or not) first. Thus, the current cell has to be *requeued* in a waiting queue of some kind. Here it is important for the termination of the algorithm that the current cell is requeued at the end of the waiting queue such that all other marked cells are processed before the current cell is processed again.

8.5 Parallelization

In the following we discuss the parallelization of the methods from the previous sections and chapters. The parallelization using the distributed memory parallelization concept is the appropriate choice for Finite Volume and Discontinuous Galerkin methods since the partitions of the mesh given by a domain decomposition method are only *weakly coupled*.

Today there are mainly two different parallel programming models *Shared Memory Parallelization* and *Distributed Memory Parallelization*. The advantages and disadvantages of these both models are listed in the following.

- *Shared Memory Parallelization*. Using this model the application makes use of many *Threads of Execution* that share a common address space in memory. This means each thread can read or write to each location in memory. This programming model can be used either by explicitly working with threads, for example by using the PThreads API. Or alternatively OpenMP [35], [126] directives can be used to spawn threads in *parallel regions*. These directives are available for the programming languages C, C++ and Fortran but they are not part of the standards of the languages and must be additionally implemented by the compiler.

  The main disadvantage of this model on the side of the hardware is that SMP machines with a large number of processors are really expensive. One problem on the software side is that many libraries are not completely *thread safe*. Another problem is that it is easy to cause effects like *cache thrashing* on modern machines since all threads have access to all memory locations. One has to be aware of these effects otherwise the parallelization gives no gain in performance.

- *Distributed Memory Parallelization*. In this model the application runs using different *Processes* on the same or different machines which communicate via *Message Passing*. The de facto standard for scientific computing applications that make use of this programming model is the *Message Passing Interface* (short MPI) [122] that defines a lot of useful routines commonly used in scientific applications. An alternative to MPI is the PVM (*Parallel Virtual Machine*) library [128], [49] which is, compared to the MPI implementations, a light weight library that provides message passing. But this comes at the cost that it is not as optimized as

MPI implementations.

The latest MPI standard (at time of this writing) is version 2.0 and can be found on the MPI-Forum website [122]. There are a number of open source implementations of MPI. All of them implement at least the 1.1 standard and some of them implement parts of, or the complete 2.0 standard. These are MPICH [53], [123], LAM/MPI [18], [120], MPICH2 [124] and OpenMPI [46], [127]. The latter two of them implement (or will do it in the near future) the complete MPI 2.0 standard. A good overview and many additional references to the Message Passing Interface are given in [54], [55].

The main advantage of this programming models is that simply a bunch of machines connected via a network can be used as a parallel computer. This is normally much cheaper and for many applications nearly equivalently efficient as using shared memory machines. Another advantage is that memory partitions are separated and effects like cache thrashing do not occur. Of course, message passing can also be used on SMP-machines. In this case modern MPI implementations communicate via shared memory which is the fastest way to communicate.

In this work we have chosen a distributed memory parallelization because of its flexibility and usability with cheap hardware. In the framework of Finite Volume and Discontinuous Galerkin methods a distributed parallelization based on a domain decomposition seems to be the most appropriate choice. Here the computational domain represented by an underlying mesh is partitioned into pieces and each of the pieces is distributed among the available processors. Figure 8.3 shows the original mesh on the left and the mesh partitioned into three pieces on the right. Additionally the *overlap of level one* is shown. These are cells from the other partitions that store the connectivity information to cells of the other partitions.

Parallel efficiency

There are mainly two different motivations for parallel implementation of software. Namely:

- The problem is too large. We have a larger memory requirement.
- The computation needs too much time to finish. We need a faster execution of the code.

In the first case we have no other choice: we need enough machines to satisfy the memory requirements of the problem. But in the latter case we have to decide how many machines we should use in order to accelerate the computation. This is discussed in the following paragraph.

For a given problem let $T(n)$ denote the time the computation using $n \in \mathbb{N}$ processors needs to finish. The speedup from n_0 processors to n_1 processors with $n_0 < n_1$ is then defined by

$$\begin{aligned} \text{speedup}(n_0, n_1) &= \frac{T(n_0)}{T(n_1)}, \\ \text{speedup}(n) &= \text{speedup}(1, n). \end{aligned}$$

The latter denotes the speedup from one to n processors. For real world applications we expect that the speedup is bounded by

$$\text{speedup}(n_0, n_1) \leq \frac{n_1}{n_0},$$

but there are examples where this is not the case. These are usually cache effects and occur normally only when the problem is *small* and the communication is really fast compared to the computation. The expression

$$\text{eff}(n_0, n_1) = \text{speedup}(n_0, n_1)\frac{n_0}{n_1}$$

is called parallel efficiency. This quantity decides whether it is worth to use n_1 processors instead of n_0 or not. If this quantity is close to one it may be worth if it is close to zero it is definitely not worth.

Parallelization of the mesh

In Section 6.1 we have discussed the structure of conform and nonconform meshes we use to approximate (possibly complex) geometries. For the implementation, especially in a parallel environment, it is convenient to restrict the set of admissible meshes a little bit. Here we restrict this set to the set of meshes that can be generated by refinement starting from a conform *macro mesh*. The macro mesh consists of cells that cannot be further coarsened. The mesh is initially assumed to be distributed among the available processors. Not all of the processors need to hold any macro cells. The part of the mesh (the partition) that is held by processor number p is called $\mathcal{T}_{h,p}$ and the part of the domain that is associated with this partition is denoted by $\Omega_{h,p}$ in the following. A partition $\mathcal{T}_{h,p}$ needs also to store the immediate neighbors of macro cells that are adjacent to other partitions in order to store connectivity information. These neighbors are called overlapping macro cells of level one. Note that for the discretization by the Discontinuous Galerkin approach the information of direct neighbors is sufficient.

Figure 8.3 shows the mesh of the unit ball in $\mathbb{R}^2$ that is hold on only one processor on the left side and on the right side the same mesh is shown that is distributed among three processors. The blue cells represent the overlapping cells from other partitions that store the connectivity information.

In the refinement and coarsening process a hierarchy of cells is constructed with the macro cell as the root cell. In the case where cells are sent from one process to another, for example when load balancing (see 8.6) is necessary because the computational cost differs too much between the processors, then the complete cell tree with the macro cell as root is sent to another process. When refinement or coarsening is performed somewhere in the hierarchy of a macro cell of process p that is part of the overlap of a process q then process p needs to inform process q about the new structure of the concerning macro cell.

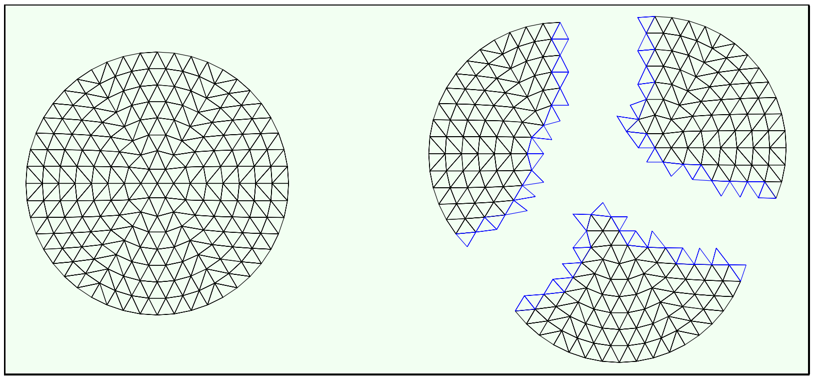

Figure 8.3: Decomposition of a mesh into three parts and the overlap of level one.

Parallelization of the Discontinuous Galerkin method

The space discretization by the Local Discontinuous Galerkin method is discussed in Chapter 6 and especially in Section 6.2. In each stage of the computation of a higher order differential operator by the LDG method the communication only between direct neighboring cells is necessary. Therefore the mesh on one partition needs only to store the information about direct neighbor cells in different partitions. As discussed in the previous paragraph, in this case the complete macro cell hierarchy is stored for simplicity.

The computation of the discrete differential operators associated with the Local Discontinuous Galerkin method in one stage is then done essentially in the following way:

- The data at the boundaries of the partitions is exchanged between processors using *nonblocking* MPI communication.
- Since nonblocking communication is used it can overlap with the computation in the inner partition (the large part of the computation). Communication is done in the background by the system.
- When all necessary data has been received from neighbor partitions, the rest of the computation that is associated with the partition boundaries can be done (the small part of the computation).

In the following we denote by $\boldsymbol{u}_{h,p}$ the part of the approximate solution that is associated with the partition $\mathcal{T}_{h,p}$, i.e., the part of the approximate solution that has support on the domain $\Omega_{h,p}$. The local parts of the discrete first order differential operators are denoted by $\mathcal{L}^1_{h,p,k}$. Using this notation the pseudocode algorithm to compute the discrete higher order differential operator in m stages in equation (6.6) can be written in a parallelized version. Each process p does the following:

Algorithm 8.5.1 (Parallel DG Method)

```
set $\boldsymbol{u}_{h,p}^0 = \boldsymbol{u}_{h,p}(\cdot, t)$;

for $k = 1, \dots, m$ {
    for $q = 0, \dots, n_{parts} - 1,\ q \neq p$ {
        if $p$ and $q$ are adjacent partitions then
        send the parts of $(\boldsymbol{u}_{h,p}^0, \dots \boldsymbol{u}_{h,p}^{k-1})$ that are associated with the partition
        boundary to process $q$ using nonblocking communication;
    }

    compute the part of $\mathcal{L}_{h,p,k}^1[(\boldsymbol{u}_h^0, \dots \boldsymbol{u}_h^{k-1})]$ using physical fluxes $\boldsymbol{f}_i^k$ and
    consistent numerical fluxes $\boldsymbol{g}^k$ for which only the inner data is necessary;

    wait until all necessary data from neighbor partitions has been received;

    compute the rest of $\mathcal{L}_{h,p,k}^1[(\boldsymbol{u}_h^0, \dots \boldsymbol{u}_h^{k-1})]$;

    set $\boldsymbol{u}_{h,p}^k = \mathcal{L}_{h,p,k}^1[(\boldsymbol{u}_h^0, \dots \boldsymbol{u}_h^{k-1})]$;
}

set $\mathcal{L}_{h,p}^m[\boldsymbol{u}_h(\cdot, t)] = \boldsymbol{u}_{h,p}^m$.
```

In the above algorithm n_{parts} denotes the number of partitions/processors in the parallel environment. Note that the treatment of nonconservative products can also be included in the algorithm.

Parallelization of linear and nonlinear solvers

The parallelization of the rest of the code is more or less straight forward. The most important remaining components are the explicit and implicit ODE solvers, nonlinear solvers and linear solvers. Parallelization of these components is done in the following way: First a local (with respect to a partition/processor) result is computed and second these local results are used to construct the global result by using global reduction operations provided by the Message Passing Interface.

ODE solvers need to compute a time step size. Explicit solvers need to do this for stability reasons and implicit solvers need to find an optimal time step. Usually this is done by computing a time step locally on each partition followed by building a minimum over all partitions. The final step is done by calling the MPI global reduction method **MPI_Allreduce(. . . , MPI_MIN)**.

Linear and nonlinear solvers need to compute dot products. Linear solvers usually need this as part of the iteration and nonlinear solvers for a stopping criterion. Dot products are computed locally for each partition. This operation is completed by building the sum over all partitions. Again this is a global reduction operation provided by MPI. The method to call in this case is **MPI_Allreduce(. . . , MPI_SUM)**. Note that computing dot products in parallel can be different from the serial computation due to roundoff errors since the operations are usually reordered. This can result in a slightly different number of iterations for the linear solvers.

Parallelization of other components

Some of the components like the interface or error indicators are not discussed in the previous paragraphs since the parallelization of these components is similar or a combination of the components discussed before. Therefore, we omit it. The parallelization of the refinement and coarsening algorithm discussed in Section 8.4 is also not discussed here. The information of refinement and coarsening has to be simply exchanged between the partitions. This process can cause additional adaption in other partitions. Thus, the process has to be iterated until all partitions have finished.

8.6 Load Balancing

Load balancing is a technique to distribute work between the available processors in a parallel environment in order to optimize and decrease computing time. In the case where the computational cost or communication cost differs too much between processes, due to local mesh adaption, the load must be balanced with respect to

- the computational cost, i.e., the number of cells of the mesh a partition holds.
- the communication cost. The number of interfaces between different partitions should be minimized.
- the redistribution cost. The exchange of parts of the partitions should not be too expensive.

For purpose of the first item we assign weights to the macro cells of the mesh which represent the count of leaf cells in the cell hierarchy of the macro cell. The second item is taken into account by assigning weights to the interfaces of the macro cells which represent the communication cost between two adjacent macro cells.

As backend for the partitioning of the macro mesh the graph partitioning library ParMetis [121], [68] is used. The graph that describes the connectivity information of the macro mesh is converted into ParMetis' data structure. This structure is a kind of a parallel CSR matrix format, see the ParMetis documentation [69] for details. The ParMetis library provides the methods **PartKway(...)** and **AdaptiveRepart(...)**. The former method is used for initial partitioning and the latter is used for repartitioning. The repartition method preserves as much of the initial structure of the mesh as possible in order to minimize the redistribution cost. The output of both ParMetis methods is a local **part** array with the information which node of the graph has to be sent to which processor, i.e., which macro cell has to be sent to which partition, in order to improve the distribution of computational cost and minimize the communication cost.

ParMetis provides additionally the possibility to assign individual weights to the partitions using the **tpwgts** array. This can be used to improve the load balancing in heterogen parallel environments, i.e., networks of machines with different computational power.

Chapter 9

Numerical Results

We apply the test cases constructed in Chapter 4 and some other tests in the following chapter. All tests throughout the following sections are related to the higher order Discontinuous Galerkin discretization of the Navier-Stokes-Korteweg equations (isothermal or temperature dependent) in multiple space dimensions discussed in Chapter 6. In the following sections we perform a lot of different tests that are dedicated to

- the quantitative behavior of numerical solutions, convergence tests with the static equilibrium solutions and traveling wave solutions in multiple space dimensions.
- the qualitative behavior of numerical solutions, decay of the total energy on the discrete level, vanishing velocity field.
- the efficiency of the used numerical techniques, parallelization, local mesh adaption.
- the qualitative behavior of solutions of the Navier-Stokes-Korteweg model, oscillating bubbles, condensation of bubbles.

In the above list one point is missing: tests with respect to the quantitative behavior of solutions. Here the problem is that no physical data of experiments are available on the temperature and length scale that can be simulated by the NSK model, see Section 2.10.

9.1 Test Case: Static Equilibrium

The first test in this chapter is the test case with the static bubble in two and three space dimensions. As initial configuration we use the profiles we have computed in Section 4.1. We compare the exact solution with the approximate solutions generated by the well balanced Discontinuous Galerkin schemes discussed in Section 6.9 for different polynomial degrees after some time of computation. For time stepping we apply the implicit Runge-Kutta schemes described in Section 7.3.

The setting is the same as in Chapter 5 except for the domain and the boundary conditions. The boundary conditions that are imposed in this test case are (2.51) and (2.53) that enforces a 90 degree contact angle of the interface at the boundary. However, the interface should not touch the boundary in this case, so it should not make a difference when boundary condition (2.53) is replaced by the more general condition (2.54).

The remaining parameters are the capillarity coefficient λ which must correspond to the computed density profile and the viscosity parameters that can be chosen arbitrarily. However, we use the value that comes from the computation of the undercompressive traveling wave solution, see Section 4.2.

$$\begin{aligned} \lambda &= 0.001, \\ \varepsilon &= 0.0136644, \quad \mu = \frac{3}{4}\varepsilon, \quad \nu = -\frac{1}{2}\varepsilon. \end{aligned}$$

Static bubble in 2d

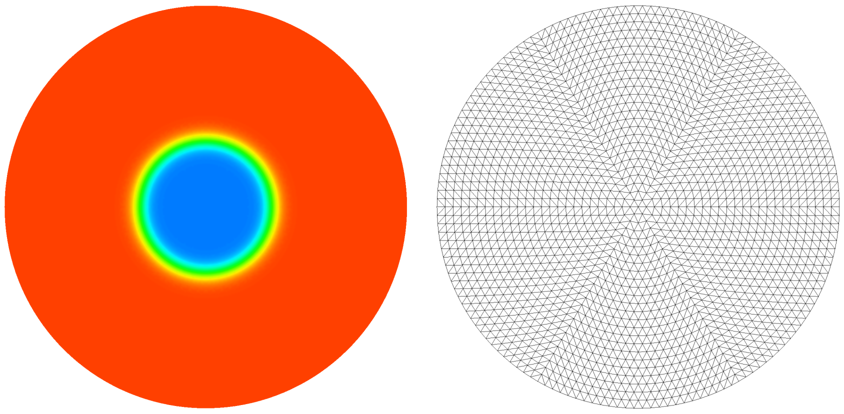

Figure 9.1: Static equilibrium bubble and a computational mesh in 2d.

Figure 9.1 shows the density distribution in the computational domain $\Omega = B_1(0) \subset \mathbb{R}^2$ of the initial configuration which is of course the solution for all times $t \geq 0$ in this case. The density values vary between approximately 0.3 (blue) and 1.8 (red). These two values are approximately the Maxwell values for the dimensionless van der Waals equation of state at temperature $\theta_{ref} = 0.85$. This display style is used throughout this chapter.

For this test we use globally refined, regular triangulations of different mesh sizes. One of them is shown in Figure 9.1. Figure 9.2 and Table 9.1 illustrate the convergence of the numerical schemes with the expected order (which is polynomial degree plus one) for $p = 1, 2, 3$.

The results of the sequence of tests in two space dimensions can be found in Table 9.1 and Figure 9.2. Computations that were not successful, because the mesh was not fine enough (this was the case for $p = 1$) or the computation was simply to expensive to finish in a reasonable time (in the case of $p = 3$ for the fine meshes) the results are marked with a * symbol in Table 9.1.

	p=1		p=2		p=3	
h	L^2 error	EOC	L^2 error	EOC	L^2 error	EOC
2.1298e-01	*		7.6272e-02		6.1959e-02	
1.3311e-01	8.0634e-02	*	2.7842e-02	2.144	1.0732e-02	3.730
8.8740e-02	3.1137e-02	2.347	6.9194e-03	3.434	2.3610e-03	3.734
6.2640e-02	1.3184e-02	2.467	4.0635e-03	1.528	3.0082e-04	5.915
4.2595e-02	7.8643e-03	1.340	9.2077e-04	3.849	1.1222e-04	2.557
2.8023e-02	3.6147e-03	1.856	2.4690e-04	3.144	2.1694e-05	3.925
1.9016e-02	1.7118e-03	1.928	7.3168e-05	3.137	5.6071e-06	3.489
1.2830e-02	7.9013e-04	1.965	2.1812e-05	3.076	1.5065e-06	3.340
8.6576e-03	3.6232e-04	1.982	6.6050e-06	3.037	4.2760e-07	3.202
5.8510e-03	1.6589e-04	1.994	2.0252e-06	3.017	*	*
3.9440e-03	7.5470e-05	1.997	6.1878e-07	3.006	*	*

Table 9.1: Static bubble in 2d. Total L^2 error and EOC for $p = 1, 2, 3$.

The numerical solutions generated by the schemes with polynomial degree $p = 0, 1, 2$ converge clearly with the expected order $p + 1$. For the fourth order scheme ($p = 3$) either the mesh size is not in the asymptotic regime or (and this is very likely the reason) the time step size has become too large (possible because of the equilibrium solution) such that the second order Runge-Kutta method destroys the order.

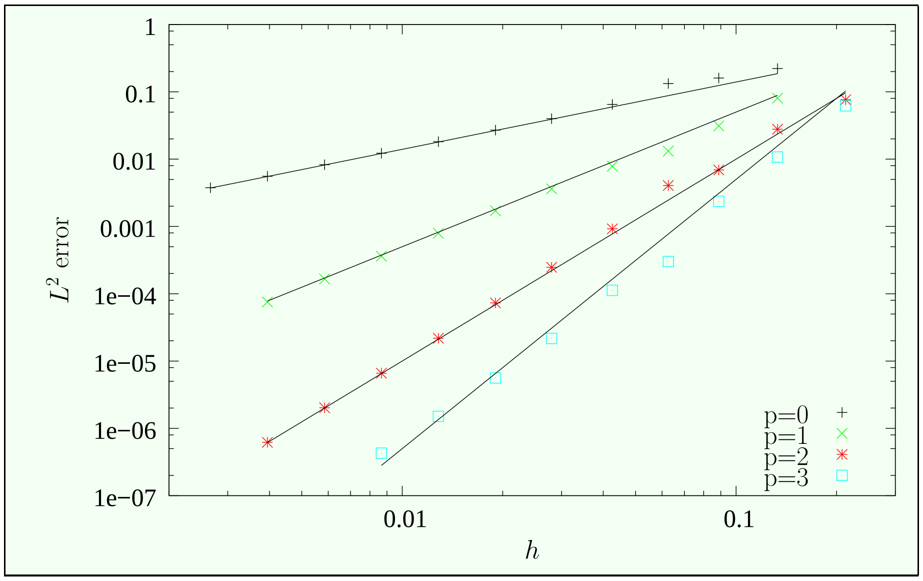

Figure 9.2: Static bubble in 2d. Mesh size versus L^2 error for $p = 0, 1, 2, 3$.

Static 2d bubble in 3d

We apply the 2d test case from the previous paragraph in three space dimensions. Therefore, we set the remaining velocity component to zero and choose as computational domain the cylinder $\Omega = B_1(0) \times [-0.1, 0.1] \subset \mathbb{R}^3$, where $B_1(0)$ denotes the unit ball in $\mathbb{R}^2$. Figure 9.3 shows the initial data and an underlying tetrahedral mesh for this test. Now it is important that at the bottom and at the top of the cylinder boundary condition (2.53) is imposed.

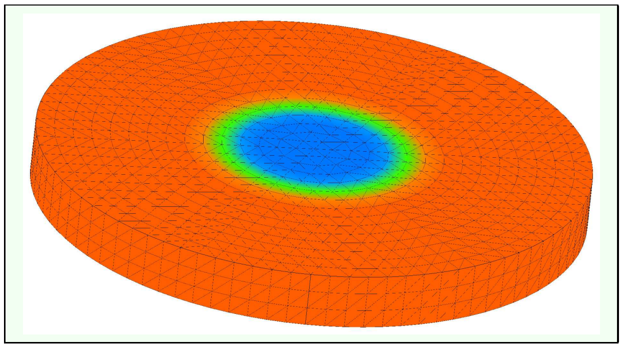

Figure 9.3: Static 2d bubble in 3d and a tetrahedral mesh.

The rest of the setting is identical to the setting in the 2d case in the previous paragraph. The results of the computations are shown in Figure 9.4 and Table 9.2 for polynomial degree (zero), one, two and three.

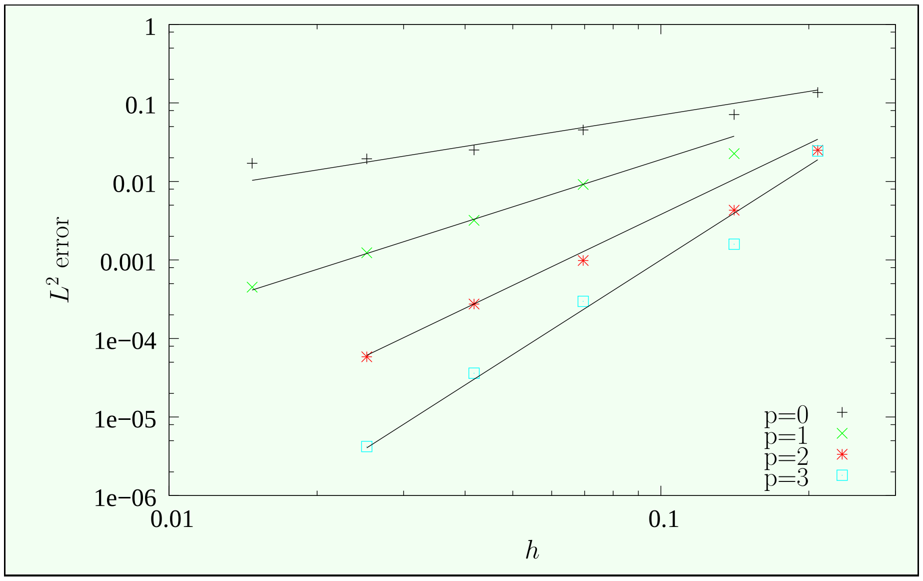

Figure 9.4: Static 2d bubble in 3d. Mesh size versus L^2 error for $p = 0, 1, 2, 3$.

The schemes with $p = 1, 2$ seem to converge with the expected order $p + 1$ but for the schemes with $p = 0$ and $p = 3$ the mesh seems to be not fine enough such that the schemes can be in the asymptotic regime. Note that the mesh is not as fine as in the purely $2d$ test case since convergence tests in three space dimensions are really expensive, regardless of parallelization.

	p=1		p=2		p=3	
h	L^2 error	EOC	L^2 error	EOC	L^2 error	EOC
2.0856e-01	*		2.4993e-02		2.4465e-02	
1.4094e-01	2.2658e-02	*	4.3036e-03	4.489	1.5906e-03	6.975
6.9520e-02	9.1541e-03	1.282	9.8393e-04	2.088	2.9740e-04	2.373
4.1712e-02	3.2035e-03	2.055	2.7513e-04	2.495	3.6137e-05	4.126
2.5236e-02	1.2346e-03	1.897	5.8482e-05	3.081	4.2149e-06	4.276
1.4757e-02	4.5120e-04	1.876	1.2013e-05	2.950	*	*

Table 9.2: Static 2d bubble in 3d. Total L^2 error and EOC for $p = 1, 2, 3$.

9.2 Test Case: Traveling Wave Solution

In this section we perform the test with the traveling wave solution computed in Section 4.2 in two and three space dimension. Therefore, the one dimensional profile is trivially extended to two and three space dimensions by setting the remaining components of the velocity to zero. A 1d test can be found in Section 9.4. The isothermal Navier-Stokes-Korteweg system is in all tests equipped with a dimensionless van der Waals equation of state and the dimensionless reference temperature is fixed to $\theta_{ref} = 0.85$.
The system is discretized using the well balanced Discontinuous Galerkin scheme as described in Section 6.9 for polynomial degrees between zero and four. Time integration is done using second and third order implicit Runge-Kutta schemes, see Section 7.3. The time step is small enough (of order $O(h^2)$, where h denotes the mesh size) such that the order of the Runge-Kutta schemes is sufficient.
Note: For polynomial degree zero a Discontinuous Galerkin scheme reduces to a first order Finite Volume scheme.

