Numerical methods for three-dimensional incompressible two-phase flow problems

Von der Fakultät für Mathematik, Informatik und Naturwissenschaften der RWTH Aachen University zur Erlangung des akademischen Grades eines Doktors der Naturwissenschaften genehmigte Dissertation

vorgelegt von

Diplom-Mathematiker Sven Groß

aus Georgsmarienhütte

Berichter: Universitätsprofessor Dr. Arnold Reusken

Universitätsprofessor Dr.-Ing. Wolfgang Marquardt

Tag der mündlichen Prüfung: 18. Juli 2008

Diese Dissertation ist auf den Internetseiten der Hochschulbibliothek online verfügbar.

Für Aga und Hanna

Acknowledgements

The present thesis originates from my work at the IGPM (Institut für Geometrie und Praktische Mathematik) at the RWTH Aachen University and was financially supported by the DFG (German Research Foundation) through the Collaborative Research Center SFB 540.

I would like to express my deep gratitude to my advisor Prof. Dr. Arnold Reusken for introducing me to the fascinating research field of two-pase flows and for his continuous support throughout my PhD studies. His unlimited positiveness and great knowledge guided me through all impassibilities on my way. I also sincerely thank my co-advisor and the head of the SFB 540 Prof. Dr.-Ing. Wolfgang Marquardt for the excellent interdisciplinary education during the seminars and countless team meetings of the SFB. The opportunity to work on such challenging research topics originating from real-life applications was always a big motivation for me.

Many thanks go to the whole staff of the SFB 540 for the excellent interdisciplinary work, many fruitful discussions and for learning so many interesting things besides mathematics. Especially I want to mention our weekly meeting of C6 & friends (Adel Mhamdi, Marcus Soemers, Maka Karalashvili and Yi Heng): thank you for your friendship, many scientific and non-scientific discussions and the delicious coffee at the AVT.PT.

I want to emphasize how much I enjoyed the encouragement and help of my colleagues and the good mood at the IGPM over all the years. In particular I'm very grateful to the core development team of DROPS, Jörg Grande, Volker Reichelt, Oliver Fortmeier and Patrick Esser, for the cooperative and pleasant working atmosphere and many interesting discussions on software design decisions and the art of standard-compliant programming.

Finally I would like to thank my friends and my family for their friendly encouragement and support. My special thanks go to my wonderful wife Agnes for her never-ending love and patience and to my lovely daughter Hanna, both making everyday's life happy.

Abstract

In this thesis a numerical approach for the simulation of three-dimensional incompressible two-phase flows is presented. It is based on a level set method for capturing the interface. The mathematical model consists of the incompressible Navier-Stokes equations and an advection equation for the level set function. The effect of surface tension is modeled by a singular force term located at the interface.

For the spatial discretization we use finite elements on a nested hierarchy of tetrahedral grids. An adaptive multilevel refinement algorithm allows for local refinement and coarsening of the grid hierarchy. By partial integration of the Laplace-Beltrami operator the weak formulation of the surface tension force term can be stated in such a way that second derivatives induced by the curvature can be avoided. It is shown that a standard Laplace-Beltrami discretization on a piecewise planar approximation of the interface only yields an order of $1/2$ w. r. t. the H^1 norm, and on the other hand that by a slight modification this order can be increased up to a value of at least 1. The pressure distribution is continuous in both phases, respectively, but has a jump across the interface due to surface tension. The approximation of such functions in standard finite element spaces yields poor results with an order of $1/2$ w. r. t. the L_2 norm. The introduction of an extended finite element (XFEM) space provides second order approximations. For this purpose a standard finite element space is augmented by additional basis functions incorporating a jump at the interface.

For the time discretization a one-step theta-scheme is applied which leads to a coupled system of level set and Navier-Stokes equations. The coupling can be treated by a Picard iteration. By applying a linearized variant of the theta-scheme the equations can be decoupled. The nonlinearity of the Navier-Stokes equations is handled by a fixed point approach. The arising Oseen problems are solved by an inexact Uzawa method or by Krylov subspace methods, where problem-adapted preconditioners are applied which account for the jump of the material properties between both phases. For the reparametrization of the level set function a Fast Marching method is used.

The methods have been implemented in the software package DROPS. The structure of the code and basic design concepts are briefly discussed. We also consider parallelization aspects, as the consumption of memory resources and computational time are typically huge for complex problems such as two-phase flows.

The correct implementation and the accuracy of several numerical components is analyzed by means of some test cases. Finally, examples originating from droplet and falling film applications are considered. These two-phase systems play an important role in chemical engineering processes and are some of the major research topics in the collaborative research center SFB 540 at the RWTH Aachen University. Some numerical results for simulations of levitated droplets, rising bubbles and a falling film are presented.

Zusammenfassung

In der vorliegenden Arbeit wird ein Ansatz zur numerischen Behandlung von inkompressiblem dreidimensionalen Zweiphasenströmungen vorgestellt, der auf einer Levelset-Methode zur Verfolgung der Phasengrenze basiert. Die Modellgleichungen bestehen aus den inkompressiblen Navier-Stokes-Gleichungen sowie einer Advektionsgleichung für die Levelset-Funktion. Die Oberflächenspannung wird durch einen singulären Kraftterm modelliert, der auf der Phasengrenze lokalisiert ist.

Zur örtlichen Diskretisierung werden Finite Elemente auf einer geschachtelten Hierarchie von Tetraedergittern eingesetzt. Ein adaptiver Multilevel-Verfeinerungsalgorithmus ermöglicht die lokale Ver- und Entfeinerung der Gitterhierarchie. Der Oberflächenspannungsterm wird in der schwachen Formulierung durch partielle Integration des Laplace-Beltrami-Operators in eine Form überführt, in der zweite Ableitungen vermieden werden, die durch die Krümmung hervorgerufen werden. Es wird gezeigt, dass mit einer Standard-Laplace-Beltrami-Diskretisierung auf einer stückweise planaren Approximation der Phasengrenze nur eine Annäherung der Ordnung 1/2 bzgl. der H^1-Norm erreicht werden kann, durch eine leichte Modifikation die Ordnung dagegen auf mindestens 1 erhöht werden kann. Der Druck ist in beiden Phasen jeweils stetig, besitzt aber aufgrund der Oberflächenspannung einen Sprung an der Phasengrenze. Die Approximation solcher Funktionen ist in Standard-Finite-Elemente-Räumen nur mit Ordnung 1/2 bzgl. der L_2-Norm möglich. Die Einführung eines erweiterten Finite-Elemente-Raumes (XFEM) ermöglicht eine Approximation zweiter Ordnung. Hierbei werden zusätzliche Basisfunktionen hinzugefügt, die einen Sprung an der Phasengrenze aufweisen.

Zur Zeitdiskretisierung kommt ein Theta-Schema zu Einsatz, das auf ein gekoppeltes System von Levelset- und Navier-Stokes-Gleichungen führt. Dies kann mit einer Picard-Iteration gelöst werden. Durch eine linearisierte Variante des Theta-Schemas kann eine Entkopplung der Gleichungen erreicht werden. Die Nichtlinearität der Navier-Stokes-Gleichungen wird durch einen Fixpunktansatz behandelt. Die auftretenden Oseen-Probleme werden durch eine inexakte Uzawa-Methode oder durch Krylov-Teilraumverfahren gelöst, wobei problemangepasste Vorkonditionierungstechniken zum Einsatz kommen, die den Sprung der Stoffdaten von der einen Phase in die andere berücksichtigen. Zur Reparametrisierung der Levelset-Funktion wird eine Fast-Marching-Methode verwendet.

Die Methoden wurden in dem Software-Werkzeug DROPS implementiert, dessen Struktur und zugrundeliegendes Design kurz dargestellt werden. Dabei

wird auch auf Parallelisierungsaspekte eingegangen, da die Speicher- und Rechenzeitanforderungen für solch komplexe Probleme wie Zweiphasenströmungen enorm groß sein können.

An einigen Testbeispielen wird die korrekte Implementierung und Genauigkeit einiger numerischer Komponenten überprüft. Schließlich werden Anwendungsbeispiele aus dem Bereich von Tropfen- und Filmsystemen behandelt, die Gegenstand der Forschung in dem verfahrenstechnisch ausgerichteten Sonderforschungsbereich SFB 540 der RWTH Aachen University sind. Dabei werden numerische Ergebnisse von Simulationen levitierter Tropfen, aufsteigender Tropfen sowie eines Fallfilmes präsentiert.

Contents

1. Introduction

Due to the advances in computer technology and improvements of numerical methods in the last decades, the simulation of *one-phase* flows in liquid or gaseous media has become state of the art and is applied as a standard tool in industry and research. However, a closer look at the simulation of *two-phase* flow problems, such as rising air or oil droplets in water, clearly reveals a different picture. One major problem is the occurrence of numerically induced oscillations of the velocity field in the vicinity of the interface, so-called *spurious currents* [WKP99, FCD+06]. We mention some other challenges in the context of two-phase flow problems:

- the different material properties of the two phases inducing a jump in the coefficients of the partial differential equations,
- the singular surface tension force term which is only defined at the interface,
- topological changes of the interface like the break-up or merging of bubbles.

As the numerical methods for two-phase problems are not yet mature in many respects, for example w. r. t. accuracy, there is an active field of current research on the improvement of existing methods or the design of new approaches.

In this thesis we describe a numerical strategy for the simulation of three-dimensional incompressible two-phase flow problems. It is based on a level set method for capturing the interface and a finite element discretization on adaptive multilevel tetrahedral grids. The main achievements of this thesis are the development and analysis of two novel methods for the numerical treatment of surface tension which feature a higher accuracy compared to standard methods in this field:

- a modified Laplace-Beltrami discretization of the singular surface tension force term which is localized at the interface,

- an extended finite element (XFEM) space for the pressure to represent the jump across the interface which is present due to surface tension.

1.1. Two-phase systems in chemical engineering applications

In this section we give a few examples of two-phase flow systems which are of interest in the area of chemical engineering and are a topic of current research in the Collaborative Research Center SFB 540 [SFB] at the RWTH Aachen University. In the SFB 540 several groups from different scientific disciplines such as chemical engineering, chemistry, physics, scientific computing and mathematics collaborate to gain more insight into kinetic phenomena arising in multi-phase systems. Among them some groups are working on numerical simulation (which is also the focus of our group), while others are conducting experiments to collect measurement data or developing adequate models for the description of the observed phenomena.

The goal of the SFB 540 is to enhance the modeling of momentum, heat and mass transport in multiphase systems, in which interfacial phenomena often play a dominant role. This is accomplished by the formulation and solution of inverse problems, which aim to match the measurement data with the simulation results as good as possible. In the most simple case this means the fitting of a few model parameters, in more elaborate cases the inverse problem consists in estimating an *unknown function* [GSM+05, KGM+08]. Questions of model structure and model identification or optimal experimental design (i. e., roughly speaking, which experiment gives most information), which are arising in this context, are also considered on the basis of inverse problems. This integrated modeling process is called 'model-based experimental analysis' [Mar05], or MEXA for short, which explains the title of the SFB 540: '**M**odel-based **ex**perimental **a**nalysis of kinetic phenomena in fluid multi-phase reactive systems'.

The main contribution from our research group is the development of the software package DROPS for the simulation of three-dimensional incompressible two-phase flow problems. Many of the numerical methods implemented in DROPS are described in Part II of this thesis. The software code is written in C++ and developed by a couple of people at the Chair of Numerical Mathematics and the Chair of Scientific Computing at the RWTH Aachen University. We refer to Chapter 9 for a compact overview of the design and

Figure 1.1.: Photo of a levitated droplet in a measurement device. The photo was taken applying an exposure time of 3 seconds to show the stability of the droplet. Provided by project B3, SFB 540.

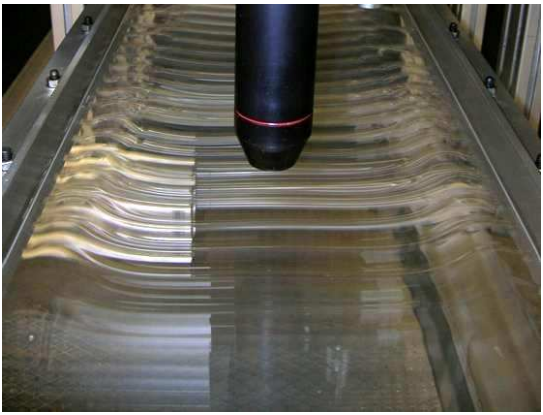

Figure 1.2.: Photo showing the surface of a falling film. The measurement device at the top is used for measuring the local film thickness. Provided by Georg Dietze, project C2, SFB 540.

structure of the DROPS code.

We are interested in the following two-phase systems, which are of interest in the chemical engineering community: The first one is a bubble in a surrounding fluid. This is related to bubble column reactors, where mass transport takes place across the phase interface of bubble swarms. For investigation purposes a single bubble is considered, the chosen substances are silicon oil, n-butanol or toluene in water. In a special measurement device, where the bubble can be held in a stable fixed position, NMR measurements of momentum (and mass) transfer are operated by our project partners [AGHK+05]. A photo of the levitated bubble in the device is shown in Figure 1.1. The goal is to improve the interface model by precise measurement in the vicinity of the interface and accurate numerical simulation of the system. The latter is our task. Some numerical results of the hydrodynamics of bubbles are presented in the Sections 11.1 and 11.2.

The second system we are interested in is the flow of thin liquid films, which are occurring in falling film apparatuses. In applications these are mainly used for heating, cooling and evaporation processes. The liquid film is flowing down an inclined wall and develops a wavy structure at the interface to the

gaseous phase, even without external excitation, see Figure 1.2. This waviness enhances heat and mass transport and is therefore an interesting field of investigation. Our project partners are measuring important parameters such as film thickness, velocity information at certain points or planar sections, surface temperature and temperature and concentration distribution inside the film by means of several measurement techniques [LASLR05, SMDK06]. The liquid phase in our examples is chosen as water or silicon oil, the gaseous phase consists of air or nitrogen. Since the material properties of this liquid-gas system differ by a factor of roughly 10^3 and the interface is very large compared to the bubble experiment, the numerical simulation is very challenging in this case. Some first results of numerical simulations are given in Section 11.3.

1.2. Numerical approach

The complexity of the aforementioned application examples defines the challenges one is facing when treating such phenomena numerically. This will give us a guideline for the development process of our numerical method, i. e., in the choice and combination of adequate numerical tools. Some of the key issues are listed below:

The transport phenomena are essentially 3D, hence we cannot restrict to 2D or 3D rotationally symmetric models. We therefore need the ability to handle *three-dimensional* incompressible flows. The high complexity of 3D problems demands the usage of *parallel computers*, otherwise a sufficient grid resolution can often not be achieved on a single processor due to memory limitations and/or huge computational times.

Many important transport phenomena occur at the interface demanding a high resolution in the interfacial region. Otherwise the development of reliable interface models is not possible. Hence we have to apply *adaptivity* locally at the interface, combined with *load distribution* in the case of parallelization.

The interface is moving in time, therefore we have to deal with a *non-stationary problem* and need some *interface localization* technique. Also the grid has to be adapted from time to time if the interface tends to move out of the refinement zone.

Surface forces are dominant. In the case of the levitated droplet, surface tension is high as the curvature is large due to the small bubble diameter. In the case of the falling film capillary forces are dominant because of

the large extent of the interface. Hence we need a *special numerical treatment of surface tension* to avoid spurious currents at the interface or keep them as small as possible.

Discontinuous pressure. In the presence of surface tension there is a pressure jump at the interface due to the Laplace-Young law. For the approximation of discontinuous functions we introduce *extended finite element ansatz functions* which are discontinuous across the interface.

Large jumps in coefficients of the PDE have to be handled, at least for the falling film problem. This demands *special quadrature* techniques for integrals with discontinuous integrands in the discretization process. *Special preconditioning* techniques for the Schur complement matrix have to be applied for the solution of the discrete problem to account for the jumping coefficients.

Based on these requirements and properties of two-phase flow problems, we have chosen several numerical methods which are in our opinion appropriate for this task. In the following we list the main ingredients of our numerical strategy:

- The level set method is applied for capturing the interface between the two phases. This method is also capable of describing topology changes of the interface.
- The spatial discretization is based on a hierarchy of three-dimensional tetrahedral grids which are constructed in such a way that they are consistent (i. e., no hanging nodes) and that the hierarchy of triangulations is stable. Local refinement and coarsening are easy to realize.
- For the discretization of level set and Navier-Stokes equations we use conforming P_2 finite elements for the velocity $\mathbf{u}$ and level set function φ as well as extended finite elements (XFEM) for the pressure p. The evaluation of integrals with discontinuous integrands arising during the assembly of the system matrices are calculated by special quadrature techniques which account for the position of the interface.
- We use a Laplace-Beltrami technique for the discretization of the surface tension force term, which avoids second derivatives induced by the interfacial curvature. By a slight modification the accuracy of the discretization can be significantly increased compared to standard Laplace-Beltrami approaches on piecewise planar interface approximations.
- The one-step theta-scheme or a linearized variant of it is applied for time integration.

- In each time step the nonlinearity of the discrete Navier-Stokes problem is treated by a fixed point defect correction. The Oseen problems are solved by an inexact Uzawa method or Krylov subspace methods, where we use special Schur complement preconditioning techniques accounting for the piecewise constant material properties ρ and μ.
- The Fast Marching method is used for reparametrization of the level set function φ.
- Most of the numerical components have been parallelized to enable the simulation of complex two-phase flow problems with sufficient resolution in affordable computational time.

1.3. Outline of the thesis

The thesis is structured in three parts describing the *mathematical model*, the *numerical methods* applied (constituting the largest part) and some *numerical results*.

In Part I the governing equations of motion for one-phase and two-phase flow are defined, cf. Chapter 2. In Section 2.2 we briefly discuss different methods to describe the unknown interface. We use the level set method, where the interface is described as the zero-level of a scalar function, the so-called level set function, cf. Section 2.2.1.

In Part II all numerical components are presented which are part of the overall numerical strategy. A short outline of the applied numerical methods is given at the beginning of Part II. We use a hierarchy of nested tetrahedral grids, a so-called multilevel triangulation, and a multilevel refinement algorithm for locally refining and coarsening the grid, cf. Chapter 3. The finite element discretization of the level set and Navier-Stokes equations is described in Chapter 4. Due to surface tension forces for the discretization of two-phase flow problems, some special aspects have to be taken into account, which are highlighted in Chapter 5. In Section 5.3 the discretization of the singular surface tension force term by a Laplace-Beltrami technique is analyzed. For the discretization of the discontinuous pressure an extended finite element space (XFEM) is applied, which is described in Section 5.4. Topics of Chapter 6 are the time discretization and coupling of level set and Navier-Stokes equations. The iterative solution of the discrete problems and corresponding preconditioning aspects are addressed in Chapter 7. Chapter 8 is concerned with the reparametrization of the level-set function and a simple volume correction

strategy to enforce conservation of mass. The structure of the software code DROPS as well as certain implementational aspects including the parallelization of some components are described in Chapter 9.

Finally, in Part III we present some results obtained by the simulation tool DROPS. In Chapter 10 the performance of several numerical components is investigated by means of specific test cases. Numerical results of 3D incompressible two-phase flow problems for real two-phase systems originating from droplet and falling film applications are given in Chapter 11.

In Chapter 12 we summarize the results of this thesis, draw some conclusions and formulate several research topics relevant for future work.

Part I.

Mathematical model

2. Governing equations

2.1. A continuous model for two-phase flow

Let Ω be a polyhedral domain in $\mathbb{R}^3$ and $[t_0, t_f]$ a time interval. In the following we derive the Navier-Stokes equations for unsteady laminar flow of two immiscible fluids. We assume the fluids to be incompressible, viscous, Newtonian and pure (i. e., no mixture of different components). Moreover we assume isothermal conditions for both fluids and therefore neglect variations of density ρ and dynamic viscosity μ due to temperature changes. Hence, μ and, due to incompressibility, also ρ are constant (and positive) in each phase.