Compressive wave in 2d

For the first test in this section we have chosen the compressive wave from Section 4.2. The profile is extended to two space dimensions by setting the second component of the velocity to zero and used as initial configuration. We impose boundary conditions known from the exact solution of the problem. The parameters for this profile are given by

$$\begin{aligned} \lambda &= 0.001, \\ \varepsilon &= 0.0056977, \quad \mu = \frac{3}{4}\varepsilon, \quad \nu = -\frac{1}{2}\varepsilon, \\ s &= -1.25273. \end{aligned}$$

Here λ denotes the capillarity parameter, μ and ν the coefficients of viscosity and s the speed of propagation of the profile. The approximate solutions are computed up to computational time $T = 0.02$. At this time the approximate solution is compared to the exact solution by computing the (total) L^2-error. All computations are done using identical machines (Pentium 4, 2.4GHz, one core per processor) and using only a single partition in order to compare the execution times. The computational domain is the channel $[-1,1] \times [-0.25, 0.25]$ and the mesh is a regular, globally refined *criss-cross* triangulation.

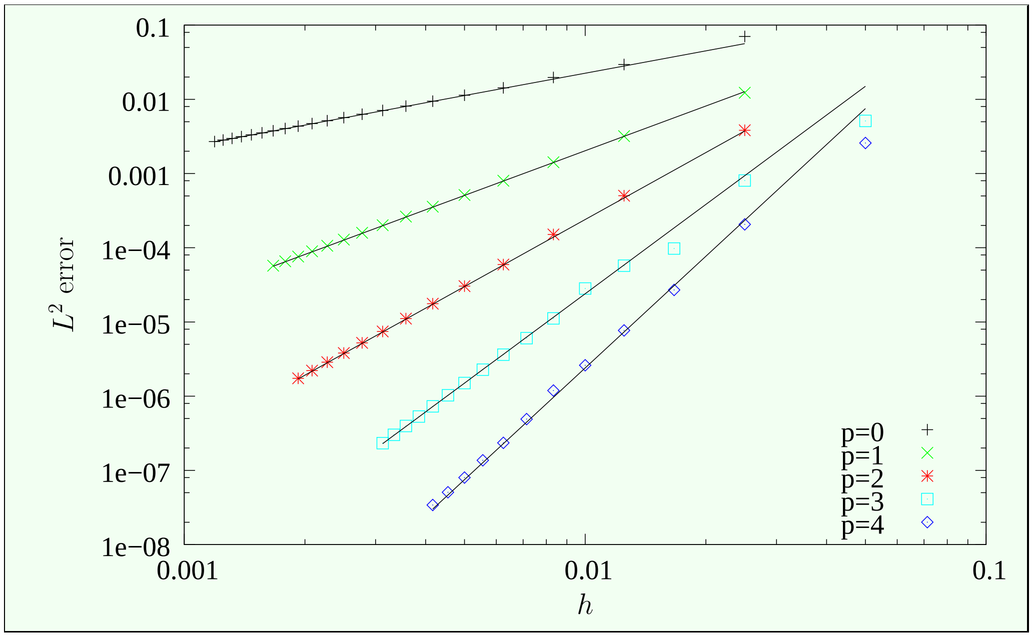

Figure 9.5: Compressive wave in $2d$. Mesh size versus L^2 error for polynomial degree $p = 0, 1, 2, 3, 4$.

Figure 9.5 shows that the approximate solutions generated by the Discontinuous Galerkin schemes with polynomial order $p = 0, 1, 2, 3$ converge to the exact solution with the expected order. In the figure the black lines indicate the expected order. Note that the expected order of the schemes is $p+1$. Table 9.3 illustrates the same. The scheme with $p = 4$ seems to have the same behavior but on closer inspection that the error is not of order five. This is due to the insufficient order of the Runge-Kutta scheme in this computation. We applied a second order Runge-Kutta scheme and the time step size is of order $O(h^2)$. This means that the resulting scheme cannot be better than order four.

	$p=1$		$p=2$		$p=3$	
h	L^2 error	EOC	L^2 error	EOC	L^2 error	EOC
2.5000e-02	1.2265e-02		3.8326e-03		5.1287e-03	
1.2500e-02	3.1980e-03	1.939	5.0307e-04	2.929	5.7384e-05	6.482
8.3333e-03	1.4234e-03	1.997	1.5131e-04	2.963	1.1166e-05	4.037
6.2500e-03	7.9961e-04	2.004	5.9397e-05	3.250	3.6240e-06	3.912
5.0000e-03	5.1395e-04	1.981	3.0369e-05	3.006	1.5009e-06	3.950
4.1667e-03	3.5756e-04	1.990	1.7597e-05	2.993	7.2917e-07	3.960
3.5714e-03	2.6292e-04	1.994	1.1096e-05	2.992	3.9572e-07	3.965
3.1250e-03	2.0137e-04	1.997	7.4416e-06	2.992	2.3285e-07	3.972

Table 9.3: Compressive wave in $2d$. L^2 error and EOC for $p = 1, 2, 3$.

From Figure 9.6 we can see that the construction of higher order schemes really leads to more efficient schemes (provided that the solution is sufficiently smooth). The figure shows the CPU time the computation needs versus the (total) L^2-error. By CPU time we mean the time the process has consumed, i.e., the user+system time on UNIX systems. We can see that the schemes with $p = 3$ and $p = 4$ are the most efficient schemes.

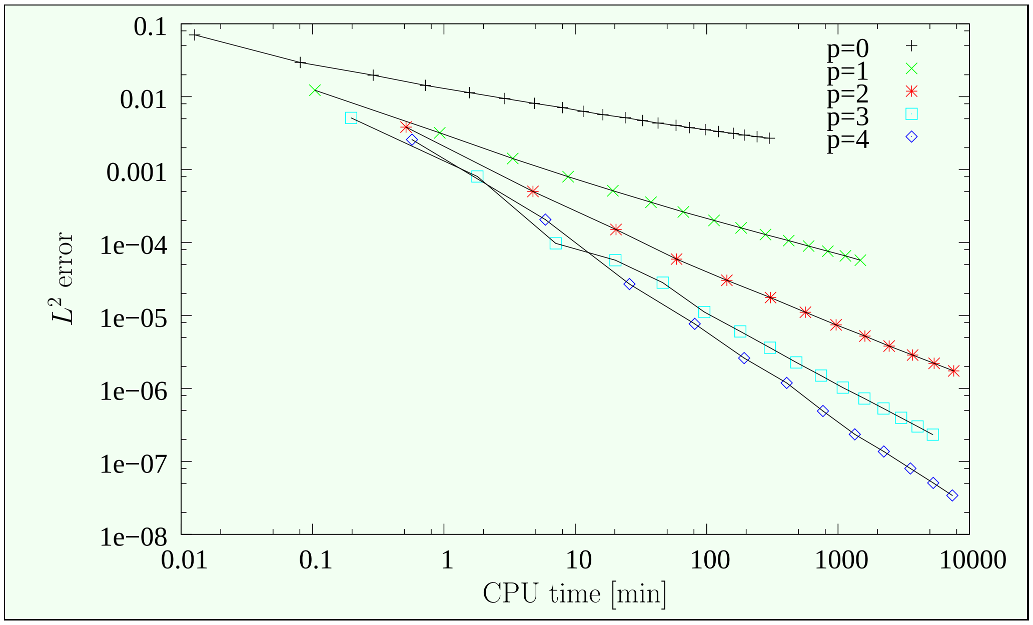

Figure 9.6: CPU time versus L^2 error for polynomial degree $p = 0, 1, 2, 3, 4$.

Undercompressive wave in 2d

As a second test in this section we repeat the test from the previous paragraph with an undercompressive wave constructed in Section 4.2 instead of a compressive wave. An undercompressive traveling wave solution is more typical for propagating phase boundaries since an interface usually propagates with subsonic speed. For the sequence of computations in this paragraph real unstructured (randomly perturbed) but uniformly fine meshes are used. One of these meshes is shown in Figure 9.7. We omit time and

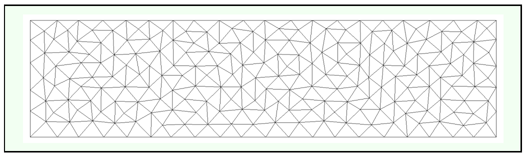

Figure 9.7: Randomly perturbed mesh in $2d$.

efficiency measurements in this test case such that the computations can be assigned to a different number of processors and different machines as it is necessary due to the different complexity when polynomial degree and mesh size vary.

The parameters for the undercompressive profile are different from the parameters for the compressive wave. They are given by

$$\begin{aligned} \lambda &= 0.001, \\ \varepsilon &= 0.0136644, \quad \mu = \frac{3}{4}\varepsilon, \quad \nu = -\frac{1}{2}\varepsilon, \\ s &= -0.32141. \end{aligned}$$

The computational end time for the computations is $T = 0.1$. At this time the total L^2-error between the numerical solutions and exact solutions are computed.

	p=1		p=2		p=3	
h	L^2 error	EOC	L^2 error	EOC	L^2 error	EOC
1.3711e-01	3.4725e-02		1.1543e-02		5.5499e-03	
8.8213e-02	1.1685e-02	2.470	3.6984e-03	2.581	5.6154e-04	5.194
6.0485e-02	7.2965e-03	1.248	1.2311e-03	2.915	2.2738e-04	2.396
4.2884e-02	3.8130e-03	1.887	4.6178e-04	2.851	6.0013e-05	3.873
3.1247e-02	2.0928e-03	1.895	1.8158e-04	2.948	1.6730e-05	4.035
2.1225e-02	9.1363e-04	2.143	5.5332e-05	3.073	3.4654e-06	4.071
1.4752e-02	4.5400e-04	1.922	1.9073e-05	2.928	8.5393e-07	3.850
1.0448e-02	2.0603e-04	2.290	6.2167e-06	3.249	2.0425e-07	4.146

Table 9.4: Undercompressive wave in 2d. Total L^2 error and EOC for $p = 1, 2, 3$.

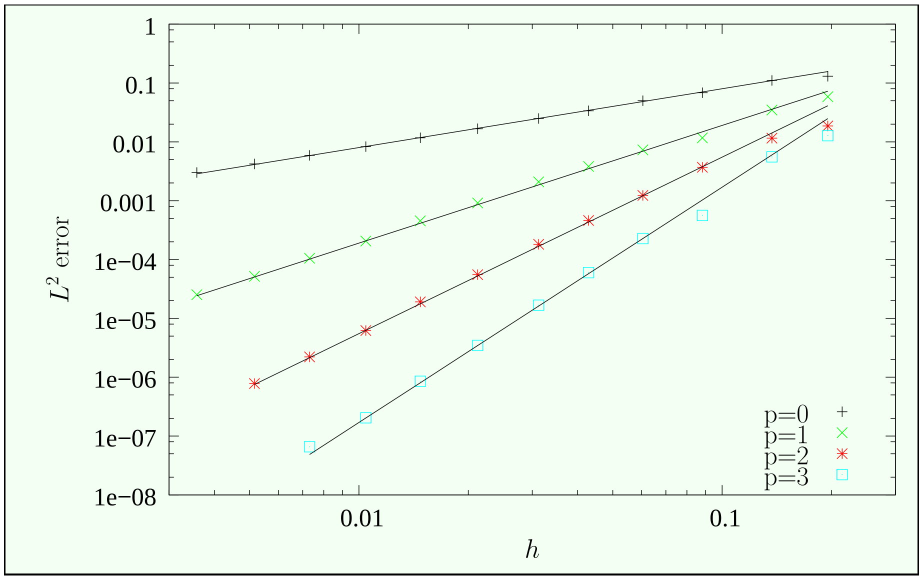

Figure 9.8: Undercompressive wave in $2d$, polynomial degree $p = 0, 1, 2, 3$.

The result of this convergence test is presented in Figure 9.8 and Table 9.4 for the Discontinuous Galerkin schemes with polynomial degree $p = 0, 1, 2, 3$ and a second order implicit Runge-Kutta method for time integration. Figure 9.9 shows the graph of such a density profile (color variation of the density distribution is also shown) and the corresponding velocity field for $p = 2$ at time $T = 0.1$.

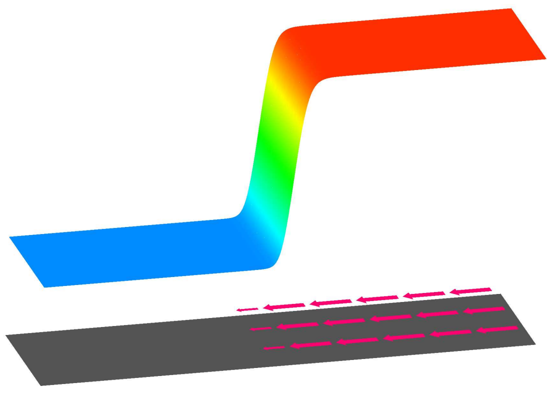

Figure 9.9: Undercompressive wave in 2d. Density distribution and velocity field.

Undercompressive wave in 3d

We extend the undercompressive traveling wave solution used in the previous paragraph trivially to three space dimensions by setting the additional momentum components to zero. The computational domain for the three dimensional test is the channel $\Omega = (-1,1)\times(-0.25,0.25)^2 \subset \mathbb{R}^3$ that is represented by regular, globally refined, tetrahedral meshes of different sizes. The computational end time is $T = 0.1$. At this time the total L^2-errors of the numerical solutions are computed. Again we omit time and efficiency measurements. The parameters are the same as in the 2d case with the undercompressive wave before.

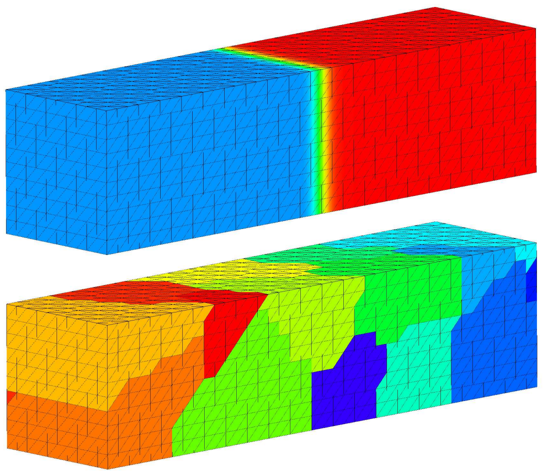

Figure 9.10: Undercompressive wave in 3d. Density distribution and the computational mesh (upper picture) and the associated partitions marked by different colors (lower picture).

The results of the tests for polynomial degree $p = 0,1,2,3$ are shown in Table 9.5 and Figure 9.11. The convergence with the expected order can clearly be seen for the second and third order schemes ($p = 1,2$). The test with the two finest meshes was too expensive for the fourth order ($p = 3$) scheme.

	p=1		p=2		p=3	
h	L^2 error	EOC	L^2 error	EOC	L^2 error	EOC
2.3208e-01	4.2703e-02		1.6035e-02		7.8913e-03	
1.5832e-01	2.3654e-02	1.544	8.3620e-03	1.702	4.8851e-03	1.254
1.1804e-01	1.6087e-02	1.313	3.8893e-03	2.607	1.4425e-03	4.155
8.0126e-02	9.7965e-03	1.280	1.4103e-03	2.618	5.9760e-04	2.274
5.5051e-02	4.8885e-03	1.852	7.0490e-04	1.848	1.1244e-04	4.451
3.8462e-02	2.3859e-03	2.000	2.3567e-04	3.055	3.1102e-05	3.584
2.7778e-02	1.2670e-03	1.945	9.1493e-05	2.907	*	*
2.0000e-02	6.6492e-04	1.963	3.3963e-05	3.017	*	*

Table 9.5: Undercompressive wave in 3d. Total L^2 error and EOC for $p = 1, 2, 3$.

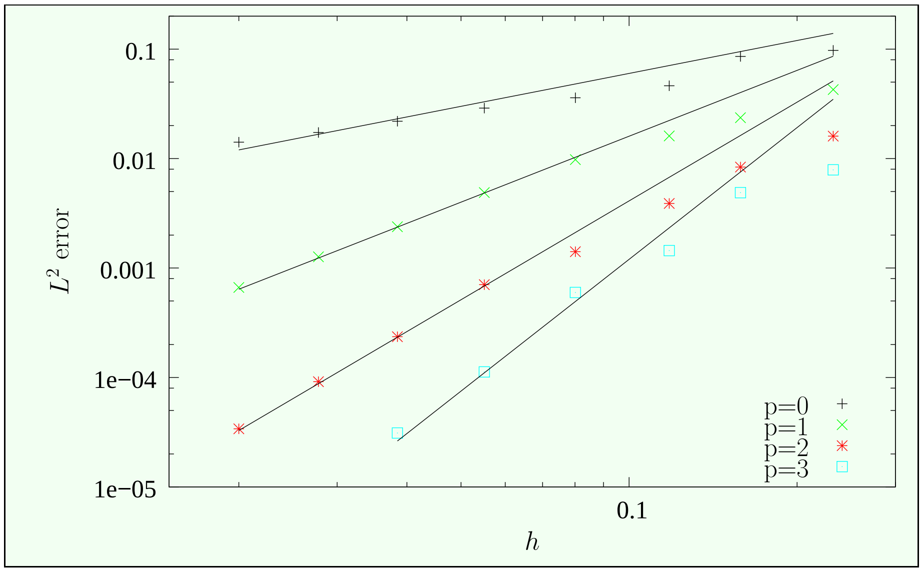

Figure 9.11: Undercompressive wave in $3d$, polynomial degree $p = 0, 1, 2, 3$.

9.3 Test Case: Towards Static Equilibrium

This is the test case proposed in Section 4.3 and the setting is similar to that in Chapter 5 but not the same.

The coefficient λ is chosen a hundred times smaller than in the tests in Chapter 5. This results in an ten times smaller interface and also the amount of surface tension is ten times smaller. This leads to slower dynamics because forces associated with surface tension are much weaker. Due to the small interface it is necessary to apply local mesh refinement using the interface indicator described in Section 8.3 with parameters

$$\begin{aligned} \eta_{low} &= 0.5, \\ \eta_{upp} &= 4.0 \cdot \eta_{low}, \\ m &= 8, \end{aligned}$$

where the parameter m controls the layer of fine cells around the interface and the load balancing is performed every 40th time step.

The isothermal NSK-system is discretized by the well balanced Discontinuous Galerkin scheme with polynomial ansatz functions of degree two and three. For polynomial degree one the above given values for the interface indicator do not provide a fine enough mesh for the complete resolution of the interface. This results in an unstable behavior of the approximate solution. The parameters for this test are given by

$$\begin{aligned}\theta_{ref} &= 0.85, \quad \text{dimensionless vdW-equation of state,}\\ \lambda &= 1.0 \cdot 10^{-5},\\ \varepsilon &= 1.366 \cdot 10^{-3}, \quad \mu = \frac{3}{4}\varepsilon, \quad \nu = -\frac{1}{2}\varepsilon.\end{aligned}$$

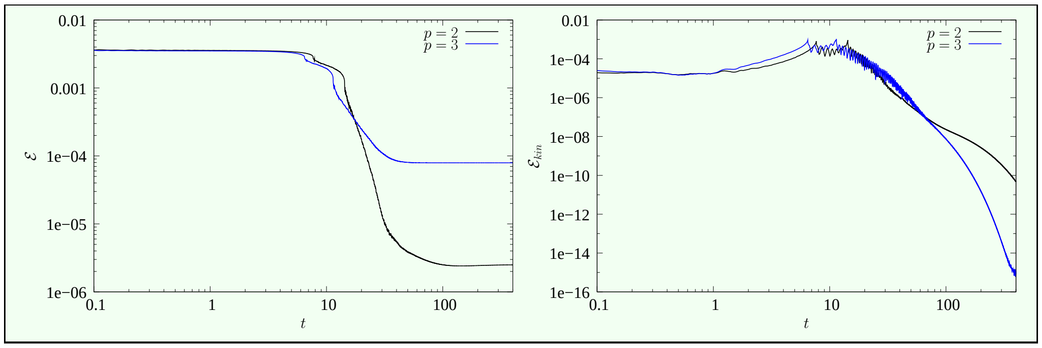

Figure 9.12: Test Case: Towards Static Equilibrium. Total and kinetic energy for polynomial degree $p = 2$ and $p = 3$.

Figure 9.12 shows the behavior of the total and kinetic energy as functions of time for the two well balanced Discontinuous Galerkin schemes with polynomial degree two and three. A constant has been added to the total energy such that it can be displayed on a logarithmic scale. The total energy $\mathcal{E}$ is an *almost* decreasing function in time with small oscillations. The scheme itself is not designed such that the total energy has this behavior. This is a side effect observed in the numerical experiments with the basic first order scheme in Chapter 5. The oscillations we can observe in the graph of the energy are mainly caused by the L^2 projection in the refinement and coarsening process. We have observed that these oscillations vanish when local mesh adaption is not applied. However, this is not possible in the case of a very small interface. Maybe a more convenient data projection should be chosen to maintain the energy decay. The right part of the figure shows that the kinetic energy converges completely (up to roundoff error) to zero as time tends to infinity. The schemes are designed to preserve static equilibrium data on the discrete level but it is not clear that if we add a small perturbation to a *stable* static equilibrium configuration that as time tends to infinity the approximate solution converges to some static equilibrium state again. This can clearly be seen from the behavior of the kinetic energy.

Figure 9.13 shows a sequence of six time steps in the evolution process from the initial data at $t = 0.0$ to the (nearly) static equilibrium configuration at time $t = 40.0$. There as still some movement at this computational time but the topological changes are completed and the large bubble in the center of the domain has an almost spherical shape. The figure shows the distribution of the density and the velocity at computational times $t = 0.0,\ 7.0,\ 7.85,\ 13.07,\ 14.6,\ 40.0$ (from upper left to lower right) for the third order Discontinuous Galerkin scheme ($p = 2$). Below the density-velocity picture the corresponding adaptively refined mesh with the distribution over the eight processors used for this computation is shown. Each of the eight colors represents one partition.

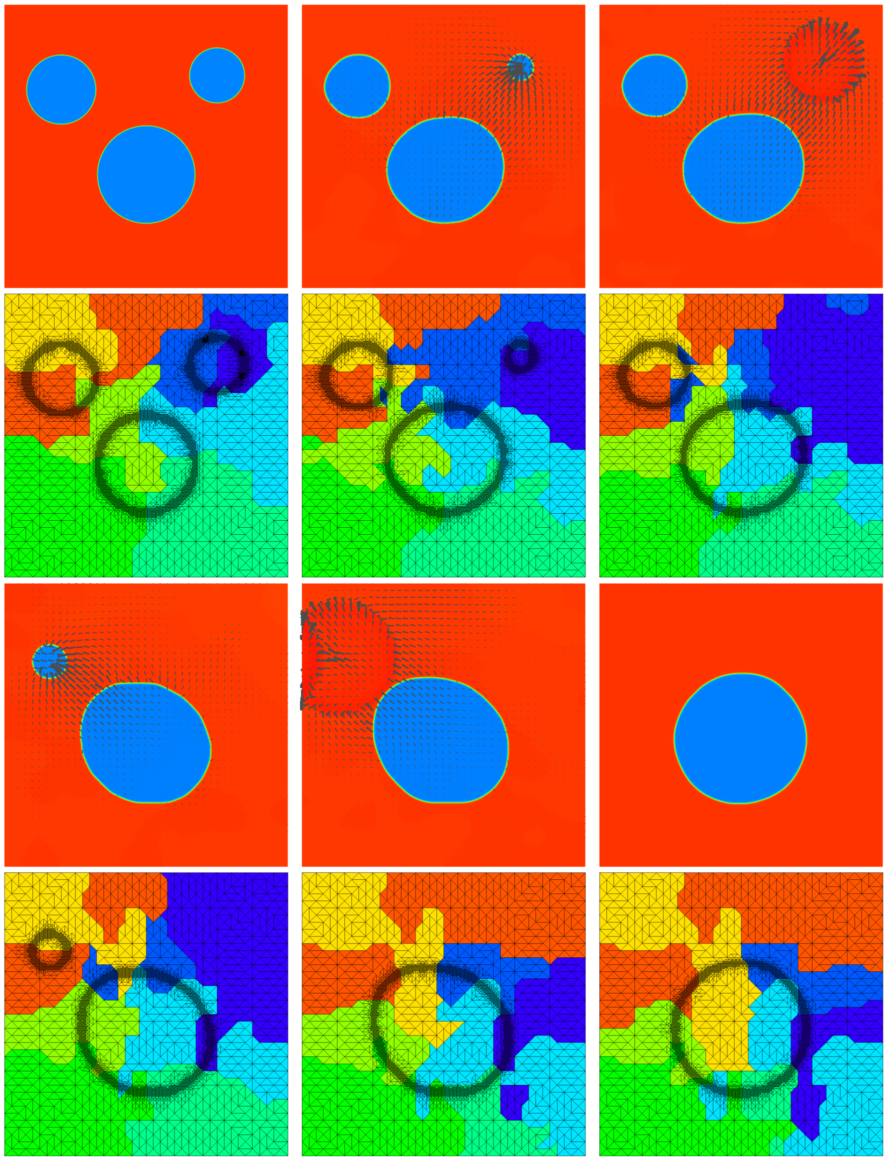

Figure 9.13: Test Case: Towards Static Equilibrium. Density, velocity and the adaptively refined meshes with distribution of the partitions. $t = 0.0,\ 7.0,\ 7.85,\ 13.07,\ 14.6,\ 40.0$ from upper left to lower right.

9.4 The Need for Artificial Viscosity

This test shows that the additional artificial viscosity, see for example Section 5.2 especially equation (5.5), in the discretization of the Navier-Stokes-Korteweg equations is really necessary to stabilize the approximate solution (at least for the higher order DG discretization). The test case with the compressive traveling wave solution from Section 9.2 is applied to the one dimensional isothermal NSK system discretized by the well balanced Discontinuous Galerkin schemes. The parameters are chosen as in Section 9.2 with the difference that in one sequence of tests the artificial viscosity parameter α_1, see (5.5), is set to zero. The approximate solutions are computed up to time $T = 0.1$ on uniform 1d grids.

Table 9.6 shows the convergence behavior of the two tests with the first order schemes, one with and the other without artificial viscosity. The errors are not in the asymptotic regime for the used mesh sizes but it can be seen that artificial viscosity is not necessary to stabilize the generated approximations for the first order schemes (at least not in this test case). The result is different in the case of higher order schemes as can be seen below.

	$\alpha_1 > 0$		$\alpha_1 = 0$	
h	L^1 error	EOC	L^1 error	EOC
1.0000e-01	2.6682e-01		3.0344e-01	
5.8824e-02	1.5707e-01	0.999	1.9467e-01	0.837
3.5088e-02	5.9956e-02	1.864	9.0428e-02	1.484
2.0619e-02	1.9982e-02	2.067	1.9096e-02	2.925
1.2121e-02	9.9869e-03	1.306	5.4978e-03	2.344
7.1685e-03	7.0067e-03	0.675	2.7571e-03	1.314
4.2373e-03	4.9088e-03	0.677	1.5106e-03	1.144

Table 9.6: Traveling wave in 1d, 1st order scheme. Total L^1 error and EOC for the scheme with artificial viscosity (left) and without (right).

The results of the tests with the corresponding second order Discontinuous Galerkin schemes can be found in Table 9.7. Almost second order convergence can be observed for the scheme with artificial viscosity. The scheme without artificial viscosity does not converge to the exact solution and Figure 9.14 clearly shows the unstable behavior of the approximate solution. The scheme produces oscillations in the vicinity of the phase boundary. The scheme that includes artificial viscosity (not shown) does not show this behavior, it converges to the exact solution.

As a result we find that except for the first order scheme artificial viscosity is necessary for the higher order schemes to produce sequences of approximate solutions that converge to the exact solutions. Of course, instead of the explicit introduction of artificial viscosity another stabilization technique could also be possible.

	$\alpha_1 > 0$		$\alpha_1 = 0$	
h	L^1 error	EOC	L^1 error	EOC
1.0000e-01	1.0310e-01		1.8342e-01	
7.1429e-02	4.9296e-02	2.193	1.5012e-01	0.595
5.1282e-02	3.9810e-02	0.645	1.1758e-01	0.737
3.7037e-02	1.9147e-02	2.249	6.0010e-02	2.067
2.6667e-02	9.1751e-03	2.239	3.1154e-02	1.996
1.9231e-02	4.4738e-03	2.197	2.5753e-02	0.582
1.3889e-02	2.0969e-03	2.329	2.3741e-02	0.250
1.0000e-02	1.2653e-03	1.538	3.0897e-02	-0.802

Table 9.7: Traveling wave in 1d, 2nd order DG-scheme. Total L^1 error and EOC for the scheme with artificial viscosity (left) and without (right).

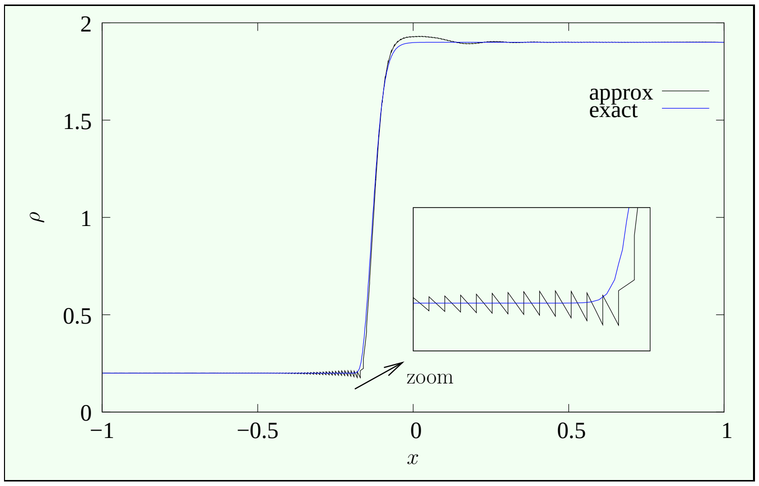

Figure 9.14: Traveling wave in 1d, 2nd order DG-scheme without artificial viscosity. Density profiles for exact and approximate solutions at time $T = 0.1$.

9.5 Different Contact Angles

In this test we impose different contact angles for the interface at a solid wall by modification of the angle φ in boundary condition (2.54). The computational domain for this test is the square/cube $[-1, 1]^n$, $n = 2, 3$ partitioned by an adaptively refined triangular/tetrahedral mesh. Initially, a bubble with 90 degree contact angle is attached to the *bottom* wall with zero velocity field, Figure 9.15 shows the initial data in two space dimensions. This means that (except for a contact angle of 90 degrees) the initial configuration is not consistent with the prescribed boundary conditions. However, this seems not to lead to instabilities in the approximate solution and as time tends to infinity the contact angle agrees with the imposed conditions.

Again we choose the dimensionless isothermal van der Waals equation of state with a reference temperature $\theta_{ref} = 0.85$. For this test we choose a much smaller interface as in the tests before. The parameters are different in two and three space dimensions.

Therefore the used capillarity and viscosity parameters are listed in the corresponding 2d and 3d paragraph below. The rest of the setting is the same regardless of the space dimension. The scheme is a 3rd order Discontinuous Galerkin scheme (polynomial degree 2) with 2nd order implicit time integration. We compute the approximate solution up to computational time $T = 50.0$. The solution does not change essentially from this time and the velocity field is close to zero.

The complete resolution of the small interface is only possible by using adaptively refined meshes. For the tracking of the interface we use the interface indicator discussed in Section 8.3, (8.10) with parameters

$$\begin{aligned} \eta_{low} &= 0.8, \\ \eta_{upp} &= 4.0 \cdot \eta_{low}, \\ m &= 8, \end{aligned}$$

and the load balancing is performed every 40th time step. This could be done less frequently because the solution does not change rapidly in this test case.

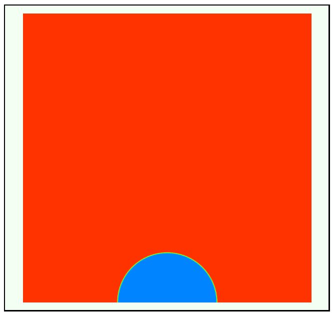

Figure 9.15: Same initial data for different contact angles.

Different contact angles in 2d

The capillarity and viscosity parameters in the two dimensional test case are given by

$$\begin{aligned} \lambda &= 1.0 \cdot 10^{-5}, \\ \varepsilon &= 1.366 \cdot 10^{-3}, \quad \mu = \frac{3}{4}\varepsilon, \quad \nu = -\frac{1}{2}\varepsilon. \end{aligned}$$

The two dimensional test is performed using two processors. The density distribution, the corresponding meshes together with the distribution of the mesh cells over the partitions is shown in Figure 9.16 for the three contact angles $\varphi = 0.25\pi,\ 0.5\pi,\ 0.75\pi$ at time $T = 50.0$.

A contact angle of 135 degree in 3d

For the three dimensional test the capillarity and viscosity parameters are chosen slightly larger for two reasons: faster computation and better display of the results. The pa-

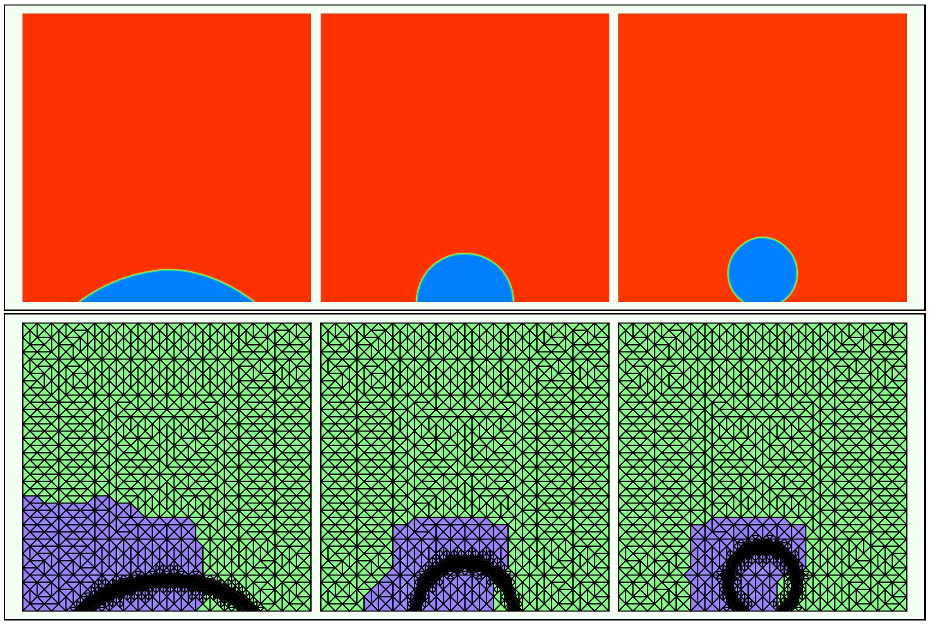
Figure 9.16: Different contact angles in $2d$. $\varphi = 0.25\pi,\ 0.5\pi,\ 0.75\pi$ from left to right.

rameters are given by

$$\begin{aligned}\lambda &= 2.5 \cdot 10^{-4},\\ \varepsilon &= 6.830 \cdot 10^{-3}, \quad \mu = \frac{3}{4}\varepsilon, \quad \nu = -\frac{1}{2}\varepsilon.\end{aligned}$$

Only one computation is performed with an adjusted contact angle of $\varphi = 0.75\pi$ using 16 processors in parallel. The result at time $T = 50.0$ is shown in Figure 9.17. This figure shows the distribution of the density on two clipping planes with $z = 0$ and $y = 0$, where x, y, z denote the spatial coordinates. Additionally a levelset of a bubble is shown.

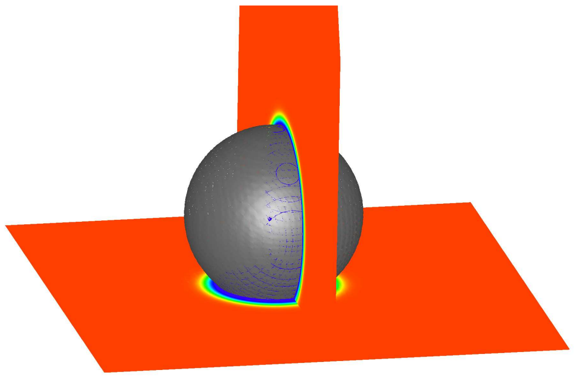
Figure 9.17: Contact angle of 135 degree in $3d$.

9.6 Implicit versus Explicit Time Stepping

We test the efficiency of implicit time stepping versus explicit time integration in this section. Note that there is no formula for the time step size restriction available that is necessary for explicit time stepping to guarantee the stability of the method. This is the main reason for using implicit Runge-Kutta methods, efficiency is the second one. Since there is no formula for controlling the time step size in the explicit case we have to figure out manually (by successively lowering) which time step size gives a stable scheme. It is guaranteed that for a *working* time step size a five percent larger time step size shows an unstable behavior.

For this test case a traveling wave solution or a static equilibrium seems to be the appropriate choice since these kinds of solutions have the same shape for all times t. Thus, the most efficient time step size for implicit schemes and the maximal possible time step for explicit schemes remain almost the same for all times t.

We have chosen both test cases in two space dimensions using different mesh sizes and a 3rd and 4th order well balanced Discontinuous Galerkin discretization ($p = 2, 3$) of the isothermal Navier-Stokes-Korteweg equations in space. The configuration is almost the same as in Sections 9.1 and 9.2. In both cases second order Runge-Kutta schemes are applied for time integration. In the explicit case the TVD2 scheme (also known as Heun scheme) is used and in the implicit case the Crank-Nicholson scheme in combination with the GMRES(15) linear solver is applied. The computations were run using a sequence of successively globally refined meshes. In order to compare the execution times of the computations without distorting the results by the overhead of parallel communication all tests were run on a single processor (AMD Athlon64, 1.8GHz).

Traveling wave solution

For the first test with the traveling wave solution as initial data in two space dimensions. We have chosen exactly the same undercompressive wave from Section 9.2. Since everything is in movement in the solution this should be the harder test for the implicit scheme. The rest of the configuration is exactly the same as in Section 9.2. The computational end time where the approximate and exact solutions are compared to each other is $T = 0.01$. The coarsest computational mesh is a relatively rough but regular triangulation of the two dimensional channel. This mesh is subsequently refined for further computations which gives meshes of the same quality.

Table 9.8 shows the results of this sequence of computations for the 3rd order DG scheme (upper part of the table) and the 4th order DG scheme (lower part). For the implicit and explicit case the total L^2-errors, the time the computation needs to finish and the time step sizes are shown. The time step size for the explicit schemes are fixed whereas the size of the time step in the implicit schemes varies and the given size in the table can be considered as a *mean* value. The errors of the approximate solutions generated by the implicit and explicit schemes are comparable at a given mesh size. We see that the implicit schemes are faster on the finer meshes and for higher polynomial degrees

$p = 2$	implicit			explicit		
h	L^2 error	Time [s]	Δt	L^2 error	Time [s]	Δt
5.77e-2	2.4377e-3	4.2	9.0e-4	2.4350e-3	1.9	2.0e-4
2.89e-2	2.8858e-4	44.5	2.0e-4	2.8848e-4	31.8	2.5e-5
1.44e-2	3.3837e-5	635.8	3.5e-5	3.3836e-5	684.2	3.1e-6
7.22e-3	4.1567e-6	10794.9	6.8e-6	4.1558e-6	18236.6	3.6e-7

$p = 3$	implicit			explicit		
h	L^2 error	Time [s]	Δt	L^2 error	Time [s]	Δt
5.77e-2	5.2560e-4	14.5	3.5e-4	5.2486e-4	10.0	5.0e-5
2.89e-2	3.4592e-5	201.3	7.0e-5	3.4591e-5	218.3	6.0e-6
1.44e-2	2.3513e-6	3083.1	1.3e-5	2.3464e-6	5122.0	7.8e-7
7.22e-3	1.7510e-7	58003.0	2.5e-6	1.5714e-7	148204.7	9.0e-8

Table 9.8: Implicit versus explicit time stepping. Traveling wave in $2d$, 3rd order scheme (upper table) and fourth order scheme (lower table).

(nearly up to three times on the finest mesh for the 4th order scheme). On the coarsest mesh the explicit schemes are faster but here the interface is not completely resolved such that this mesh is not usable in practical applications. Note that a necessary time step size control, which is also time consuming, is not included in the explicit schemes since there is no formula for time step size restriction available.

Static equilibrium solution

The second test with the static bubble in two space dimensions uses exactly the same configuration as in Section 9.1. Here only the third order Discontinuous Galerkin scheme is applied. The computations use successively refined meshes and the errors are computed at the computational end time $T = 1.0$.

$p = 2$	implicit			explicit		
h	L^2 error	Time [s]	Δt	L^2 error	Time [s]	Δt
5.32e-2	2.3135e-3	107.9	3.0e-3	2.3134e-3	199.6	1.5e-4
2.66e-2	2.0993e-4	2118.1	6.0e-4	2.0993e-4	6511.2	1.8e-5
1.33e-2	2.4471e-5	47580.3	1.3e-4	2.4471e-5	224191.2	2.1e-6

Table 9.9: Implicit versus explicit time stepping. Static equilibrium in $2d$, 3rd order scheme.

Table 9.9 shows the result of these computations. Again we see that the errors produced by the implicit and explicit scheme at a given mesh size are comparable and that the implicit scheme is faster on the finer meshes (by factor 4.8 for the finest mesh). It can be observed that the implicit scheme becomes much faster at the end of the computation because the discrete initial data provided by L^2-projection is not a discrete equilibrium but a discrete equilibrium is approached during the computation.

9.7 Adaptive Efficiency

This test will show the gain in efficiency that local mesh adaption can give. In order to test only the adaptive efficiency the sequence of tests should be run on a single processor only (AMD Athlon64, 1.8GHz). Adaptive mesh refinement and coarsening is employed to resolve small diffuse interfaces and to reduce the error between the approximate and exact solution. We compare approximate solutions generated on globally refined meshes and on adaptively refined and coarsened meshes. For a fair test we need a solution with rapid changes.

For this test we have chosen an undercompressive traveling wave solution similar to Section 9.2 and trivially extended to two space dimensions. The chosen profile is a little bit sharper than that in Section 9.2 such that some levels of refinement are necessary to resolve the interface completely. The parameters for this wave are

$$\begin{aligned}\lambda &= 0.0001,\\ \varepsilon &= 0.0025773, \quad \mu = \frac{3}{4}\varepsilon, \quad \nu = -\frac{1}{2}\varepsilon,\\ s &= -0.65691.\end{aligned}$$

The equation of state is again the dimensionless van der Waals equation of state with reference temperature $\theta_{ref} = 0.85$. The domain is the channel $\Omega = [-1,1] \times [-0.2, 0.2]$ and the approximate solutions are computed up to time $T = 1.0$. At this time the total L^2-errors are compared. We use the rough macro mesh shown in Figure 9.18.

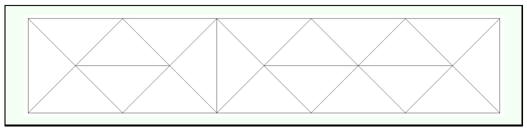

Figure 9.18: Macro grid of the $2d$ channel.

For the computations that use a globally refined mesh this macro mesh is refined three or four times before the initial data is L^2-projected to the corresponding Finite Element space. The corresponding tests are denoted by nonadapt(3) and nonadapt(4) respectively.

The adaptive computations use the interface indicator discussed in Section 8.3, (8.10) with parameters

$$\begin{aligned}\eta_{low} &= 0.5,\\ \eta_{upp} &= 4.0 \cdot \eta_{low},\\ m = 3 \quad &\text{or} \quad m = 8,\end{aligned}$$

where m controls the size of the layer of fine cells around the interface. A smaller value leads to a faster computation and a larger value gives a smaller error and a more robust scheme. These tests are denoted by adapt(3) and adapt(8). The initial mesh is chosen such that the interface indicator does not mark any cells for refinement and coarsening applied to the L^2-projected values.

In both cases, adaptive and non adaptive, The Navier-Stokes-Korteweg equations are discretized by the well balanced Discontinuous Galerkin schemes of order three and four. In both cases the implicit second order Crank-Nicholson scheme is applied for the time integration. As linear solver the GMRES solver with the same Krylov space dimension 15 is used in all cases.

Test	L^2-error	Time [s]	number of cells
adapt(3)	1.84569e-3	814.9	377
adapt(8)	1.86002e-3	1300.3	713
nonadapt(3)	1.72694e-2	696.7	1280
nonadapt(4)	1.88873e-3	9522.1	5120

Test	L^2-error	Time [s]	number of cells
adapt(3)	4.25129e-4	3178.7	377
adapt(8)	4.17148e-4	5419.8	713
nonadapt(3)	6.52487e-3	2717.7	1280
nonadapt(4)	4.28355e-4	41551.9	5120

Table 9.10: Comparison of adaptive and non adaptive third order (upper table) and fourth order (lower table) DG schemes.

The results of the computations can be found in table 9.10. The tests with the globally refined meshes use a fixed number of mesh cells. These numbers are shown in the tables. In the adaptive cases the number of cells can vary (but not significantly) during the computation and the number shown in the tables is taken at the end of the computation. The nonadapt(4) tests have the same resolution of the interface as the adaptive tests, the resolution of the nonadapt(3) tests is not that fine which results in a larger error. For the 3rd order and 4th order DG schemes the tests adapt(3) are 11-13 times faster than the corresponding nonadapt(4) tests at a comparable error. This is a significant speedup of the adaptive algorithm and this factor becomes even more significant the smaller the interface is. The result of these computations is that very small interfaces cannot be resolved by uniform fine meshes in practical applications. Thus, adaptive mesh refinement is mandatory.

Figure 9.19 shows the density distribution and the adaptively refined mesh (velocity field is omitted) of the adapt(3) test using the 3rd order Discontinuous Galerkin Discretization.

9.8 Parallel Efficiency

In this section we test the parallel performance of the Discontinuous Galerkin code applied to the isothermal Navier-Stokes-Korteweg system. We have chosen two different three dimensional settings for this test, both small enough to fit on a single processor. Of course, larger problems show a much better scaling but the intention of this section is to show also the limits of parallelization. Nevertheless, even for these small problems

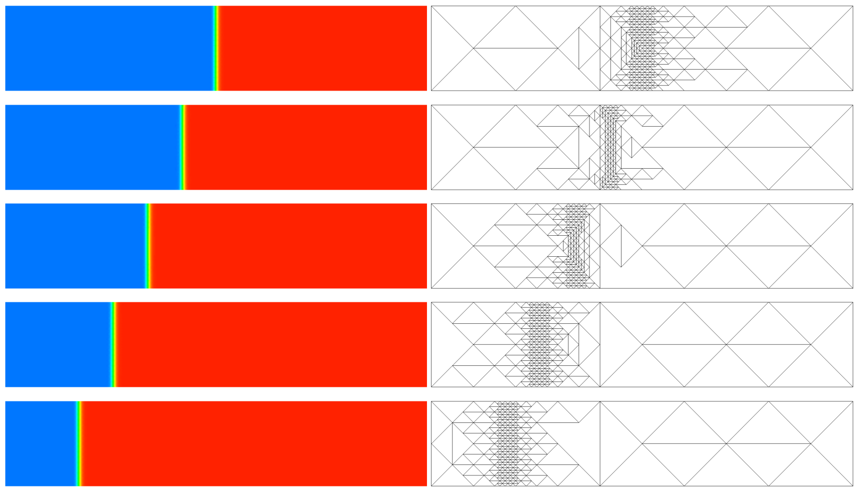

Figure 9.19: Density distribution and the adaptively refined mesh for the 3rd order test adapt(3) at times $t = 0.0, 0.25, 0.5, 0.75, 1.0$ from first to last picture.

the Discontinuous Galerkin discretization is very well suited for parallel computation. Within the class of DG methods parallel efficiency increases when the order of the method is increased since the local workload becomes higher and communication can be bundled. All computations in this section are done on the XC4000 Cluster at the computing center of the university of Karlsruhe. This cluster consists of 2.6 GHz Dual Core Opteron processors (AMD64-NUMA architecture), 2 Dual Core CPUs per node and InfiniBand network interconnects.

For both tests we have chosen the parameters of the $3d$ test case in Section 9.5. As computational domain we consider the domain $\Omega = [-1, 1]^3$. The domain is partitioned into a macro mesh of $6.000 = 10 \times 10 \times 10 \times 6$ macro cells. The first test uses a globally refined mesh starting from this macro mesh. Provided that the initial partitions are equally well distributed over the available processors, this is mainly a test for the parallel performance of the third order Discontinuous Galerkin code in combination with a second order implicit Runge-Kutta time discretization. The second test uses an adaptively refined mesh. Besides the Discontinuous Galerkin code the quality of the load balancer is also tested in this example.

Globally refined mesh

The setting is the same as for the $3d$ contact angle example from Section 9.5 except that for the globally refined mesh the bubble has no contact to the wall since the used mesh is not fine enough in this case. The macro mesh is twice globally refined which results in 384.000 cells.

The computational end time is $T = 0.2$. Provided that the initial mesh is equally well distributed a redistribution is not necessary in this test case. Nevertheless, load balancing is done every 40th time step for completeness of the algorithm. This results

np	Time [s]	Speedup to $np/2$	Speedup to $np = 1$
1	75058.3		
2	37520.3	2.00	2.00
4	19965.9	1.87	3.74
8	10134.2	1.97	7.37
16	5118.92	1.98	14.59
32	2628.77	1.95	28.45
64	1367.68	1.92	54.62
128	702.316	1.95	106.51
256	362.371	1.94	206.62
512	190.991	1.90	392.58
1024	133.534	1.43	561.39

Table 9.11: Speedup for the globally refined mesh.

in a slightly modified mesh distribution at the beginning of the computation and does not further alter the mesh after a few calls of the load balancer since the weights of the macro cells do not change.

For the sequence of computations using the globally refined mesh the number of processors np varied between $1, 2, 4, \dots, 1024$. We have measured the real time in seconds the computation needed to finish. Table 9.11 shows the result of these timings and the speedup compared to the previous computation with half the number of processors and compared to the first computation that was run on one processor only. The results show that even for this small example a parallelization using 512 and maybe 1024 processors is appropriate. With a factor of 1.43 the speedup from 512 to 1024 processors cannot be as good as previous speedups because only 6.000 macro cells have to be distributed over 1024 processors and even when the load balancer provides the optimal distribution of the partitions (which is in general not possible) the computational work cannot be equally distributed since the problem is too small.

Adaptive, load balanced mesh

For this test the setting is identical to $3d$ contact angle example from Section 9.5. This includes the setting of the interface indicator. Since the initial data does not satisfy the enforced contact angle there is a movement of the interface which requires an adapted and repartitioned mesh every few time steps. Therefore load balancing is done every 40th time step. The computational end time $T = 0.1$ which is of course not the computational end time from Section 9.5 since the motivation for this test in this section is not to generate a bubble with the correct contact angle. The same macro mesh as in the previous example with 6.000 macro cells is used. The locally refined mesh has approximately 30.000 cells. Therefore the problem is much smaller than the previous one and we cannot expect the same parallel scaling as before.

Table 9.12 shows the results of the sequence of computations using $np = 1, 2, 4, \dots, 256$

number of processors. For this small problem it is not appropriate to use more than 32 or 64 processors. In the case of 64 processors some of the processors were not assigned to a partition and therefore they did no work. Using more processors worsens the situation a lot.

np	Time [s]	Speedup to $np/2$	Speedup to $np = 1$
1	25058.1		
2	13083.0	1.92	1.92
4	6951.35	1.88	3.60
8	3555.19	1.96	7.07
16	1872.53	1.89	13.37
32	1022.23	1.83	24.47
64	661.953	1.54	37.68
128	420.659	1.57	59.16
256	286.044	1.47	86.97

Table 9.12: Speedup for the locally refined, load balanced mesh.

As a result we can conclude that parallelization of Discontinuous Galerkin code can be very effective and efficient even for small problems where memory consumption is not the bottleneck. This also holds for the two dimensional case and even for one dimensional problems parallelization can be a gain in efficiency. Here efficiency means runtime is reduced.