2.1.1. One-phase flow

We first consider the Navier-Stokes equations for unsteady laminar flow of *one phase*. Let $\mathbf{u} = \mathbf{u}(\mathbf{x}, t) \in \mathbb{R}^3$ and $p = p(\mathbf{x}, t) \in \mathbb{R}$ denote velocity and pressure, respectively. We introduce a function

$$\mathbf{X} : \Omega \times [t_0, t_f] \to \Omega$$

with the following meaning. For a particle with the initial spatial position $\mathbf{x}_0 \in \Omega$ at the initial time t_0, $\mathbf{X}(\mathbf{x}_0, t)$ describes the spatial position of the particle at the time t. This is the Eulerian description of motion in spatially fixed coordinates. By definition, $\mathbf{X}(\mathbf{x}_0, t_0) = \mathbf{x}_0$ for all $\mathbf{x}_0 \in \Omega$ and

$$\mathbf{X}_t(\mathbf{x}_0, t) := \frac{d}{dt}\mathbf{X}(\mathbf{x}_0, t) = \mathbf{u}(\mathbf{X}(\mathbf{x}_0, t), t)$$

as the particles are moving with velocity $\mathbf{u}$. Let $W_0 \subset \Omega$ be an arbitrary, bounded subset and

$$W(t) := \{\, \mathbf{X}(\mathbf{x}_0, t) : \mathbf{x}_0 \in W_0 \,\}.$$

$W(t)$ describes the position of the particles for time t, which were located in W_0 at initial time t_0. Then for a C^1 function $f = f(x, t)$ the following

transport theorem holds: For all $t \in [t_0, t_f]$,

$$\frac{d}{dt}\int_{W(t)} f(\mathbf{x},t)\,d\mathbf{x} = \int_{W(t)} f_t(\mathbf{x},t) + \operatorname{div}(f\,\mathbf{u})(\mathbf{x},t)\,d\mathbf{x}. \tag{2.1}$$

Considering conservation of mass we choose $f = \rho$ and obtain

$$0 = \frac{d}{dt}\int_{W(t)} \rho\,d\mathbf{x} = \int_{W(t)} \rho_t + \operatorname{div}(\rho\mathbf{u})\,d\mathbf{x}.$$

Since $W(t)$ is arbitrary, this is equivalent to

$$\rho_t + \operatorname{div}(\rho\mathbf{u}) = 0$$

for all $(\mathbf{x},t) \in \Omega \times [t_0, t_f]$. Due to our assumption $\rho = const$ this simplifies to

$$\operatorname{div}\mathbf{u} = 0. \tag{2.2}$$

(2.2) is also called *continuity equation*.

We now consider conservation of momentum. The momentum of mass contained in $W(t)$ is given by

$$M(t) = \int_{W(t)} \rho\mathbf{u}\,d\mathbf{x}.$$

Due to Newton's law the change of momentum $M(t)$ is equal to the force $F(t)$ acting on $W(t)$. This force is decomposed in a volume force $F_1(t)$ and a boundary force $F_2(t)$. The external volume force $F_1(t)$ is given by the gravitational force,

$$F_1(t) = \int_{W(t)} \rho\mathbf{g}\,d\mathbf{x},$$

where $\mathbf{g} \in \mathbb{R}^3$ is the vector of gravitational acceleration. The boundary force $F_2(t)$ is modeled by the surface integral

$$F_2(t) = \int_{\partial W(t)} \boldsymbol{\sigma}\mathbf{n}\,ds,$$

where $\boldsymbol{\sigma} = \boldsymbol{\sigma}(\mathbf{x},t) \in \mathbb{R}^{3\times 3}$ is the stress tensor and $\mathbf{n} = \mathbf{n}(\mathbf{x},t) \in \mathbb{R}^3$ the outer normal on $\partial W(t)$. Summarizing, Newton's law yields

$$\begin{aligned}\frac{d}{dt}M(t) &= F_1(t) + F_2(t) \\ &= \int_{W(t)} \rho\mathbf{g} + \operatorname{div}\boldsymbol{\sigma}\,d\mathbf{x},\end{aligned} \tag{2.3}$$

where we applied Stokes' theorem for $F_2(t)$. Using the transport theorem (2.1) in the left-hand side of (2.3) with $f = \rho\, u_i$, $i = 1, 2, 3$, we obtain

$$\int_{W(t)} (\rho\, u_i)_t + \operatorname{div}(\rho\, u_i\, \mathbf{u})\, d\mathbf{x} = \int_{W(t)} \rho\, \mathbf{g} + \operatorname{div} \boldsymbol{\sigma}_i\, d\mathbf{x}, \qquad i = 1, 2, 3,$$

with $\boldsymbol{\sigma}_i$ the i-th row of $\boldsymbol{\sigma}$. In vector notation,

$$\int_{W(t)} (\rho\, \mathbf{u})_t + \operatorname{div}(\rho\, \mathbf{u} \otimes \mathbf{u})\, d\mathbf{x} = \int_{W(t)} \rho\, \mathbf{g} + \operatorname{div} \boldsymbol{\sigma}\, d\mathbf{x}.$$

Since $W(t)$ is arbitrary, this is equivalent to

$$(\rho\, \mathbf{u})_t + \operatorname{div}(\rho\, \mathbf{u} \otimes \mathbf{u}) = \rho\, \mathbf{g} + \operatorname{div} \boldsymbol{\sigma}$$

for all $(\mathbf{x}, t) \in \Omega \times [t_0, t_f]$. Note that $\operatorname{div}(\rho\, \mathbf{u} \otimes \mathbf{u}) = (\rho\, \mathbf{u} \cdot \nabla)\mathbf{u} + (\rho\, \mathbf{u})(\operatorname{div} \mathbf{u})$ and due to the continuity equation (2.2), the last summand vanishes, yielding the so-called *momentum equation*

$$(\rho\, \mathbf{u})_t + (\rho\, \mathbf{u} \cdot \nabla)\mathbf{u} = \rho\, \mathbf{g} + \operatorname{div} \boldsymbol{\sigma}. \tag{2.4}$$

For viscous Newtonian fluids the stress tensor $\boldsymbol{\sigma}$ is modeled as

$$\boldsymbol{\sigma} = -p\mathbf{I} + \mu\, \mathbf{D}(\mathbf{u})$$

where $\mathbf{D}(\mathbf{u}) = \nabla\mathbf{u} + (\nabla\mathbf{u})^T$ is the deformation tensor. Summarizing the equations from above, we end up with the well-known Navier-Stokes equations for incompressible flow:

$$\rho\Big(\frac{\partial \mathbf{u}}{\partial t} + (\mathbf{u} \cdot \nabla)\mathbf{u}\Big) - \operatorname{div}(\mu \mathbf{D}(\mathbf{u})) + \nabla p = \rho \mathbf{g}, \tag{2.5}$$

$$\operatorname{div} \mathbf{u} = 0 \qquad \text{in } \Omega \times [t_0, t_f]. \tag{2.6}$$

Remark 2.1
If μ is constant (which is the case for isothermal one-phase flows of a pure substance) then the term $\operatorname{div}\big(\mu \mathbf{D}(\mathbf{u})\big)$ simplifies to

$$\operatorname{div}\big(\mu \mathbf{D}(\mathbf{u})\big) = \mu \operatorname{div}\big(\mathbf{D}(\mathbf{u})\big) = \mu\big(\Delta \mathbf{u} + \nabla(\operatorname{div} \mathbf{u})\big) = \mu\, \Delta \mathbf{u}$$

taking into account that $\operatorname{div} \mathbf{u} = 0$. ◇

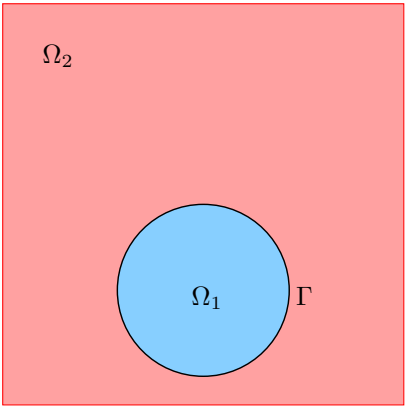

Figure 2.1.: 2D illustration of the computational domain Ω consisting of two phases Ω_1 and Ω_2 and interface Γ.

2.1.2. Two-phase flow

We now consider two-phase flows, i. e., Ω contains two different immiscible incompressible phases (fluid-fluid or fluid-gas) which are moving in time and have different material properties ρ_i and μ_i, $i = 1, 2$. Therefore we assume that for each point in time, $t \in [t_0, t_f]$, Ω is partitioned into two subdomains $\Omega_1(t)$ and $\Omega_2(t)$, $\overline{\Omega} = \overline{\Omega}_1(t) \cup \overline{\Omega}_2(t)$, $\Omega_1(t) \cap \Omega_2(t) = \emptyset$, each of them containing one of the phases, respectively. Both phases are separated from each other by the interface $\Gamma(t) = \overline{\Omega}_1(t) \cap \overline{\Omega}_2(t)$, cf. Figure 2.1. As mentioned before, we assume isothermal conditions and that both phases are pure substances. Moreover, we do *not* consider reaction, mass transfer or phase transition.

In each of the phases conservation of mass and momentum has to hold, yielding separate Navier-Stokes equations on the two domains $\Omega_i, i = 1, 2$. Additionally we have to consider transition conditions at the interface. As the phases are viscous and no phase transition is taking place, the velocity can be assumed to be continuous at the interface:

$$[\mathbf{u}]_\Gamma = 0. \tag{2.7}$$

Here for $\mathbf{x} \in \Gamma$ and a function f defined on Ω we use the notation

$$[f]_\Gamma(\mathbf{x}) := f_1(\mathbf{x}) - f_2(\mathbf{x}), \qquad f_i(\mathbf{x}) := \lim_{\xi \to \mathbf{x}} f(\xi) \text{ in } \Omega_i,\ i = 1, 2.$$

For the interface force we choose a standard model incorporating surface tension, i. e., we assume that the jump of the normal stress along the interface

Γ is proportional to the local curvature $\kappa = \kappa(\mathbf{x}), \mathbf{x} \in \Gamma$, cf., for example, [BKZ92, Scr60]:

$$[\boldsymbol{\sigma}\mathbf{n}]_\Gamma = \tau\kappa\mathbf{n}. \tag{2.8}$$

This is a free boundary condition at the interface. Here $\mathbf{n}$ denotes the outer normal on Γ pointing from Ω_1 to Ω_2. The curvature is defined by

$$\kappa(\mathbf{x}) = -\operatorname{div}\mathbf{n}(\mathbf{x}), \qquad \mathbf{x} \in \Gamma,$$

thus for a convex interior of Γ we have the convention that κ is *negative*. τ is called the surface tension coefficient, which is a material property of the two-phase system. In combination with conservation of mass and momentum in each phase this yields the following standard model for two-phase flows:

$$\begin{cases} \rho_i(\dfrac{\partial \mathbf{u}}{\partial t} + (\mathbf{u}\cdot\nabla)\mathbf{u}) = \rho_i\mathbf{g} + \operatorname{div}\boldsymbol{\sigma}_i \\ \operatorname{div}\mathbf{u} = 0 \end{cases} \quad \text{in } \Omega_i \times [t_0, t_f], \quad i = 1,2, \tag{2.9}$$

$$[\boldsymbol{\sigma}\mathbf{n}]_\Gamma = \tau\kappa\mathbf{n}, \quad [\mathbf{u}]_\Gamma = 0, \tag{2.10}$$

initial condition $\mathbf{u}|_{t=t_0} = \mathbf{u}_0$ in Ω,

suitable boundary conditions at $\partial\Omega$.

The density and viscosity, ρ_i and $\mu_i, i = 1,2$, are assumed to be constant in each phase.

Instead of two separate Navier-Stokes equations on the computational domains Ω_i, $i = 1,2$, and additional interface conditions it would be advantageous to consider an equivalent system of PDEs on the whole domain Ω. In fact, the so-called continuum surface force (CSF) model (cf. [BKZ92, CHMO96]) is such a model. It consists of the Navier-Stokes equation on Ω with jumping coefficients ρ, μ,

$$\rho = \begin{cases} \rho_1 & \text{in } \Omega_1, \\ \rho_2 & \text{in } \Omega_2, \end{cases} \qquad \mu = \begin{cases} \mu_1 & \text{in } \Omega_1, \\ \mu_2 & \text{in } \Omega_2. \end{cases} \tag{2.11}$$

The free boundary condition (2.8) is expressed in terms of a localized force term f_Γ, which appears on the right-hand side of the momentum equation. The CSF term f_Γ is given by

$$f_\Gamma = \tau\kappa\,\delta_\Gamma\mathbf{n}_\Gamma, \tag{2.12}$$

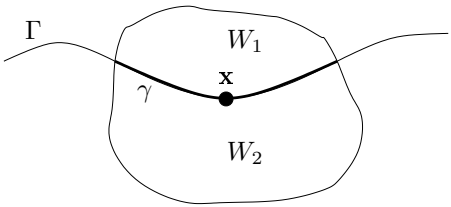

Figure 2.2.: 2D illustration of a neighborhood $W = W_1 \cup W_2$ for an interface point $\mathbf{x} \in \Gamma$.

where δ_Γ is a Dirac δ-distribution defined by

$$\int_\Omega \delta_\Gamma(\mathbf{x})\phi(\mathbf{x})\, d\mathbf{x} = \int_\Gamma \phi(s)\, ds$$

for a smooth function ϕ.

Summarizing, the CSF model is as follows:

$$\begin{cases} \rho(\dfrac{\partial \mathbf{u}}{\partial t} + (\mathbf{u}\cdot\nabla)\mathbf{u}) = \rho\mathbf{g} + \operatorname{div}\boldsymbol{\sigma} + f_\Gamma & \text{in } \Omega \times [t_0, t_f], \\ \operatorname{div}\mathbf{u} = 0 \end{cases} \tag{2.13}$$

initial condition $\mathbf{u}|_{t=t_0} = \mathbf{u}_0$ in Ω,

suitable boundary conditions at $\partial\Omega$.

Under reasonable smoothness assumptions one can show that (2.13) is in fact equivalent to (2.9)–(2.10). In the following lemma we show that the CSF model can be derived from conservation of momentum and mass in the whole domain Ω.

Lemma 2.2

Let $\mathbf{u}, p$ be a solution of (2.9)–(2.10) such that $\boldsymbol{\sigma}(\mathbf{u})$ is differentiable. Then we have

$$\int_\Omega \mathbf{u}\cdot\nabla q\, d\mathbf{x} = 0 \qquad \text{for all } q \in C_0^\infty(\Omega), \tag{2.14}$$

i. e., the conservation equation for mass, $\operatorname{div}\mathbf{u} = 0$, holds in Ω (in the sense of distributions). Furthermore, conservation of momentum in an arbitrary subdomain $W \subset \Omega$ yields

$$\int_W \rho\left(\mathbf{u}_t + (\mathbf{u}\cdot\nabla)\mathbf{u}\right)\, d\mathbf{x} = \int_W (\rho\mathbf{g} + \operatorname{div}\boldsymbol{\sigma})\, d\mathbf{x} + \int_{W\cap\Gamma} \tau\kappa\mathbf{n}\, ds. \tag{2.15}$$

Proof. We first consider (2.14). Using $\operatorname{div}\mathbf{u} = 0$ in Ω_i, $i = 1, 2$, and $[\mathbf{u}\cdot\mathbf{n}]_\Gamma = 0$, for $q \in C_0^\infty(\Omega)$ we obtain

$$\begin{aligned}\int_\Omega \mathbf{u}\cdot\nabla q\,d\mathbf{x} &= \sum_{i=1}^{2}\int_{\Omega_i}\mathbf{u}\cdot\nabla q\,d\mathbf{x}\\ &= \sum_{i=1}^{2}\left(\int_{\partial\Omega_i} q\,\mathbf{u}\cdot\mathbf{n}\,ds - \int_{\Omega_i} q\,\operatorname{div}\mathbf{u}\,d\mathbf{x}\right)\\ &= \int_\Gamma q\,[\mathbf{u}\cdot\mathbf{n}]_\Gamma\,ds = 0.\end{aligned}$$

Now we consider (2.15). For $W \cap \Gamma = \emptyset$ we have the situation $W \subset \Omega_i$ with $i = 1$ or $i = 2$. The derivation of conservation of momentum for one-phase flows was already discussed in Section 2.1.1 yielding (2.15), where the boundary integral is missing due to $W \cap \Gamma = \emptyset$. Thus we only have to consider the case $\gamma := W \cap \Gamma \neq \emptyset$. Then W is subdivided by Γ into two subdomains $W_i := W \cap \Omega_i$, $i = 1, 2$, cf. Figure 2.2. Repeating the steps from Section 2.1.1 but without integrating by parts, we end up with

$$\int_W \rho\,(\mathbf{u}_t + (\mathbf{u}\cdot\nabla)\mathbf{u})\;d\mathbf{x} = \int_W \rho\mathbf{g}\,d\mathbf{x} + \int_{\partial W}\boldsymbol{\sigma}\mathbf{n}\,ds. \tag{2.16}$$

Note that applying Stokes' theorem to the boundary integral on the right-hand side of (2.16) yields

$$\begin{aligned}\int_{\partial W}\boldsymbol{\sigma}\mathbf{n}\,ds &= \left(\sum_{i=1}^{2}\int_{\partial W_i}\boldsymbol{\sigma}\mathbf{n}\,ds\right) + \int_\gamma [\boldsymbol{\sigma}\mathbf{n}]_\Gamma\,ds\\ &= \left(\sum_{i=1}^{2}\int_{W_i}\operatorname{div}\boldsymbol{\sigma}\,d\mathbf{x}\right) + \int_\gamma \tau\kappa\mathbf{n}\,ds\\ &= \int_W \operatorname{div}\boldsymbol{\sigma}\,d\mathbf{x} + \int_\gamma \tau\kappa\mathbf{n}\,ds.\end{aligned}$$

Combined with (2.16) this proves the result. □

One important issue is hidden in the formulation of problem (2.13), namely that the location of the interface $\Gamma(t)$ has to be known for each time instant $t \in [t_0, t_f]$. This topic is discussed in Section 2.2.

Note that the classical strong formulation in (2.13) has to be treated with care as the coefficients ρ, μ are discontinuous across Γ and thus, e. g., $\operatorname{div}\boldsymbol{\sigma}$ is not defined. It should rather be interpreted in a weak sense. The weak formulation of (2.13) is given below in Section 2.1.4.

2.1.3. Boundary conditions

In the two-phase flow models (2.9)–(2.10) and (2.13) suitable boundary conditions on $\partial\Omega$ have to be added. We distinguish between *essential* and *natural* boundary condition. Let the boundary $\Sigma = \partial\Omega$ be partitioned in a part $\Sigma_N \subset \Sigma$ where natural boundary conditions are imposed and a remaining part $\Sigma_D = \Sigma \setminus \Sigma_N$ with essential boundary conditions.

The essential boundary conditions are of Dirichlet type and are used for modeling inflow (*inflow boundary condition*) and walls (*no-slip boundary condition*). The inflow boundary condition prescribes the velocity at the inflow boundary $\Sigma_{in} \subset \Sigma_D$,

$$\mathbf{u}(\mathbf{x},t) = \mathbf{u}_{in}(\mathbf{x},t), \qquad (\mathbf{x},t) \in \Sigma_{in} \times [t_0, t_f],$$

with $\mathbf{u}_{in}$ such that $\mathbf{u}_{in} \cdot \mathbf{n} \leq 0$ on Σ_{in} holds. The no-slip boundary condition prescribes the velocity on $\Sigma_{wall} \subset \Sigma_D$ to be equal to the tangential velocity $\mathbf{u}_{wall}$ of the related wall,

$$\mathbf{u}(\mathbf{x},t) = \mathbf{u}_{wall}(\mathbf{x},t), \qquad (\mathbf{x},t) \in \Sigma_{wall} \times [t_0, t_f],$$

with $\mathbf{u}_{wall} \cdot \mathbf{n} = 0$ on Σ_{wall}. For a fixed wall we have $\mathbf{u}_{wall} = 0$, thus homogeneous Dirichlet boundary conditions on Σ_{wall}. In the following we represent inflow and no-slip boundary conditions in a combined way by

$$\mathbf{u}(\mathbf{x},t) = \mathbf{u}_D(\mathbf{x},t), \qquad (\mathbf{x},t) \in \Sigma_D \times [t_0, t_f]. \tag{2.17}$$

Natural boundary conditions are usually applied to model outflow (*outflow boundary condition*):

$$\boldsymbol{\sigma}\mathbf{n} = -p_{ext}\mathbf{n} \qquad \text{on } \Sigma_N \times [t_0, t_f].$$

Here p_{ext} is a given external pressure. One usually chooses $p_{ext} = 0$, thus we get a homogeneous natural boundary condition on Σ_N. In the following we assume

$$\boldsymbol{\sigma}\mathbf{n} = 0 \qquad \text{on } \Sigma_N \times [t_0, t_f]. \tag{2.18}$$

A discussion on alternative outflow boundary conditions can be found in [Tur99].

2.1.4. Weak formulation

We first collect some useful results on partial integration. We assume that $p : \Omega \to \mathbb{R}$ and $\mathbf{u}, \mathbf{v} : \Omega \to \mathbb{R}^3$ are sufficiently smooth functions. Then

$$\int_\Omega \nabla p \cdot \mathbf{u}\, d\mathbf{x} = -\int_\Omega p \operatorname{div} \mathbf{u}\, d\mathbf{x} + \int_{\partial\Omega} p\mathbf{u} \cdot \mathbf{n}\, ds.$$

A simple calculation shows

$$-\int_\Omega \big(\operatorname{div}\mathbf{D}(\mathbf{u})\big)\cdot\mathbf{v}\,d\mathbf{x} = \frac{1}{2}\int_\Omega \operatorname{tr}\big(\mathbf{D}(\mathbf{u})\mathbf{D}(\mathbf{v})\big)\,d\mathbf{x} - \int_{\partial\Omega}\big(\mathbf{D}(\mathbf{u})\,\mathbf{n}\big)\cdot\mathbf{v}\,ds,$$

where tr denotes the trace operator for matrices, i. e., $\operatorname{tr} M = \sum_{i=1}^N M_{ii}$ for $M \in \mathbb{R}^{N\times N}$.