9.9 Bubble Ensembles

Instead of a single or a few bubbles we consider the dynamics of a whole bubble ensemble in this section. The configuration of the isothermal Navier-Stokes-Korteweg equations and the interface indicator is the same as in Section 9.3. The initial configuration is a randomly distributed ensemble of 200 bubbles in the domain $\Omega = [-1, 1]^2 \subset \mathbb{R}^2$. The radiuses of the bubbles vary (randomly) between 0.02 and 0.06.

The sequence of pictures in Figure 9.20 shows the distribution of vapor bubbles at times $t = 0.0, 0.2, 1.0, 4.0, 15.0, 100.0$. The bubbles merge and grow until there is only one large bubble left. The final bubble stays in a static equilibrium configuration and the interface has a 90 degree contact angle with the container wall. The velocity field is not shown in the sequence of pictures.

Figure 9.21 shows the number of vapor bubbles as a function of time. The left part of the figure shows the decay of bubbles in the time interval $(0, 50)$ where the number of objects decreases from 200 bubbles at $t = 0$ to only one bubble at $t = 50$. The right part of the figure is a zoom of the time interval $(0, 5)$. The number of bubbles decreases rapidly at the beginning of the computation. An exponential decay from 200 to 30 objects during the time interval $(0, 1)$ can clearly be seen.

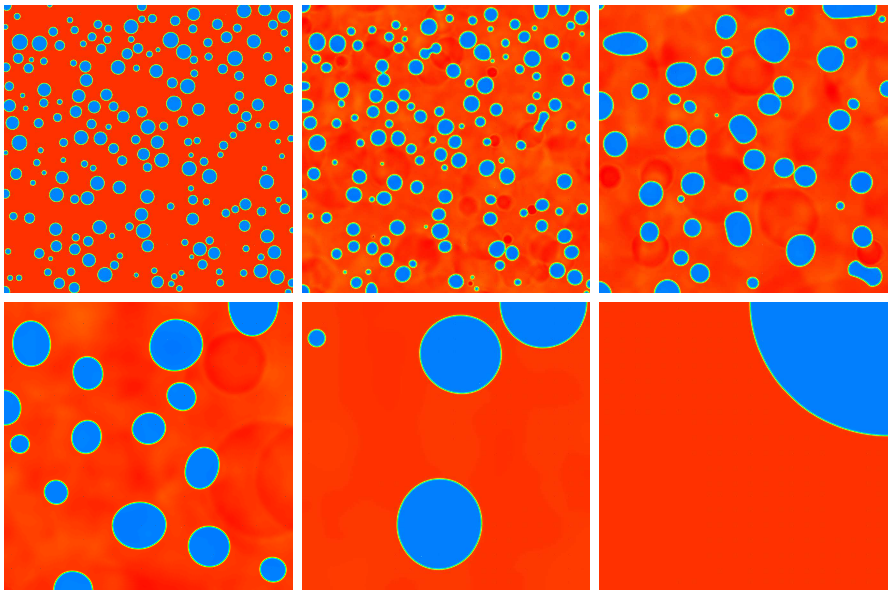

Figure 9.20: Bubble Ensemble, initially randomly distributed. Density distribution at times $t = 0.0, 0.2, 1.0, 4.0, 15.0, 100.0$ from top left to lower right picture.

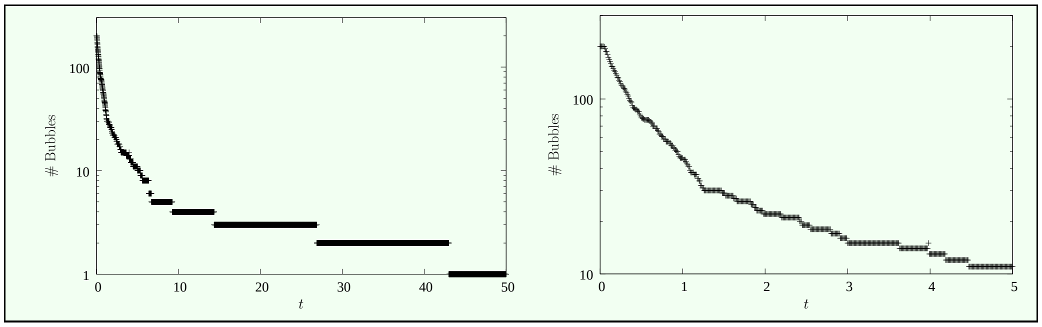

Figure 9.21: Bubble Ensemble. Time versus number of bubbles.

The number of bubbles are counted by counting the number of contiguous objects in the vapor phase.

9.10 The Temperature Dependent Model

The static equilibrium solution of the isothermal Navier-Stokes-Korteweg model constructed in Section 4.1 is also a solution of the temperature dependent NSK model (6.31) with boundary condition (2.52) when the wall temperature θ_b is set to the constant reference temperature. Note that this is not the case for the traveling wave solutions. In this section we use the static equilibrium solutions as initial data to perform convergence tests with the Discontinuous Galerkin discretization of the temperature dependent version of the two dimensional NSK system discussed in Section 6.9.4.

The setting in this section is the same as in Section 9.1. The temperature in the initial data and at the boundary are set to the reference temperature $\theta_{ref} = 0.85$. The remaining parameters in the temperature dependent model are the heat capacity at constant volume c and the heat conduction coefficient κ. Note that the static equilibrium configuration is a solution of the system independent of the choice of these parameters. In this test we choose the parameters

$$\begin{aligned} c &= 6.6, \\ \kappa &= 0.01366. \end{aligned}$$

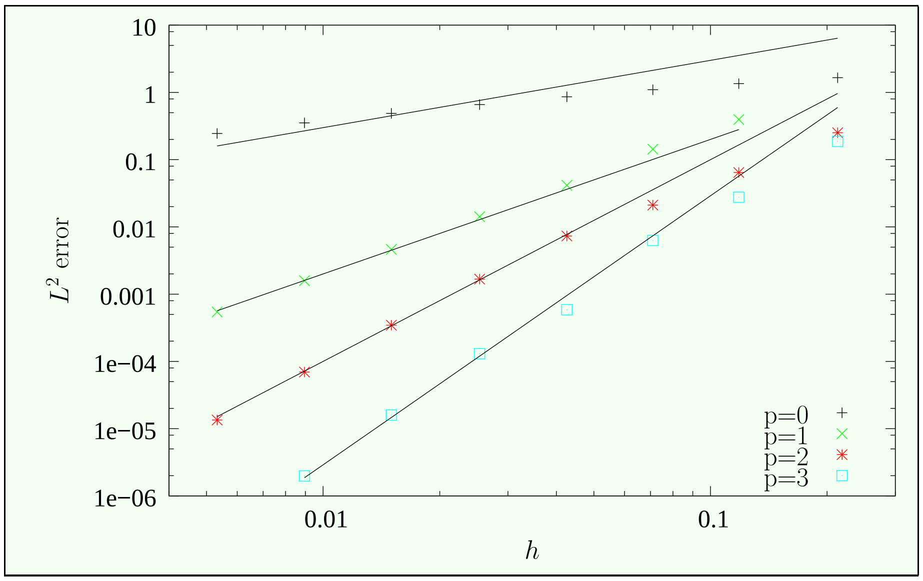

Figure 9.22: Static bubble, temperature dependent NSK model in 2d. Mesh size versus L^2 error for $p = 0, 1, 2, 3$.

	p=1		p=2		p=3	
h	L^2 error	EOC	L^2 error	EOC	L^2 error	EOC
2.1298e-01	*		2.5147e-01		1.8718e-01	
1.1832e-01	3.9576e-01	*	6.4436e-02	2.317	2.7605e-02	3.256
7.0992e-02	1.4322e-01	1.990	2.1098e-02	2.186	6.2847e-03	2.897
4.2595e-02	4.1457e-02	2.427	7.3170e-03	2.073	5.8590e-04	4.645
2.5354e-02	1.4174e-02	2.069	1.6734e-03	2.844	1.3003e-04	2.902
1.4998e-02	4.6256e-03	2.133	3.4556e-04	3.005	1.5942e-05	3.998
8.9486e-03	1.5875e-03	2.071	6.9507e-05	3.105	1.9765e-06	4.042
5.3244e-03	5.4365e-04	2.064	1.3450e-05	3.163	*	*

Table 9.13: Static bubble, temperature dependent NSK model in 2d. Total L^2 error and EOC for $p = 1, 2, 3$.

Figure 9.22 and Table 9.13 show the results of these computations for polynomial degree $p = 0, 1, 2, 3$. The first order scheme seems to be not in the asymptotic regime for the tested mesh sizes (this can be seen from Figure 9.22) whereas the higher order schemes achieve the expected order $p + 1$.

9.11 Condensation, Evaporation

The setting in this section consists of a static bubble or a drop in a spherical cylinder initially at a dimensionless temperature $\theta = 0.85$. For $t > 0$ the temperature at the solid wall of the container is raised to the constant $\theta_{wall} = 0.95$ immediately. The initial data is chosen such that initially the mean density lies between the Maxwell states with respect to $\theta = 0.85$ which admits a stable bubble or drop at this temperature. At temperature $\theta = 0.95$ which corresponds to the wall temperature. The mean density lies in the vapor or liquid phase respectively but not between the Maxwell states with respect to $\theta = 0.95$. The boundary condition for the temperature implies that the solution approaches the constant wall temperature in the domain Ω as time tends to infinity. Therefore the bubble or the drop is not a stable configuration as $t \to \infty$ which results in a condensing bubble and an evaporating drop. At the end of the computation there is only a constant vapor or liquid state at $t \approx \infty$.

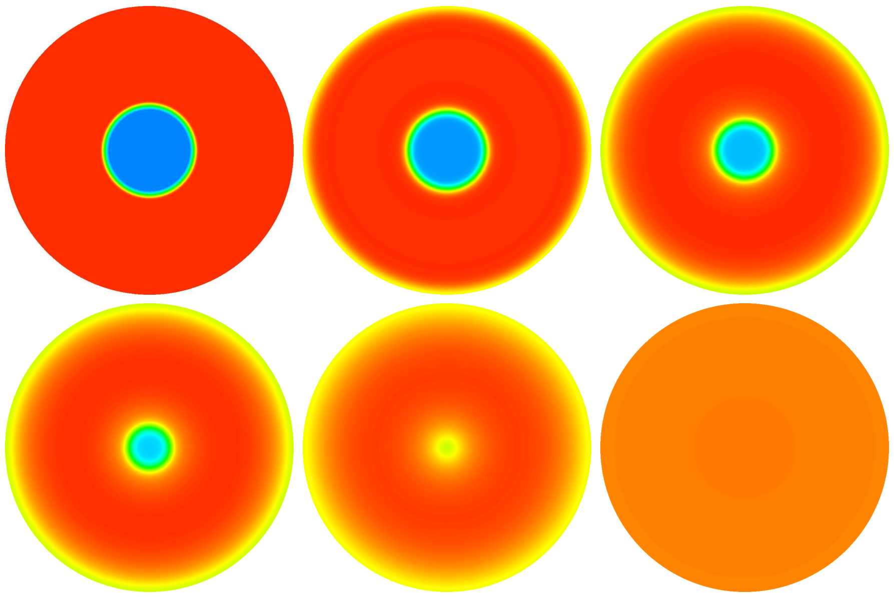

Figure 9.23: Condensating bubble in a spherical container. Density distribution at times $t = 0, 50, 250, 450, 850, 10000$ from top left to lower right picture.

The model considered in this section is again the two dimensional temperature dependent Navier-Stokes-Korteweg model with the same boundary conditions as in Section 9.10.

A similar experiment was proposed in [5]. In this work the temperature dependent Navier-Stokes-Korteweg model was used together with the assumption that the data stays spherical symmetric for all times $t \geq 0$. This assumption results in a time dependent one dimensional system that is approximately solved by a higher order finite difference scheme. However, in one space dimension a much smaller interface can be resolved by the mesh than in two space dimensions because of computational complexity and therefore we have to choose a smaller domain (which then gives a larger interface that can be resolved).

In [5] the fluid parameters were approximately these of the noble gas Argon. Most of the physical parameters differ by a factor of about ten between the vapor and the liquid phase. These parameters have to be fixed to some constants in between the vapor and liquid states.

For our experiment we have chosen the following physical parameters.

$$\begin{array}{rcll} L &=& 1.0 \cdot 10^{-7}\, m & \text{radius of the domain,} \\ c_{phys} &=& 4.0 \cdot 10^{2}\, \frac{K\,kg}{Nm} & \text{heat capacity at constant volume,} \\ \mu_{phys} &=& 3.0 \cdot 10^{-5}\, \frac{Ns}{m^2} & \text{viscosity,} \\ \kappa_{phys} &=& 4.0 \cdot 10^{-2}\, \frac{W}{mK} & \text{heat conductivity,} \\ \sigma_{phys} &=& 5.0 \cdot 10^{-3}\, \frac{N}{m} & \text{surface tension.} \end{array}$$

These are approximately the parameters of Argon at a dimensionless reference temperature $\theta_{ref} = 0.85$, as in [5], and can be found in Section B.2. Note that some constant states between the vapor and liquid states have been chosen.

The corresponding dimensionless parameters are then given by

$$\begin{array}{rcl} c &=& 6.63, \\ \mu &=& 5.87 \cdot 10^{-3}, \quad \nu = -3.91 \cdot 10^{-3}, \\ \lambda &=& 3.95 \cdot 10^{-4}, \\ \kappa &=& 1.30 \cdot 10^{-1}. \end{array}$$

These parameters are obtained by the physical parameters from above together with the scaling given in Section B.1.

In this experiment the velocity field in the whole computation is rather small since the temperature propagation from the wall is mainly driven by heat conduction and thus, very slow. The computational end time is $T = 10000.0$, really large compared to the experiments in the previous sections and therefore it is not possible finish the computation within an acceptable time frame with the same radius of a domain as in [5]. In our simulation the radius of the domain is ten times smaller than in [5].

Figure 9.23 shows the density distribution of a sequence of snapshots for a condensating bubble at times $t = 0, 50, 250, 450, 850, 10000$ and the corresponding temperature distribution can be found in Figure 9.24. As usual the density varies between approximately 0.3 and 1.8. The temperature has values in the range $(0.85, 0.95)$, values between initial and wall temperature.

The density distribution of an evaporating drop is shown in Figure 9.25. The snapshots are taken at times $t = 0, 500, 1500, 2500, 3100, 10000$

9.12 Oscillating Bubble

In this section we investigate the dynamics of a single spherical bubble that oscillates due to perturbation of the velocity field at the boundary. We compare the radius of the

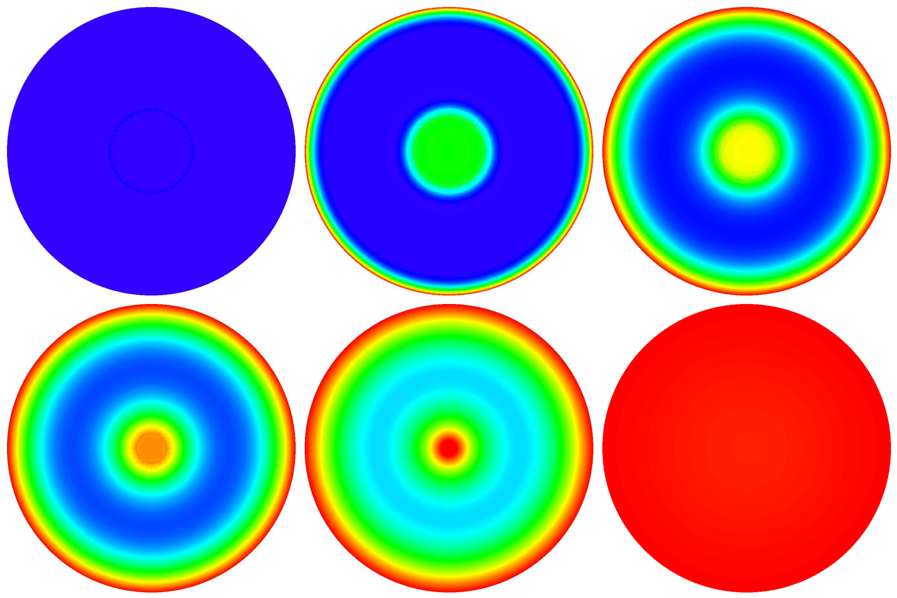

Figure 9.24: Condensating bubble in a spherical container. Temperature distribution in the range 0.85 (blue) to 0.95 (red) at times $t = 0, 50, 250, 450, 850, 10000$ from top left to lower right picture.

bubble given by the numerical simulation using the isothermal Navier-Stokes-Korteweg model in two space dimensions with the predicted radiuses given by the Rayleigh-Plesset formula (4.22) and the *Incompressibility* formula (4.25). We cannot expect that the results of the simulation using the NSK model matches exactly with the results given by the formulas since effects like compressibility and mass transfer over the liquid-vapor interface are neglected. But if there are qualitatively agreements with the formulas these could be used to predict a certain behavior of the solution like a bubble collapse. The Rayleigh-Plesset equation is usually used to predict such a behavior.

We consider the domain $\Omega = B_L(0) \subset \mathbb{R}^2$ with $L = 1.0$. Instead of using the boundary condition $\boldsymbol{u} = \boldsymbol{0}$ on $\partial\Omega$ we *simulate* a vibrating container by application of the boundary conditions

$$\begin{array}{rcl} \boldsymbol{u} \cdot \boldsymbol{n} & = & x(t), \\ \boldsymbol{u} \cdot \boldsymbol{\tau} & = & 0, \end{array} \quad \text{on } \partial\Omega \tag{9.1}$$

where $\boldsymbol{n}$ denotes the normal and $\boldsymbol{\tau}$ the tangent on the boundary $\partial\Omega$. The *vibrating container* experiment in Section 4.4.2 requires the complicated treatment of a moving domain. To avoid this we simulate the vibrating container experiment using the boundary conditions (9.1) on a fixed domain which means that we have a mass transfer over the boundary of the domain. In practice the difference between both experiments should be negligible as long the mean of the mass in Ω over a period of oscillation does not change. This holds for our computation.

The configuration of the Navier-Stokes-Korteweg model is completely the same as in

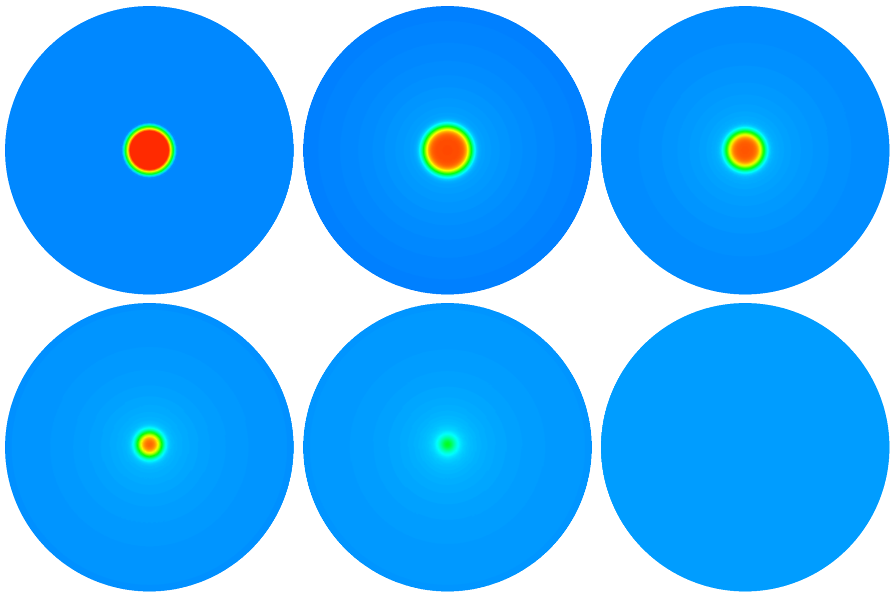
Figure 9.25: Evaporating drop in a spherical container. Density distribution at times $t = 0, 500, 1500, 2500, 3100, 10000$ from top left to lower right picture.

Section 9.3 including the settings of the interface indicator (not shown below), namely

$$\begin{aligned}
\theta_{ref} &= 0.85, \\
\lambda &= 1.0 \cdot 10^{-5}, \\
\varepsilon &= 1.366 \cdot 10^{-3}, \quad \mu = \frac{3}{4}\varepsilon, \quad \nu = -\frac{1}{2}\varepsilon.
\end{aligned}$$

The initial data is given by a rotationally (with respect to the origin) symmetric static bubble with an equilibrium radius, vapor and liquid density states given by

$$R_{eq} = 0.345, \quad \rho_v = 0.3208, \quad \rho_l = 1.8088.$$

The oscillation in the velocity field by the boundary conditions (9.1) is imposed by the function

$$x(t) = -0.005 \ \cos(0.5 \ \pi \ t). \tag{9.2}$$

Given the density distribution from the computation at a time t by the function ρ_h we compute the radius of the vapor bubble at time t by the relation

$$\pi R(t)^2 = \int_\Omega \eta(\boldsymbol{x}, t) \, d\boldsymbol{x}, \qquad \eta(\boldsymbol{x}, t) = \begin{cases} 1 & \text{if } \rho_h(\boldsymbol{x}, t) \leq 1, \\ 0 & \text{else.} \end{cases}$$

Note that here the density value 1 is the threshold for the vapor density values.

Comparison with the Rayleigh-Plesset formula

We compare the radius of the bubble computed using the NSK model with the radius predicted by the two dimensional Rayleigh-Plesset equation (4.22). First we have to provide the *input* for the Rayleigh-Plesset equation.

The pressure oscillation in the liquid phase $p_L(t)$ close to the boundary of the domain is taken from the computation using the NSK model. We assume that there is no mass transfer over the liquid-vapor interface and we further assume that the density inside the bubble does not depend on the spatial variable. This leads to a density in the bubble that depends only on the radius of the bubble and the initial configuration. The pressure inside the bubble is then given by the function $p_B(t)$ stated below as well the rest of the missing parameters for the two dimensional Rayleigh-Plesset formula (4.22).

$$\begin{aligned}
n &= 2, \\
L &= 1.0, \\
\sigma &= c_0(\theta_{ref}) \cdot \sqrt{\lambda} = 0.5238 \cdot \sqrt{\lambda}, \\
p_B(t) &= p\left(\rho_v \left(\frac{R_{eq}}{R(t)}\right)^n\right), \\
p_L(t) &= 0.495 - 0.005 \cdot \cos(0.5\pi \cdot t).
\end{aligned}$$

Here the surface tension coefficient σ is computed by application of the formula (2.68).

Figure 9.26 shows the radius of the bubble taken from the computation using the Navier-Stokes-Korteweg model compared with the radius predicted by the Rayleigh-Plesset formula.

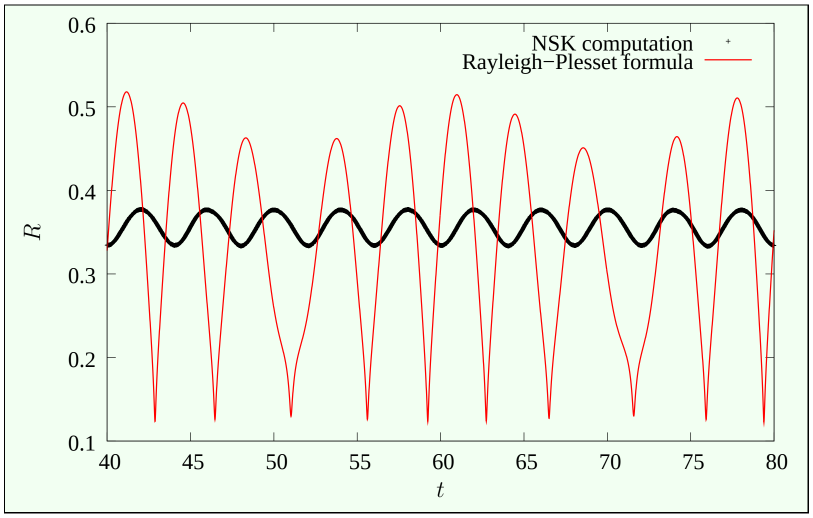

Figure 9.26: Oscillating bubble. Comparison with the Rayleigh-Plesset formula.

The radius computed by the NSK simulation oscillates with the frequency of the perturbation with an almost constant amplitude whereas the radius given by the Rayleigh-Plesset equation shows a completely different behavior. It seems that the frequency of the pressure perturbation in the liquid phase and another frequency that is associated with the bubble interfere with each other. Since mass transfer over the liquid-vapor interface is neglected in the Rayleigh-Plesset equation the gas phase is completely compressed and relaxed. This results in a force term that determines, together with the force term that comes from the pressure perturbation in the liquid phase, the position of the bubble interface. In contrast to that we observe in the computation using the NSK model that during the oscillation there is almost no compression in the vapor phase. The vapor close to the interface condensates immediately and the bubble interface can freely move. It is unclear whether this behavior is physically correct or not but it attracts the attention to the fact that there is no free parameter in the Navier-Stokes-Korteweg model left that can control the amount of mass transfer over the liquid-vapor interface. As a result we see that the assumptions on the density and pressure in the vapor phase that serve as input for the Rayleigh-Plesset formula are completely wrong and the oscillation of a bubble from the NSK simulation can neither quantitatively nor qualitatively be predicted by the Rayleigh-Plesset formula since frequency and amplitude are totally different. It is also unclear which of the computations is closer to real world behavior since the size of the domain and the reference temperature is totally different with respect to the settings of existing experimental data.

Comparison with the Incompressibility formula

The only input for the Incompressibility formula (4.25) is the equilibrium radius R_{eq} of the bubble at time $t = 0$ and the perturbation in the velocity field given by equation (9.2).

Figure 9.27 shows the resulting radiuses of the bubble given by the Navier-Stokes-Korteweg simulation and by formula (4.25) respectively.

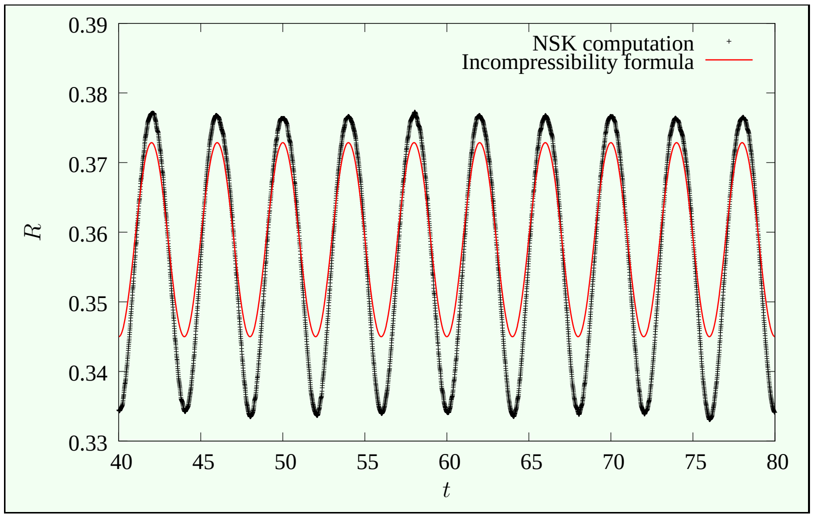

Figure 9.27: Oscillating bubble. Comparison with the Incompressibility formula.

We can see that the liquid-vapor interface oscillates with the frequency of the perturbation at the boundary of the domain in both simulations. The amplitude given by formula (4.25) differs from the amplitude that comes from the NSK-computation.

We summarize the results of the computations done in this section as follows.

- The behavior of a *NSK* bubble is not predictable by the Rayleigh-Plesset equation, it is qualitatively predictable by the Incompressibility formula.
- There is almost no change in the density and pressure in the vapor phase when a *NSK* bubble oscillates.
- There is no free parameter in the Navier-Stokes-Korteweg model left to control the mass transfer over the phase interface.
- The correct physical behavior is unclear since experimental data is not available on the temperature and length scale of our simulation.

Appendix A

Notation and Definitions

A.1 Notation

This section gives a summary of frequently used notational conventions concerning Thermodynamical and Kinematic variables and differential operators.

Thermodynamic and Kinematic quantities

t	$\geq$	0	time variable,
$\boldsymbol{x}$	$\in$	$\mathbb{R}^n$	spatial variable,
ρ	$=$	$\rho(\boldsymbol{x},t) > 0$	density of the fluid,
$\boldsymbol{u}$	$=$	$\boldsymbol{u}(\boldsymbol{x},t) \in \mathbb{R}^n$	velocity of the fluid,
$\mathcal{E}$	$=$	$\mathcal{E}(\boldsymbol{x},t) \in \mathbb{R}$	total energy of the fluid,
θ	$=$	$\theta(\boldsymbol{x},t) > 0$	temperature of the fluid,
f	$=$	$f(\theta,\rho) \in \mathbb{R}$	(Helmholtz) free energy, denotes also the extended free energy $f(\theta,\rho,\alpha)$ with $\alpha = \frac{1}{2}\|\nabla\rho\|^2$,
e	$=$	$e(\theta,\rho) \in \mathbb{R}$	internal energy or extended internal energy $e(\theta,\rho,\alpha)$,
s	$=$	$s(\theta,\rho) \in \mathbb{R}$	specific entropy or extended specific entropy $s(\theta,\rho,\alpha)$,
p	$=$	$p(\theta,\rho) \in \mathbb{R}$	pressure,
μ	$=$	$\mu(\theta,\rho) \in \mathbb{R}$	chemical potential, same as Gibbs free energy for a one component fluid,
μ	$>$	$0,\ \nu \in \mathbb{R}$	viscosity coefficients,
ε	$=$	$2\mu + \nu$	one dimensional viscosity coefficients,
λ	$>$	0	capillarity coefficient,
κ	$>$	0	heat conduction coefficient,
$(0,\overline{\rho}_v)$			vapor phase,
$(\underline{\rho}_l, b)$			liquid phase,
ρ_v^M,		ρ_l^M	vapor and liquid Maxwell states.

Differential Operators

Commonly used differential operators with respect to the spatial variable $\boldsymbol{x} = (x_1, \dots, x_n)^T \in \mathbb{R}^n$ are defined in the following list.

$\nabla u = \left(\frac{\partial}{\partial x_1} u, \dots, \frac{\partial}{\partial x_n} u\right)^T$. Denotes the gradient of a scalar, real valued function $u : \mathbb{R}^n \to \mathbb{R}$, i.e., the transposed Jacobian.

$\nabla \boldsymbol{u} = \left(\frac{\partial}{\partial x_j} u_i\right)_{i,j}$. Denotes the gradient of a vector valued function $\boldsymbol{u} : \mathbb{R}^n \to \mathbb{R}^m$, i.e., the Jacobian (not transposed).

$\nabla \cdot \boldsymbol{u} = \sum_{i=1}^{n} \frac{\partial}{\partial x_i} u_i$. The divergence of a vector field $\boldsymbol{u} = (u_1, \dots, u_n)^T : \mathbb{R}^n \to \mathbb{R}^n$.

$\nabla \cdot \boldsymbol{A} = \left(\sum_{j=1}^{n} \frac{\partial}{\partial x_j} A_{1,j} \;, \dots, \sum_{j=1}^{n} \frac{\partial}{\partial x_j} A_{n,j}\right)^T$. Denotes the divergence of a tensor field $\boldsymbol{A} : \mathbb{R}^n \to \mathbb{R}^{n \times n}$. Here the $A_{i,j}$ denote the entries of the matrix A.

$\frac{D}{Dt}\varphi = \varphi_t + \boldsymbol{u} \cdot \nabla \varphi$. The material derivative with respect to the velocity field $\boldsymbol{u}$ of a function $\varphi : \mathbb{R}^n \times \mathbb{R}_{\geq 0} \to \mathbb{R}$.

For a function $\varphi = \varphi(\rho, \alpha)$ where α stands for $\frac{1}{2}|\nabla \rho|^2$ the variational derivative with respect to ρ is denoted by

$$[\varphi]_\rho = \varphi_\rho - \nabla \cdot (\varphi_\alpha \nabla \rho)$$

as used in standard textbooks as [32].

A.2 General Definitions

Definition A.2.1 *(Experimental order of convergence)*
Let $(h_n)_{n \in \mathbb{N}}$ be a monotonically decreasing sequence that converges to zero and $\varphi \in C^0([0, \infty), \mathbb{R}_{>0})$. Then for $n > 0$ the experimental order of convergence is defined by

$$EOC(\varphi, h_n) = \frac{\log\left(\frac{\varphi(h_n)}{\varphi(h_{n-1})}\right)}{\log\left(\frac{h_n}{h_{n-1}}\right)} \tag{A.1}$$

Definition A.2.2 *(Kronecker Product)*
For two matrices $Q \in \mathbb{R}^{s \times r}$ and $M \in \mathbb{R}^{n \times m}$ we define the Kronecker product matrix $Q \otimes M \in \mathbb{R}^{sn \times rm}$ by

$$Q \otimes M = \begin{pmatrix} q_{0,0}M & \dots & q_{0,s-1}M \\ \vdots & & \vdots \\ q_{s-1,0}M & \dots & q_{s-1,s-1}M \end{pmatrix}, \tag{A.2}$$

where the scalar values $q_{i,j}$ denote the entries of the matrix Q.

In the following lemma we summarize some useful properties of the Kronecker product. The proof is a straightforward calculation, so we omit it.

Lemma A.2.3
For $s, n \in \mathbb{N}$ let $A, C \in \mathbb{R}^{s\times s}$ and $B, D \in \mathbb{R}^{n\times n}$. Then we have the following properties of the Kronecker product:

(i) $(A \otimes B)(C \otimes D) = AC \otimes BD$.

(ii) If the matrices A and B are invertible then $A \otimes B$ is also invertible and we have the identity $(A \otimes B)^{-1} = A^{-1} \otimes B^{-1}$.

(iii) $I_s \otimes I_n = I_{sn}$, where $I_k \in \mathbb{R}^{k\times k}$ for $k \in \mathbb{N}$ denotes the unit matrix.

A.3 Characterization of the Maxwell States

We give a definition and equivalent characterizations of the Maxwell values for some general W-shaped free energy.

Definition A.3.1
Let the constants $\overline{\rho}_v, \underline{\rho}_l, b \in \mathbb{R}$ with $0 < \overline{\rho}_v < \underline{\rho}_l < b$ and $W \in C^2((0, b))$ with

$$\begin{gathered} W'' > 0 \text{ in } (0, \overline{\rho}_v) \cup (\underline{\rho}_l, b) \quad \text{and} \quad W'' < 0 \text{ in } (\overline{\rho}_v, \underline{\rho}_l), \\ \lim_{\rho\to 0} W(\rho) = \infty \quad \text{and} \quad \lim_{\rho\to b} W(\rho) = \infty, \end{gathered} \tag{A.3}$$

be given. Then by the shape of W it is clear that there exist unique states $\rho_v^M \in (0, \overline{\rho}_v)$ and $\rho_l^M \in (\underline{\rho}_l, b)$ with the property

$$W'(\rho_v^M) = W'(\rho_l^M), \tag{A.4}$$

$$W(\rho_l^M) = W(\rho_v^M) + W'(\rho_v^M)(\rho_l^M - \rho_v^M). \tag{A.5}$$

These states are called Maxwell states.

We define functions p and μ by

$$p(\rho) = \rho W'(\rho) - W(\rho), \tag{A.6}$$

$$\mu(\rho) = W'(\rho). \tag{A.7}$$

With the definition of $W(\rho) = \rho f^{vdW}(\rho)$ where f^{vdW} denotes the Helmholtz free energy of an isothermal van der Waals fluid, see Section 2.1, $\overline{\rho}_v$ and $\underline{\rho}_l$ denote the phase boundaries given in definition 2.1.5 and W has the properties stated in (A.3). The functions p and μ are equal to the pressure and chemical potential of a van der Waals fluid. The definition of the Maxwell states is equivalent to that given in definition 2.1.6 as we will see in lemma A.3.2. This lemma gives three equivalent characterizations of the Maxwell states.

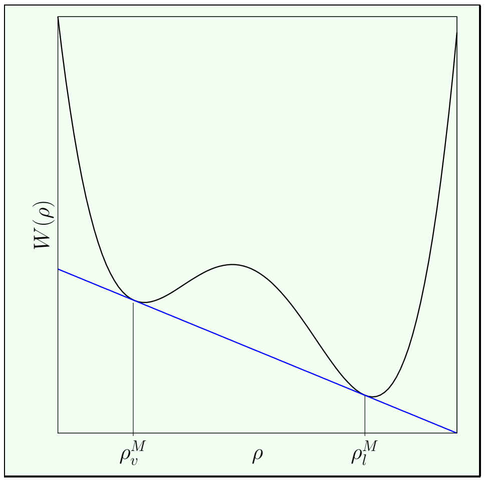

Figure A.1: Energy W and the associated Maxwell states.

Lemma A.3.2
With the above definitions the Maxwell states can be characterized equivalently by

(i) *equations (A.4) and (A.5).*

(ii) *the equations*

$$p(\rho_v^M) = p(\rho_l^M), \tag{A.8}$$
$$\mu(\rho_v^M) = \mu(\rho_l^M). \tag{A.9}$$

In this way the Maxwell states are defined in definition 2.1.6 for a van der Waals fluid.

(iii) *the equations*

$$p(\rho_v^M) = p(\rho_l^M), \tag{A.10}$$
$$\int_{\rho_v^M}^{\rho_l^M} \frac{p(\rho) - p(\rho_v^M)}{\rho^2}\, d\rho = 0. \tag{A.11}$$

Proof. For notational simplicity we denote the Maxwell states by ρ_v and ρ_l.
(i) $\Leftrightarrow$ (ii): By definition of μ equations (A.9) and (A.4) are the same. Using this identity, the above definition for the function p and property (A.5) we get

$$\begin{aligned} p(\rho_v) &= \rho_v W'(\rho_v) - W(\rho_v) \\ &= \rho_v W'(\rho_v) - W(\rho_l) + W'(\rho_l)(\rho_l - \rho_v) \\ &= W'(\rho_l)\rho_l - W(\rho_l) \\ &= p(\rho_l). \end{aligned}$$

Thus, we have property (A.8). The opposite direction is done analogously.

(ii) $\Leftrightarrow$ (iii): Using the above definitions and integration by parts we have

$$\begin{aligned}
\mu(\rho_l) - \mu(\rho_v) &= \int_{\rho^v}^{\rho_l} \mu'(\rho)\, d\rho \\
&= \int_{\rho^v}^{\rho_l} \frac{p'(\rho)}{\rho}\, d\rho \\
&= \int_{\rho^v}^{\rho_l} \frac{p(\rho)}{\rho^2}\, d\rho \;+\; p(\rho_v)\left(\frac{1}{\rho_l} - \frac{1}{\rho_v}\right) \\
&= \int_{\rho^v}^{\rho_l} \frac{p(\rho) - p(\rho_v)}{\rho^2}\, d\rho.
\end{aligned}$$

Hence, we have the equivalence of equations (A.9) and (A.11). This completes the proof.

Lemma A.3.3
With the notation above let the function ϕ be given by

$$\phi(\rho) = \int_{\rho_v^M}^{\rho} \frac{p(s) - p(\rho_v^M)}{s^2}\, ds.$$

Then we have

$$\phi(\rho_v^M) = 0, \quad \phi(\rho_l^M) = 0 \quad \textit{and} \quad \phi(\rho) > 0 \textit{ for all } \rho \in (\rho_v^M, \rho_l^M).$$

Proof. $\phi(\rho_v^M) = 0$ is trivial, $\phi(\rho_l^M) = 0$ because of the characterization of the Maxwell states, see lemma A.3.2. Since p is monotonically increasing in the vapor phase we have $\phi(\rho) > 0$ in $(\rho_v^M, \rho_v^M + \varepsilon)$ for some sufficiently small value $\varepsilon > 0$. Because of the shape of the function p it is not possible for ϕ to have another zero in the interval $[\rho_v^M, \rho_l^M]$ except the Maxwell states (the integrand changes the sign only once in the interval). This completes the proof.

A.4 Definition of Nonconservative Products

In this section we give a definition of *nonconservative products*, i.e., products of the form $f(u) \cdot \frac{d}{dx} v$. Products of this form appear in the formulation of the Discontinuous Galerkin method (see Chapter 6) and cannot be defined as functions in the case where the function u and v are discontinuous. In the case where u and v are discontinuous functions we can define the nonconservative product in the sense of measures following the work of Dal Maso, LeFloch and Murat [36] in the one dimensional case. We give a (formal) multidimensional generalization of this definition to products of the form $\sum_{i=1}^{n} f_i(u) \cdot \frac{\partial}{\partial x_i} v$. Here we do not claim that the measure we construct in the multidimensional case is well defined as it is in the one dimensional case ensured by the work of Dal Maso, LeFloch and Murat.

We start with the description of *paths* ϕ. This is an object the resulting measure will depend on. Let $\phi : [0,1] \times \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}^d$ be a locally Lipschitz continuous map with the following three properties

(i) $\phi(0; u^-, u^+) = u^-$ and $\phi(1; u^-, u^+) = u^+$ for all $u^-, u^+ \in \mathbb{R}^d$,

(ii) $\phi(t; u, u) = u$ for all $u \in \mathbb{R}^d$, $t \in [0,1]$,

(iii) for all bounded sets $U \subset \mathbb{R}^d$ there exists a constant $c \geq 1$, such that for all $u^-, u^+, v^-, v^+ \in U$ and almost all $t \in [0,1]$ we have

$$|\phi'(t; u^-, u^+) - \phi'(t; v^-, v^+)| \leq c|(u^- - v^-) - (u^+ - v^+)|.$$

In the above statement ϕ' denotes the derivative with respect to t which exists for almost all $t \in [0,1]$.

Theorem A.4.1
Let $a < b$, $u, v \in BV((a,b), \mathbb{R}^d)$ and let f be locally bounded in the sense that for all $U \subset \mathbb{R}^d$ bounded there exists a constant $c > 0$ such that for all $u \in U$ and $x \in (a,b)$ we have $|g(u,x)| \leq c$. Then there exists a unique bounded Borel measure μ on (a,b) characterized by the following two properties

(i) If the function u is continuous in $B \subset (a,b)$, then

$$\mu(B) = \int_B f(u(x), x) \cdot d(v')(x),$$

where the integral is defined with respect to vector-valued Borel measure (v')

(ii) For $x \in (a,b)$ we have

$$\mu(\{x\}) = \int_0^1 g(\phi(t; u(x^-), u(x^+)), x) \cdot \phi'(t; v(x^-), v(x^+)) dt.$$

$u(x^-)$ and $u(x^+)$ denote the limit (which exists for functions of bounded variation in one space dimension) from the left and right respectively.

Definition A.4.2
The measure μ introduced in the above theorem is called the nonconservative product of $f(u(\cdot), \cdot)$ and v' and is denoted by

$$\left[f(u, \cdot) \cdot (v')\right]_\phi = \mu,$$

which in general depends on the paths ϕ.

We consider the interval $\Omega = (a, b)$ and a partition (mesh) $\mathcal{T}$ of this interval as defined in definition 6.1.1. Let $V = \{\varphi : \Omega \to \mathbb{R} \mid \varphi|_{\Delta_j} \in C^1(\Delta_j),\ \Delta_j \in \mathcal{T}\}$ and $u, v \in V^d$. According to the above theorem the measure μ applied to the whole interval Ω can be computed as

$$\begin{aligned}
&\int_\Omega d\left[f(u,\cdot)\cdot(v')\right]_\phi(x) \\
&\quad= \sum_{j=0}^{|\mathcal{T}|-1} \int_{\Delta_j} f(u(x),x)\cdot v'(x)\,dx \\
&\qquad+ \sum_{j=1}^{|\mathcal{T}|-1} \int_0^1 f\big(\phi(t;u(x_{j-}^-),u(x_{j-}^+)),x_{j-}\big)\cdot\phi'(t;v(x_{j-}^-),v(x_{j-}^+))\,dt.
\end{aligned}$$

Here x_{j-} and x_{j+} denote the left and right vertices of cell Δ_j. The vertices at the boundary of the interval give no contribution to the measure because there is no discontinuity.

We give a n-dimensional generalization of the measure μ as we need it to define the Discontinuous Galerkin method for conservative as well as for nonconservative equations in Section 6.2. Let $\Omega \subset \mathbb{R}^n$ be an open bounded set such that a mesh $\mathcal{T}$ that partitions Ω exists. Let $V = \{\varphi : \Omega \to \mathbb{R} \mid \varphi|_{\Delta_j} \in C^1(\Delta_j),\ \Delta_j \in \mathcal{T}\}$ and $u, v \in V^d$. The generalization of the measure

$$\mu = \left[\sum_{i=1}^n f_i(u,\cdot)\cdot\frac{\partial}{\partial x_i}v\right]_\phi$$

applied to the set Ω can be computed as

$$\begin{aligned}
&\int_\Omega d\left[\sum_{i=1}^n f_i(u,\cdot)\cdot\frac{\partial}{\partial x_i}v\right]_\phi(x) \\
&= \sum_{j=0}^{|\mathcal{T}|-1}\int_{\Delta_j}\sum_{i=1}^n f_i(u(x),x)\cdot\frac{\partial}{\partial x_i}v(x)\,dx \qquad\qquad (A.12)\\
&\quad+\frac{1}{2}\sum_{j=0}^{|\mathcal{T}|-1}\int_{\partial\Delta_j\setminus\partial\Omega}\int_0^1\sum_{i=1}^n \nu_i f_i\big(\phi(t;u(x_j),u(x_{j'})),x_j\big)\cdot\phi'(t;v(x_j),v(x_{j'}))\,dt\,d\sigma(x).
\end{aligned}$$

In the above equation $u(x_j)$ stands for $u|_{\Delta_j}(x)$ and $u(x_{j'})$ for $u|_{\Delta_{j'}}(x)$ where $\Delta_{j'}$ denotes a corresponding neighboring cell. ν_i denotes the i-th component of the normal vector ν. The factor $\frac{1}{2}$ in front of the last term appears because all interfaces are counted twice, except the boundary interfaces.

Note that we do not claim that in the multidimensional case the measure μ is well defined as it is in one space dimension guaranteed by theorem A.4.1. We only define an object $\mu(\Omega)$ by the right hand side of equation (A.12), where Ω is partitioned by an underlying mesh and u, v are functions from the space V^d. This is what we need for the definition of the Discontinuous Galerkin method in Chapter 6.

Sometimes it is more convenient to work with the notion of *numerical fluxes* instead of the notion of paths ϕ, i.e., in the Finite Volume and Discontinuous Galerkin Framework. Therefore we replace the term

$$\int_0^1 \sum_{i=1}^n \nu_i f_i\big(\phi(t; u(x_j), u(x_{j'})), x_j\big) \cdot \phi'(t; v(x_j), v(x_{j'}))\, dt$$

by the expression

$$g\big(u(x_j), u(x_{j'}), x_j, \nu\big) \cdot \big(v(x_{j'}) - v(x_j)\big)$$

with a *suitable* function g. In order to be an approximation in some sense the function g has to satisfy at least the relation

$$g(u, u, x, \nu) = \sum_{i=1}^n \nu_i f_i(u, x)$$

for all $u \in \mathbb{R}^d, x \in \mathbb{R}^n$ and $n \in \{x \in \mathbb{R}^n \mid |x| = 1\}$. In the Finite Volume framework such a function g is called *numerical flux* function and is usually supposed to be locally Lipschitz continuous. Using the above expression we get

$$\begin{aligned}\mu(\Omega) \quad \approx \quad & \sum_{j=0}^{|\mathcal{T}|-1} \int_{\Delta_j} \sum_{i=1}^n f_i(u(x), x) \cdot \frac{\partial}{\partial x_i} v(x)\, dx \\ & + \frac{1}{2} \sum_{j=0}^{|\mathcal{T}|-1} \int_{\partial\Delta_j \setminus \partial\Omega} g\big(u(x_j), u(x_{j'}), x_j, \nu\big) \cdot \big(v(x_{j'}) - v(x_j)\big)\, d\sigma(x).\end{aligned}$$

As before, the factor $\frac{1}{2}$ appears because all interfaces are double counted. Note that the dependence on the path ϕ is dropped in favor of the dependence on the numerical flux g.

Appendix B

Fluid Properties

In this chapter we summarize the nondimensionalization procedure given in Chapter 2 and provide necessary fluid parameters for the fluids Argon, Butane and Water.

B.1 Dimensionless Scaling

For the dimensionless version of the Navier-Stokes-Korteweg system derived in Chapter 2 we have to provide a reference length L in m. Usually L is chosen to be the diameter of the domain Ω. This is the only parameter that does not depend on the fluid.

We need the critical values of the fluid, i.e., the critical temperature θ_{crit} in K, the critical density ρ_{crit} in $\frac{kg}{m^3}$ and the critical pressure in $\frac{N}{m^2}$. Table B.1 shows these values for the fluids Argon, Butane and Water. Critical values of other fluids can be obtained for example from the NIST website [125].

Up to now all fluid parameters were constants, parameters like heat capacity, viscosity, heat conductivity and surface tension depend on temperature and density in general but we will fix them to some *reference constants* for simplicity. Figures B.1 - B.4 show these parameters on the *saturation curve*, i.e., at the Maxwell states for different temperatures.

We choose a dimensionless reference temperature $\tilde{\theta}_{ref}$ which corresponds to the physical θ temperature by the relation $\theta_{crit}\,\tilde{\theta}_{ref} = \theta$. For example, if the boundary temperature is fixed to a constant $\tilde{\theta}_b$ we choose $\tilde{\theta}_{ref} = \tilde{\theta}_b$. Using the reference temperature we can determine the surface tension σ in $\frac{N}{m}$. Further we choose reference values for the heat capacity at constant volume c in $\frac{Nm}{kg\,K}$, heat conductivity κ in $\frac{W}{mK}$ and viscosity μ in $\frac{Ns}{m^2}$.

Using reference length and critical values we can define the reference time

$$T = L\sqrt{\frac{\rho_{crit}}{p_{crit}}}.$$

With the reference time we have also defined the reference velocity $\frac{L}{T}$. In the case of a perfect gas the reference velocity is equal to the sound speed at some reference state. In the case of a van der Waals fluid the reference velocity is not directly linked to the sound speed because $\frac{p}{\rho}$ is in general not equal to p_ρ in a van der Waals fluid as it is in the case of a perfect gas.

Now the following table summarizes the relations between the physical and dimensionless (tilde) values. The units of the corresponding physical values are given in the last column.

$\tilde{\boldsymbol{x}}$	$= \frac{1}{L}\boldsymbol{x}$	spatial variable	m
$\tilde{t}$	$= \frac{1}{T}t$	time variable	s
$\tilde{\rho}$	$= \frac{1}{\rho_{crit}}\rho$	density	$\frac{kg}{m^3}$
$\tilde{\boldsymbol{u}}$	$= \frac{T}{L}\boldsymbol{u}$	velocity	$\frac{m}{s}$
$\tilde{\theta}$	$= \frac{1}{\theta_{crit}}\theta$	temperature	K
$\tilde{\mu}$	$= \frac{1}{T\,p_{crit}}\,\mu$	viscosity	$\frac{Ns}{m^2}$
$\tilde{\nu}$	$= -\frac{2}{3}\tilde{\mu}$	viscosity	$\frac{Ns}{m^2}$
$\tilde{\kappa}$	$= \frac{\theta_{crit}\,\rho_{crit}}{T\,p_{crit}^2}\kappa$	heat conductivity	$\frac{W}{mK}$
$\tilde{\lambda}$	$= \left(\frac{\sigma(\tilde{\theta}_{ref})}{L\,p_{crit}\,\tilde{c}_0(\tilde{\theta}_{ref})}\right)^2$	capillarity	$\frac{m^7}{kg\,s^2}$
$\tilde{c}$	$= \frac{\theta_{crit}\,\rho_{crit}}{p_{crit}}c$	heat capacity at constant volume	$\frac{Kkg}{Nm}$
$\tilde{\boldsymbol{g}}$	$= \frac{T^2}{L}\boldsymbol{g}$	gravity	$\frac{m}{s^2}$

The coefficient $\tilde{c}_0(\tilde{\theta}_{ref})$ can be computed using the approximative formula (2.69)

$$\tilde{c}_0(\tilde{\theta}) = \sqrt{2}\cdot\sqrt{1.0-\tilde{\theta}}\cdot\left(6.4\cdot(1.0-\tilde{\theta}) - 0.7\cdot(1.0-\tilde{\theta})^2\right).$$

B.2 Equation of State

In this section we collect all important parameters we need for the nondimensionalization procedure for the fluids Argon, Butane and Water. Finally, as an example, we give the set of dimensionless parameters of water in a *micrometer container*. All data is taken from the NIST website [125].