For the weak formulation of (2.13) we introduce the spaces

$$\begin{aligned}\mathbf{V} &:= (H^1(\Omega))^3,\\ \mathbf{V}_0 &:= \{\,\mathbf{v}\in\mathbf{V}:\ \mathbf{v}=0 \text{ on } \Sigma_D\,\},\\ \mathbf{V}_D &:= \{\,\mathbf{v}\in\mathbf{V}:\ \mathbf{v}=\mathbf{u}_D \text{ on } \Sigma_D\,\},\\ Q &:= L_{2,0}(\Omega) = \{\,q\in L_2(\Omega):\ \int_\Omega q\,d\mathbf{x}=0\,\}.\end{aligned}$$

We define the bilinear forms

$$\begin{aligned} m:\mathbf{V}\times\mathbf{V}\to\mathbb{R}: \qquad & m(\mathbf{u},\mathbf{v}) := \int_\Omega \rho\,\mathbf{u}\mathbf{v}\,d\mathbf{x},\\ a:\mathbf{V}\times\mathbf{V}\to\mathbb{R}: \qquad & a(\mathbf{u},\mathbf{v}) := \frac{1}{2}\int_\Omega \mu\,\operatorname{tr}\big(\mathbf{D}(\mathbf{u})\mathbf{D}(\mathbf{v})\big)\,d\mathbf{x},\\ & \qquad = \frac{1}{2}\int_\Omega \mu\sum_{i,j=1}^3 \big[\mathbf{D}(\mathbf{u})\big]_{ij}\big[\mathbf{D}(\mathbf{v})\big]_{ij}\,d\mathbf{x}\\ b:\mathbf{V}\times Q\to\mathbb{R}: \qquad & b(\mathbf{v},q) := -\int_\Omega q\,\operatorname{div}\mathbf{v}\,d\mathbf{x}, \end{aligned}$$

and the trilinear form

$$n:\mathbf{V}\times\mathbf{V}\times\mathbf{V}\to\mathbb{R}: \qquad n(\mathbf{u};\mathbf{v},\mathbf{w}) := \int_\Omega \rho(\mathbf{u}\cdot\nabla\mathbf{v})\mathbf{w}\,d\mathbf{x}.$$

For the weak formulation of the CSF term we introduce the linear form

$$f_\Gamma:\mathbf{V}\to\mathbb{R}: \qquad f_\Gamma(\mathbf{v}) := \int_\Gamma \tau\kappa\mathbf{n}\cdot\mathbf{v}\,ds. \tag{2.19}$$

The L_2 scalar product in $L_2(\Omega)$ is denoted by

$$(g,h)_0 := \int_\Omega g\cdot h\,d\mathbf{x}.$$

Note that for $\mathbf{u}, \mathbf{v} \in \mathbf{V}, p \in Q$ we get by partial integration

$$-\int_\Omega \operatorname{div}(\boldsymbol{\sigma})\mathbf{v}\, d\mathbf{x} = a(\mathbf{u}, \mathbf{v}) + b(\mathbf{v}, p) - \int_{\partial\Omega} \boldsymbol{\sigma}\mathbf{n}\mathbf{v}\, ds. \tag{2.20}$$

Assuming homogeneous natural boundary conditions on Σ_N, i. e., $\boldsymbol{\sigma}\mathbf{n}|_{\Sigma_N} = 0$, all boundary integrals caused by partial integration vanish for $\mathbf{v} \in \mathbf{V}_0$. Then the weak formulation of (2.13) is as follows:

Find $(\mathbf{u}, p) \in \mathbf{V}_D \times Q$ such that for all $t \in [t_0, t_f]$

$$m(\frac{\partial \mathbf{u}}{\partial t}, \mathbf{v}) + n(\mathbf{u}; \mathbf{u}, \mathbf{v})$$
$$+ a(\mathbf{u}, \mathbf{v}) + b(\mathbf{v}, p) = (\rho\mathbf{g}, \mathbf{v})_0 + f_\Gamma(\mathbf{v}) \qquad \text{for all } \mathbf{v} \in \mathbf{V}_0, \tag{2.21}$$
$$b(\mathbf{u}, q) = 0 \qquad \text{for all } q \in Q, \tag{2.22}$$

initial condition $\mathbf{u}|_{t=t_0} = \mathbf{u}_0$ in Ω.

2.2. Locating the interface

The CSF model (2.13) has to be supplemented by some interface localization technique, because the surface force f_Γ and the coefficients ρ, μ are depending on Γ. Several interface localization techniques can be found in the literature. Most of these methods are either of *front-tracking* or of *front-capturing* type. We refer to [Smo01] for a survey on this topic. Most of these methods are strongly connected to discretization concepts and often they cannot be formulated in a continuous manner. Therefore discretization notions are used in the discussion of these methods below.

The *front-tracking* methods are based on an explicit representation of the interface as a discretized manifold, which is evolved in time. Either the grid is fitted to the interface and deformed according to the flow field *(Lagrangian approach)*, or a separate representation is used for the interface, e. g., some interface mesh, while the grid discretizing the computational domain is kept fixed *(Eulerian approach)*.

The Lagrangian front-tracking approach is mostly used for simulations of free-surface flows (cf. [Bän01, Beh02]), where the computational domain covers only one phase and the moving interface is part of the domain's boundary. There are only few applications of this method to two-phase flows (e. g.,

[JT94, TBM92, Tez07]). The main difficulty of this method is to prevent grid deterioration over time. Non-local updating strategies ranging from simple projecting methods to complete remeshing are used to keep the quality of the elements.

In Eulerian front-tracking methods (e. g., [UT92, TBE+01, GGL+98, MCN03, DFG+06, Mao07]) the interface is represented by a separate data structure storing the position and connectivity of marker points on the interface. These marker points are individually advected by the local velocity field. During the evolution of the interface, points have to be inserted or removed to provide an accurate representation of the interface. If the topology of the interface changes (e. g., when bubbles merge or break up), a proper update of the interface representation is hard to realize, in particular for the 3D case, which is a major drawback of this method. However, in [ZALC05] such a method is applied to three-dimensional droplet break up yielding satisfactory results.

The most popular *front-capturing* methods are the *Volume of Fluids* (VOF) method and the *level set* method. In both cases the interface is defined implicitly by some indicator function, the so-called VOF or level set function, respectively.

The VOF method (see e. g., [HN81, GW01, BKW04]) is a volume-tracking approach which uses a cell-wise constant function to indicate the volume fraction of a certain phase for each cell. If combined with conservative finite volume schemes, one benefit of the method is its conservation of mass property. A major disadvantage is the non-uniqueness of the interface location. For the advection of the VOF function and the computation of normals and curvature a sharp interface has to be reconstructed locally from the volume fraction information in each time step. If one tries to keep the interface relatively smooth, this task is not straight-forward. Therefore, several interface reconstruction algorithms have been developed and improved over the years. A comparison of some VOF methods is given in [Rud97].

For the level set method (cf. [CHMO96, OF01, OS88, Sus03]) the interface is given by the zero-level set of a continuous function, which is positive in the one phase and negative in the other. Thus the position of the interface is uniquely described and can be reconstructed from its implicit representation (if needed). Moreover, topology changes can be handled without further effort. A drawback of this method is that conservation of mass is not inherently incorporated in it.

The level set technique has been successfully used in many two-phase incompressible flow simulations. By far most of these simulations use finite

difference or finite volume discretization methods (cf. [OF01, Set99] and the references therein and [Sus03]). There are relatively few publications in which the level set method is combined with finite element discretization techniques. Such a combination for a 2D simulation is presented in [Tor00, TE00] and [Smo01, Smo05]. Other references are [PS01, Hys06]. In [MR06, CMR08] the level set equation is discretized by a discontinuous Galerkin method. In Section 2.2.1 a more detailed description of the level set method is given.

2.2.1. The level set method

The level set method was introduced by Sethian and Osher [OS88]. An overview of the method and its applications is given in [Set99, OF01].

The basic idea of the level set method is to represent the interface Γ implicitly by the zero level set of some continuous, scalar function $\varphi : \Omega \times [t_0, t_f] \to \mathbb{R}$, i. e.,

$$\Gamma(t) = \{\, \mathbf{x} \in \Omega : \varphi(\mathbf{x}, t) = 0 \,\}, \qquad t \in [t_0, t_f].$$

Moreover, the sign of $\varphi(\mathbf{x}, t)$ indicates in which phase $\mathbf{x}$ is located, i. e., whether $\mathbf{x} \in \Omega_1(t)$ or $\mathbf{x} \in \Omega_2(t)$. As a convention, we assume $\varphi(\cdot, t) < 0$ in $\Omega_1(t)$ and $\varphi(\cdot, t) > 0$ in $\Omega_2(t)$. There are many possible choices of such functions φ. From the computational point of view the most convenient one is the signed distance function, i. e.,

$$|\varphi(\mathbf{x}, t)| = \operatorname{dist}(\mathbf{x}, \Gamma(t)).$$

For this choice we have $\|\nabla\varphi\| = 1$. In practice, we will use an approximative signed distance function for φ. A 2D example of a level set function for the setting shown in Figure 2.1 is given in Figure 2.3.

We assume that the initial location $\Gamma|_{t=t_0}$ of the interface is known, given by an initial value for the level set function $\varphi(\mathbf{x}, t_0) = \varphi_0(\mathbf{x})$. The evolution of the interface is determined by the local flow field $\mathbf{u}|_\Gamma$ as the interface is transported by the moving fluid. Formulated in a Lagrangian manner of a moving coordinate system, if $\mathbf{x}(t) \in \Omega$ is the position of a particle moving with the fluid and

$$\text{if } \mathbf{x}(t_0) \in \Gamma(t_0), \quad \text{then } \mathbf{x}(t) \in \Gamma(t) \qquad \text{for all } t \in [t_0, t_f]. \tag{2.23}$$

Rewriting (2.23) in terms of the level set function φ, for each $\mathbf{x}(t_0) \in \Gamma(t_0)$ we obtain

$$\varphi(\mathbf{x}(t), t) = \varphi(\mathbf{x}(t_0), t_0) = 0 \qquad \text{for all } t \in [t_0, t_f].$$

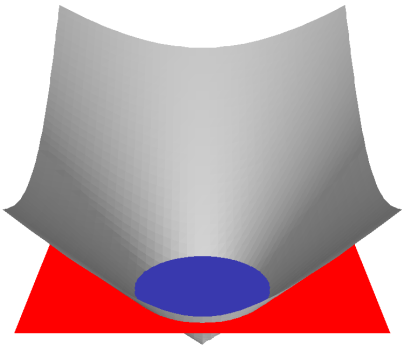

Figure 2.3.: 2D level set function φ for two-phase example from Figure 2.1.

Extending this to the whole domain Ω, for each $\mathbf{x}(t_0) \in \Omega$

$$\varphi(\mathbf{x}(t), t) = \varphi(\mathbf{x}(t_0), t_0) = const \qquad \text{for all } t \in [t_0, t_f].$$

Hence,

$$\begin{aligned} 0 &= \frac{d}{dt}\varphi(\mathbf{x}(t), t) \\ &= \varphi_{\mathbf{x}}(\mathbf{x}, t)\, \dot{\mathbf{x}}(t) + \varphi_t(\mathbf{x}, t) \qquad \text{for all } (\mathbf{x}(t), t) \in \Omega \times [t_0, t_f]. \end{aligned} \tag{2.24}$$

Note that $\dot{\mathbf{x}}(t) = \mathbf{u}(\mathbf{x}(t), t)$, as $\mathbf{u}$ denotes the velocity of particles. Substituting this in (2.24) we obtain the following evolution equation for the level set function φ,

$$\varphi_t + \mathbf{u} \cdot \nabla\varphi = 0 \qquad \text{in } \Omega \times [t_0, t_f]. \tag{2.25}$$

(2.25) is called the *level set equation*. Note that (2.25) is a pure hyperbolic problem. We introduce the Sobolev space $V := H^1(\Omega)$. Then a weak formulation of (2.25) is given by

Find $\varphi \in V$ such that for all $t \in [t_0, t_f]$

$$(\varphi_t, v)_0 + (\mathbf{u} \cdot \nabla\varphi, v)_0 = 0 \qquad \text{for all } v \in V. \tag{2.26}$$

Other weak formulations for hyperbolic problems can be found in [QV94].

One advantage of the level set approach is the fact that geometric properties of the interface Γ such as normals and curvature can be easily computed:

$$\mathbf{n} = \frac{\nabla\varphi}{\|\nabla\varphi\|}, \tag{2.27}$$

$$\kappa = -\operatorname{div}\mathbf{n} = -\operatorname{div}\frac{\nabla\varphi}{\|\nabla\varphi\|}. \tag{2.28}$$

Remark 2.3
The computation of curvature is needed for the evaluation of the surface force term f_Γ, cf. (2.19). Instead of formula (2.28), which requires the computation of second order derivatives of φ, we use a Laplace-Beltrami technique described in Section 5.3 which only needs first order derivatives of φ. As we assume $\varphi \in H^1(\Omega)$, clearly the latter approach is more suitable. ◇

The jumps in the coefficients ρ and μ can also be described by using the level set function φ. Introducing the Heaviside function $H : \mathbb{R} \to \mathbb{R}$,

$$H(x) = \begin{cases} 0 & x < 0, \\ 0.5 & x = 0, \\ 1 & x > 0, \end{cases}$$

we define

$$\rho(\varphi) := \rho_1 + (\rho_2 - \rho_1)H(\varphi), \qquad \mu(\varphi) := \mu_1 + (\mu_2 - \mu_1)H(\varphi). \tag{2.29}$$

Even though the level set method is a very elegant method from the mathematical point of view, it also suffers from some nasty features. Firstly, during the evolution of the level set function, the signed distance property is lost and has to be reestablished from time to time. This *reparametrization* has to be handled with care as the zero level set should be kept fixed or moved by only a 'small' amount. Secondly, the discretization of the Navier-Stokes equations induces loss or gain of mass as the method is not inherently mass-conserving for a discrete divergence-free velocity field $\mathbf{u}_h$. This has to be corrected by some suitable strategy. These issues are discussed in Chapter 8.

Part II.

Numerical methods

Overview

In the following the main features of the numerical methods are presented, which are used for solving the two-phase flow problem discussed in the previous chapter. The description is rather short and intends to give a schematic overview of the overall structure of the solution strategy, see also Figure 2.4 for a sketch of the outline. A more detailed description of the several components is given in the Chapters 3–9.

To be able to discretize the phase interface with a high resolution, an adaptive refinement algorithm based on multilevel tetrahedral grids is applied. This allows for a highly refined mesh close to the interface and a relatively coarse mesh in other regions of the domain. If the interface is moving (like in the example of the rising bubble), the mesh can easily be adapted to follow the interface by appropriate refinement and coarsening of the tetrahedral elements from time to time. The refinement algorithm is described in Chapter 3.

Both Navier-Stokes and level set equations are discretized in space by finite elements. We use P_2 finite elements for the level set function which are stabilized by streamline diffusion (SDFEM). For the discretization of the Navier-Stokes equations P_2-FE for the velocity and P_1-FE or extended P_1^Γ-FE for the pressure are used. The Taylor-Hood (P_2-P_1) finite element pair is known to be LBB stable. For the $P_2 - P_1^\Gamma$ finite element pair this is an open question. In our simulations we did not experience any stability problems. The jumping coefficients are treated by means of a special quadrature strategy. More details can be found in Chapter 4.

The numerical treatment of surface tension raises (at least) two challenging issues which are addressed in Chapter 5.

- For the weak formulation of the CSF term a Laplace-Beltrami technique is used avoiding the calculation of second derivatives when computing the curvature. The CSF term is discretized as a surface integral, so there is no need for numerically approximating the Dirac delta distribution in the volume integral formulation. In Section 5.3 we treat this issue.
- Due to the discontinuous pressure jump across the interface in pres-

ence of surface tension forces, known as the Laplace-Young law, it is advantageous to switch to another FE space for the pressure variable (P_1^{Γ}-FE instead of P_1-FE). The construction of appropriate FE spaces is discussed in Section 5.4.

For the time discretization implicit schemes like the one-step theta scheme or the Fractional Step scheme are applied. For the Fractional Step scheme two different variants — one with operator splitting and one without — are presented. Especially the time discretization of the CSF term has to be handled with care, as it leads to restrictive time step sizes if it is treated explicitly. In each (macro) time step a coupled Navier-Stokes and level set problem has to be solved. Here the problem is decoupled by introducing a fixed point approach, which can be seen as some kind of Picard iteration. Time discretization and coupling strategy are discussed in Chapter 6.

The solution of the discrete problems is topic of Chapter 7. The nonlinearity of the Navier-Stokes problem is treated by a defect correction approach. Krylov subspace methods are used as iterative solvers for the linear problems. For the saddle point problems an inexact Uzawa method (Schur complement approach) is applied. Preconditioning of the Schur complement operator has to take into account the jumping coefficients caused by different material properties in the two phases.

Even though the level set approach is very attractive for interface capturing, there are some disadvantages that have to be overcome. During the advection process, the signed distance function property gets lost and has to be reestablished by a certain reparametrization. A variant of the fast marching method has turned out to be a favorable choice for this task. As conservation of mass is not inherently incorporated into the method, it is enforced artificially by a simple correction of the level set function. These topics are addressed in Chapter 8.

Finally, Chapter 9 describes the structure and design of the software package DROPS [DRO], where all the aforementioned methods have been implemented. DROPS is written in C++ and developed by a couple of people at the IGPM, RWTH Aachen University, Germany. The software is applied to simulate the hydrodynamics and heat and mass transfer in two-phase flow problems arising in the Collaborative Research Center SFB 540 [SFB].

Remark 2.4
In the following we briefly motivate the choice of the numerical methods.

Due to the nested multilevel hierarchy of tetrahedral meshes which allows simple refinement and also coarsening routines we can realize a high resolution

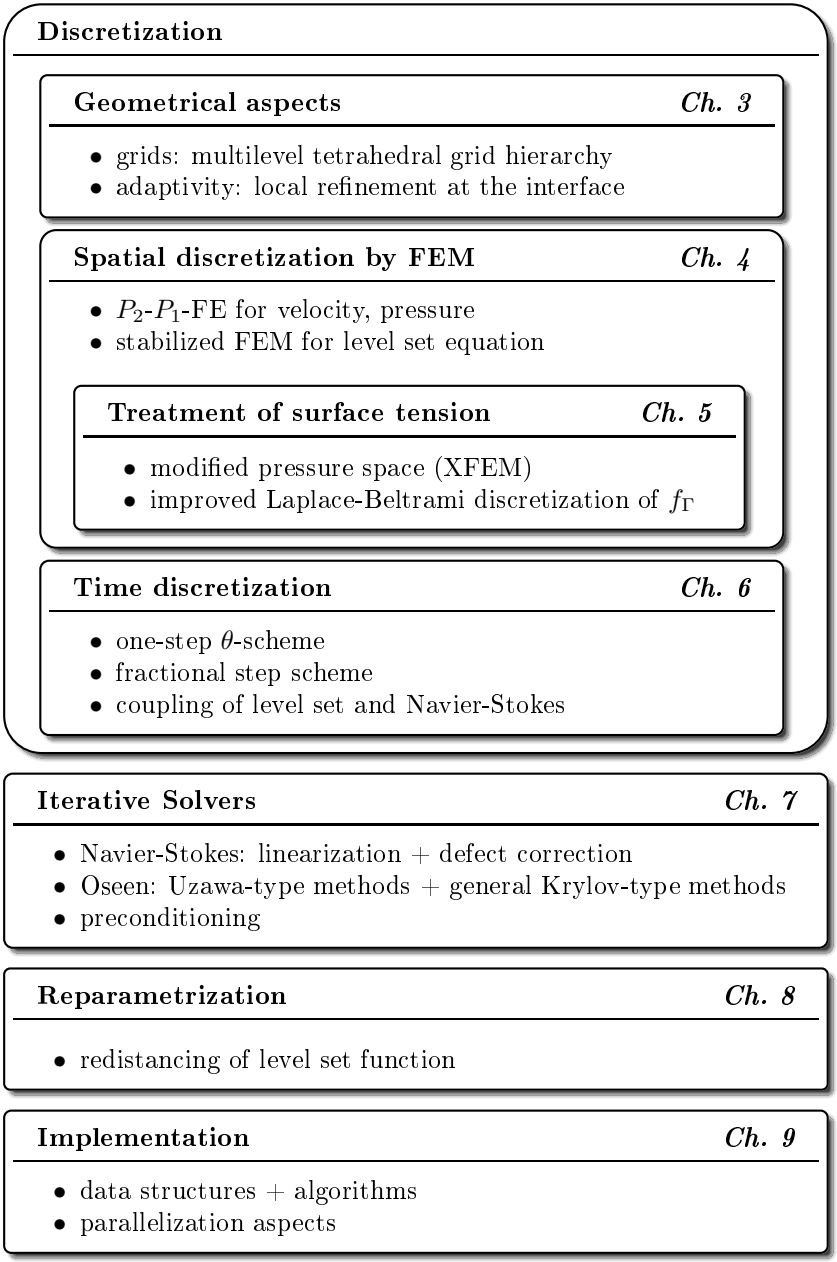

Figure 2.4.: Overview of numerical methods and outline of Part II of the thesis.

close to the interface Γ. For the local refinement in that area we need a suitable marking strategy. For this the level set function is very well suited, because it yields a good approximation of the distance to the interface. Another important reason why we use the level set technique is the fact that topological changes of the interface (e. g., occurring during droplet-droplet interaction) can be handled without further effort. For interface tracking approaches this is a more delicate task, as the interface mesh has to be completely restructured. Also the VOF method suffers from a bad interface reconstruction in this case.

The finite element method is a flexible discretization method, which can deal with complex geometries. Due to the use of finite element discretizations on a nested multilevel grid hierarchy, multigrid solution techniques can be applied. A further nice property of the finite element approach is that we can apply partial integration to the Laplace-Beltrami operator and thus eliminate the second order derivatives that occur in the curvature κ. A disadvantage compared to finite volume methods is the fact that the discretization has to be stabilized for convection-dominated problems and that the finite element method is not conservative.

We use implicit time integration schemes to avoid the time step restrictions of explicit methods for small grid sizes h.

3. Adaptive multilevel refinement

3.1. Multilevel grid hierarchy

We first introduce some notions for the geometric entities by the following definitions.

Definition 3.1 (Triangulation)
A finite collection $\mathcal{T}$ of tetrahedra $T \subset \overline{\Omega}$ is called a *triangulation* of Ω (or $\overline{\Omega}$) if the following holds:

1. $\text{meas}_3(T) > 0$ for all $T \in \mathcal{T}$,
2. $\bigcup_{T \in \mathcal{T}} T = \overline{\Omega}$,
3. $\text{int}(S) \cap \text{int}(T) = \emptyset$ for all $S, T \in \mathcal{T}$ with $S \neq T$.

Here $\text{int}(U)$ means the interior of the set $U \subset \overline{\Omega}$. ◇

Definition 3.2 (Consistency)
A triangulation $\mathcal{T}$ is called *consistent* if the intersection of any two tetrahedra in $\mathcal{T}$ is either empty, a common face, a common edge or a common vertex. ◇

Definition 3.3 (Stability)
A sequence of triangulations $(\mathcal{T}_0, \mathcal{T}_1, \mathcal{T}_2, \ldots)$ is called *stable* if all angles of all tetrahedra in this sequence are uniformly bounded away from zero. ◇

It is known that for finite element discretizations in many cases the weaker (maximal angle) condition "all angles of all tetrahedra are uniformly bounded away from π" would be sufficient. However, using the latter condition, stronger requirements on the robustness of iterative solvers are needed, which can be avoided when using the minimal angle condition in Definition 3.3.

Definition 3.4 (Refinement)
For a given tetrahedron T a triangulation $\mathcal{K}(T)$ of T is called a *refinement* of T if $|\mathcal{K}(T)| \geq 2$ and any vertex of any tetrahedron $T' \in \mathcal{K}(T)$ is either a

vertex or an edge midpoint of T. In this case T' is called a *child* of T and T is called the *parent* of T'.

A refinement $\mathcal{K}(T)$ of T is called *regular* if $|\mathcal{K}(T)| = 8$, otherwise it is called *irregular*.

A triangulation $\mathcal{T}_{k+1}$ is called *refinement* of a triangulation $\mathcal{T}_k \neq \mathcal{T}_{k+1}$ if for every $T \in \mathcal{T}_k$ either $T \in \mathcal{T}_{k+1}$ or $\mathcal{K}(T) \subset \mathcal{T}_{k+1}$ for some refinement $\mathcal{K}(T)$ of T. ◇

Definition 3.5 (Multilevel triangulation)
A sequence of consistent triangulations $\mathcal{M} = (\mathcal{T}_0, \ldots, \mathcal{T}_J)$ is called a *multilevel triangulation* of Ω if the following holds:

1. For $0 \leq k < J$: $\mathcal{T}_{k+1}$ is a refinement of $\mathcal{T}_k$.
2. For $0 \leq k < J$: if $T \in \mathcal{T}_k \cap \mathcal{T}_{k+1}$, then $T \in \mathcal{T}_J$.

The tetrahedra $T \in \mathcal{T}_J$ are called the *leaves* of $\mathcal{M}$. Note that T is a leaf iff T has no children in $\mathcal{M}$.

A tetrahedron $T \in \mathcal{M}$ is called *regular* if $T \in \mathcal{T}_0$ or T resulted from a regular refinement of its parent. Otherwise T is called *irregular*.

A multilevel triangulation $\mathcal{M}$ is called *regular* if all irregular $T \in \mathcal{M}$ are leaves (i. e., have no children in $\mathcal{M}$).

$\mathcal{T}_0$ is called the *coarsest* or *initial triangulation*, $\mathcal{T}_J$ is called the *finest triangulation*. ◇

Remark 3.6
Let $\mathcal{M}$ be a multilevel triangulation and V_k $(0 \leq k \leq J)$ be the corresponding finite element spaces of continuous functions $p \in C(\bar{\Omega})$ such that $p_{|T} \in \mathcal{P}_q$ for all $T \in \mathcal{T}_k$ $(q \geq 1)$. The refinement property 1 in Definition 3.5 implies nestedness of these finite element spaces: $V_k \subset V_{k+1}$. ◇

Definition 3.7 (Hierarchical decomposition of $\mathcal{M}$)
Let $\mathcal{M} = (\mathcal{T}_0, \ldots, \mathcal{T}_J)$ be a multilevel triangulation of Ω. For every tetrahedron $T \in \mathcal{M}$ a unique level number $\ell(T)$ is defined by

$$\ell(T) := \min\{\, k : T \in \mathcal{T}_k \,\}.$$

The set $\mathcal{G}_k \subset \mathcal{T}_k$,

$$\mathcal{G}_k := \{\, T \in \mathcal{T}_k : \ell(T) = k \,\}$$

is called the *hierarchical surplus* on level k, $k = 0, 1, \ldots, J$. Note that

$$\mathcal{G}_0 = \mathcal{T}_0, \qquad \mathcal{G}_k = \mathcal{T}_k \setminus \mathcal{T}_{k-1} \quad \text{for } k = 1, \ldots, J.$$

The sequence $\mathcal{H} = (\mathcal{G}_0, \ldots, \mathcal{G}_J)$ is called the *hierarchical decomposition* of $\mathcal{M}$. Note that the multilevel triangulation $\mathcal{M}$ can be uniquely reconstructed from its hierarchical decomposition due to refinement property 2 in Definition 3.5.$\diamond$

Remark 3.8
The hierarchical decomposition induces simple data structures in a canonical way. The tetrahedra of each hierarchical surplus $\mathcal{G}_k$ are stored in a separate list. Thus every tetrahedron $T \in \mathcal{M}$ is stored exactly once since T has a unique level number $\ell(T)$. By introducing unique level numbers also for vertices, edges and faces, these sub-simplices can be stored in the same manner: For a sub-simplex S the level number $\ell(S)$ is defined as the level of its first appearance. Additionally, the objects are linked to certain corresponding objects by pointers (e. g., a tetrahedron is linked to its vertices, edges, faces, children and parent). $\diamond$

3.2. Adaptive refinement

In this section we describe a refinement algorithm which is, apart from some minor modifications, the algorithm presented in [Bey95, Bey98]. This method is based on similar ideas as the refinement algorithms in [BSW83, BBJ+97]. We restrict ourselves to tetrahedral meshes. However, the method can easily be modified such that it is applicable to other element types such as, for example, hexahedra and pyramids.

The refinement strategy is based on a set of regular and irregular refinement rules (also called *red* and *green* rules, due to [BSW83]), which are described in the following two sections. The regular and irregular rules are local in the sense that they are applied to a single tetrahedron. These rules are applied in a (global) refinement algorithm that describes how the local rules can be combined to ensure consistency and stability, cf. Definitions 3.2 and 3.3.

3.2.1. The regular refinement rule

Let T be a given tetrahedron. For the construction of a regular refinement of T it is natural to connect midpoints of the edges of T by subdividing each of the faces into four congruent triangles. This yields four sub-tetrahedra at the corners of T (all similar to T) and an octahedron in the middle. This octahedron is further subdivided into four sub-tetrahedra with equal volume

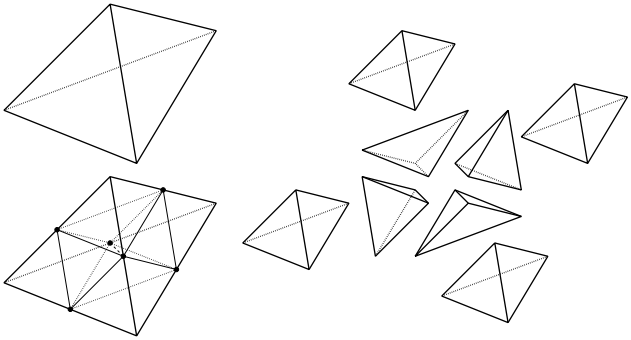

Figure 3.1.: Regular refinement.

(cf. Figure 3.1). A stable tetrahedral regular refinement strategy, based on an idea from [Fre42], is presented in [Bey95, Bey00]. We recall this method.

Let $T = [x^{(1)}, x^{(2)}, x^{(3)}, x^{(4)}]$ be a tetrahedron with *ordered* vertices $x^{(1)}$, $x^{(2)}$, $x^{(3)}$, $x^{(4)}$ and

$$x^{(ij)} := \frac{1}{2}(x^{(i)} + x^{(j)}) \ , \quad 1 \le i < j \le 4,$$

the midpoint of the edge between $x^{(i)}$ and $x^{(j)}$. The regular refinement $\mathcal{K}(T) := \{T_1, \ldots, T_8\}$ of T is constructed by the (red) rule

$$\begin{aligned} T_1 &:= [x^{(1)}, x^{(12)}, x^{(13)}, x^{(14)}] \ , & T_5 &:= [x^{(12)}, x^{(13)}, x^{(14)}, x^{(24)}] \ , \\ T_2 &:= [x^{(12)}, x^{(2)}, x^{(23)}, x^{(24)}] \ , & T_6 &:= [x^{(12)}, x^{(13)}, x^{(23)}, x^{(24)}] \ , \\ T_3 &:= [x^{(13)}, x^{(23)}, x^{(3)}, x^{(34)}] \ , & T_7 &:= [x^{(13)}, x^{(14)}, x^{(24)}, x^{(34)}] \ , \\ T_4 &:= [x^{(14)}, x^{(24)}, x^{(34)}, x^{(4)}] \ , & T_8 &:= [x^{(13)}, x^{(23)}, x^{(24)}, x^{(34)}] \ . \end{aligned} \tag{3.1}$$

$T_1, \ldots, T_4$ are the sub-tetrahedra at the corners of T, and $T_5, \ldots, T_8$ form the octahedron in the middle of T. In [Bey00] it is shown that for any T the repeated application of this rule produces a sequence of consistent triangulations of T which is stable. For a given T all tetrahedra that are generated in such a recursive refinement process form at most three similarity classes.

3.2.2. Irregular refinement rules

Let $\mathcal{T}$ be a given consistent triangulation. We select a subset $\mathcal{S}$ of tetrahedra from $\mathcal{T}$ and assume that the regular refinement rule is applied to each of the tetrahedra from $\mathcal{S}$. In general the resulting triangulation $\mathcal{T}'$ will not be consistent. The irregular (or green) rules are used to make this new triangulation

consistent. For this we introduce the notion of an edge refinement pattern. Let $E_1, \ldots, E_6$ be the ordered edges of $T \in \mathcal{T}$. We define the 6-tuple

$$R(T) = (r_1, \ldots, r_6) \in \{0,1\}^6$$

by:

- $r_i = 1$ if E_i is an edge of a tetrahedron $S \in \mathcal{S}$ (i. e., edge E_i is refined and has two sub-edges in $\mathcal{T}'$) and
- $r_i = 0$ otherwise (i. e., edge E_i is not refined) .

For $T \in \mathcal{S}$ we have $R(T) = (1, \ldots, 1)$. For $T \in \mathcal{T} \backslash \mathcal{S}$ the case $R(T) = (0, \ldots, 0)$ corresponds to the situation that the tetrahedron T does not contain any vertices from $\mathcal{T}'$ at the midpoints of its edges. For each of the $2^6 - 1$ possible patterns $R \neq (0, \ldots, 0)$ there exists a corresponding refinement $\mathcal{K}(T)$ of T (in the fashion of (3.1)) for which the vertices of the children coincide with vertices of T or with the vertices at the midpoints on the edges E_i with $r_i = 1$. This refinement, however, is not always unique. This is illustrated in Figure 3.2.

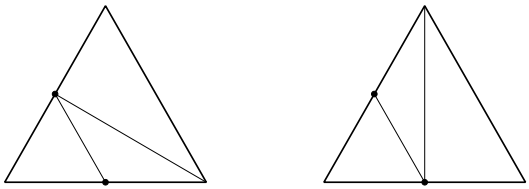

Figure 3.2.: Non-unique face refinement.

To obtain a consistent triangulation in which the subdivision of adjacent faces of neighboring tetrahedra matches special care is needed. One way to ensure consistency is by introducing a so-called consistent vertex numbering:

Definition 3.9 (Consistent vertex numbering)
Let T_1 and T_2 be two adjacent tetrahedra with a common face $F = T_1 \cap T_2$ and local vertex ordering

$$T_l = [x_l^{(1)}, x_l^{(2)}, x_l^{(3)}, x_l^{(4)}], \qquad l = 1, 2.$$

The pair (T_1, T_2) has a *consistent vertex numbering*, if the ordering of the vertices of F induced by the vertex ordering of T_1 coincides with the one induced by the vertex ordering of T_2. A consistent triangulation $\mathcal{T}$ has a *consistent vertex numbering* if every two neighboring tetrahedra have this property. ◇

Remark 3.10
We note that a consistent vertex numbering can be constructed in a rather simple way. Consider an (initial) triangulation $\tilde{\mathcal{T}}$ with an arbitrary numbering of its vertices. This global numbering induces a canonical local vertex ordering which is a consistent vertex numbering of $\tilde{\mathcal{T}}$. Furthermore, each refinement rule can be defined such that the consistent vertex numbering property of the parent is inherited by its children by prescribing suitable local vertex orderings of the children. (3.1) is an example of such a rule. Using such a strategy a consistent triangulation $\tilde{\mathcal{T}}'$ that is obtained by refinement of $\tilde{\mathcal{T}}$ according to these rules also has a consistent vertex numbering. ◇

Assume that the given triangulation $\mathcal{T}$ has a consistent vertex numbering. For a face with a pattern as in Figure 3.2 one can then define a unique face refinement by connecting the vertex with the smallest number with the midpoint of the opposite edge. *For each edge refinement pattern $R \in \{0,1\}^6$ we then have a unique rule.* We emphasize that if for a given tetrahedron T the edge refinement pattern $R(T)$ is known, then *for the application of the regular or irregular rules to this tetrahedron no information from neighboring tetrahedra is needed.* Clearly, for parallelization this is a very nice property.

3.2.3. Multilevel refinement algorithm

Up to now we discussed how the *consistency* of a triangulation can be achieved by the choice of suitable irregular refinement rules based on the consistent vertex numbering property. We will now explain how the regular and irregular rules can be combined in a repeated refinement procedure to obtain a *stable* sequence of consistent triangulations. The crucial point is to *allow only the refinement of regular tetrahedra*, i. e., children of irregularly refined tetrahedra, also called *green children*, are never refined. If such a green child T is marked for refinement, instead of refining T the irregular refinement of the parent will be replaced by a regular one. As the application of the regular rule (3.1) creates tetrahedra of at most 3 similarity classes (cf. [Fre42, Bey00]), the tetrahedra created by a refinement procedure according to this strategy belong to an a-priori bounded number of similarity classes. Hence the obtained sequence of triangulations is stable.

The idea of the so called red-green refinement strategy can be best explained by a simple 2D example: for ease of presentation we use triangles instead of tetrahedra and show the action of a *one-level* refinement method. Consider the following multilevel triangulation $\mathcal{M} = (\mathcal{T}_0, \mathcal{T}_1)$ as depicted in Figure 3.3.

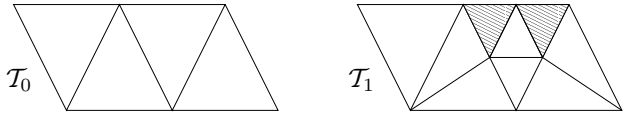

Figure 3.3.: Initial multilevel triangulation with some leaf tetrahedra marked for refinement (indicated by shading).

In $\mathcal{T}_1$ two triangles are marked (by shading) for refinement. A one-level refinement algorithm (like the one described in [BSW83]) only uses the finest triangulation $\mathcal{T}_1$ as input. It first applies the regular refinement rule (the so called "red refinement") to marked regular triangles and to the parents of green children, which are either marked or neighbors of marked triangles — green children are never refined because of stability reasons. This red refinement of course yields an inconsistent triangulation (cf. Figure 3.4 in the middle). Thus in the next step appropriate irregular refinement rules are applied to avoid hanging nodes ("green closure"). The output of the one-level refinement algorithm is the new triangulation $\mathcal{T}_2$ (cf. Figure 3.4 on the right).

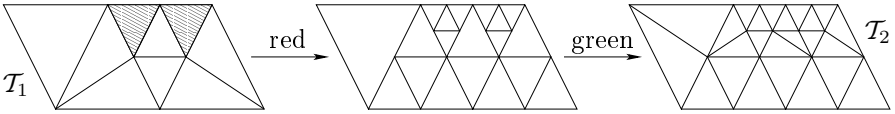

Figure 3.4.: One-level red/green refinement.

The new triangulation $\mathcal{T}_2$ is consistent, but not a refinement of $\mathcal{T}_1$ in the sense of Definition 3.4. Related to this, the corresponding FE spaces are not nested, which is not favorable if one wants to use multigrid solvers for the solution of the linear systems. Another important disadvantage is the fact that it is not obvious how to treat coarsening of the grid, which is also an important task, if the refinement zones are moving in time. This, for example, occurs in the rising bubble problem, where tetrahedra in the lower part of the grid have to be unrefined, when the interface has moved further upwards.

In multilevel refinement algorithms both input and output are multilevel triangulations (cf. Definition 3.5). That means that in general the algorithm not only affects the finest triangulation like in the case of the one-level method, but the whole multilevel triangulation, which can be seen in the next example. The multilevel method is more complicated than the one-level algorithm, but offers important advantages: Property 1 of Definition 3.5 assures the nestedness of the belonging FE spaces, cf. Remark 3.6. The multilevel structure also allows

to treat local refinement and *coarsening* in a similar way. In view of these advantages we implemented a *multilevel* refinement algorithm in DROPS.

For the description of the multilevel refinement algorithm we introduce the notions of status and mark of a tetrahedron. Let $\mathcal{M} = (\mathcal{T}_0, \ldots, \mathcal{T}_J)$ be a multilevel triangulation that has been constructed by applying the regular and irregular refinement rules and let $\mathcal{H} = (\mathcal{G}_0, \ldots, \mathcal{G}_J)$ be the corresponding hierarchical decomposition. Every tetrahedron $T \in \mathcal{H}$ is either a leaf of $\mathcal{M}$ (i.e., $T \in \mathcal{T}_J$) or it has been refined. The label status is used to describe this property of T:

$$\text{For} \quad T \in \mathcal{H}: \quad \mathsf{status}(T) = \begin{cases} \mathsf{NoRef} & \text{if } T \text{ is a leaf of } \mathcal{M}, \\ \mathsf{RegRef} & \text{if } T \text{ is regularly refined in } \mathcal{M}, \\ \mathsf{IrregRef} & \text{if } T \text{ is irregularly refined in } \mathcal{M}. \end{cases}$$

The label IrregRef also contains the number of the irregular refinement rule (one out of 63) that has been used to refine T, i.e., the binary representation of $\mathsf{status}(T)$ coincides with the edge refinement pattern $R(T)$ of T.

In adaptive refinement an error estimator (or indicator) is used to mark certain elements of $\mathcal{T}_J$ for further refinement or for deletion. For this the label mark is used:

$$\text{For} \quad T \in \mathcal{H}: \quad \mathsf{mark}(T) = \begin{cases} \mathsf{Ref} & \text{if } T \in \mathcal{T}_J \text{ is marked for refinement}, \\ \mathsf{Del} & \text{if } T \in \mathcal{T}_J \text{ is marked for deletion}, \\ \mathsf{status}(T) & \text{otherwise.} \end{cases}$$

We describe a multilevel refinement algorithm known in the literature. The basic form of this method was introduced by Bastian [Bas96] and developed further in the UG-group [BBJ+97, BBJ+99, UG]. We use the presentation as in [Bey95, Bey98], which is shown in Algorithm 3.11.