Table B.1 shows the critical temperature, density and pressure of all three fluids.
Figures B.1 - B.3 show the heat capacity at constant volume, viscosity and heat conductivity of the three fluids on the *saturation curve*, this means at the Maxwell states in the vapor and liquid phase. The independent variable is the dimensionless temperature $\tilde{\theta} = \frac{1}{\theta_{crit}}\theta$.

Figure B.5 shows the pressure as graph of the density at 95% of the critical temperature for all three fluids and the van der Waals approximation (2.13). Argon and Butane are better approximated by the van der Waals equation of state than Water.

	Argon	Butane	Water
θ_{crit}	$150.687\ K$	$425.125\ K$	$647.096\ K$
ρ_{crit}	$535.599\ \frac{kg}{m^3}$	$228.000\ \frac{kg}{m^3}$	$322.000\ \frac{kg}{m^3}$
p_{crit}	$4.863 \cdot 10^6\ \frac{N}{m^2}$	$3.796 \cdot 10^6\ \frac{N}{m^2}$	$22.064 \cdot 10^6\ \frac{N}{m^2}$

Table B.1: Critical values of Argon, Butane and Water.

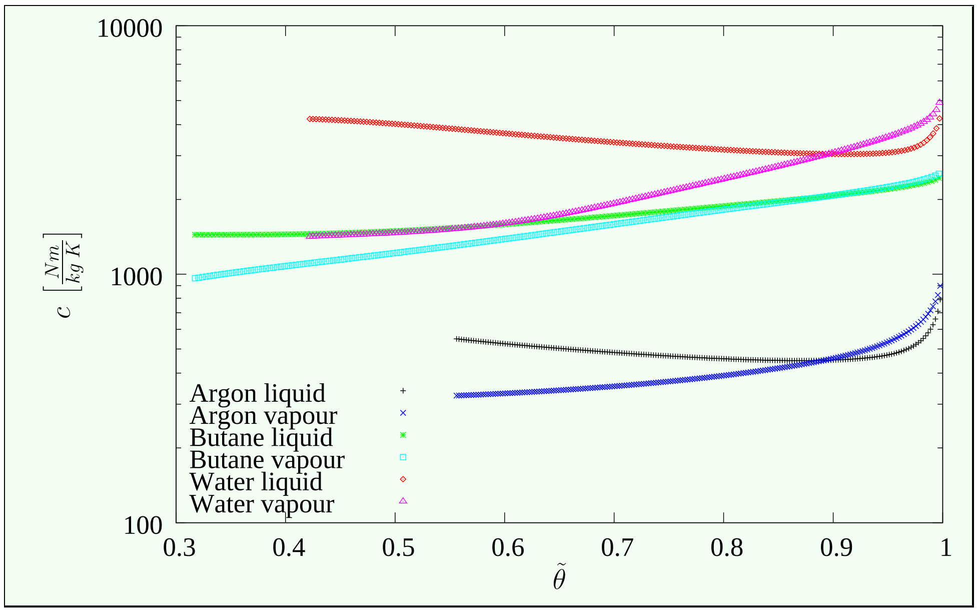

Figure B.1: Dimensionless temperature versus heat capacity at constant volume in $\left[\frac{Nm}{kg\ K}\right]$ for Argon, Butane and Water.

As an example we provide the dimensionless quantities of Water in a *micrometer container* at a reference temperature of $550K$, i.e., 85% of the critical temperature. The critical values of Water are given in Table B.1. The *mean* values of the heat capacity, heat conductivity and viscosity are fixed to some constant between the corresponding values of the vapor and liquid phases respectively.

Example B.2.1 *(Water at* $550K$*)*

$$L = 10^{-6}\ m$$

$$\begin{aligned} c &= 3.0 \cdot 10^3\ \frac{Nm}{kg\ K} & \tilde{c} &= 2.833 \cdot 10^1 \\ \mu &= 5.0 \cdot 10^{-5}\ \frac{Ns}{m^2} & \tilde{\mu} &= 5.931 \cdot 10^{-5}, \quad \tilde{\nu} = -3.954 \cdot 10^{-5} \\ \kappa &= 1.0 \cdot 10^{-1}\ \frac{W}{m\ K} & \tilde{\kappa} &= 1.120 \cdot 10^{-3} \\ \sigma &= 2.0 \cdot 10^{-2}\ \frac{Nm}{kg\ K} & \tilde{\lambda} &= 3.071 \cdot 10^{-8} \end{aligned}$$

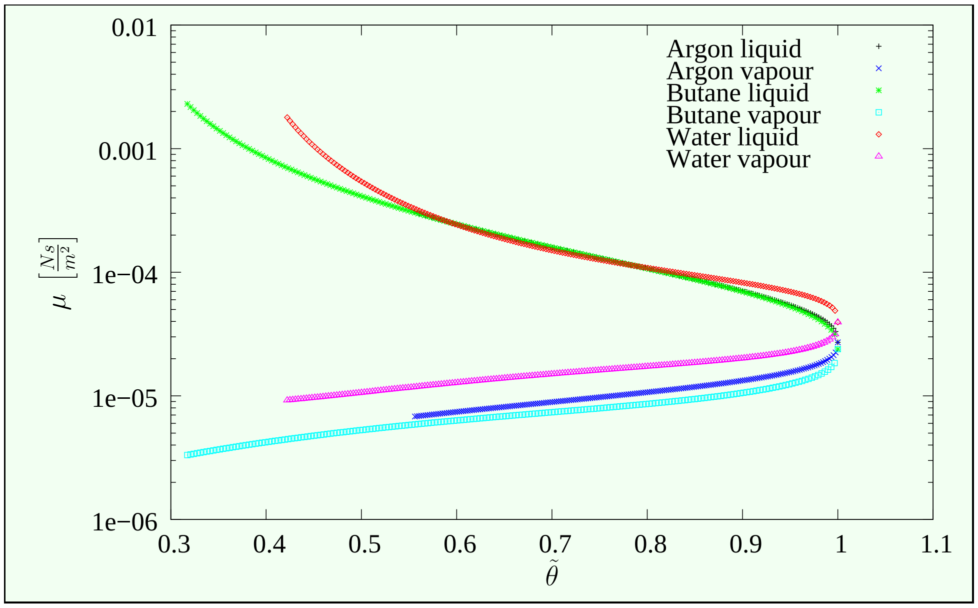

Figure B.2: Dimensionless temperature versus viscosity in $\left[\frac{Ns}{m^2}\right]$ for Argon, Butane and Water.

In this setting we have a reference velocity and reference time of

$$\begin{aligned} \frac{L}{T} &= 261.767 \, \frac{m}{s}, \\ T &= 3.820 \, ns. \end{aligned}$$

At this temperature the sound speed in the vapor phase and in the liquid phase (at the Maxwell values) are approximately

$$\begin{aligned} c^v_{snd} &= 493 \, \frac{m}{s}, \\ c^l_{snd} &= 1025 \, \frac{m}{s}. \end{aligned}$$

This means the reference velocity is approximately half the sound speed in the vapor phase.

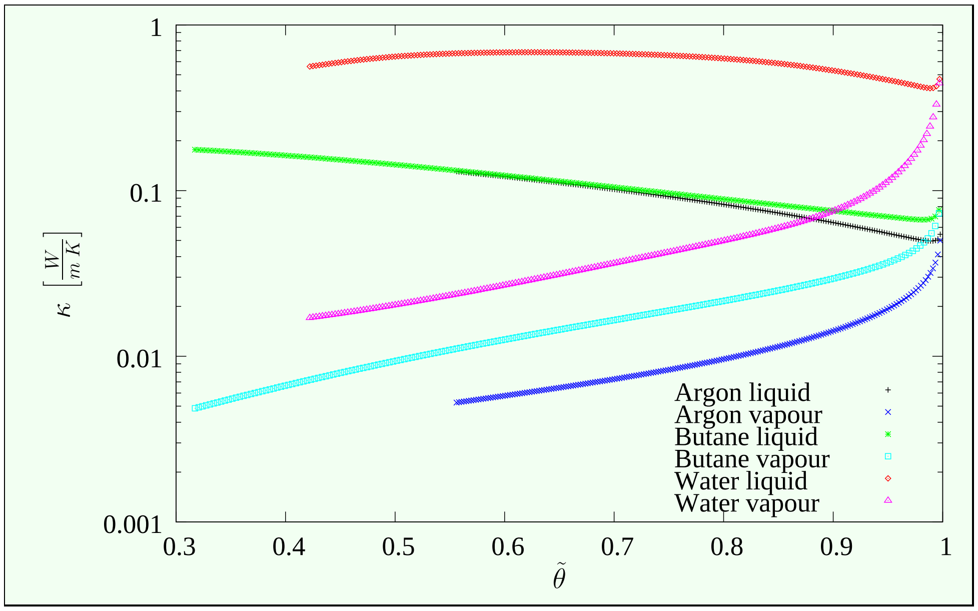

Figure B.3: Dimensionless temperature versus thermal conduction in $\left[\frac{W}{m\ K}\right]$ for Argon, Butane and Water.

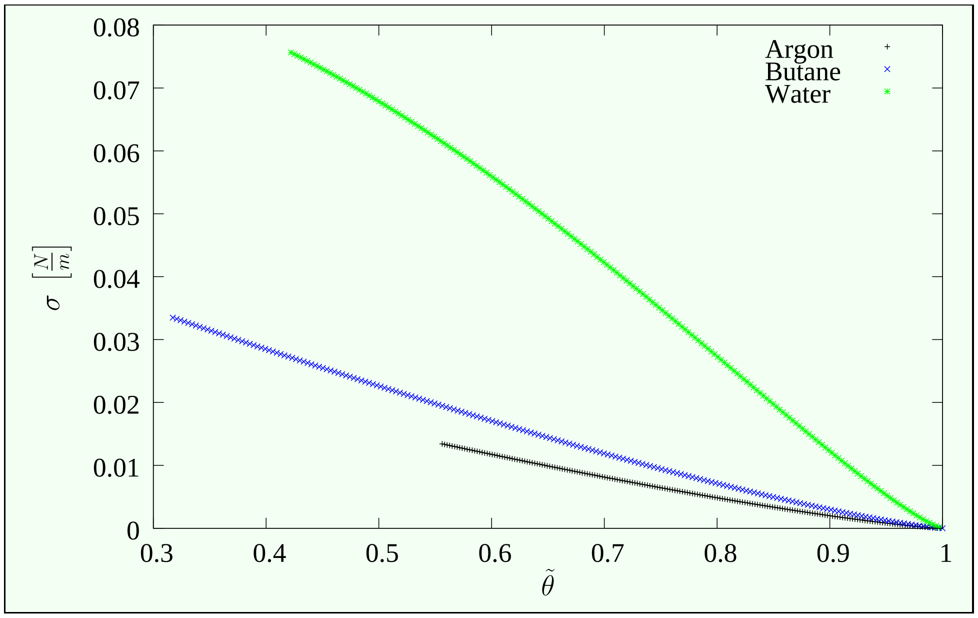

Figure B.4: Dimensionless temperature versus surface tension in $\left[\frac{N}{m}\right]$ of Argon, Butane and Water.

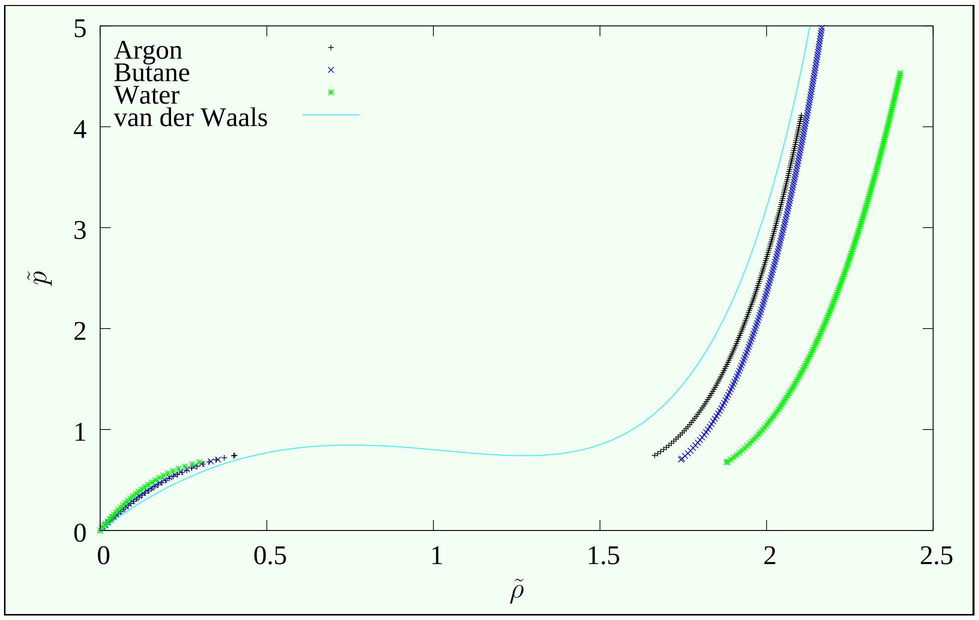

Figure B.5: Dimensionless density versus dimensionless pressure at dimensionless temperature 0.95 for Argon, Butane, Water and the dimensionless van der Waals equation of state (2.13).

Appendix C

Description of the Software Package

We give a basic description of the parDG (**par**allel **D**iscontinuous **G**alerkin) software package developed within the framework of this thesis. This chapter gives an overview of the software package and it provides all necessary knowledge for a user to discretize general time dependent partial differential equations by the Discontinuous Galerkin and Local Discontinuous Galerkin method together with a higher order time discretization in a parallel environment. It does not provide a detailed documentation on the implementation of the methods.

The package has a modular design completely written in C++ and relies on the standard libraries as well as on two external packages. The first necessary library is an implementation of the Message Passing Interface (MPI). There are a number different freely available and commercial implementations. Section 8.5 gives an overview and references to some freely available implementations. The second necessary external package is the ParMETIS library [68], [121] that provides graph partitioning algorithms in a parallel MPI-based environment. This package is used for partitioning and repartitioning of distributed meshes, i.e., for load balancing. The ParMETIS library is copyrighted by the Regents of the University of Minnesota. It can be freely used for educational and research purposes by non-profit institutions. Section 8.6 provides an overview of the ParMETIS library. The parDG package itself is released under the GNU GENERAL PUBLIC LICENSE, version 2.

Optionally an external BLAS (basic linear algebra subprograms [14], [119]) library can be used, for example the freely available package from the ATLAS project [129], [118] or some vendor provided library. The software package itself comes with its own implementation of the necessary CBLAS calls which are implemented as inline functions. However, on the tested architectures (x86 and amd64) external BLAS libraries do not give an extra performance gain.

In the following sections we give an overview of the most important classes and member functions that are necessary to apply this package. All classes and functions are declared inside the namespace *pardg*. For simplicity this namespace is omitted in the definition of the classes that follow in the next sections. Some classes have a deeper inheritance hierarchy, for simplicity this is neglected and the inherited member functions are assigned to the derived class.

At the end of this chapter we give two (stripped down) examples of usage. The first example is the DG discretization of the linear advection equation in one space dimension. The second example provides the basic DG discretization of the isothermal Navier-Stokes-Korteweg equations in two space dimensions including the higher order implicit Runge-Kutta time stepping.

C.1 Communicator Class

The parDG package does not provide serial algorithms. Everything is done in parallel. For the communication between processes Communicator objects are necessary. This also holds for computations using a single partition only. The Communicator class definition is listed below.

```
class Communicator
{
public:
   Communicator(int argc, char *argv[]);
   virtual ~Communicator();
   void set_output(std::ostream &os);
   int id() const;
   int size() const;

   // global reduction
   void allreduce(int n, double *in, double *out, MPI_Op op);

   // modification of send / receive buffers
   template<class T> void put(int dest, const T& content);
   template<class T> void get(int source, T& content);
   template<class T> void put(int dest, const T* content, int num);
   template<class T> void get(int source, T* content, int num);

   // communication with other processes
   void send_request(int dest);
   void receive_request(int source);
   virtual void start_communication(const char comment[] = "");
   virtual bool finish_communication();
};
```

The Communicator class is the most important class for all objects that communicate over process boundaries. This holds for almost all non trivial tasks in a parallel environment. The Communicator is responsible for communication, basic I/O operations and memory management for send and receive buffers. Thus, it is a comfortable wrapper for *MPI_ Communicator*s that hides the growing and shrinking of message buffers from the user.

The constructor takes the variables argc and argv that come from the main method. These variables are passed to the underlying **MPI_Init()** method to setup the parallel environment. Actually not all MPI implementations make use of these variables to build up the environment.

The destructor calls the method **MPI_Finalize()** that closes the parallel environment. The method **id()** returns the number of the local process and the method **size()** returns the total number of processes in the parallel environment.

In order to send some data from process p_s to p_d the user calls one of the **put(dest, ...)** methods with dest=p_d to fill the send buffer with the data (all memory management is done automatically) and calls **send_request**(p_d) on the source process p_s. Process p_d is aware of the message that it will receive from process p_s and calls **receive_request**(p_s). Both processes call the methods **start_communication()** and **finish_communication()**. After that the data resides in the receive buffer of process p_d and can be read by using the **get(source, ...)** methods with source=p_s.

The communication is split into the two methods **start_communication()** and **finish_communication()** to allow for computation during the communication phase. The user must not touch the send and receive buffers during the communication phase, i.e., until the method **finish_communication()** has been called.

Global reduction operations like **allreduce()** are also available. These are simply wrappers to the MPI equivalents and they are used in the same way.

C.2 Triang(1,2,3)d Classes

The Triang1d, Triang2d and Triang3d classes represent the underlying simplicial meshes for one, two and three space dimensions respectively. They are inherited from the template class Triang<int d> and share the same code except the parts concerning the mesh generators. The meshes can contain *nonconformities of level one.* The access to neighbor cells is accomplished by a STL style *intersection iterator* provided by the Simplex class (not shown in the code section). The refinement and coarsening of the meshes is done as described in Chapter 8.

We start with the description of the Triang2d class.

```
class Triang2d
{
public:
   Triang2d(Communicator &comm);
   virtual ~Triang2d();

   // I/O
   void read_triangle_files(const char basename[]);
   void write(const char filename[]);
   void read(const char filename[]);

   // modifications
   void partition();
   void repartition();
   void reorder();
   void refine_all();
   void coarsen_all();
   void adaption(std::set<int> &refine, std::set<int> &coarsen);
};
```

The constructor takes a reference to the communicator as argument. Communication with other partitions is established using this object.

Modifications of an existing mesh can be done by the methods **partition()** and **repartition()** these methods are wrappers to the corresponding methods in the ParMETIS library [68], [121] and are used to distribute or redistribute the mesh cells over the processes in the parallel environment (Load balancing). These methods additionally provide all data structures the ParMETIS library uses to generate a partition of the underlying mesh.

The method **reorder()** provides a (local) Cuthill-McKee ordering of the mesh. This can speed up the convergence process of iterative solvers due to a better ordering of the unknowns but it can also speed up explicit solvers because of the reduction of *cache mismatches*.

Refinement and coarsening of the mesh is done either globally using the methods **refine_all()** and **coarsen_all()** or each cell can be selected individually by storing the identification number of the cells in a refinement and coarsening list and pass these lists to the method **adaption(refinement, coarsening)**. A cell in the refinement list is guaranteed to be refined, a cell in the coarsening list is only coarsened if this is *possible*, see Section 8.4 for details.

The class Triang2d provides an STL style iterator class to access each cell of the mesh in a sequential way. Random access of mesh cells is also possible. The methods that are necessary to accomplish these tasks are not shown in the class definition above for simplicity.

All data that is associated with the mesh by registering is also partitioned, repartitioned, reordered and refined/coarsened automatically when one of these methods is called.

For input and output of the native mesh format the methods **read()** and **write()** can be used. The output results in a raw binary format file that is not compatible between different machine architectures (byte order is important). The file can be read back using the **read()** method. The number of processes that wrote that file and the number of processes that read this file back do not need to match.

Triang2d objects can be constructed using the output of the *Triangle* mesh generator [101]. Therefore, the basename of the Triangle files must be passed to the method **read_triangle_files(basename)**. Necessary files are basename.node, basename.ele, basename.edge and basename.neigh. Boundary markers are mapped to negative numbers. In the case of negative boundary markers the numbers are preserved.

```
class Triang3d
{
public:
  ⋮
  // I/O
  void read_tetgen_files(const char basename[]);
  ⋮
};
```

```
class Triang1d
{
public:
  ⋮
  // I/O
  void make(double x0, double x1, int n);
  ⋮
};
```

The Triang3d and Triang1d are very similar to the Triang2d class. The only difference is the generation of meshes.

In three space dimensions the output of the *TetGen* [105] mesh generator can be used to construct Triang3d objects. The mesh generator uses a similar syntax as the 2d mesh generator Triangle and the output files are read by passing the basename to the method **read_tetgen_files(basename)**. The necessary files are basename.node, basename.ele, basename.face and basename.neigh. As in the 2d case boundary markers are preserved if the numbers are negative otherwise they are mapped to negative numbers.

Mesh generation in one space dimension is an almost trivial task. In this case we provide the method **make(x_0, x_1, n)** that constructs an equidistant mesh of n cells of the interval (x_0, x_1).

C.3 Function Class

```
class Function
{
public:
   virtual void operator()(const double *u, double *f, int i = 0) = 0;
   virtual int dim_of_argument(int i = 0) const = 0;
   virtual int dim_of_value(int i = 0) const = 0;
   double& time();
   double time() const;
};
```

Functions are the central objects in this implementation. The class Function has pure virtual functions that must be overloaded by inherited classes. The member function **operator()(u, f)** takes an argument u and returns the value of the function f. The functions **dim_of_argument()** and **dim_of_value()** return the dimension of the argument and value respectively. A Function object can optionally depend on parameters. One important parameter is the time. The parameter time can be get and set by using the method **time()**.

C.4 Data Classes

The FeData classes are an abstraction of the Discontinuous Galerkin space V_h^d introduced in Section 6.2. The classes provide I/O operations for data, an interface for adaption of data and methods for projection of data and computation of errors.

```
template<int n>
class FeData : public Data
{
public:
   FeData(Triang<n> &mesh);
   FeData(Triang<n> &mesh, int dim_system, int poly_order);
   virtual ~FeData();

   void L2_projection(Function &u);
   double Lp_distance(double p, Function &u);
   void eval(const Simplex<n> &tr, const double x[n], double *result) const;

   // adaptivity
   void adaption(ErrorIndicator &error_indicator);

   // I/O
   void write(const char filename[]) const;
   void read(const char filename[]);
};
```

The FeData class is parameterized by the template argument n which is the dimension of the coordinate system. The constructor takes a simplex mesh Triang<n> (which is the base class of Triang(1,2,3)d respectively) and optionally the dimension of the state space d of V_h^d which is denoted by **dim_system** and the polynomial order of V_h^d **poly_order**. After calling the constructor the data class is registered by the mesh. This means all operations like refinement, coarsening, partitioning, etc., that are applied to the mesh, are implicitly applied to the corresponding data. Data is essentially a vector and is automatically converted (by a conversion operator not shown in the class definition) to double*.

An instance U of the FeData<n> class stores the coefficients that represents some object $u_h \in V_h^d$ of the Finite Element space given by

$$u_h|_{\Delta_j}(\boldsymbol{x}) = \sum_{l=0}^{n_p-1} \varphi^j(\boldsymbol{x}) \begin{pmatrix} \alpha_{l,0}^j \\ \vdots \\ \alpha_{l,d-1}^j \end{pmatrix}, \quad \boldsymbol{x} \in \mathbb{R}^n,$$

on the j-th cells of the underlying mesh (with identity number j) in the following way

$$\alpha_{l,k}^j = U[d \cdot (n_p \cdot j + l) + k], \quad k = 0, \ldots, d-1.$$

Here n_p denotes the number of the local basis polynomials.

The method **L2_projection(u)** provides a L^2 projection of a given function u to the Discontinuous Galerkin space V_h^d. The L^p distance between the data and a given function u for $p \in [1, \infty)$ can be computed using the method **Lp_distance(p, u)**. Quadrature formulas are selected automatically for these methods but can also be explicitly set.

Writing and reading to and from a file is done using the methods **write(filename)** and **read(filename)**. The file filename must be (at least) accessible from the comm.master() process where comm denotes the Communicator.

By the use of an ErrorIndicator object, see Section C.8, the mesh and the data can be refined and coarsened.

C.5 DG Class

The Discontinuous Galerkin class implements the Discontinuous Galerkin space discretization in one, two and three space dimensions. DG class is parameterized by the space dimension and is therefore formally not limited to three space dimensions, but some necessary implementations like basis polynomials are provided for one, two and three space dimensions only.

The input for a m-stage Discontinuous Galerkin scheme is a differential equation of the form

$$\begin{aligned} u^0 &= u, \\ u^1 &+ \mathcal{L}_1^c[u^0] + \mathcal{L}_1^{ncs}[u^0] = 0, \\ u^2 &+ \mathcal{L}_2^c[u^0, u^1] + \mathcal{L}_2^{ncs}[u^0, u^1] = 0, \\ &\vdots \\ u^m &+ \mathcal{L}_m^c[u^0, \dots, u^{m-1}] + \mathcal{L}_m^{ncs}[u^0, \dots, u^{m-1}] = 0. \end{aligned}$$

This is the form we discussed in Section 6.2 generalized by additional nonconservative and source terms. The operators $\mathcal{L}_k^c[u^0, \dots, u^{k-1}]$ for $k = 1, \dots, m$ represent the conservative differential operators of the form

$$\mathcal{L}_k^c[u^0, \dots, u^{k-1}](x) = \sum_{i=1}^{n} \frac{\partial}{\partial x_i} f_i^k(u^0(x), \dots, u^{k-1}(x), x)$$

and the operators $\mathcal{L}_k^{ncs}[u^0, \dots, u^{k-1}]$ denote the nonconservative operators in combination with source terms of the form

$$\mathcal{L}_k^{ncs}[u^0, \dots, u^{k-1}](x) = a_k\left((u^0(x), \dots, u^{k-1}(x)), \nabla(u^0(x), \dots, u^{k-1}(x)), x\right),$$

where the functions a_k are linear in the gradients. More precisely a_k have the form

$$\begin{aligned} a_k(u, \nabla u, x) &= B(u, x) + \sum_{i=1}^{n} A_i^k(u) \frac{\partial}{\partial x_i} u, \\ u &= (u^0, \dots, u^{k-1}). \end{aligned}$$

All of the above functions can additionally depend on further parameters such as time. The output of a m-stage Discontinuous Galerkin scheme is the function u^m which is the projection of the m-th order differential operator as discussed in Section 6.2.

The variables in the above equations have the following dimensions:

$$x \in \mathbb{R}^n, \quad u^k \in \mathbb{R}^{d_k}, k = 0, \dots, m. \tag{C.1}$$

The DG class has three pure virtual functions namely **flux(...)**, **num_flux(...)** and **bnd_flux(...)** that have to be implemented by the inherited class. The method **flux(...)** implements the physical fluxes f_i^k, nonconservative and source terms a for each of the m stages in the method.

For the complete method all physical fluxes f_i^k and nonconservative terms need associated numerical fluxes g_j^k and g_n^k. Here g_j^k denote the numerical fluxes associated with the current (the j-th) cell and g_n^k the numerical fluxes associated with the corresponding neighbor cell which is in general not $g_n^k = -g_j^k$ since nonconservative terms are also taken into account. The implementation of the numerical fluxes is provided by the method **num_flux(...)**. The numerical fluxes at the boundary of the computational domain and with this the treatment of boundary conditions are provided by the method **bnd_flux(...)**. For more details see the two examples in Section C.9.

The Discontinuous Galerkin class has the following definition, here only the most important member functions are listed.

```
template<int n>
class DG : public Function
{
public:
   DG(Communicator &comm, Triang<n> &mesh, int dim_value,
         int poly_order, int num_stages, const int *dim_flux);

   // from Function
   virtual void operator()(const double *U, double *result, int i=0);
   virtual int dim_of_argument(int i=0) const;
   virtual int dim_of_value(int i=0) const;

   void codegen(char classname[]) const;

protected:
   // fluxes and numerical fluxes
   virtual void flux(int stage, const double *u, const double * const grad_u[n],
                     double *f[n], double *a) = 0;
   virtual void num_flux(int stage, const double *uj, const double *un,
                         const double normal[n], double *gj, double *gn) = 0;
   virtual void bnd_flux(int stage, const double *uj,
                         const double normal[n], double *gj) = 0;
};
```

The DG class is derived from the class Function and therefore it implements the methods **operator()(U, result)**, **dim_of_argument()** and **dim_of_value()**. As a function the **operator()(U, result)** returns the coefficients of the discrete differential operator in the variable **result** and the variable **U** provides the coefficients (with respect to the basis functions of the Finite Element space) of some discrete function.

The constructor of the DG<n> class takes as arguments a reference to a Communicator, a reference to a mesh which can be a one dimensional, two dimensional or three dimensional Simplex grid. With the notation from above, see (C.1), the remaining parameters are given by

$$\begin{aligned}
\textbf{poly\_order} &= \text{polynomial degree of the method,} \\
\textbf{dim\_value} &= d_0, \\
\textbf{dim\_flux[m]} &= \{d_1, d_2, \ldots, d_m\}, \\
\textbf{num\_stages} &= m.
\end{aligned}$$

By default the DG class provides all states u at the integration points and all gradients ∇u that are available in the stages. Very often not all of the states are necessary, especially most gradients are usually not necessary since they are only used in non-conservative products. In order to render the method more efficient certain values can be *unset* but the methods that are necessary for this task are not shown in the class definition for simplicity. The degree of the quadrature formulas can also be chosen freely. By default the degree of the quadrature formulas are chosen according to the recommendation by Cockburn and Shu, see Section 6.4.

For further improvement of the efficiency of the method the method **codegen(...)** can be used to generate highly optimized code. Using generated code can lead to long compilation times.

C.6 ODE Solver Classes

Several ODE solver classes are available to perform time stepping for ordinary initial value problems. The base class for all ODE solvers is the class ODESolver. Available Solver classes belong to the classes of explicit, implicit, semi-implicit Runge-Kutta methods, explicit and implicit Extrapolation methods and SSP methods.

```
class ODESolver
{
public:
   ODESolver(Communicator &comm, int num_tmpobj);
   virtual ~ODESolver();
   void set_limiter(Limiter &limiter);

   // user-interface for solving
   virtual bool step(double t, double dt, double *u) = 0;

protected:
   Limiter *limiter;
};
```

Inherited classes, i.e., implementations of ODE solvers, have to overload the virtual function **step(**t**,** Δt**,** u**)** which performs the time stepping of some data u from time t to time $t + \Delta t$. The method returns true on success or false if it fails to perform the time stepping. In the latter case it is guaranteed that the data u remains unmodified. An explicit method for time stepping is always successful, so it always returns true, but an implicit method can fail to converge in which case it returns false and a smaller time step as to be chosen.

Optionally a *limiter* can be set. A limiter is a function that performs some post processing on the data u in order to maintain the stability of the underlying numerical method. This is usually be done in combination with explicit higher order Runge-Kutta Discontinuous Galerkin schemes.

C.6.1 ExplicitRungeKutta Class

The simplest class of ODE solvers is the class of explicit Runge-Kutta methods discussed in Section 7.2.

The first order one-stage explicit Euler scheme has the following definition.

```
class ExplicitEuler : public ExplicitRungeKutta (: public ODESolver)
{
public:
   ExplicitEuler(Communicator &comm, Function &f);
};
```

The Constructor takes a reference to a Communicator object and the right hand side of the ordinary differential equation given by the Function f. The Function f can be the discrete differential operator constructed by a Discontinuous Galerkin class for example.

For the computation of the evolution in time the method **step(t, dt, u)** from the ODESolver base class is used.

There are several other higher order explicit Runge-Kutta methods available. The class definition of these methods is the same as for the ExplicitEuler class. The following methods are available.

- ExplicitModifiedEuler, 2nd order, 2 stages.
- ExplicitTVD2, 2nd order, 2 stages.
- ExplicitRK3, 3rd order, 3 stages.
- ExplicitTVD3, 3rd order, 3 stages.
- ExplicitRK4, 4th order, 4 stages.
- ExplicitButcher6, 6th order, 7 stages.

For the details concerning these methods see Section 7.2.

C.6.2 ImplicitRungeKutta Class

The class of diagonally implicit Runge-Kutta methods, discussed in Section 7.3, has exactly the same class definition as explicit Runge-Kutta methods except the additional method for the choice of a linear solver. Here for example the definition for the implicit Euler scheme.

```
class ImplicitEuler : public DIRK (: public ODESolver)
{
public:
   ImplicitEuler(Communicator &comm, Function &f);
   void set_linear_solver(IterativeLinearSolver &ls);
};
```

The implicit methods need a linear solver to perform the Newton iteration. This can be set by the method **set_linear_solver(ls)**. Available linear solvers are listed in Section C.7.

Other diagonally implicit Runge-Kutta methods are

- Gauss2 (Crank-Nicholson), 2nd order, 1 stage.
- DIRK3, 3rd order, 2 stages.

C.6.3 SemiImplicitRungeKutta Class

For the semi-implicit Runge-Kutta schemes the constructor takes two Functions as arguments **fex** which is discretized explicitly and **fim** which is discretized in an implicit fashion, see Section 7.4 for details. Again, the implicit part needs a linear solver. The simplest scheme of this class is the first order semi-implicit Euler scheme which has the following definition.

```
class SemiImplicitEuler : public SIRK (: public ODESolver)
{
public:
   SemiImplicitEuler(Communicator &comm, Function &fim, Function &fex);
   void set_linear_solver(IterativeLinearSolver &ls);
};
```

The schemes from Section 7.4 that are available at the moment besides the semi-implicit euler scheme are

- SIRK23, 2nd order, three stages.
- SIRK33 (YZ33), 3rd order, 3 stages.
- IMEX_SPP222, 2nd order, 2 stages.

C.6.4 Other Classes

Additionally to the different kinds of Runge-Kutta classes implicit and explicit extrapolation schemes are also available but not discussed in detail here. The use of this kind of methods is less common in the framework of Finite Volume and Discontinuous Galerkin approximation but they have the advantage that arbitrary order methods can be constructed.

C.7 Linear Solver Classes

The implicit and semi-implicit ODE solver classes use a Newton type nonlinear iteration for solving the corresponding systems of nonlinear equations. Newton type methods need to solve linear systems of equations. In the framework of Finite Element, Finite Volume and Discontinuous Galerkin discretizations of partial differential equations these linear systems are usually large but sparse. Some methods from the class of Krylov space solvers are very efficient methods for these tasks. The Conjugate Gradient (CG) method for symmetric problems belongs to this class. For non symmetric problems the GMRES and BiCGSTAB methods are a good choice.

The GMRES class implements the restarted GMRES algorithm given in [98]. The FGMRES class has exactly the same class definition (therefore we omit it) and implements the flexible variant of the restarted GMRES algorithm [96]. The advantage is that the preconditioner can vary in each step of the iteration. The disadvantage is that it needs twice the amount of memory compared to standard GMRES.

```
class GMRES : public IterativeLinearSolver
{
public:
   GMRES(Communicator &comm, int m);
   virtual ~GMRES();
   virtual void set_preconditioner(Function &preconditioner);
   void set_tolerance(double tol, bool relative = true);
   void set_max_number_of_iterations(int iter);

   // from IterativeLinearSolver, solve Au = b, Au = op(u)
   virtual bool solve(Function &op, double *u, const double *b);
};
```

The constructor takes a Communicator and the Krylov space dimension m as arguments. The choice of the Krylov space dimension is crucial in the GMRES method. The efficiency of the method depends heavily on this parameter. It must not be chosen to small to otherwise the method may fail to converge. A value between 5 and 15 is usually a good choice.

An optional preconditioner, i.e., a function $u \mapsto Mu$, can be set using the method **set_preconditioner(preconditioner)** to speed up the convergence process. The matrix M is chosen to approximate the matrix A^{-1} in some sense.

Using the method **set_tolerance(tol, relative)** a tolerance for the stopping criterion of the iteration can be set. With the boolean value relative=true/false it can be controlled whether this tolerance is interpreted as relative or absolute tolerance. It is a good idea to choose an absolute tolerance in combination with a Newton method because the right hand side of the linear system tends to zero as the Newton method converges. The maximum number of iterations can be controlled using the **set_max_number_of_iterations(iter)** method.

Now, the linear system $Au = b$ is solved using the method **solve(op, u, b)**. The linear operator $u \mapsto Au$ is denoted by op, the right hand side by b. On entry the vector u carries an initial guess of the solution. Upon success the method returns true and the solution of the system is stored in the vector u. If it fails to converge it returns false and it is guaranteed that the initial guess u is not modified in that case.

Another efficient Krylov space method is the BiCGSTAB algorithm given in [113]. The storage requirement of this algorithm does not depend on an extra parameter. Hence, in most cases this method needs less memory than the GMRES method which is important for large scale simulations. The class definition is exactly the same as for the GMRES class (except the parameter m). For completeness we list it below.

```
class BICGSTAB : public IterativeLinearSolver
{
public:
   BICGSTAB(Communicator &comm);
   virtual ~BICGSTAB();
   void set_tolerance(double tol, bool relative = true);
   void set_max_number_of_iterations(int iter);

   // from IterativeLinearSolver, solve Au = b, Au = op(u)
   virtual bool solve(Function &op, double *u, const double *b);
};
```

It depends on the problem whether the GMRES or the BiCGSTAB method performs better. In our test cases we have observed that the GMRES method usually performs about 10-20% better.

For symmetric problems the Conjugate Gradient method (CG) should be used because in this case this algorithm is much more efficient than the other methods for nonsymmetric problems. The interface is exactly the same as for the BiCGSTAB method.

C.8 ErrorIndicator Classes

The ErrorIndicator class is an abstract class and serves as an interface for problem dependent indicators. An inherited class, that implements some indicator or estimator, has to implement the method **operator()(refine, coarsen)**. Here **refine** is the set of cell ids that have to be refined and **coarsen** is the list of cell ids that should be coarsened *if possible*. The class definition follows below.

```
class ErrorIndicator
{
public:
   virtual void operator()(std::set<int> &refine, std::set<int> &coarsen) = 0;
};
```

As an example for an ErrorIndicator we have the space gradient indicator SpaceGradIndicator proposed in Section 8.3. This is actually not an error indicator because it has nothing to do with errors but is used to track the liquid-vapor interface in the simulations using the Navier-Stokes-Korteweg system.

```
template<int dim>
class SpaceGradIndicator : public ErrorIndicator
{
public:
   SpaceGradIndicator(Communicator &comm, Triang<dim> &tr,
                        Data &U, Function &F);
   virtual ~SpaceGradIndicator();
   void set(double eta_low, double eta_upp, int num_iter);

   // from ErrorIndicator
   virtual void operator()(std::set<int> &refine, std::set<int> &coarsen);
};
```

C.9 Examples of Usage

In this section we give two examples of usage of the software package discussed above. The first application is the Discontinuous Galerkin discretization of the linear advection equation in one space dimension. Quite simple for better understanding. The second application is the DG discretization of the Navier-Stokes-Korteweg system using implicit time stepping as we use it in our computations (at some points stripped down a little bit for simplicity).

C.9.1 Example 1: Linear Advection in 1d

The linear advection equation in one space dimensions we consider in this example is given by

$$\begin{aligned} u_t + (su)_x &= 0 \quad \text{in } (-1,1)\times(0,T), \\ u(x,0) &= \begin{cases} u_{left} & \text{if } x<0, \\ u_{right} & \text{else,} \end{cases} \qquad \text{for } x\in(-1,1), \\ u(-1,t) &= u_{left}. \end{aligned}$$

Here we choose $s = 1.0$, $u_{left} = 0.5$ and $u_{right} = 1.0$. The interval $(-1, 1)$ is partitioned into a uniform mesh using the method **Triang1d::make(. . .)**. By default the left boundary of the mesh has the *boundary id* -1 and the right boundary the id -2. The Discontinuous Galerkin method uses the upwind flux as numerical flux and for time integration the explicit third order Runge-Kutta method TVD3 is applied. The complete implementation is given in the following one hundred lines of C++ code.

```
#include <iostream>
#include <cmath>
#include "communicator.hpp"
#include "triang.hpp"
#include "dg.hpp"
#include "ode_solver.hpp"
```

```
#include "data.hpp"


using namespace std;
using namespace pardg;

// global variables
const double s = 1.0;
const double u_left = 0.5;
const double u_right = 1.0;


// initial data
class InitialData :  public Function
{
public:
  virtual void operator()(const double *x, double *result, int i=0)
  {
    const double tau = x[0] - s*time();
    result[0] = (tau < 0)?  u_left :  u_right;
  }

  virtual int dim_of_argument(int i) const { return 1;}
  virtual int dim_of_value(int i) const { return 1;}
};


// Discontinuous Galerkin discretization
const int num_stages1d = 1;
const int dim_value1d = 1;
const int dim_flux1d[num_stages1d] = {1};

class DG1dLinAdv :  public DG<1>
{
public:
  DG1dLinAdv(Communicator &comm, Triang<1> &mesh, int poly_order) :
    DG<1>(comm, mesh, dim_value1d, poly_order, num_stages1d, dim_flux1d)
{}

private:
  // physical flux
  virtual void flux(int stage, const double *u, const double *const grad_u[1],
                    double *f[1], double *a)
  {
    f[0][0] = s * u[0];
  }
  // upwind flux = Lax-Friedrichs flux
  virtual void num_flux(int stage, const double *uj, const double *un,
                        const double n[1], double *gj, double *gn)
  {
    gj[0] = 0.5*n[0]*s*( uj[0]+un[0] ) - 0.5*fabs(s)*( un[0]-uj[0] );
    gn[0] = -gj[0];
  }

  // boundary treatment
  virtual void bnd_flux(int stage, const double *uj, const double n[1],
                        double *gj)
  {
    double ub;
    if (bnd_id == -1) ub = u_left; // left boundary
    else ub = u_right; // right boundary
    gj[0] = 0.5*n[0]*s*( uj[0]+ub ) - 0.5*fabs(s)*( ub-uj[0] );
  }
};


int main(int argc, char *argv[])
{
  // Communicator
  Communicator comm(argc, argv);

  // construct mesh and distribute it over the available processors
  const int n = 200;
  Triang1d mesh(comm);
  mesh.make(-1.0, 1.0, n);
  mesh.partition();
  const double h = mesh.h();

  // setup DG scheme & Runge-Kutta scheme
  const int poly_order = 2;
  DG1dLinAdv dg_linadv(comm, mesh, poly_order);
  ExplicitTVD3 ode_solver(comm, dg_linadv);

  // setup data and projection of initial data
  FeData<1> U(mesh, 1, poly_order);
  InitialData u0;
  U.L2_projection(u0);

  // perform time stepping
  const double T = 0.2;
  const double cfl = 0.45 / (1+2*poly_order); // Cockburn & Shu formula
  double dt = cfl * h / fabs(s);
  double t = 0.0;
  while (t < T){
    ode_solver.step(t, dt, U);
    t += dt;
  }

  // output
  u0.time() = t;
  const double L2error = U.Lp_distance(2.0, u0);
  if ( comm.id() == comm.master() ){
    cout << "L2 error:  "<< L2error << "   h:  "<< h << endl;
  }
}
```

The approximate solution is computed up to computational end time $T = 0.2$. At the end of the computation the L^2-error to the exact solution is computed.

C.9.2 Example 2: Isothermal Navier-Stokes-Korteweg in 2d

In this final example we discuss the implementation of the higher order discretization of the isothermal Navier-Stokes-Korteweg equations in two space dimensions.

$$\begin{aligned} \rho_t + \nabla \cdot (\rho \boldsymbol{u}) &= 0, \\ (\rho \boldsymbol{u})_t + \nabla \cdot (\rho \boldsymbol{u}\boldsymbol{u}^T) + \rho \nabla \kappa &= \nabla \cdot \boldsymbol{\tau}, \end{aligned}$$

where, as usual, $\boldsymbol{\tau}$ denotes the viscous part of the stress tensor and $\kappa = \mu(\rho) - \lambda \Delta \rho$. For this example we have chosen the boundary conditions

$$\boldsymbol{u} = \boldsymbol{0} \quad \text{and} \quad \nabla \rho \cdot \boldsymbol{n} = 0 \quad \text{on } \partial\Omega$$

and as initial data we provide an almost static bubble. Physical and numerical fluxes are implemented as discussed in section 6.9.2. The inherited DG class DG2dNSK is also derived from the class VanDerWaalsIsothermal which provides the equations of state. Time stepping is done using a second order implicit Runge-Kutta scheme (Gauss2/Crank-Nicholson) equiped with the GMRES(15) linear solver. The computational domain $\Omega = (-1, 1)^2$ is represented by a triangular mesh stored in the box2d.1.* files. This mesh constructed by the using the *Triangle* mesh generator [101].

In this example local mesh adaption is omitted for simplicity and the time step size is fixed to some small enough constant. The complete implementation is given by the following 250 lines of C++ code.

```
#include <iostream>
#include <cmath>
#include "communicator.hpp"
#include "triang.hpp"
#include "dg.hpp"
```

```
#include "ode_solver.hpp"
#include "data.hpp"
#include "vdw.hh"



using namespace std;
using namespace pardg;

// global variables
static const double T_ref = 0.85;      // reference temperature
static const double lambda = 0.001;   // capillarity
static const double eps = 0.0136644; // viscosity
static const double nu = 0.75*eps;    // viscosity


// initial data, bubble of radius R with center 0
class Bubble :  public Function
{
public:
  virtual void operator()(const double *x, double *result, int i=0)
  {
    const double width = 5.4*T_ref*T_ref * sqrt(lambda); // appr. formula
    const double r0 = R - 0.5*width;
    const double r1 = R + 0.5*width;
    const double r = sqrt(x[0]*x[0] + x[1]*x[1]);

    // density
    if (r < r0) result[0] = rho_v;
    else if(r < r1) {
      const double phi = (2.0*(r-r0)/(r1-r0) - 1.0) * M_PI/2.0;
      result[0] = 0.5*(rho_v+rho_l) + 0.5*(rho_l-rho_v) * tanh(tan(phi));
    }
    else result[0] = rho_l;

    // momentum
    result[1] = result[2] = 0.0;
  }

  virtual int dim_of_argument(int i) const { return 2;}
  virtual int dim_of_value(int i) const { return 3;}

private:
  static const double rho_v = 0.3;
  static const double rho_l = 1.8;
  static const double R = 0.3;
};
```

```
// Discontinuous Galerkin discretization
const int num_stages2d = 3;
const int dim_value2d = 3;
const int dim_flux2d[num_stages2d] = {4, 1, 3};

class DG2dNSK : public DG<2>, public VanDerWaalsIsothermal
{
public:
  DG2dNSK(Communicator &comm, Triang<2> &mesh, int poly_order) :
    DG<2>(comm, mesh, dim_value2d, poly_order, num_stages2d, dim_flux2d),
    VanDerWaalsIsothermal(T_ref)
  {
    // some std values
    alpha_1 = 0.648676;
    alpha_2 = 1.86921;
  }

private:
  virtual void flux(int stage, const double *u, const double *const grad_u[2],
                    double *f[2], double *a);
  virtual void num_flux(int stage, const double *uj, const double *un,
                        const double n[2], double *gj, double *gn);
  virtual void bnd_flux(int stage, const double *uj, const double n[2],
                        double *gj);

  double alpha_1, alpha_2;
};


// physical flux
void DG2dNSK::flux(int stage, const double *u, const double *const grad_u[2],
                   double *f[2], double *a)
{
  // u[0]=rho, u[1]=rho_u,   u[2]=rho_v,
  // u[3]=rho_x, u[4]=rho_y, u[5]=u_x+v_y, u[6]=u_y-v_x,
  // u[7]=kappa
  if (stage == 0){ // reconstruct 1st derivatives
    f[0][0] = -u[0];
    f[0][1] = 0.0;
    f[0][2] = -u[1]/u[0];
    f[0][3] = u[2]/u[0];

    f[1][0] = 0.0;
    f[1][1] = -u[0];
    f[1][2] = -u[2]/u[0];
    f[1][3] = -u[1]/u[0];
  }
  else if (stage == 1){ // reconstruct kappa
    f[0][0] = lambda*u[3];
    f[1][0] = lambda*u[4];

    a[0] = -potential(u[0]);
  }
  else if (stage == 2){ // evaluate flux
    const double rho = u[0];
    const double rho_u = u[1];
    const double rho_v = u[2];
    const double ru_rv_r = rho_u*rho_v/rho;
    const double ux_vy = u[5];
    const double uy_vx = u[6];

    f[0][0] = u[1];
    f[0][1] = rho_u*rho_u/rho - eps*ux_vy;
    f[0][2] = ru_rv_r + nu*uy_vx;

    f[1][0] = u[2];
    f[1][1] = ru_rv_r - nu*uy_vx;
    f[1][2] = rho_v*rho_v/rho - eps*ux_vy;

    a[1] = rho * grad_u[0][7];
    a[2] = rho * grad_u[1][7];
  }
}


// numerical flux
void DG2dNSK::num_flux(int stage, const double *uj, const double *un,
                       const double n[2], double *gj, double *gn)
{
  // u[0]=rho, u[1]=rho_u,   u[2]=rho_v,
  // u[3]=rho_x, u[4]=rho_y, u[5]=u_x+v_y, u[6]=u_y-v_x,
  // u[7]=kappa
  if (stage == 0){ // reconstruct 1st derivatives
    const double rho_j = uj[0];
    const double rho_n = un[0];
    const double u_j = uj[1]/rho_j;
    const double u_n = un[1]/rho_n;
    const double v_j = uj[2]/rho_j;
    const double v_n = un[2]/rho_n;

    gj[0] = -0.5*(rho_n+rho_j) * n[0];
    gj[1] = -0.5*(rho_n+rho_j) * n[1];
    gj[2] = -0.5*(u_n+u_j)*n[0] - 0.5*(v_n+v_j)*n[1];
    gj[3] = 0.5*(v_n+v_j)*n[0] - 0.5*(u_n+u_j)*n[1];

    gn[0] = -gj[0];
    gn[1] = -gj[1];
    gn[2] = -gj[2];
    gn[3] = -gj[3];
  }
  if (stage == 1){ // reconstruct kappa
    gj[0] = lambda * 0.5*( (un[3]+uj[3])*n[0] + (un[4]+uj[4])*n[1]);
    gn[0] = -gj[0];
  }
  if (stage == 2){ // eval flux
    const double rho_j = uj[0];
    const double rho_n = un[0];
    const double rho_u_j = uj[1];
    const double rho_u_n = un[1];
    const double rho_v_j = uj[2];
    const double rho_v_n = un[2];
    const double ru_rv_r_j = rho_u_j * rho_v_j / rho_j;
    const double ru_rv_r_n = rho_u_n * rho_v_n / rho_n;
    const double kappa_j = uj[7];
    const double kappa_n = un[7];
    const double ux_vy_n = un[5];
    const double ux_vy_j = uj[5];
    const double uy_vx_n = un[6];
    const double uy_vx_j = uj[6];

    gj[0] = 0.5*( (rho_u_j + rho_u_n)*n[0] + (rho_v_j + rho_v_n)*n[1] )
      -0.5*alpha_1*(kappa_n - kappa_j);

    gj[1] = 0.5*( (rho_u_j*rho_u_j/rho_j + rho_u_n*rho_u_n/rho_n)*n[0]
                  +(ru_rv_r_j + ru_rv_r_n)*n[1] )
      -0.5*alpha_2*(rho_u_n - rho_u_j)
      -0.5*eps*(ux_vy_n+ux_vy_j)*n[0] - 0.5*nu*(uy_vx_n+uy_vx_j)*n[1];

    gj[2] = 0.5*( (ru_rv_r_j + ru_rv_r_n)*n[0]
                  +(rho_v_j*rho_v_j/rho_j + rho_v_n*rho_v_n/rho_n)*n[1] )
      -0.5*alpha_2*(rho_v_n - rho_v_j)
      +0.5*nu*(uy_vx_n+uy_vx_j)*n[0] - 0.5*eps*(ux_vy_n+ux_vy_j)*n[1];

    gn[0] = -gj[0];
    gn[1] = -gj[1];
    gn[2] = -gj[2];

    const double jump = 0.25*(rho_j + rho_n) * (kappa_n - kappa_j);
    gj[1] += jump*n[0];
    gn[1] += jump*n[0];
    gj[2] += jump*n[1];
    gn[2] += jump*n[1];
  }
}


// boundary treatment
void DG2dNSK::bnd_flux(int stage, const double *uj, const double n[2],
                       double *gj)
{
  // u[0]=rho, u[1]=rho_u,   u[2]=rho_v,
  // u[3]=rho_x, u[4]=rho_y, u[5]=u_x+v_y, u[6]=u_y-v_x,
  // u[7]=kappa
  if (stage == 0){ // reconstruct 1st derivatives
    const double rho_j = uj[0];
    const double u_j = uj[1]/rho_j;
    const double v_j = uj[2]/rho_j;

    gj[0] = -rho_j * n[0];
    gj[1] = -rho_j * n[1];
    gj[2] = -0.5*(u_j*n[0] + v_j*n[1] + 0.0);
    gj[3] = -0.5*(u_j*n[0] - v_j*n[1] + 0.0);
  }
  else if (stage == 1){ // reconstruct kappa
    gj[0] = 0.0;
  }
  if (stage == 2){ // eval flux
    const double ux_vy = uj[5];
    const double uy_vx = uj[6];

    gj[0] = 0.0;
    gj[1] = -eps*ux_vy*n[0] - nu*uy_vx*n[1];
    gj[2] = nu*uy_vx*n[0] - eps*ux_vy*n[1];
  }
}



int main(int argc, char *argv[])
{
  // Communicator
  Communicator comm(argc, argv);

  // read mesh and distribute it over the available processors
  Triang2d mesh(comm);
  mesh.read_triangle_files("./box2d.1");
  mesh.partition();

  // setup DG scheme
  const int poly_order = 2;
  DG2dNSK dg_nsk(comm, mesh, poly_order);

  // Linear Solver
  GMRES linear_solver(comm, 15);
  linear_solver.set_tolerance(1.0e-6, false);

  // Implicit Runge-Kutta method
  Gauss2 ode_solver(comm, dg_nsk);
  ode_solver.set_linear_solver(linear_solver);

  // setup data and projection of initial data
  FeData<2> U(mesh, 3, poly_order);
  Bubble u0;
  U.L2_projection(u0);

  // perform time stepping
  const double T = 0.1;
  double dt = 1.0e-4; // small enough timestep
  double t = 0.0;
  while (t < T){
    cout << comm.id() << "  "<< t << endl;
    bool convergence = ode_solver.step(t, dt, U);
    assert(convergence);
    t += dt;
  }
}
```

The approximate solution is computed up to computational end time $T = 0.1$. Nothing is done with the approximate solution. It would be more convenient to write the approximate solution to a data file at some points in computational time but this is omitted here for simplicity.