Algorithm 3.11 (Multilevel refinement)

Algorithm *SerRefinement*$(\mathcal{G}_0, \ldots, \mathcal{G}_J)$
for $k = J, \ldots, 0$ **do** // phase I
DetermineMarks$(\mathcal{G}_k)$; (1)
MarksForClosure$(\mathcal{G}_k)$; (2)
for $k = 0, \ldots, J$ **do if** $\mathcal{G}_k \neq \emptyset$ **then** // phase II
if $k > 0$ **then** *MarksForClosure*$(\mathcal{G}_k)$; (3)
if $k < J$ **then** *Unrefine*$(\mathcal{G}_k)$; (4)
Refine$(\mathcal{G}_k)$; (5)
if $\mathcal{G}_J = \emptyset$ **then** $J := J - 1$; (6)
else if $\mathcal{G}_{J+1} \neq \emptyset$ **then** $J := J + 1$; (7)

The input of *SerRefinement* consists of a hierarchical decomposition

$$\mathcal{H} = (\mathcal{G}_0, \ldots, \mathcal{G}_J)$$

in which all refined tetrahedra T are labeled by $\mathsf{mark}(T) = \mathsf{status}(T)$ according to their status and the unrefined $T \in \mathcal{T}_J$ have $\mathsf{mark}(T) \in \{\mathsf{NoRef}, \mathsf{Ref}, \mathsf{Del}\}$. The output is again a hierarchical decomposition, where all tetrahedra are marked according to their status.

The main idea underlying the algorithm *SerRefinement* is illustrated using the multilevel triangulation $(\mathcal{T}_0, \mathcal{T}_1)$ from above. The hierarchical decomposition $\mathcal{H}$ and the corresponding marks are shown in Figure 3.5.

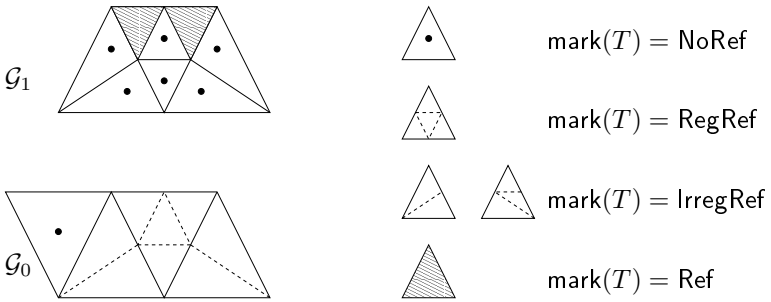

Figure 3.5.: Input hierarchical decomposition.

Note that for the two shaded triangles in $\mathcal{G}_1$ we have $\mathsf{status}(T) \neq \mathsf{mark}(T)$. For all other triangles $\mathsf{status}(T) = \mathsf{mark}(T)$ holds. In phase I of the algorithm (top–down: (1),(2)) only marks are changed. In *DetermineMarks* some

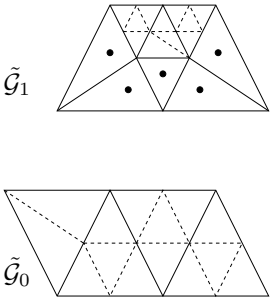

Figure 3.6.: After phase I.

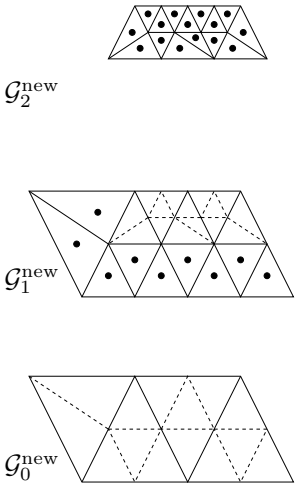

Figure 3.7.: Output hierarchical decomposition.

tetrahedra are labeled with new marks, which are of the type RegRef (for red refinement) or NoRef (for coarsening). The green closure marks are set in *MarksForClosure*, where appropriate irregular refinement marks are determined from the edge refinement patterns to avoid hanging nodes.

Once phase I has been completed the marks have been changed such that $\mathsf{mark}(T) \in \{\mathsf{NoRef}, \mathsf{RegRef}, \mathsf{IrregRef}\}$ holds for all $T \in \mathcal{H}$, cf. Figure 3.6. We emphasize that all green children in $\tilde{\mathcal{G}}_1$ have $\mathsf{mark}(T) = \mathsf{NoRef}$, as they are not refined because of stability reasons. Instead the corresponding irregular refined parents in $\tilde{\mathcal{G}}_0$ are labeled by $\mathsf{mark}(T) = \mathsf{RegRef}$.

In the second phase (bottom–up: (3)-(5)) the actual refinement (coarsening is not needed in our example) is constructed: A call of *Unrefine*$(\mathcal{G}_k)$ deletes all tetrahedra, faces, edges and vertices on level $k+1$, which are not needed anymore due to changed marks. In the subroutine *Refine*$(\mathcal{G}_k)$ all $T \in \mathcal{G}_k$ with $\mathsf{mark}(T) \neq \mathsf{status}(T)$ are refined according to $\mathsf{mark}(T)$ and new objects (tetrahedra, faces, edges, vertices) on level $k+1$ are created. A subsequent call to *MarksForClosure* in (3) computes the appropriate refinement marks for the new created tetrahedra in the next sweep of the **for**-loop.

In the output hierarchical decomposition

$$\mathcal{H}^{\text{new}} = (\mathcal{G}_0^{\text{new}}, \mathcal{G}_1^{\text{new}}, \mathcal{G}_2^{\text{new}})$$

we have $\mathsf{mark}(T) = \mathsf{status}(T)$ for all $T \in \mathcal{H}^{\text{new}}$, cf. Figure 3.7. The output multilevel triangulation $\mathcal{M}^{\text{new}} = (\mathcal{T}_0^{\text{new}}, \mathcal{T}_1^{\text{new}}, \mathcal{T}_2^{\text{new}})$ is *regular* (cf. Definition 3.5) and is given by

$$\mathcal{T}_0^{\text{new}} = \mathcal{G}_0^{\text{new}}, \quad \mathcal{T}_1^{\text{new}} = \mathcal{G}_1^{\text{new}}, \quad \mathcal{T}_2^{\text{new}} = \mathcal{G}_2^{\text{new}} \cup \{\, T \in \mathcal{G}_1^{\text{new}} \,:\, \mathsf{mark}(T) = \mathsf{NoRef} \,\}.$$

Note that $\mathcal{T}_0^{\text{new}} = \mathcal{T}_0$, $\mathcal{T}_1^{\text{new}} \neq \mathcal{T}_1$ (!) and that the new finest triangulation $\mathcal{T}_2^{\text{new}}$ is the same as the triangulation $\mathcal{T}_2$ in Figure 3.4 resulting from the one-level algorithm.

A more detailed discussion of the subroutines in algorithm *SerRefinement* (cf. Algorithm 3.11) is given in [Bey95, Bey98, Gro02].

Remark 3.12
A parallelized version of the algorithm, called *ParRefinement*, has been developed and is described in [Gro02, GR05]. It is based on a formal description of the distributed geometric data which is very suitable for parallelization. This formal description was introduced in [Gro02] and is called an *admissible hierarchical decomposition*, cf. Definition 9.2. It was proved that the application of the multilevel refinement algorithm *ParRefinement* to an input admissible hierarchical decomposition again yields an admissible hierarchical decomposition. The same holds for a suitable load balancing strategy described in [Gro02]. Both parallel refinement algorithm and load balancing strategy have been implemented and were successfully applied up to a number of 64 processors. This implementation has served as a starting point for a further parallelization of DROPS [For07] which is currently conducted by our partners at the Chair of Scientific Computing, RWTH Aachen University. ◇

4. Spatial discretization by Finite Elements

Let $\mathcal{T}$ be a consistent triangulation of Ω, e. g., $\mathcal{T} = \mathcal{T}_J$ the finest triangulation introduced in the previous chapter. For $k \geq 1$ we introduce the spaces of piecewise polynomial continuous functions,

$$\mathbb{X}^k := \{\, v_h \in H^1(\Omega) : \, v_h|_T \in \mathcal{P}_k \quad \forall T \in \mathcal{T} \,\}, \tag{4.1}$$

$$\mathbb{X}^k_D := \mathbb{X}^k \cap H^1_{0,\Sigma_D}(\Omega) \tag{4.2}$$

with $H^1_{0,\Sigma_D}(\Omega)$ the space of all functions in $H^1(\Omega)$ vanishing on the Dirichlet boundary Σ_D (in the sense of traces). For the discretization of the Navier-Stokes and level set equations we will consider the spaces for $1 \leq k \leq 2$, so called P_1 (piecewise linear) and P_2 (piecewise quadratic) finite elements.

4.1. Discretization of the Navier-Stokes equations

For the finite element discretization of the Navier-Stokes equations we choose finite dimensional subspaces $\mathbf{V}_h \subset \mathbf{V}_0$ and $Q_h \subset Q$ for velocity and pressure, respectively. Here we choose the Hood-Taylor finite element pair

$$\mathbf{V}_h \times Q_h := \left(\mathbb{X}^2_D\right)^3 \times \mathbb{X}^1,$$

which fulfills the *inf-sup condition* (also known as *LBB stability*)

$$\inf_{q_h \in Q_h} \sup_{\mathbf{v}_h \in \mathbf{V}_h} \frac{b(\mathbf{v}_h, q_h)}{\|\mathbf{v}_h\|_1 \|q_h\|_0} \geq \beta > 0 \tag{4.3}$$

with $\beta > 0$ independent of h.

Remark 4.1
In Section 5.4 we will introduce an alternative finite element space Q_h^Γ for the pressure which allows for discontinuities at the interface Γ and is thus more appropriate to approximate pressure jumps induced by surface tension. Certain theoretical questions like the LBB stability of the pair $\mathbf{V}_h \times Q_h^\Gamma$ are still unanswered and topics of current research. ◇

We consider the continuous problem in weak formulation, cf. (2.21)–(2.22). For the time being we address the simple case of *homogeneous boundary conditions*, i. e., $\mathbf{u}_D(\mathbf{x},t) = 0$ for all $(\mathbf{x},t) \in \Sigma_D \times [t_0,t_f]$ and $\boldsymbol{\sigma}\mathbf{n} = 0$ on $\Sigma_N \times [t_0,t_f]$, cf. Section 2.1.3. The treatment of non-homogeneous boundary conditions will be discussed in Section 4.1.1. The associated Galerkin discretization is given as follows:

Find $\mathbf{u}_h(t) \in \mathbf{V}_h$ and $p_h(t) \in Q_h$ such that for (almost every) $t \in [t_0,t_f]$

$$m(\mathbf{u}_h'(t),\mathbf{v}_h) + n(\mathbf{u}_h(t);\mathbf{u}_h(t),\mathbf{v}_h) + a(\mathbf{u}_h(t),\mathbf{v}_h) + b(\mathbf{v}_h,p_h(t)) = (\rho\mathbf{g},\mathbf{v}_h)_0 + f_\Gamma(\mathbf{v}_h) \quad \text{for all } \mathbf{v}_h \in \mathbf{V}_h, \tag{4.4}$$

$$b(\mathbf{u}_h(t),q_h) = 0 \quad \text{for all } q_h \in Q_h, \tag{4.5}$$

initial condition $\mathbf{u}_h|_{t=t_0} = \mathbf{u}_0$ in Ω.

Here we use the notation from Section 2.1.4 for the bilinear forms $m(\cdot,\cdot)$, $a(\cdot,\cdot)$, $b(\cdot,\cdot)$, the trilinear form $n(\cdot;\cdot,\cdot)$ and the linear form $f_\Gamma(\cdot)$.

Let $N_{\mathbf{V}_h} := \dim \mathbf{V}_h$ and $N_{Q_h} := \dim Q_h$ be the dimensions of the finite element spaces. For tetrahedral meshes the nodes of the P_1 finite element are located at the vertices of the triangulation, cf. Figure 4.1. Let $\hat{\mathbf{x}}_i \in \mathbb{R}^3$ denote the spatial coordinate of the i-th P_1 node, $i = 1,\ldots,N_{Q_h}$. For the P_2 finite element the nodes are located at the vertices and the midpoint of the edges of the triangulation, its spatial coordinates are denoted by $\mathbf{x}_1,\ldots,\mathbf{x}_{N_{\mathbf{V}_h}}$. Note that, due to $\mathbf{v}_h|_{\Sigma_D} = 0$ for all $\mathbf{v}_h \in \mathbf{V}_h$, nodes on Dirichlet boundaries are not taken into account for the construction of $\mathbf{V}_h$. We introduce *nodal bases* $\{\mathbf{v}_i\}_{i=1,\ldots,N_{\mathbf{V}_h}}$ and $\{q_i\}_{i=1,\ldots,N_{Q_h}}$ of $\mathbf{V}_h$ and Q_h, respectively. Then by construction, $\mathbf{v}_i(\mathbf{x}_j) = 0$ and $q_i(\hat{\mathbf{x}}_j) = 0$ for $i \neq j$ and $\mathbf{v}_i(\mathbf{x}_i) = 1$, $q_i(\hat{\mathbf{x}}_i) = 1$.

By means of the nodal bases, the Galerkin problem (4.4)–(4.5) can be equivalently written in matrix-vector notation. For this we define the isomorphisms

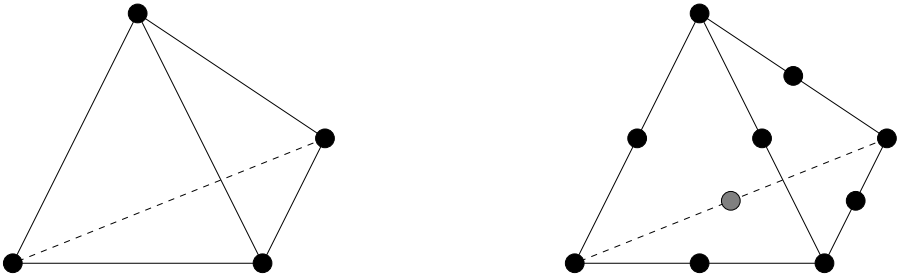

Figure 4.1.: P_1 (4 nodes, on the left) and P_2 finite element (10 nodes, on the right).

$J_{\mathbf{V}_h} : \mathbb{R}^{N_{\mathbf{V}_h}} \to \mathbf{V}_h$ and $J_{Q_h} : \mathbb{R}^{N_{Q_h}} \to Q_h$ by

$$\begin{aligned} J_{\mathbf{V}_h}(\underline{\mathbf{x}}) &:= \sum_{i=1}^{N_{\mathbf{V}_h}} \underline{\mathbf{x}}_i \, \mathbf{v}_i, \\ J_{Q_h}(\underline{y}) &:= \sum_{i=1}^{N_{Q_h}} \underline{y}_i \, q_i, \end{aligned} \tag{4.6}$$

for all vectors $\underline{\mathbf{x}} \in \mathbb{R}^{N_{\mathbf{V}_h}}$, $\underline{y} \in \mathbb{R}^{N_{Q_h}}$. These isomorphisms represent the link between coefficient vectors and associated finite element functions. Based on that, the matrices $A, M, N(\underline{\mathbf{w}}) \in \mathbb{R}^{N_{\mathbf{V}_h} \times N_{\mathbf{V}_h}}$ and $B \in \mathbb{R}^{N_{Q_h} \times N_{\mathbf{V}_h}}$ are defined by

$$\begin{aligned} \langle M\,\underline{\mathbf{u}},\ \underline{\mathbf{v}}\rangle &:= m\big(J_{\mathbf{V}_h}(\underline{\mathbf{u}}),\ J_{\mathbf{V}_h}(\underline{\mathbf{v}})\big) && \text{(mass matrix)}, \\ \langle A\,\underline{\mathbf{u}},\ \underline{\mathbf{v}}\rangle &:= a\big(J_{\mathbf{V}_h}(\underline{\mathbf{u}}),\ J_{\mathbf{V}_h}(\underline{\mathbf{v}})\big) && \text{(discrete diffusion)}, \\ \langle N(\underline{\mathbf{w}})\,\underline{\mathbf{u}},\ \underline{\mathbf{v}}\rangle &:= n\big(J_{\mathbf{V}_h}(\underline{\mathbf{w}});\ J_{\mathbf{V}_h}(\underline{\mathbf{u}}),\ J_{\mathbf{V}_h}(\underline{\mathbf{v}})\big) && \text{(discrete convection)}, \\ \langle B\,\underline{\mathbf{v}},\ \underline{q}\rangle &:= b\big(J_{\mathbf{V}_h}(\underline{\mathbf{v}}),\ J_{Q_h}(\underline{q})\big) && \text{(discrete divergence)} \end{aligned}$$

for all $\underline{\mathbf{u}}, \underline{\mathbf{v}}, \underline{\mathbf{w}} \in \mathbb{R}^{N_{\mathbf{V}_h}}$ and $\underline{q} \in \mathbb{R}^{N_{Q_h}}$. Here $\langle \cdot, \cdot \rangle$ denotes the Euclidean inner product of two vectors.

We rewrite the Galerkin problem (4.4)–(4.5) in matrix-vector notation:

Find $\underline{\mathbf{u}}_h(t) \in \mathbb{R}^{N_{\mathbf{V}_h}}$, $\underline{p}_h(t) \in \mathbb{R}^{N_{Q_h}}$ such that for (almost every) $t \in [t_0, t_f]$

$$\begin{pmatrix} M\underline{\mathbf{u}}_h'(t) \\ 0 \end{pmatrix} + \begin{pmatrix} [N(\underline{\mathbf{u}}_h(t)) + A] & B^T \\ B & 0 \end{pmatrix} \begin{pmatrix} \underline{\mathbf{u}}_h(t) \\ \underline{p}_h(t) \end{pmatrix} = \begin{pmatrix} \underline{\mathbf{b}} \\ 0 \end{pmatrix}, \qquad (4.7)$$

$$\text{initial condition} \qquad \underline{\mathbf{u}}_h|_{t=t_0} = \underline{\mathbf{u}}_0,$$

where the upper right-hand side $\underline{\mathbf{b}}$ is given by

$$\underline{\mathbf{b}}_i := (\rho\mathbf{g}, \mathbf{v}_i)_0 + f_\Gamma(\mathbf{v}_i), \qquad i = 1, \ldots, N_{\mathbf{V}_h}$$

and for the initial condition $\underline{\mathbf{u}}_0$ the property $J_{\mathbf{V}_h}(\underline{\mathbf{u}}_0) = \mathbf{u}_0$ holds.

4.1.1. Non-homogeneous boundary conditions

In the previous section we only considered the discretization for homogeneous boundary conditions. We now discuss the case with general Dirichlet boundary conditions

$$\mathbf{u} = \mathbf{u}_D \qquad \text{on } \Sigma_D \times [t_0, t_f]$$

and general natural boundary conditions

$$\boldsymbol{\sigma}\mathbf{n} = \mathbf{g}_N \qquad \text{on } \Sigma_D \times [t_0, t_f].$$

Define the finite element space $\mathbf{V}_h^D := (\mathbb{X}^2)^3 \supset \mathbf{V}_h$, which also has nodes on the Dirichlet boundary Σ_D. Let $N_{\Sigma_D} := \dim \mathbf{V}_h^D - \dim \mathbf{V}_h$ be the number of nodes on Σ_D and $\{\mathbf{v}_i^D\}_{i=1,\ldots,N_{\Sigma_D}}$ the corresponding nodal basis functions. Note that $\mathbf{V}_h^D = \mathbf{V}_h \oplus \text{span}(\mathbf{v}_1^D, \ldots, \mathbf{v}_{N_{\Sigma_D}}^D)$. We denote by $\mathbf{x}_i^D$ the spatial coordinate of the location of the i-th node on Σ_D, i. e., $\mathbf{x}_i^D$ is either the coordinate of a vertex on Σ_D or the midpoint of an edge on Σ_D.

For $t \in [t_0, t_f]$ let

$$\mathbf{u}_h^D(t) = \sum_{i=1}^{N_{\Sigma_D}} \alpha_i \mathbf{v}_i^D$$

such that $\mathbf{u}_h^D(t)(\mathbf{x}_i^D) = \mathbf{u}_D(\mathbf{x}_i^D, t)$ for all $i = 1, \ldots, N_{\Sigma_D}$. By construction, $\mathbf{u}_h^D(t)(\mathbf{x}_i) = 0$ for all $i = 1, \ldots, N_{\mathbf{V}_h}$ and $t \in [t_0, t_f]$. We introduce $J_{\mathbf{V}_h^D} : \mathbb{R}^{N_{\mathbf{V}_h}} \times [t_0, t_f] \to \mathbf{V}_h^D$ with

$$J_{\mathbf{V}_h^D}(\underline{\mathbf{x}}, t) := J_{\mathbf{V}_h}(\underline{\mathbf{x}}) + \mathbf{u}_h^D(t) \qquad (4.8)$$

The Galerkin problem with non-homogeneous boundary conditions is as follows:

Find $\mathbf{u}_h(t) \in \mathbf{V}_h^D$ with $\mathbf{u}_h(t)|_{\Sigma_D} \equiv \mathbf{u}_h^D(t)$ and $p_h(t) \in Q_h$ such that for (almost every) $t \in [t_0, t_f]$

$$m(\mathbf{u}_h'(t), \mathbf{v}_h) + n(\mathbf{u}_h(t); \mathbf{u}_h(t), \mathbf{v}_h) + a(\mathbf{u}_h(t), \mathbf{v}_h) + b(\mathbf{v}_h, p_h(t)) = (\rho \mathbf{g}, \mathbf{v}_h)_0 + f_\Gamma(\mathbf{v}_h) + \int_{\Sigma_N} \mathbf{g}_N \mathbf{v}_h \, ds \qquad \text{for all } \mathbf{v}_h \in \mathbf{V}_h, \tag{4.9}$$

$$b(\mathbf{u}_h(t), q_h) = 0 \qquad \text{for all } q_h \in Q_h, \tag{4.10}$$

$$\text{initial condition} \qquad \mathbf{u}_h|_{t=t_0} = \mathbf{u}_0 \qquad \text{in } \Omega.$$

Note that the surface integral over Σ_N in the right-hand side of (4.9) arises from partial integration and substitution of the natural boundary condition, cf. (2.20). In practice, we do not use this formulation, but the following equivalent one. Let $\mathbf{u}_h^0 := \mathbf{u}_h - \mathbf{u}_h^D \in \mathbf{V}_h$ be the *homogeneous part* of the finite element solution $\mathbf{u}_h$. Replacing $\mathbf{u}_h(t)$ by $\mathbf{u}_h^0(t)$ in (4.9)–(4.10), we obtain

Find $\mathbf{u}_h^0(t) \in \mathbf{V}_h$ and $p_h(t) \in Q_h$ such that for (almost every) $t \in [t_0, t_f]$

$$\begin{aligned} m((\mathbf{u}_h^0)'(t), \mathbf{v}_h) + n(\mathbf{u}_h(t); \mathbf{u}_h^0(t), \mathbf{v}_h) + n(\mathbf{u}_h^0(t); \mathbf{u}_h^D(t), \mathbf{v}_h) \\ + a(\mathbf{u}_h^0(t), \mathbf{v}_h) + b(\mathbf{v}_h, p_h(t)) \\ = (\rho \mathbf{g}, \mathbf{v}_h)_0 + f_\Gamma(\mathbf{v}_h) + \int_{\Sigma_N} \mathbf{g}_N \mathbf{v}_h \, ds \\ - m((\mathbf{u}_h^D)'(t), \mathbf{v}_h) - n(\mathbf{u}_h^D(t); \mathbf{u}_h^D(t), \mathbf{v}_h) - a(\mathbf{u}_h^D(t), \mathbf{v}_h) \qquad \text{for all } \mathbf{v}_h \in \mathbf{V}_h, \end{aligned} \tag{4.11}$$

$$b(\mathbf{u}_h^0(t), q_h) = -b(\mathbf{u}_h^D(t), q_h) \quad \text{for all } q_h \in Q_h, \tag{4.12}$$

$$\text{initial condition} \qquad \mathbf{u}_h^0|_{t=t_0} = \mathbf{u}_0 - \mathbf{u}_h^D(t_0) \qquad \text{in } \Omega.$$

Then the seeked finite element solution is given by

$$\mathbf{u}_h = \mathbf{u}_h^0 + \mathbf{u}_h^D.$$

Note that the right-hand side of (4.11) contains additional terms accounting for the non-homogeneous Dirichlet boundary values. On the left-hand side

there are two occurrences of the trilinear form $n(\cdot;\cdot,\cdot)$, hence, the matrix N has to be replaced by $\tilde{N} \in \mathbb{R}^{N_{\mathbf{V}_h} \times N_{\mathbf{V}_h}}$, where

$$\begin{aligned}\langle \tilde{N}(\underline{\mathbf{w}}(t))\,\underline{\mathbf{u}}(t),\ \underline{\mathbf{v}}(t)\rangle := n\big(J_{\mathbf{V}_h^D}(\underline{\mathbf{w}}(t),t);\ J_{\mathbf{V}_h}(\underline{\mathbf{u}}(t)),\ J_{\mathbf{V}_h}(\underline{\mathbf{v}}(t))\big)\\ + n\big(J_{\mathbf{V}_h}(\underline{\mathbf{u}}(t));\ \mathbf{u}_h^D(t),\ J_{\mathbf{V}_h}(\underline{\mathbf{v}}(t))\big)\end{aligned}$$

for all $\underline{\mathbf{u}}(t), \underline{\mathbf{v}}(t), \underline{\mathbf{w}}(t) \in \mathbb{R}^{N_{\mathbf{V}_h}}$, $t \in [t_0, t_f]$.

Writing (4.11)–(4.12) in equivalent matrix-vector notation we obtain

Find $\underline{\mathbf{u}}_h^0(t) \in \mathbb{R}^{N_{\mathbf{V}_h}}$, $\underline{p}_h(t) \in \mathbb{R}^{N_{Q_h}}$ such that for (almost every) $t \in [t_0, t_f]$

$$\begin{pmatrix} M(\underline{\mathbf{u}}_h^0)'(t) \\ 0 \end{pmatrix} + \begin{pmatrix} [\tilde{N}(\underline{\mathbf{u}}_h(t)) + A] & B^T \\ B & 0 \end{pmatrix} \begin{pmatrix} \underline{\mathbf{u}}_h^0(t) \\ \underline{p}_h(t) \end{pmatrix} = \begin{pmatrix} \tilde{\underline{\mathbf{b}}} \\ \tilde{\underline{c}} \end{pmatrix}, \qquad (4.13)$$

$$\text{initial condition} \qquad \underline{\mathbf{u}}_h^0|_{t=0} = \underline{\mathbf{u}}_0,$$

with

$$\begin{aligned}\underline{\mathbf{u}}_h &= \underline{\mathbf{u}}_h^0 + \underline{\mathbf{u}}_h^D,\\ \tilde{\underline{\mathbf{b}}} &= \underline{\mathbf{b}} + \underline{\mathbf{v}}^N - \underline{\mathbf{v}}^D,\\ \underline{\mathbf{v}}_i^N &= \int_{\Sigma_N} \mathbf{g}_N \mathbf{v}_i\, ds, \qquad i = 1, \ldots, N_{\mathbf{V}_h},\\ \underline{\mathbf{v}}_i^D &= m((\mathbf{u}_h^D)'(t), \mathbf{v}_i) + n(\mathbf{u}_h^D(t); \mathbf{u}_h^D(t), \mathbf{v}_i) + a(\mathbf{u}_h^D(t), \mathbf{v}_i), \quad i = 1, \ldots, N_{\mathbf{V}_h},\\ \tilde{\underline{c}}_j &= -b(\mathbf{u}_h^D, q_j), \qquad j = 1, \ldots, N_{Q_h}.\end{aligned}$$

Thus, the discretization is very similar to (4.7), but with a different right-hand side and a slightly changed discrete convection matrix $\tilde{N}$.

4.1.2. Treatment of jumping coefficients

The material coefficients ρ and μ have to be handled with care as they are discontinuous across Γ. They occur in integrals of the form

$$I = \int_\Omega \alpha\, G(\mathbf{x})\, d\mathbf{x},$$

where $\alpha \in \{\rho, \mu\}$ is piecewise constant and G is a continuous smooth function on each element $T \in \mathcal{T}_h$. There are two possible ways to deal with the computation of such integrals with discontinuous integrands:

(A) Integration on parts. Split the integral into two integrals on the subdomains,

$$\int_\Omega \alpha\, G(\mathbf{x})\, d\mathbf{x} = \alpha_1 \int_{\Omega_1} G(\mathbf{x})\, d\mathbf{x} + \alpha_2 \int_{\Omega_2} G(\mathbf{x})\, d\mathbf{x}.$$

The integrands on the right-hand side are continuous and smooth on each tetrahedron and thus standard quadrature rules can be used. However, technical difficulties arise for tetrahedra $T \in \mathcal{T}$ which are intersected by Γ, as we have to integrate over its two parts $\Omega_i \cap T$, $i = 1, 2$ which are not tetrahedral in general, cf. Figure 5.9. This issue is further discussed in Section 5.4.4.

(B) Integration of regularized integrands. Replace the discontinuous α by a continuous smoothed α_ε. This can be achieved by replacing the Heaviside function H in (2.29) by a smoothed Heaviside function $H_\varepsilon : \mathbb{R} \to \mathbb{R}$,

$$H_\varepsilon(x) = \begin{cases} 0 & x \le -\varepsilon, \\ \nu(\frac{x}{\varepsilon}) & x \in (-\varepsilon, \varepsilon), \\ 1 & x \ge \varepsilon. \end{cases}$$

with

$$\nu(\xi) = \frac{1}{2} + \frac{1}{32}(45\xi - 50\xi^3 + 21\xi^5), \tag{4.14}$$

cf. [Tor00]. Also other choices of smooth transition functions $\nu(\xi)$ can be found in the literature, e. g., $\nu(\xi) = \frac{1+\sin(\xi\frac{\pi}{2})}{2}$ [SSO94]. For the approximation of I, we apply quadrature to the integral

$$I_\varepsilon := \int_\Omega \alpha_\varepsilon(\varphi(\mathbf{x}))\, G(\mathbf{x})\, d\mathbf{x}.$$

For the first approach we have the following error bound.

Remark 4.2 (Discretization error for approach A)
Assume that Γ is approximated by a piecewise planar interface approximation Γ_h with the property

$$|d(\mathbf{x})| \le c\, h^2 \qquad \text{for all } \mathbf{x} \in \Gamma_h,$$

where $d(\mathbf{x}) := \operatorname{dist}(\mathbf{x}, \Gamma)$ is the distance function for Γ. For the construction of such an interface approximation Γ_h we refer to Section 5.1.2 where more

details are given. Γ_h subdivides Ω into two subdomains $\Omega_{i,h}$, $i = 1, 2$. The integral I is approximated by integrating on both subdomains,

$$I_h := \alpha_1 \int_{\Omega_{1,h}} G(\mathbf{x})\, d\mathbf{x} + \alpha_2 \int_{\Omega_{2,h}} G(\mathbf{x})\, d\mathbf{x}.$$

Usually the integration is performed tetrahedron by tetrahedron. If a tetrahedron T is cut by the planar interface approximation Γ_h, the domains of integration $T \cap \Omega_{i,h}$, $i = 1, 2$, are not necessarily tetrahedral. For more details on how to integrate over the two parts of a cut tetrahedron we refer to Section 5.4.4.

In the following the error $|I - I_h|$ is analyzed in terms of h. We introduce the sets $D_h^+ := \Omega_1 \setminus \Omega_{1,h}$, $D_h^- := \Omega_{1,h} \setminus \Omega_1$ and $D_h := D_h^+ \cup D_h^-$. Note that D_h contains all points between Γ and Γ_h and that $\text{meas}_3(D_h) \leq c\, h^2$. Then for $G \in L_\infty(D_h)$ we have

$$\begin{aligned} |I - I_h| &\leq |\alpha_1 - \alpha_2|\, \|G\|_{L_1(D_h)} \leq |\alpha_1 - \alpha_2|\, \|G\|_{L_\infty(D_h)}\, \text{meas}_3(D_h) \\ &\leq c\, h^2. \end{aligned} \tag{4.15}$$

◇

We now turn to the second approach. Clearly, the second method is much easier to implement than the first one, but introduces a new parameter ε. For the choice of $\nu(\xi)$ as in (4.14), an extensive analysis and comparison with the first approach is given in [Tor00, Tor02] for the 2D case. Based on these investigations the second approach is used in [TE00]. We give here the main discretization error results from [Tor02]. Before that we have to introduce some notions for polynomial transition functions.

Definition 4.3
Let $\nu : [-1, 1] \to \mathbb{R}$ be a polynomial with $\nu(-1) = 0$ and $\nu(1) = 1$. Then ν is called a *transition polynomial*. ν has $m \geq 0$ *vanishing moments*, if

$$\int_{-1}^{1} \nu(\xi)\, \xi^\alpha\, d\xi = \frac{1}{\alpha + 1} \qquad \text{for all } \alpha = 0, 1, \ldots, m.$$

ν has an *transition smoothness of order* $k \geq 0$, if

$$\nu^{(\beta)}(\pm 1) = 0 \qquad \text{for all } \beta = 1, \ldots, k.$$

◇

Theorem 4.4 (Discretization error for approach B, 2D case)
We consider the 2D case $\Omega \subset \mathbb{R}^2$. Let Q_T be a quadrature formula for a triangle T such that $Q_T f = \int_T f(\mathbf{x})\, d\mathbf{x}$ for all polynomials f up to the order

n. We introduce the regularization error

$$E_\varepsilon := \int_\Omega (H - H_\varepsilon)(\varphi(\mathbf{x}))\, G(\mathbf{x})\, d\mathbf{x}$$

and the quadrature error

$$E_{\text{quad}} := \int_\Omega H_\varepsilon(\varphi(\mathbf{x}))\, G(\mathbf{x})\, d\mathbf{x} - \sum_{T\in\mathcal{T}_h} Q_T(H_\varepsilon G).$$

Then the total error $E_{\text{tot}} := \int_\Omega H(\varphi(\mathbf{x}))\, G(\mathbf{x})\, d\mathbf{x} - \sum_{T\in\mathcal{T}_h} Q_T(H_\varepsilon G)$ is the sum of regularization and quadrature error,

$$E_{\text{tot}} = E_\varepsilon + E_{\text{quad}}. \tag{4.16}$$

Assuming that $\varepsilon \cdot \max_{\mathbf{x}\in\Gamma} |\kappa(x)| < 1$, where κ is the local curvature of Γ, and that $\nu(\xi)$ has m vanishing moments, for the regularization error we have

$$E_\varepsilon \sim \varepsilon^{\beta+2}, \qquad \text{with } \beta = 2\left\lfloor \frac{m+1}{2} \right\rfloor. \tag{4.17}$$

If $G \in C^k$, $H_\varepsilon \in C^k$ and $n > k$, where n is the order of the quadrature rule Q_T, then

$$E_{\text{quad}} \sim \frac{h^{k+2}}{\varepsilon^{k+1}}. \tag{4.18}$$

Proof. Given in [Tor02]. □

Remark 4.5
For the transition function $\nu(\xi)$ mentioned above in (4.14) we have $m = 2$ vanishing moments and a transition smoothness of order $k = 1$. Thus we have to apply a quadrature rule which is exact up to the order of at least $n = 2$, yielding

$$E_{\text{tot}} \sim \varepsilon^4 + \frac{h^3}{\varepsilon^2}.$$

When the grid size h changes due to refinement the regularization parameter ε should be scaled with h such that $\varepsilon \sim h^{1/2}$. In that case we have

$$E_{\text{tot}} \sim h^2.$$

This is the same order of convergence as for the first approach, cf. (4.15) in Remark 4.2. ◇

Even though the second approach seems to be quite convenient, we experienced some *critical* problems. When discretizing the mass matrix M,

$$M_{ij} = \int_\Omega \rho_\varepsilon(\varphi(\mathbf{x}))\, \mathbf{v}_i \mathbf{v}_j \, d\mathbf{x}, \qquad 1 \le i,j \le N_{\mathbf{V}_h},$$

using a quadrature rule of order 2 or even of order 5 yields a matrix which is not always positive definite. This undesired effect was also observed for other matrices involving discontinuous coefficients and has of course a significant impact on the convergence behavior of the iterative solvers. We therefore favor the first approach, although its implementation is more tedious, as it avoids the additional smoothing parameter ε and guarantees a positive definite discretization of elliptic operators.

4.2. Discretization of the level set equation

The level set equation (2.26) is also discretized by finite elements. For this purpose we use P_2 finite elements and introduce the finite-dimensional space $V_h := \mathbb{X}^2 \subset H^1(\Omega)$. Note that there are no boundary conditions stated for the level set function φ, hence $N_{V_h} := \dim V_h$ is equal to the number of vertices and edges of the corresponding triangulation $\mathcal{T}_h$. Let $\{v_i\}_{i=1,\dots,N_{V_h}}$ be the nodal basis of V_h and $J_{V_h} : \mathbb{R}^{N_{V_h}} \to V_h$ the isomorphism defined by

$$J_{V_h}(\underline{x}) := \sum_{i=1,\dots,N_{V_h}} \underline{x}_i \, v_i \tag{4.19}$$

for all vectors $\underline{x} \in \mathbb{R}^{N_{V_h}}$.

As the level set equation is purely hyperbolic, the standard Galerkin discretization should not be used and requires some stabilization. We apply a streamline diffusion stabilization which can be seen as a Petrov-Galerkin method with trial space V_h and special test functions $\hat{v}_h$. For each tetrahedron $T \in \mathcal{T}_h$ a stabilization parameter $\delta_T = \delta_T(h_T, \mathbf{u}_h|_T)$ is chosen, where h_T denotes the maximal diameter of T. The test functions are then defined as

$$\hat{v}_h|_T := v_h + \delta_T \mathbf{u}_h \cdot \nabla v_h, \qquad T \in \mathcal{T}_h,$$

which induces additional diffusion in streamline direction explaining the name of the method. For an analysis of the streamline diffusion method and reasonable choices of the stabilization parameter δ_T we refer to [RST96]. We use $\delta_T = c\, h_T$ with a suitable constant $c > 0$. If the velocity field $\mathbf{u}_h$ shows strong

local fluctuations, the choice $\delta_T = c\,\frac{h_T}{\|\mathbf{u}_h\|_{\infty,T}}$ is suggested in [Pri06]. As this is not well-defined for $\mathbf{u}_h = 0$ and tends to infinity for $\mathbf{u} \to 0$ we recommend to use

$$\delta_T = c\,\frac{h_T}{\max\{\varepsilon_0/h_T, \|\mathbf{u}_h\|_{\infty,T}\}}$$

instead for some small $\varepsilon_0 > 0$.

The streamline diffusion finite element discretization of the level set equation is given by

$$\sum_{T\in\mathcal{T}_h} (\varphi_h'(t) + \mathbf{u}_h(t)\cdot\nabla\varphi_h(t),\ v_h + \delta_T\mathbf{u}_h(t)\cdot\nabla v_h)_{0,T} = 0 \qquad \text{for all } v_h \in V_h, \tag{4.20}$$

where $t \in [t_0, t_f]$. Introducing the matrices $E = E(\mathbf{u}_h) \in \mathbb{R}^{N_{V_h}\times N_{V_h}}$ and $H = H(\mathbf{u}_h) \in \mathbb{R}^{N_{V_h}\times N_{V_h}}$ given by

$$E_{ij} := \sum_{T\in\mathcal{T}_h} \big(v_j,\ v_i + \delta_T\mathbf{u}_h\cdot\nabla v_i\big)_{0,T} \qquad \text{(stabilized mass matrix)},$$

$$H_{ij} := \sum_{T\in\mathcal{T}_h} \big(\mathbf{u}_h\cdot\nabla v_j,\ v_i + \delta_T\mathbf{u}_h\cdot\nabla v_i\big)_{0,T} \qquad \text{(stabilized discrete convection)},$$

where $1 \le i, j \le N_{V_h}$, we rewrite (4.20) in matrix-vector notation:

Find $\underline{\varphi}(t) \in \mathbb{R}^{N_{V_h}}$ such that for (almost every) $t \in [t_0, t_f]$

$$E\,\underline{\varphi}'(t) + H\,\underline{\varphi}(t) = 0. \tag{4.21}$$

5. Numerical treatment of surface tension

Due to the Laplace-Young law, typically the pressure has a jump across the interface, when surface tension forces are present ($\tau \neq 0$), cf. Remark 5.1 below. In numerical simulations, this discontinuity and inadequate approximation of the localized surface force term often lead to strong unphysical oscillations of the velocity $\mathbf{u}_h$ at the interface, so called *spurious velocities* or *spurious currents*, cf. , e. g., [LNS+94, FCD+06]. In this chapter we present an alternative finite element discretization approach which significantly reduces the size of these spurious velocities compared to known methods. For the motivation and analysis of our approach we further simplify (2.21)–(2.22) and consider a stationary Stokes problem with a constant viscosity ($\mu_1 = \mu_2 = \mu$ in Ω). We emphasize, however, that the methods that we present are *not* restricted to this simplified problem but apply to the general Navier-Stokes model (2.21)–(2.22) as well. We introduce the following Stokes problem: find $(\mathbf{u}, p) \in \mathbf{V}_0 \times Q$ such that

$$\begin{aligned} a(\mathbf{u}, \mathbf{v}) + b(\mathbf{v}, p) &= (\rho \mathbf{g}, \mathbf{v}) + f_\Gamma(\mathbf{v}) \qquad \text{for all } \mathbf{v} \in \mathbf{V}_0, \\ b(\mathbf{u}, q) &= 0 \qquad\qquad\qquad\qquad\quad \text{for all } q \in Q, \end{aligned} \tag{5.1}$$

where

$$a(\mathbf{u}, \mathbf{v}) := \int_\Omega \mu \nabla \mathbf{u} \nabla \mathbf{v} \, d\mathbf{x}, \qquad b(\mathbf{v}, q) = -\int_\Omega q \operatorname{div} \mathbf{v} \, d\mathbf{x},$$

with a viscosity $\mu > 0$ that is constant in Ω. The unique solution of this problem is denoted by $(\mathbf{u}^*, p^*) \in \mathbf{V}_0 \times Q$.