Bibliography

[1] D.M. Anderson, G.B. McFadden, and A.A. Wheeler. Diffuse interface methods in fluid mechanics. *Ann. Rev. Fluid Mech.*, 30:139–165, 1998.

[2] L.K. Antanovskii. Microscale theory of surface tension. *Phys. Rev.*, E54:6285–6290, 1996.

[3] D. N. Arnold, F. Brezzi, B. Cockburn, and L. D. Marini. Unified analysis of discontinuous Galerkin methods for elliptic problems. *SIAM J. Numer. Anal.*, 39(5):1749–1779, 2002.

[4] H. L. Atkins and C.-W. Shu. Quadrature-free implementation of discontinuous galerkin method for hyperbolic equations. Technical Report TR-96-51, 1996.

[5] V. Babin and R. Holyst. Condensation of a vapor bubble in submicrometer container. *J. Chem. Phys.*, 123, 2005.

[6] G. Bader and U. Ascher. A new basis implementation for a mixed order boundary value ode solver. *SIAM J. Sci. Stat. Comput.*, 8:483–500, 1987.

[7] J.M. Ball. Material instabilities and the calculus of variations. In *Phase transformations and material instabilities in solids, Proc. Conf.*, pages 1–19. Madison/Wis. 1983, Publ. Math. Res. Cent. Univ. Wis. Madison 52, 1984.

[8] F. Bassi and S. Rebay. A high-order accurate discontinuous finite element method for the numerical solution of the compressible Navier-Stokes equations. *J. Comput. Phys.*, 131(2):267–279, 1997.

[9] C. E. Baumann and J. T. Oden. A discontinuous *hp* finite element method for convection-diffusion problems. *Comput. Methods Appl. Mech. Eng.*, 175(3-4):311–341, 1999.

[10] M. Beckers and A. Haegemans. The construction of cubature formulae for the tetrahedron, 1990. Report TW 128, Dept. of Computer Science, K.U. Leuven.

[11] S. Benzoni-Gavage. *Contributions à l'étude des solutions régulières par morceaux des systèmes hyperboliques de lois de conservation.* Habilitation, Lyon I, 1998.

[12] S. Benzoni-Gavage. Stability of subsonic planar phase boundaries in a van der Waals fluid. *Arch. Ration. Mech. Anal.*, 150:23–55, 1999.

[13] M. J. Berger and Colella P. Local Adaptive Mesh Refinement for Shock Hydrodynamics. *J. Comput. Phys.*, 82(1):64–84, 1989.

[14] L.S. Blackford, J. Demmel, J. Dongarra, I. Duff, S. Hammarling, G. Henry, M. Heroux, L. Kaufman, A. Lumsdaine, A. Petitet, R. Pozo, K. Remington, and R.C. Whaley. An updated set of basic linear algebra subprograms (blas). *ACM Trans. Math. Soft.*, 28(2):135–151, 2002.

[15] T. Blesgen. *Eine Verallgemeinerung der Navier-Stokes-Gleichungen auf Zweiphasenströmungen.* Doctoral dissertation, Rheinische Friedrich–Wilhelms–Universität Bonn, 1997.

[16] C. E. Brennen. *Cavitation and bubble dynamics.* Oxford University Press, 1995.

[17] D. Bresch, B. Desjardins, and C.-K. Lin. On some compressible fluid models: Korteweg, lubrication, and shallow water systems. *Commun. Partial Differ. Equations*, 28(3-4):843–868, 2003.

[18] Greg Burns, Raja Daoud, and James Vaigl. LAM: An Open Cluster Environment for MPI. In *Proceedings of Supercomputing Symposium*, pages 379–386, 1994.

[19] A. Burri, A. Dedner, D. Diehl, R. Kloefkorn, and M. Ohlberger. A general object oriented framework for discretizing nonlinear evolution equations. In *Advances in High Performance Computing and Computational Sciences. The 1st Kazakh-German Advanced Research Workshop, Almaty, Kazakhstan, September 25 - October 1, 2005.* Springer Series: Notes on Numerical Fluid Mechanics and Multidisciplinary Design (NNFM) , Vol. 93. Shokin, Y.I.; Resch, M.; Danaev, N.; Orunkhanov, M.; Shokina, N. (Eds.), 2006.

[20] Paul Castillo. Performance of discontinuous Galerkin methods for elliptic PDEs. *SIAM J. Sci. Comput.*, 24(2):524–547, 2002.

[21] Cockburn and C.-W. Shu. TVB Runge-Kutta local projection discontinuous Galerkin finite element method for conservation laws II: General framework. *Math. Comp.*, 82, 1989.

[22] Cockburn and C.-W. Shu. The Runge-Kutta local projection P^1-discontinuous Galerkin finite element method for scalar conservation laws. M^2AN, 25, 1991.

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

[24] B. Cockburn, S. Y. Lin, and C.-W. Shu. TVB Runge-Kutta local projection discontinuous Galerkin finite element method for conservation laws III: One dimensional systems. *J. Comput. Phys.*, 84, 1989.

[25] B. Cockburn and C.-W. Shu. The Local Discontinuous Galerkin method for time-dependent convection-diffusion systems. *SIAM J. Numer. Anal.*, 35(6):2440–2463, 1998.

[26] B. Cockburn and C.-W. Shu. TVB Runge-Kutta local projection discontinuous Galerkin finite element method for conservation laws V: Multidimensional systems. *J. Comput. Phys.*, 141:199–224, 1998.

[27] B. Cockburn and C.-W. Shu. Runge-Kutta Discontinuous Galerkin methods for convection-dominated problems. *J. Sci. Comput.*, 16(3):173–261, 2001.

[28] R. Colls. An encyclopaedia of cubature formulas. *J. Complexity*, 19:445–453, 2003.

[29] F. Coquel, D. Diehl, C. Merkle, and C. Rohde. Sharp and diffuse interface methods for phase-transition problems in liquid-vapour flows. In *E. Sonnendrücker (Ed.), Numerical Methods for Hyperbolic and Kinetic Problems.* IRMA Series, 2004.

[30] F. Coquel, D. Diehl, and C. Rohde. Static equilibrium solutions of the Navier-Stokes-Korteweg system and relaxation schemes. 2004. Preprint.

[31] F. Coquel and B. Perthame. Relaxation of energy and approximate Riemann solvers for general pressure laws in fluid dynamics. *SIAM J. Numer. Anal.*, 35(6):2223–2249, 1998.

[32] R. Courant and D. Hilbert. *Methoden der mathematischen Physik. (Methods of mathematical physics). 4. Aufl.* Springer, 1993.

[33] J.K. Cullum and M. Tuma. Matrix-free preconditioning using partial matrix estimation. *BIT*, 46(4):711–729, 2006.

[34] C.M. Dafermos. *Hyperbolic Conservation Laws in Continuum Physics.* Springer, Berlin, 2000.

[35] Leonardo Dagum. Openmp: A proposed industry standard api for shared memory programming. http://www.openmp.org/mp-documents/paper/paper.ps, October 1997.

[36] G. Dal Maso, P.G. LeFloch, and F. Murat. Definition and weak stability of nonconservative products. *J. Math. Pures Appl.*, 74(6):483–548, 1995.

[37] J. F. Dannenhofer and J. R. Baron. Grid Adaption for the 2–D Euler Equations. *AIAA Paper 85–0484*, 1985.

[38] A. Dedner, C. Makridakis, and M. Ohlberger. Error control for a class of Runge Kutta Discontinuous Galerkin methods for nonlinear conservation laws. *Preprint 05-01, Mathematisches Institut, Freiburg*, 2005.

[39] D. Diehl and C. Rohde. On the Structure of MHD Shock Waves in Diffusive-Dispersive Media. *Journal of Mathematical Fluid Mechanics*, 8:120–145, February 2006.

[40] D.A. Dunavant. High degree efficient symmetrical gaussian quadrature rules for the triangle. *Int. J. Numer. Methods Eng.*, 21:1129–1148, 1985.

[41] J.E. Dunn and J. Serrin. On the thermomechanics of interstitial working. *Arch. Ration. Mech. Anal.*, 88:95–133, 1985.

[42] M. Feistauer. *Mathematical methods in fluid dynamics.* Longman Scientific & Technical, Harlow, 1993.

[43] H. Freistühler and C. Rohde. Numerical computation of viscous profiles for hyperbolic conservation laws. *Math. Comput.*, 71(239):1021–1042, 2002.

[44] H. Freistühler, C. Fries, and C. Rohde. Existence, bifurcation, and stability of profiles for classical and non-classical shock waves. In *B. Fiedler (Ed.), Ergodic Theory, Analysis, and Efficient Simulation of Dynamical Systems*, pages 287–309. Springer, Heidelberg, 2001.

[45] H. Frenzel and H. Schultes. *Z. Phys. Chem.*, B27(421), 1934.

[46] Edgar Gabriel, Graham E. Fagg, George Bosilca, Thara Angskun, Jack J. Dongarra, Jeffrey M. Squyres, Vishal Sahay, Prabhanjan Kambadur, Brian Barrett, Andrew Lumsdaine, Ralph H. Castain, David J. Daniel, Richard L. Graham, and Timothy S. Woodall. Open MPI: Goals, concept, and design of a next generation MPI implementation. In *Proceedings, 11th European PVM/MPI Users' Group Meeting*, pages 97–104, Budapest, Hungary, September 2004.

[47] R. Geisler, T. Kurz, and W. Lauterborn. Acoustic bubble traps. In *W. Lauterborn, T. Kurz (eds.), Nonlinear Acoustics at the Turn of the Millennium, Proceedings of the 15th Int. Symp. on Nonlinear Acoustics,ISNA 15, 1-4 September, 1999, Göttingen, Germany.* AIP Conference Proceedings 524 (2000), American Institute of Physics, Melville (2000), 2000.

[48] R. Homepage Geisler. http://www.macgeisler.de/.

[49] G. A. Geist, J. A. Kohla, and P. M. Papadopoulos. PVM and MPI: A Comparison of Features. *Calculateurs Paralleles*, 8(2):137–150, 1996.

[50] T. Geßner. *Dynamic Mesh Adaption for Supersonic Combustion Waves modeled with Detailed Reaction Mechanisms.* Doctoral dissertation, Albert–Ludwigs–Universität Freiburg, 2001.

[51] E. Godlewski and P.-A. Raviart. *Hyperbolic Systems of Conservation Laws.* Ellipses, 1991.

[52] E. Godlewski and P.-A. Raviart. *Numerical Approximation of Hyperbolic Systems of Conservation Laws.* Springer, 1996.

[53] W. Gropp, E. Lusk, N. Doss, and A. Skjellum. A high-performance, portable implementation of the MPI message passing interface standard. *Parallel Computing*, 22(6):789–828, September 1996.

[54] W. Gropp, E. Lusk, and A. Skjellum. *Using MPI (2nd ed.): portable parallel programming with the message-passing interface.* MIT Press, Cambridge, MA, USA, 1999.

[55] W. Gropp, R. Thakur, and E. Lusk. *Using MPI-2: Advanced Features of the Message Passing Interface.* MIT Press, Cambridge, MA, USA, 1999.

[56] M. E. Gurtin. On the structure of equilibrium phase transitions within the gradient theory of fluids. *Q. Appl. Math.*, 46(2):301–317, 1988.

[57] H. Hattori and D. Li. Solutions for two-dimensional system for materials of Korteweg type. *SIAM J. Math. Anal.*, 25(1):85–98, 1994.

[58] H. Hattori and D. Li. The existence of global solutions to a fluid dynamic model for materials of Korteweg type. *J. Partial Differ. Equations*, 9(4):323–342, 1996.

[59] Magnus R. Hestenes and Eduard Stiefel. Methods of conjugate gradients for solving linear systems. *J. Res. Natl. Bur. Stand.*, 49:409–436, 1952.

[60] T.Y. Hou and P.G. LeFloch. Why nonconservative schemes converge to wrong solutions: Error analysis. *Math. Comput.*, 62(206):497–530, 1994.

[61] L. Howle, D.G. Schaeffer, M. Shearer, and P. Zhong. Lithotripsy: The treatment of kidney stones with shock waves. *SIAM Rev.*, 40(2):356–371, 1998.

[62] G. M.-K. Hui and H. Swann. On orthogonal polynomial bases for triangles and tetrahedra invariant under the symmetric group. In *[CA] Mandel, Jan (ed.) et al., Domain decomposition methods 10. The 10th international conference, Boulder, CO, USA, August 10–14, 1997. Providence.* RI: AMS, American Mathematical Society. Contemp. Math. 218, 438-446, 1998.

[63] M.A. Hulsen. The discontinuous Galerkin method with explicit Runge-Kutta time integration for hyperbolic and parabolic systems with source terms. Technical report, Delft University of Technology, 1991.

[64] D. Jamet, O. Lebaigue, N. Coutris, and J.M. Delhaye. The second gradient method for the direct numerical simulation of liquid-vapor flows with phase change. *J. Comput. Phys.*, 169(2):624–651, 2001.

[65] D. Jamet, D. Torres, and J.U. Brackbill. On the theory and computation of surface tension: The elimination of parasitic currents through energy conservation in the second-gradient method. *J. Comput. Phys.*, 182(1):262–276, 2002.

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

[67] V. Jovanovic and C. Rohde. Error estimates for finite volume approximations of classical solutions for nonlinear systems of hyperbolic balance laws. *SIAM J. Numer. Anal.*, 43(6):2423–2449, 2006.

[68] G. Karypis and V. Kumar. Parallel multilevel k-way partitioning scheme for irregular graphs. *SIAM Review*, 41(2):278–300, 1999.

[69] G. Karypis, K. Schloegel, and V. Kumar. ParMETIS: Parallel graph partitioning and sparse matrix ordering library. University of Minnesota, 2003.

[70] P. Keast. Moderate-degree tetrahedral quadrature formulas. *Comput. Methods Appl. Mech. Engrg.*, 55:339–348, 1986.

[71] D.A. Knoll and D.E. Keyes. Jacobian-free Newton-Krylov methods: A survey of approaches and applications. *J. Comput. Phys.*, 193(2):357–397, 2004.

[72] D.J. Korteweg. Sur la forme que prennent les équations du mouvement des fluides si l'on tient compte des forces capillaires causées par des variations de densité considérables mais continues et sur la théorie de la capillarité dans l'hypothèse d'une variation continue de la densité. *Arch. Néerl. (2)*, 6:1–24, 1901.

[73] M. Kotschote. Strong well-posedness for a Korteweg type model for the dynamics of a compressible non-isothermal fluid. *Preprint, Mathematisches Institut, Leipzig.*

[74] M. Kotschote. Strong solutions for a compressible fluid model of Korteweg type. *Preprint, Mathematisches Institut, Leipzig*, 2006.

[75] C. Kraus and W. Dreyer. The sharp interface limit of the van der waals–cahn–hilliard phase model for fixed and time dependent domains. *WIAS Preprint No. 1103*, 2006.

[76] D. Kröner. *Numerical Schemes for Conservation Laws.* Verlag Wiley & Teubner, Stuttgart, 1997.

[77] D. Kröner and M. Ohlberger. A Posteriori Error Estimates for Upwind Finite Volume Schemes for Nonlinear Conservation Laws in Multi Dimensions. *Math. Comput.*, 69:25–39, 2000.

[78] L.D. Landau and E.M. Lifshitz. *Statistical Physics*, volume 5 of *Course of Theoretical Physics.* Pergamon Press, London, 1958.

[79] L.D. Landau and E.M. Lifshitz. *Hydrodynamik*, volume 4 of *Lehrbuch der Theoretischen Physik.* Akademie Verlag, 1991.

[80] P. LeSaint and P.A. Raviart. On a finite element method for solving the neutron transport equation. Math. Aspects finite Elem. partial Differ. Equat., Proc. Symp. Madison 1974, 89-123 (1974), 1974.

[81] R. LeVeque. *Numerical Methods for Conservation Laws.* Birkhäuser, Basel, 1990.

[82] GNU Multiple Precision Arithmetic Library. http://www.swox.com/gmp/.

[83] D. Lockard and H. Atkins. Efficient implementations of the quadrature-free discontinous galerkin method, 1999.

[84] P. R. McHugh and D. A. Knoll. Comparison of standard and matrix-free implementations of several Newton- Krylov solvers. *AIAA J.*, 32(12):2394–2400, 1994.

[85] C. Merkle. *Phase Transitions in Compressible Media, a Sharp–Interface Model.* Doctoral dissertation, Albert–Ludwigs–Universität Freiburg, 2006.

[86] L. Modica. The gradient theory of phase transitions and the minimal interface criterion. *Arch. Ration. Mech. Anal.*, 98:123–142, 1987.

[87] I. Müller. *Grundzüge der Thermodynamik: mit historischen Anmerkungen.* Springer, 1999.

[88] Encyclopaedia of Cubature Formulas website. http://www.cs.kuleuven.ac.be/ nines/research/ecf/.

[89] M. Ohlberger. A Posteriori Error Estimates for Finite Volume Approximations to Singularly Perturbed Nonlinear Convection–Diffusion Equations. *Numer. Math.*, 87(4):737–761, 2001.

[90] M. Ohlberger and J. Vovelle. Error Estimate for the Approximation of Non–Linear Conservation Laws on Bounded Domains by the Finite Volume Method. *Math. Comput.*, 75(253):113–150, 2006.

[91] L. Pareschi and G. Russo. Implicit-explicit Runge-Kutta schemes and applications to hyperbolic systems with relaxation. *Journal of Scientific Computing*, 25(1):129–155, 2005.

[92] W.H. Press, S.A. Teukolsky, W.T. Vetterling, and B.P. Flannery. *Numerical recipes in C++. The art of scientific computing. 2nd ed.* Cambridge University Press, 2002.

[93] Lord Rayleigh. On the pressure developed in a liquid during the collapse of a spherical cavity. *Phil. Mag.*, 34(199), 1917.

[94] W.H. Reed and T.R. Hill. Triangular mesh methods for the neutron transport equation. Technical Report LA-UR-73-479, Los Alamos National Laboratory, 1973.

[95] C. Rohde. *Approximation of Solutions of Conservation Laws by Non-Local Regularization and Discretization.* Habilitation, Albert–Ludwigs–Universität Freiburg, 2004.

[96] Y Saad. A flexible inner-outer preconditioned gmres algorithm. *SIAM J. Sci. Comput.*, 14(2):461–469, 1993.

[97] Y. Saad. *Iterative methods for sparse linear systems. 2nd ed.* Philadelphia, PA: SIAM Society for Industrial and Applied Mathematics. xviii, 528 p. $ 89.00 , 2003.

[98] Y. Saad and M.H. Schultz. Gmres: A generalized minimal residual algorithm for solving nonsymmetric linear systems. *SIAM J. Sci. Stat. Comput.*, 7:856–869, 1986.

[99] E. Sainz de la Maza and J.I. Maeztu. An invariant quadrature rule of degree 11 for the tetrahedron. *C. R. Acad. Sci., Paris, Sér. I*, 321(9):1263–1267, 1995.

[100] D. Serre. *Systems of Conservation Laws.* Cambridge University Press, 1999.

[101] J.R. Shewchuk. Triangle: Engineering a 2D Quality Mesh Generator and Delaunay Triangulator. In Ming C. Lin and Dinesh Manocha, editors, *Applied Computational Geometry: Towards Geometric Engineering*, volume 1148 of *Lecture Notes in Computer Science*, pages 203–222. Springer-Verlag, May 1996. From the First ACM Workshop on Applied Computational Geometry.

[102] C.-W. Shu. Different formulations of the Discontinuous Galerkin method for the viscous terms.

[103] C.-W. Shu and S. Osher. Efficient implementation of essentially nonoscillatory shock-capturing schemes. *J. Comput. Phys.*, 77(2):439–471, 1988.

[104] C.-W. Shu and S. Osher. Efficient implementation of essentially nonoscillatory shock-capturing schemes. II. *J. Comput. Phys.*, 83(1):32–78, 1989.

[105] H. Si. TetGen, a quality tetrahedral mesh generator and three-dimensional delaunay triangulator. Technical report, Berlin, Germany: Res. Group Numer. Math. Sci. Comput., Weierstrass Inst. Appl. Anal. Stochast., 2005.

[106] M. Slemrod. Admissibility criteria for propagating phase boundaries in a van der Waals fluid. *ARCRM*, 81:301–315, 1983.

[107] J. Stoer. *Numerische Mathematik 1.* Springer, Berlin, Heidelberg, New York, 1999.

[108] K. Strehmel and R. Weiner. *Numerik gewöhnlicher Differentialgleichungen.* Teubner Studienbücher: Mathematik. Stuttgart: Teubner. 462 p., 1995.

[109] A.H. Stroud. *Approximate calculation of multiple integrals.* Prentice-Hall Series in Automatic Computation. Englewood Cliffs, New Jersey: Prentice-Hall, Inc. XIII, 1971.

[110] I. Suliciu. On modelling phase transitions by means of rate type constitutive equations. *Int. J.Engng. Sci.*, 28:827–841, 1990.

[111] E.F. Toro. *Riemann solvers and numerical methods for fluid dynamics. A practical introduction. 2nd ed.* Berlin: Springer, 1999.

[112] D. Trescher. *Development of an efficient 3-D CFD software to simulate and visualize the scavenging of a two-stroke engine.* Doctoral dissertation, Albert–Ludwigs–Universität Freiburg, 2005.

[113] H.A. van der Vorst. BiCGSTAB: A fast and smoothly converging variant of BiCG for the solution of non-symmetric linear systems. *SIAM Journal on Scientific and Statistical Computing*, 13, 1992.

[114] J.D. van der Waals. *Die Kontinuität des gasförmigen und flüssigen Zustandes.* PhD thesis, Leiden, 1873.

[115] J.D. van der Waals. Thermodynamische Theorie der Kapillarität unter Voraussetzung stetiger Dichteänderung. *Z. Phys. Chem.*, 13:657–725, 1894.

[116] R. Vilsmeier and D. Hänel. Adaptive Methods on Unstructured Grids for Euler and Navier–Stokes Equations. *Computers Fluids*, 22(4/5):485–499, 1993.

[117] N.J. Walkington. Quadrature on simplices of arbitrary dimension. Technical report, Carnegie Mellon University, Pittsburgh, 2000.

[118] ATLAS website. http://math-atlas.sourceforge.net/.

[119] BLAS website. http://www.netlib.org/blas/.

[120] LAM/MPI website. http://www.lam-mpi.org/.

[121] METIS/ParMETIS website. http://www-users.cs.umn.edu/ karypis/metis/.

[122] MPI-Forum website. http://www.mpi-forum.org/.

[123] MPICH website. http://www-unix.mcs.anl.gov/mpi/mpich1/.

[124] MPICH2 website. http://www-unix.mcs.anl.gov/mpi/mpich/.

[125] NIST website. http://webbook.nist.gov/chemistry/fluid/.

[126] OpenMP website. http://www.openmp.org/.

[127] OpenMPI website. http://www.open-mpi.org/.

[128] PVM website. http://www.csm.ornl.gov/pvm/.

[129] R.C. Whaley, A. Petitet, and J.J. Dongarra. Automated empirical optimization of software and the ATLAS project. *Parallel Computing*, 2(1-2):3–25, 2001.

[130] J. Yan and C.-W. Shu. A local discontinuous galerkin method for kdv type equations. *SIAM Journal on Numerical Analysis*, 40:769–791, 2002.

[131] J. J. Yoh and X. Zhong. New hybrid Runge-Kutta methods for unsteady reactive flow simulation. *AIAA Journal*, 42(8):1593–1600, 2004.

[132] J. J. Yoh and X. Zhong. New hybrid Runge-Kutta methods for unsteady reactive flow simulation: Applications. *AIAA Journal*, 42(8):1600–1611, 2004.

[133] Q. Zhang and C.-W. Shu. Error estimates to smooth solutions of Runge-Kutta discontinuous Galerkin methods for scalar conservation laws. *SIAM J. Numer. Anal.*, 42(2):641–666, 2004.

[134] S. Zhang. Successive subdivisions of tetrahedra and multigrid methods on tetrahedral meshes. *Houston J. Math.*, 21(3):541–556, 1995.