Remark 5.1
The problem (5.1) has a *smooth* velocity solution $\mathbf{u}^* \in \mathbf{V}_0 \cap \left(H^2(\Omega)\right)^3$ and a *piecewise smooth* pressure solution p with $p_{|\Omega_i} \in H^1(\Omega_i)$, $i = 1, 2$, which has a jump across Γ. These smoothness properties can be derived as follows. The curvature κ is a smooth function (on Γ). Thus there exist $\hat{p}_1 \in H^1(\Omega_1)$ such that $(\hat{p}_1)_{|\Gamma} = \kappa$ (in the sense of traces). Define $\hat{p} \in L^2(\Omega)$ by $\hat{p} = \hat{p}_1$ in Ω_1,

$\hat{p} = 0$ on Ω_2. Note that for all $\mathbf{v} \in \mathbf{V}_0$,

$$\begin{aligned} f_\Gamma(\mathbf{v}) &= \tau \int_\Gamma \kappa \mathbf{n}_\Gamma \cdot \mathbf{v}\, ds = \tau \int_\Gamma \hat{p}_1 \mathbf{n}_\Gamma \cdot \mathbf{v}\, ds \\ &= \tau \int_{\Omega_1} \hat{p}_1 \operatorname{div} \mathbf{v}\, d\mathbf{x} + \tau \int_{\Omega_1} \nabla \hat{p}_1 \cdot \mathbf{v}\, d\mathbf{x} = \tau \int_\Omega \hat{p} \operatorname{div} \mathbf{v}\, d\mathbf{x} + \tau \int_\Omega \tilde{\mathbf{g}} \cdot \mathbf{v}\, d\mathbf{x}, \end{aligned}$$

with $\tilde{\mathbf{g}} \in L^2(\Omega)^3$ given by $\tilde{\mathbf{g}} = \nabla \hat{p}_1$ in Ω_1, $\tilde{\mathbf{g}} = 0$ on Ω_2. Thus $(\mathbf{u}^*, p^* + \tau\hat{p})$ satisfies the standard Stokes equations

$$\begin{aligned} a(\mathbf{u}^*, \mathbf{v}) + b(\mathbf{v}, p^* + \tau\hat{p}) &= (\rho\mathbf{g} + \tau\tilde{\mathbf{g}}, \mathbf{v}) \quad \text{for all } \mathbf{v} \in \mathbf{V}_0, \\ b(\mathbf{u}^*, q) &= 0 \qquad\qquad\qquad\quad \text{for all } q \in Q. \end{aligned}$$

From regularity results on Stokes equations and the fact that Ω is convex we conclude that $\mathbf{u}^* \in H^2(\Omega) \cap H_0^1(\Omega)$ and $p^* + \tau\hat{p} \in H^1(\Omega)$. Thus $[p^* + \tau\hat{p}]_\Gamma = 0$ (a.e. on Γ) holds, which implies

$$[p^*]_\Gamma = -\tau[\hat{p}]_\Gamma = -\tau\kappa,$$

i.e., p^* has a jump across Γ of the size $\tau\kappa$. ◇

Example 5.2 (Static Bubble)
A simple example that is used in the numerical experiments in Section 10.4 is the following. Let $\Omega := (-1, 1)^3$ and Ω_1 a sphere with center at the origin and radius $r < 1$. We take $\mathbf{g} = 0$. In this case the curvature is constant, $\kappa = -\frac{2}{r}$, and the solution of the Stokes problem (5.1) is given by $\mathbf{u}^* = 0$, $p^* = \tau\frac{2}{r} + c_0$ on Ω_1, $p^* = c_0$ on Ω_2 with a constant c_0 such that $\int_\Omega p^*\, d\mathbf{x} = 0$. ◇

The outline of this chapter is as follows. In Section 5.1 we introduce an interface approximation Γ_h of the interface Γ and formulate some abstract properties of the interface approximation our analysis is based on. Furthermore, we describe how the interface approximation is implemented in our code such that the desired properties are fulfilled. In Section 5.2 the spurious velocities are traced back to two major error sources, the discretization error of the surface tension force and the approximation error of the discontinuous pressure. Both are analyzed in the subsequent sections. Section 5.3 describes the discretization of the surface tension force f_Γ based on a Laplace-Beltrami technique. For this approach a discretization error of $\mathcal{O}(\sqrt{h})$ is proved and some slight modification with an improved $\mathcal{O}(h)$ behavior is introduced. In Section 5.4 it is shown that standard finite element spaces are not very suitable for the approximation of functions with a jump across Γ due to an approximation error of size $\mathcal{O}(\sqrt{h})$. We introduce a new finite element space which is more suitable for this task, based on the extended finite element method (XFEM) by Belytschko [MDB99, BMUP01].

5.1. Interface approximation

Recalling the definition of f_Γ,

$$f_\Gamma(\mathbf{v}) = \tau \int_\Omega \kappa \delta_\Gamma \mathbf{v}\mathbf{n}\, d\mathbf{x} = \tau \int_\Gamma \kappa \mathbf{v}\mathbf{n}\, ds \qquad \text{for all } \mathbf{v} \in \mathbf{V}_0,$$

we see, that this term can either be discretized by computing a volume integral where the integrand contains a (regularized) delta function δ_Γ or by computing a surface integral over an approximation of the interface Γ. Both approaches can be found in the literature, see for instance [TE00, Hys06, PS01, MGCR07] for the volume integral approach and [MCN03, GRR06, Smo05] for the surface integral approach. We favor the surface integral approach as it seems to be more natural and avoids the difficulties arising from the numerical treatment of the delta function. For evaluating the surface integral we need to know the location of the interface Γ, which is only implicitly given by the level set function. Hence, a local interface reconstruction method has to be applied which provides an approximative interface Γ_h.

Before describing how an approximation Γ_h of the interface Γ can be constructed in practice, we first give some abstract conditions which the interface approximation Γ_h should fulfill (cf. Section 5.1.1). Our theoretical analysis of the discretization of the surface tension force f_Γ in Section 5.3 will be based on these abstract conditions. We note that due to this fact the analysis is not only restricted to our concrete interface reconstruction method described in Section 5.1.2 but applies to any interface reconstruction method that meets the requirements formulated in the conditions (5.5)–(5.7) below.

5.1.1. Assumptions on Γ_h

For the formulation of assumptions on the approximate interface Γ_h it is convenient to introduce the signed distance function

$$d : U \to \mathbb{R}, \qquad |d(\mathbf{x})| := \operatorname{dist}(\mathbf{x}, \Gamma) \quad \text{for all } \mathbf{x} \in U.$$

Thus Γ is the zero level set of d. We assume $d < 0$ on the interior of Γ (that is, in Ω_1) and $d > 0$ on the exterior. Note that $\mathbf{n}_\Gamma = \nabla d$ on Γ. We define $\mathbf{n}(\mathbf{x}) := \nabla d(\mathbf{x})$ for all $\mathbf{x} \in U$. Thus $\mathbf{n} = \mathbf{n}_\Gamma$ on Γ and $\|\mathbf{n}(\mathbf{x})\| = 1$ for all $\mathbf{x} \in U$. Here and in the remainder of the section $\|\cdot\|$ denotes the Euclidean norm.

Remark 5.3
In our approach we use the discrete level set function φ_h as approximation for the distance function d, which is not available. ◇

The Hessian of d is denoted by $\mathbf{H}$:

$$\mathbf{H}(\mathbf{x}) = D^2 d(\mathbf{x}) \in \mathbb{R}^{3\times 3} \quad \text{for all } \mathbf{x} \in U. \tag{5.2}$$

The eigenvalues of $-\mathbf{H}(\mathbf{x})$ are denoted by $\kappa_1(\mathbf{x}), \kappa_2(\mathbf{x})$ and 0. For $\mathbf{x} \in \Gamma$ the eigenvalues $\kappa_i(\mathbf{x})$, $i = 1, 2$, are the *principal curvatures*, and $\kappa(\mathbf{x}) = \kappa_1(\mathbf{x}) + \kappa_2(\mathbf{x})$ is the *mean curvature*.

We will need the orthogonal projection $\mathbf{P}$ onto the tangential space of Γ,

$$\mathbf{P}(\mathbf{x}) = \mathbf{I} - \mathbf{n}(\mathbf{x})\mathbf{n}(\mathbf{x})^T \quad \text{for } \mathbf{x} \in U. \tag{5.3}$$

Using the distance function d we introduce assumptions on the approximate interface Γ_h. In Section 5.1.2 below we indicate how in practice an approximate interface Γ_h can be constructed which satisfies these assumptions. Let $\{\Gamma_h\}_{h_\Gamma>0}$ be a family of polygonal approximations of Γ. Each Γ_h is contained in U and consists of a set $\mathcal{F}_h$ of *triangular faces*:

$$\Gamma_h = \bigcup_{F\in\mathcal{F}_h} F. \tag{5.4}$$

For $F_1, F_2 \in \mathcal{F}_h$ with $F_1 \neq F_2$ we assume that $F_1 \cap F_2$ is either empty or a common edge or a common vertex. The parameter h_Γ denotes the maximal diameter of the triangles in $\mathcal{F}_h$:

$$h_\Gamma = \max_{F\in\mathcal{F}_h} \operatorname{diam}(F).$$

By $\mathbf{n}_h(\mathbf{x})$ we denote the outward pointing unit normal on Γ_h. This normal is piecewise constant with possible discontinuities at the edges of the triangles in $\mathcal{F}_h$.

The approximation Γ_h is assumed to be close to Γ in the following sense:

$$|d(\mathbf{x})| \leq c h_\Gamma^2 \quad \text{for all } \mathbf{x} \in \Gamma_h, \tag{5.5}$$

$$\operatorname{ess\,inf}_{\mathbf{x}\in\Gamma_h} \mathbf{n}(\mathbf{x})^T \mathbf{n}_h(\mathbf{x}) \geq c > 0, \tag{5.6}$$

$$\operatorname{ess\,sup}_{\mathbf{x}\in\Gamma_h} \|\mathbf{P}(\mathbf{x})\mathbf{n}_h(\mathbf{x})\| \leq c h_\Gamma. \tag{5.7}$$

Here c denotes a generic constant independent of h_Γ.

Remark 5.4
The conditions (5.6), (5.7) are satisfied if

$$\operatorname{ess\,sup}_{\mathbf{x}\in\Gamma_h} \|\mathbf{n}(\mathbf{x}) - \mathbf{n}_h(\mathbf{x})\| \leq \min\{c_0, c h_\Gamma\}, \quad \text{with } c_0 < \sqrt{2}, \tag{5.8}$$

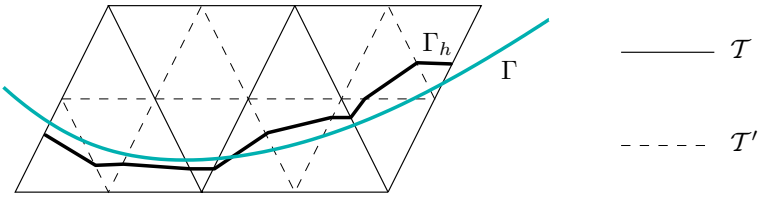

Figure 5.1.: Construction of approximate interface for 2D case.

holds. This easily follows from

$$\|\mathbf{n}(\mathbf{x}) - \mathbf{n}_h(\mathbf{x})\|^2 = 2\big(1 - \mathbf{n}(\mathbf{x})^T \mathbf{n}_h(\mathbf{x})\big),$$

and

$$\|\mathbf{P}(\mathbf{x})\mathbf{n}_h(\mathbf{x})\| = \|\mathbf{P}(\mathbf{x})\big(\mathbf{n}(\mathbf{x}) - \mathbf{n}_h(\mathbf{x})\big)\| \leq \|\mathbf{n}(\mathbf{x}) - \mathbf{n}_h(\mathbf{x})\|. \qquad \diamond$$

5.1.2. Implementation

We briefly explain the approach that is used in our implementation DROPS (cf. [DRO]) for computing Γ_h. Let $\mathcal{S}$ be the (locally refined) triangulation of Ω, consisting of tetrahedra, that is used for the discretization of the flow variables with finite elements, cf. (2.21)–(2.22). The level set equation for d is discretized with continuous piecewise quadratic finite elements on a triangulation $\mathcal{T}$, cf. Section 4.2. This triangulation is either equal to $\mathcal{S}$ or obtained from one or a few refinements of $\mathcal{S}$, i. e., $\mathcal{T} = \mathcal{T}_J$ is the finest and $\mathcal{S} = \mathcal{T}_k$, $0 \leq k \leq J$ is a possibly coarser triangulation of the multilevel triangulation $\mathcal{M}$, cf. Chapter 3. The piecewise *quadratic* finite element approximation of d on $\mathcal{T}$ is denoted by d_h.

We now introduce one further regular refinement of $\mathcal{T}$ (subdivision of each tetrahedron in 8 child tetrahedra), resulting in $\mathcal{T}'$. Let $I(d_h)$ be the continuous piecewise *linear* function on $\mathcal{T}'$ which interpolates d_h at all vertices of all tetrahedra in $\mathcal{T}'$. Note that the degrees of freedom of the P_1 FE on $\mathcal{T}'$ (located at the vertices) coincide with the degrees of freedom of the P_2 FE on $\mathcal{T}$ (located at the vertices and midpoints of edges).

The approximation Γ_h of the interface Γ is defined by

$$\Gamma_h := \{\, \mathbf{x} \in \Omega : \, I(d_h)(\mathbf{x}) = 0 \,\} \tag{5.9}$$

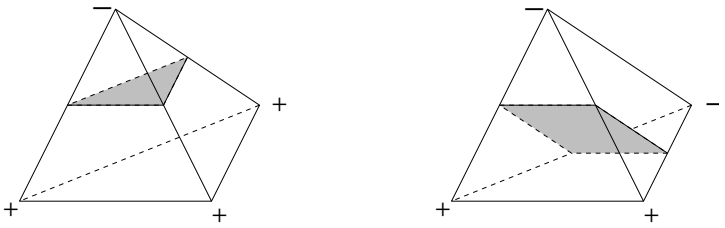

Figure 5.2.: Sign pattern of d_h on $T \in \mathcal{T}'$ and corresponding interface segment $\Gamma_T = T \cap \Gamma$ (in gray): either a triangle or a quadrilateral.

which consists of piecewise planar segments $F = \Gamma_T \subset \Gamma_h$, where

$$\Gamma_T := T \cap \Gamma_h \tag{5.10}$$

for $T \in \mathcal{T}'$.

The interface mesh size parameter h_Γ is the maximal diameter of these segments. This (maximal) diameter is approximately the (maximal) diameter of the tetrahedra in $\mathcal{T}'$ that contain the discrete interface, i.e., h_Γ is approximately the maximal diameter of the tetrahedra in $\mathcal{T}'$ *that are close to the interface*. In Figure 5.1 we illustrate this construction for the two-dimensional case. Note that in general the segments of Γ_h *are not aligned with the faces of the tetrahedral triangulation* $\mathcal{T}'_h$.

Each of the planar segments of Γ_h is either a triangle or a quadrilateral, depending on the sign pattern of d_h on the corresponding $T \in \mathcal{T}'$, cf. Figure 5.2. By construction the vertices of a planar segment Γ_T are located on those edges of T along which d_h changes its sign. If there are two positive and two negative values of d_h on the vertices of T, then the corresponding interface segment Γ_T is a quadrilateral. In all other cases Γ_T is a triangle. The quadrilaterals can (formally) be divided into two triangles. Thus Γ_h consists of a set $\mathcal{F}_h$ of triangular faces.

Special cases may occur if some of the values of d_h on the vertices of T are equal to zero. Let $0 \leq n_0 \leq 4$ be the number of these zero values. In the following we discuss the shape of Γ_T in all the cases $n_0 = 0, 1, 2, 3, 4$.

- $n_0 = 0$ is not a special case, the situation is as depicted in Figure 5.2 which was discussed in the foregoing paragraph.
- For $n_0 = 1, 2$ we distinguish two cases: If the other $4 - n_0$ non-zero values have the same sign, then Γ_T is a point ($n_0 = 1$) or a line segment

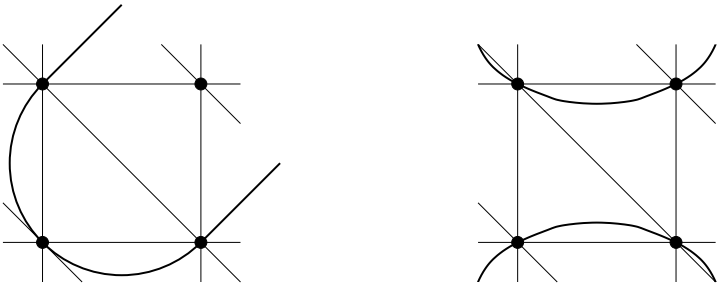

Figure 5.3.: 2D examples for interface degeneration, where the proposed interface reconstruction fails. Left: curvature κ too large for grid resolution ($|\kappa| \geq \frac{2}{h_\Gamma}$). Right: distance d between interfaces too small for grid resolution ($d \leq h_\Gamma$).

($n_0 = 2$) and can be ignored as $\text{meas}_2\ \Gamma_T = 0$. Otherwise the non-zero values are of different sign yielding $3 - n_0$ edges with a change of sign, as a simple case differentiation shows. Thus Γ_T has 3 vertices, hence Γ_T is a triangle.

- In the case $n_0 = 3$ the interface segment Γ_T is equal to a face of T. Then one has to take care that this face is not counted twice (additionally by the neighboring tetrahedron in $\mathcal{T}'$ which also has Γ_T as one of its faces) when computing a surface integral on Γ_h.
- If $n_0 = 4$ then the interface segment is not 2D but 3D ($\Gamma_T = T$) which, of course, makes not much sense. If such a situation occurs, the corresponding segment is ignored and a warning is given. This is typically an indication that the grid is too coarse to represent the interface properly, cf. Figure 5.3.

For the example 5.2, in which Γ is a sphere, the resulting polygonal approximations Γ_h for $h = \frac{1}{5}$ and $h = \frac{1}{10}$ resp. are shown in Figure 5.4. Here the radius is chosen as $r = \frac{1}{2}$, see also the numerical experiment presented in Section 10.3.

Remark 5.5
Related to the assumptions (5.5)-(5.7) we note the following. If we assume

$$|I(d_h)(\mathbf{x}) - d(\mathbf{x})| \leq c\, h_\Gamma^2$$

for all $\mathbf{x}$ in a neighborhood of Γ, which is reasonable for a smooth d and piecewise quadratic d_h, then for $\mathbf{x} \in \Gamma_h$ we have $|d(\mathbf{x})| = |d(\mathbf{x}) - I(d_h)(\mathbf{x})| \leq c\, h_\Gamma^2$ and thus (5.5) is satisfied. Instead of (5.6), (5.7) we consider the sufficient

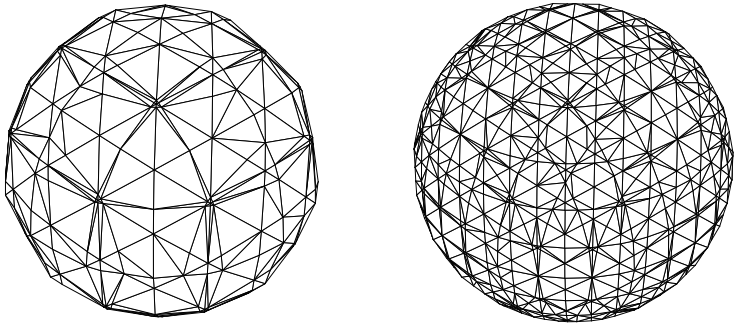

Figure 5.4.: Approximate interface Γ_h for the example from Section 10.3 on a coarse grid (left) and after one refinement (right).

condition (5.8). We assume

$$\|\nabla I(d_h)(\mathbf{x}) - \nabla d(\mathbf{x})\| \leq c\, h_\Gamma$$

for all $\mathbf{x}$ in a neighborhood of Γ ($\mathbf{x}$ not on an edge), which again is reasonable for a smooth d and piecewise quadratic d_h. Due to $\|\nabla d\| = 1$ we then also have $\|\nabla I(d_h)(\mathbf{x})\| = 1 + \mathcal{O}(h)$, in a neighborhood of Γ. For $\mathbf{x} \in \Gamma_h$ (not on an edge) we obtain

$$\begin{aligned}
\|\mathbf{n}_h(\mathbf{x}) - \mathbf{n}(\mathbf{x})\| &= \left\| \frac{\nabla I(d_h)(\mathbf{x})}{\|\nabla I(d_h)(\mathbf{x})\|} - \nabla d(\mathbf{x}) \right\| \\
&\leq \left| \frac{1}{\|\nabla I(d_h)(\mathbf{x})\|} - 1 \right| \cdot \|\nabla I(d_h)(\mathbf{x})\| + \|\nabla I(d_h)(\mathbf{x}) - \nabla d(\mathbf{x})\| \\
&\leq c\, h_\Gamma,
\end{aligned}$$

and thus (5.8) is satisfied (for h_Γ sufficiently small). ◇

5.2. Consequences of Strang's Lemma

We assume that a piecewise planar surface Γ_h is known, which is close to the interface Γ in the sense of (5.5)–(5.7). The induced polyhedral approximations of the subdomains are $\Omega_{1,h} = \mathrm{int}(\Gamma_h)$ (region in the interior of Γ_h) and $\Omega_{2,h} = \Omega \setminus \overline{\Omega}_{1,h}$. Furthermore, we define the piecewise constant approximation of the

density ρ_h by $\rho_h = \rho_i$ on $\Omega_{i,h}$. We assume that for $\mathbf{v}_h \in \mathbf{V}_h$ the integrals in

$$(\rho_h \mathbf{g}, \mathbf{v}_h) = \rho_1 \int_{\Omega_{1,h}} \mathbf{g} \cdot \mathbf{v}_h \, d\mathbf{x} + \rho_2 \int_{\Omega_{2,h}} \mathbf{g} \cdot \mathbf{v}_h \, d\mathbf{x}$$

can be computed with high accuracy. This can be realized efficiently in our implementation because if one applies the standard finite element assembling strategy by using a loop over all tetrahedra $T \in \mathcal{T}_h$, then $T \cap \Omega_{i,h}$ is either empty or T or a relatively simple polygonal subdomain (due to the construction of Γ_h). For more details we refer to Section 5.4.4.

The discretization of (5.1) is as follows: determine $(\mathbf{u}_h, p_h) \in \mathbf{V}_h \times Q_h$ such that

$$\begin{aligned} a(\mathbf{u}_h, \mathbf{v}_h) + b(\mathbf{v}_h, p_h) &= (\rho_h \mathbf{g}, \mathbf{v}_h) + f_{\Gamma_h}(\mathbf{v}_h) \qquad \text{for all } \mathbf{v}_h \in \mathbf{V}_h, \\ b(\mathbf{u}_h, q_h) &= 0 \qquad \text{for all } q_h \in Q_h. \end{aligned} \tag{5.11}$$

The approximation $f_{\Gamma_h}(\mathbf{v}_h)$ of $f_\Gamma(\mathbf{v}_h)$ is discussed in Section 5.3.1 below. Using standard finite element error analysis (Strang lemma) we get a discretization error bound. In our applications we are particularly interested in problems with $\mu \ll 1$. Therefore, in the next theorem we give a discretization error bound that shows the dependence on μ.

Theorem 5.6
Let $(\mathbf{u}^*, p^*)$, $(\mathbf{u}_h, p_h)$ be the solution of (5.1) and (5.11), respectively. Then the error bound

$$\begin{aligned} \mu \, \|\mathbf{u}_h - \mathbf{u}^*\|_1 + \|p_h - p^*\|_{L^2} \le \; c\Big(& \mu \inf_{\mathbf{v}_h \in \mathbf{V}_h} \|\mathbf{v}_h - \mathbf{u}^*\|_1 + \inf_{q_h \in Q_h} \|q_h - p^*\|_{L^2} \\ & + \sup_{\mathbf{v}_h \in \mathbf{V}_h} \frac{|(\rho \mathbf{g}, \mathbf{v}_h) - (\rho_h \mathbf{g}, \mathbf{v}_h)|}{\|\mathbf{v}_h\|_1} \\ & + \sup_{\mathbf{v}_h \in \mathbf{V}_h} \frac{|f_\Gamma(\mathbf{v}_h) - f_{\Gamma_h}(\mathbf{v}_h)|}{\|\mathbf{v}_h\|_1} \Big) \end{aligned} \tag{5.12}$$

holds with a constant c independent of h, μ and ρ.

Proof. The result follows from a scaling argument. For $f \in \mathbf{V}_0'$, $f_h \in \mathbf{V}_h'$ let $(\hat{\mathbf{u}}, \hat{p})$, $(\hat{\mathbf{u}}_h, \hat{p}_h)$ be the solutions of the μ-independent Stokes problems

$$\begin{aligned} \int_\Omega \nabla \hat{\mathbf{u}} \, \nabla \mathbf{v} \, d\mathbf{x} + b(\mathbf{v}, \hat{p}) &= f(\mathbf{v}) \quad \text{for all } \mathbf{v} \in \mathbf{V}_0, \\ b(\hat{\mathbf{u}}, q) &= 0 \quad \text{for all } q \in Q, \end{aligned} \tag{5.13}$$

$$\begin{aligned} \int_\Omega \nabla \hat{\mathbf{u}}_h \, \nabla \mathbf{v}_h \, d\mathbf{x} + b(\mathbf{v}_h, \hat{p}_h) &= f_h(\mathbf{v}_h) \quad \text{for all } \mathbf{v}_h \in \mathbf{V}_h, \\ b(\hat{\mathbf{u}}_h, q_h) &= 0 \quad \text{for all } q_h \in Q_h. \end{aligned} \tag{5.14}$$

Standard error analysis for Stokes equations, using the Strang lemma, yields

$$\|\hat{\mathbf{u}}_h - \hat{\mathbf{u}}\|_1 + \|\hat{p}_h - \hat{p}\|_{L^2} \le c \Big(\inf_{\mathbf{v}_h \in \mathbf{V}_h} \|\mathbf{v}_h - \hat{\mathbf{u}}\|_1 + \inf_{q_h \in Q_h} \|q_h - \hat{p}\|_{L^2} + \sup_{\mathbf{v}_h \in \mathbf{V}_h} \frac{|f(\mathbf{v}_h) - f_h(\mathbf{v}_h)|}{\|\mathbf{v}_h\|_1} \Big), \tag{5.15}$$

with a constant c independent of f, f_h and h. Now note that $(\mathbf{u}^*, p^*)$ satisfies (5.13) with $\hat{\mathbf{u}} = \mathbf{u}^*$, $\hat{p} = \frac{1}{\mu} p^*$, $f(\mathbf{v}) = \frac{1}{\mu}\big((\rho \mathbf{g}, \mathbf{v}) + f_\Gamma(\mathbf{v})\big)$ and $(\mathbf{u}_h, p_h)$ satisfies (5.14) with $\hat{\mathbf{u}}_h = \mathbf{u}_h$, $\hat{p}_h = \frac{1}{\mu} p_h$, $f_h(\mathbf{v}_h) = \frac{1}{\mu}\big((\rho_h \mathbf{g}, \mathbf{v}_h) + f_{\Gamma_h}(\mathbf{v}_h)\big)$. The result in (5.15) then yields (5.12). □

Remark 5.7
We assume Ω to be convex and thus the problem (5.13) is H^2-regular. Using a standard duality argument it follows that

$$\|\hat{\mathbf{u}} - \hat{\mathbf{u}}_h\|_{L_2} \le ch \left(\|\hat{\mathbf{u}} - \hat{\mathbf{u}}_h\|_1 + \|\hat{p} - \hat{p}_h\|_{L_2} \right).$$

Due to $\hat{\mathbf{u}} = \mathbf{u}^*$, $\hat{\mathbf{u}}_h = \mathbf{u}_h$, $\hat{p} = \frac{1}{\mu} p^*$, $\hat{p}_h = \frac{1}{\mu} p_h$, (cf. proof of Theorem 5.6) we get

$$\|\mathbf{u}^* - \mathbf{u}_h\|_{L_2} \le ch \left(\|\mathbf{u}^* - \mathbf{u}_h\|_1 + \frac{1}{\mu} \|p^* - p_h\|_{L_2} \right)$$

with a constant c independent of μ and h. ◇

Corollary 5.8
Let $(\mathbf{u}^*, p^*)$, $(\mathbf{u}_h, p_h)$ be as in Theorem 5.6 and define

$$r_h := \sup_{\mathbf{v}_h \in \mathbf{V}_h} \frac{|(\rho \mathbf{g}, \mathbf{v}_h) - (\rho_h \mathbf{g}, \mathbf{v}_h)|}{\|\mathbf{v}_h\|_1} + \sup_{\mathbf{v}_h \in \mathbf{V}_h} \frac{|f_\Gamma(\mathbf{v}_h) - f_{\Gamma_h}(\mathbf{v}_h)|}{\|\mathbf{v}_h\|_1}.$$

The following holds:

$$\|\mathbf{u}_h - \mathbf{u}^*\|_1 \le c \Big(\inf_{\mathbf{v}_h \in \mathbf{V}_h} \|\mathbf{v}_h - \mathbf{u}^*\|_1 + \frac{1}{\mu} \inf_{q_h \in Q_h} \|q_h - p^*\|_{L^2} + \frac{1}{\mu} r_h \Big), \tag{5.16}$$

$$\|\mathbf{u}_h - \mathbf{u}^*\|_{L_2} \le ch \Big(\inf_{\mathbf{v}_h \in \mathbf{V}_h} \|\mathbf{v}_h - \mathbf{u}^*\|_1 + \frac{1}{\mu} \inf_{q_h \in Q_h} \|q_h - p^*\|_{L^2} + \frac{1}{\mu} r_h \Big), \tag{5.17}$$

$$\|p_h - p^*\|_{L^2} \le c \Big(\mu \inf_{\mathbf{v}_h \in \mathbf{V}_h} \|\mathbf{v}_h - \mathbf{u}^*\|_1 + \inf_{q_h \in Q_h} \|q_h - p^*\|_{L^2} + r_h \Big), \tag{5.18}$$

with constants c independent of h, μ and ρ. We observe that if $\mu \ll 1$ then in the velocity error we have an error amplification effect proportional to $\frac{1}{\mu}$. This effect does not occur in the discretization error of the pressure.

Remark 5.9
For small μ values the discretization can be improved by adding a grad-div stabilization term to the Stokes equations. In [OR04] it is shown that with this term the velocity errors (in $\|\cdot\|_1$) are proportional to $\mu^{-1/2}$ (instead of μ^{-1}) and that for small μ values the discretization errors for the velocity are significantly smaller than without this grad-div term. ◇

We comment on the terms occurring in the bound in (5.12). As explained above (Remark 5.1), the solution $\mathbf{u}^*$ of (5.1) is smooth and thus with standard finite element spaces $\mathbf{V}_h$ for the velocity (e.g., P_1 or P_2) we obtain

$$\inf_{\mathbf{v}_h \in \mathbf{V}_h} \|\mathbf{v}_h - \mathbf{u}^*\|_1 \leq ch.$$

Due to (5.5) we get $|\operatorname{meas}_3(\Omega_i) - \operatorname{meas}_3(\Omega_{i,h})| \leq ch_\Gamma^2$, $i = 1, 2$, and using this we obtain

$$\begin{aligned} |(\rho\mathbf{g}, \mathbf{v}_h) - (\rho_h\mathbf{g}, \mathbf{v}_h)| &\leq \sum_{i=1}^{2} \rho_i \left| \int_{\Omega_i} \mathbf{g} \cdot \mathbf{v}_h \, d\mathbf{x} - \int_{\Omega_{i,h}} \mathbf{g} \cdot \mathbf{v}_h \, d\mathbf{x} \right| \\ &\leq c(\rho_1 + \rho_2)\, h_\Gamma \|\mathbf{v}_h\|_1, \end{aligned}$$

and thus an $\mathcal{O}(h_\Gamma)$ bound for the third term in (5.12).

The remaining two terms in (5.12) are less easy to handle. In Section 5.3 we treat the fourth term. It is shown that a (not so obvious) approximation method based on a Laplace-Beltrami representation results in a $\mathcal{O}(h_\Gamma)$ bound for this term whereas a naive Laplace-Beltrami approximation, which is used in the literature, only yields $\mathcal{O}(\sqrt{h_\Gamma})$ if it is applied to a piecewise planar interface approximation.

The second term in (5.12) is discussed in Section 5.4.1. It is shown that standard finite element spaces (e.g., P_0 or P_1) lead to an error $\inf_{q_h \in Q_h} \|q_h - p^*\|_{L^2} \sim \sqrt{h_\Gamma}$. This motivates the use of another pressure finite element space, as explained in Section 5.4.2, which has much better approximation properties for functions that are piecewise smooth but discontinuous across Γ_h.

Remark 5.10
Consider the problem as in Example 5.2. Then $\mathbf{u}^* = 0$, $\mathbf{g} = 0$ and the bound in (5.12) simplifies to

$$\begin{aligned} &\mu \|\mathbf{u}_h\|_1 + \|p_h - p^*\|_{L^2} \\ &\qquad \leq\ c\Big(\inf_{q_h \in Q_h} \|q_h - p^*\|_{L^2} + \sup_{\mathbf{v}_h \in \mathbf{V}_h} \frac{|f_\Gamma(\mathbf{v}_h) - f_{\Gamma_h}(\mathbf{v}_h)|}{\|\mathbf{v}_h\|_1} \Big). \end{aligned} \tag{5.19}$$

◇

5.3. Discretization of the surface tension force

In this section we discuss the discretization of the surface tension force f_Γ by a Laplace-Beltrami technique and analyze the discretization error $\|f_\Gamma - f_{\Gamma_h}\|_{\mathbf{V}_h'}$. Based on this analysis we introduce an improved discretization $\tilde{f}_{\Gamma_h}$ which has a higher order of convergence. The results are also presented in [GR07b].

5.3.1. Laplace-Beltrami discretization

In this section we explain how the localized surface tension force term $f_\Gamma(\mathbf{v}_h)$ in (2.21) is approximated. We use the technique presented in [Bän01, Dzi91, GRR06]. For this we first need some notions from differential geometry.

Let U be an open subset in $\mathbb{R}^3$ and Γ a connected C^2 compact hypersurface contained in U. For a sufficiently smooth function $g : U \to \mathbb{R}$ the tangential derivative (along Γ) is defined by projecting the derivative on the tangent space of Γ, i. e.

$$\nabla_\Gamma g = \nabla g - (\nabla g \cdot \mathbf{n}_\Gamma)\,\mathbf{n}_\Gamma. \tag{5.20}$$

Note that the tangential derivative can be written as $\nabla_\Gamma g = \mathbf{P}\nabla g$ with $\mathbf{P}$ defined as in (5.3).

The *Laplace-Beltrami operator* of g on Γ is defined by

$$\Delta_\Gamma g := \nabla_\Gamma \cdot \nabla_\Gamma g.$$

It can be shown that $\nabla_\Gamma g$ and $\Delta_\Gamma g$ depend only on values of g on Γ. For vector valued functions f, $g : \Gamma \to \mathbb{R}^3$ we define

$$\Delta_\Gamma f := (\Delta_\Gamma f_1, \Delta_\Gamma f_2, \Delta_\Gamma f_3)^T, \qquad \nabla_\Gamma f \cdot \nabla_\Gamma g := \sum_{i=1}^{3} \nabla_\Gamma f_i \cdot \nabla_\Gamma g_i.$$

We recall the following basic result from differential geometry.

Theorem 5.11
Let $\mathrm{id}_\Gamma : \Gamma \to \mathbb{R}^3$ be the identity on Γ and $\kappa = \kappa_1 + \kappa_2$ the sum of the principal curvatures. For all sufficiently smooth vector functions $\mathbf{v}$ on Γ the following holds:

$$\int_\Gamma \kappa\,\mathbf{n}_\Gamma \cdot \mathbf{v}\,ds = \int_\Gamma (\Delta_\Gamma\,\mathrm{id}_\Gamma) \cdot \mathbf{v}\,ds = -\int_\Gamma \nabla_\Gamma\,\mathrm{id}_\Gamma \cdot \nabla_\Gamma \mathbf{v}\,ds. \tag{5.21}$$

In a finite element setting (which is based on a weak formulation) it is natural to use the expression on the right-hand side in (5.21) as a starting point for the discretization. This idea is used in, for example, [Dzi91, Bän01, GT05, GRR06, Hys06, MCN03]. In this discretization we use the approximation Γ_h of Γ.

Given an approximate interface Γ_h the localized force term $f_\Gamma(\mathbf{v}_h)$ is approximated by

$$f_{\Gamma_h}(\mathbf{v}_h) := -\tau \int_{\Gamma_h} \nabla_{\Gamma_h} \mathrm{id}_{\Gamma_h} \cdot \nabla_{\Gamma_h} \mathbf{v}_h \, ds, \qquad \mathbf{v}_h \in \mathbf{V}_h. \tag{5.22}$$

Under the assumptions (5.5)-(5.7) on the family $\{\Gamma_h\}_{h_\Gamma>0}$ we will derive, in Section 5.3.3, a bound for the approximation error

$$\sup_{\mathbf{v}_h \in \mathbf{V}_h} \frac{f_\Gamma(\mathbf{v}_h) - f_{\Gamma_h}(\mathbf{v}_h)}{\|\mathbf{v}_h\|_1}, \quad \text{with } f_{\Gamma_h}(\mathbf{v}_h) \text{ as in (5.22).} \tag{5.23}$$

Remark 5.12
From Theorem 5.11, the fact that $f_\Gamma(\mathbf{v}) = \tau \int_\Gamma \kappa \mathbf{v} \cdot \mathbf{n} \, ds$ is a bounded linear functional on $\mathbf{V}_0$ and a density argument it follows that the linear functional

$$f_\Gamma : \ \mathbf{v} \to -\tau \int_\Gamma \nabla_\Gamma \mathrm{id}_\Gamma \cdot \nabla_\Gamma \mathbf{v} \, ds \ , \qquad \mathbf{v} \in \left(C_0^\infty(\Omega)\right)^3, \tag{5.24}$$

has a unique bounded extension to $\mathbf{V}_0$. Therefore, for $f_\Gamma : \mathbf{V}_0 \to \mathbb{R}$ we can use both the representation in (2.19) and the one in (5.24) (these are the same on a dense subset). This, however, is *not* the case for f_{Γ_h}. Because Γ_h is not sufficiently smooth, a partial integration result as in Theorem 5.11 does not hold. The linear functional

$$\mathbf{v} \to -\tau \int_{\Gamma_h} \nabla_{\Gamma_h} \mathrm{id}_{\Gamma_h} \cdot \nabla_{\Gamma_h} \mathbf{v} \, ds$$

is *not* necessarily bounded on $\mathbf{V}_0$. For this reason the restriction to $\mathbf{v}_h$ from the finite element space $\mathbf{V}_h$ in (5.22) and (5.23) is essential. ◇

Remark 5.13
At many places in this section, for example in (5.21), (5.2) and (implicitly) in (5.5), and also in the analysis presented in the next section the assumption that Γ is a C^2 smooth interface plays a crucial role. We do not know any literature in which for a Navier-Stokes incompressible two-phase flow problem with surface tension smoothness properties of the interface are analyzed. In [AMY00] and [AMS01] a two-phase *Stokes* flow problem *without* surface

tension in which the evolution is driven by the gravity force is analyzed. In [AMY00] it is proved that if the initial configuration has a C^2 smooth interface $\Gamma = \Gamma(0)$ then for arbitrary finite time $t > 0$ the interface $\Gamma(t)$ is a surface of class $C^{2-\varepsilon}$ for arbitrary $\varepsilon \in (0, 2]$. In [AMS01] it is shown that if $\Gamma(0)$ is a $C^{2+\ell}$ smooth surface, with $\ell > 0$, then $\Gamma(t)$ is of class $C^{2+\ell}$, too, for all $t \in [0, T]$ and $T > 0$ sufficiently small. ◇

5.3.2. Extensions and projections

In this section we collect some results that will be used in the analysis in Section 5.3.3. The techniques that we use come from the paper [DD07]. For proofs of certain results we will refer to that paper.

We introduce a locally (in a neighborhood of Γ) orthogonal coordinate system by using the projection $\mathbf{p} : U \to \Gamma$:

$$\mathbf{p}(\mathbf{x}) = \mathbf{x} - d(\mathbf{x})\mathbf{n}(\mathbf{x}) \quad \text{for all } \mathbf{x} \in U.$$

We assume that the decomposition $\mathbf{x} = \mathbf{p}(\mathbf{x}) + d(\mathbf{x})\mathbf{n}(\mathbf{x})$ is unique for all $\mathbf{x} \in U$. Note that

$$\mathbf{n}(\mathbf{x}) = \mathbf{n}(\mathbf{p}(\mathbf{x})) \quad \text{for all } \mathbf{x} \in U.$$

We use an extension operator defined as follows. For a (scalar) function v defined on Γ we define

$$v^e_\Gamma(\mathbf{x}) := v(\mathbf{x} - d(\mathbf{x})\mathbf{n}(\mathbf{x})) = v(\mathbf{p}(\mathbf{x})) \quad \text{for all } \mathbf{x} \in U,$$

i.e., v is extended along normals on Γ. We will also need extensions of functions defined on Γ_h to U. This is done again by extending along normals $\mathbf{n}(\mathbf{x})$. For v defined on Γ_h we define, for $\mathbf{x} \in \Gamma_h$,

$$v^e_{\Gamma_h}(\mathbf{x} + \alpha\mathbf{n}(\mathbf{x})) := v(\mathbf{x}) \quad \text{for all } \alpha \in \mathbb{R} \quad \text{with } x + \alpha\mathbf{n}(\mathbf{x}) \in U. \tag{5.25}$$

The projection $\mathbf{p}$ and the extensions v^e_Γ, $v^e_{\Gamma_h}$ are illustrated in Figure 5.5.

We define a discrete analogon of the orthogonal projection $\mathbf{P}$:

$$\mathbf{P}_h(\mathbf{x}) := \mathbf{I} - \mathbf{n}_h(\mathbf{x})\mathbf{n}_h(\mathbf{x})^T \quad \text{for } \mathbf{x} \in \Gamma_h, \ \mathbf{x} \text{ not on an edge.}$$

The tangential derivative along Γ_h can be written as $\nabla_{\Gamma_h} g = \mathbf{P}_h \nabla g$. In the analysis a further technical assumption is used, namely that the neighborhood U of Γ is sufficiently small in the following sense. We assume that U is a strip of width $\delta > 0$ with

$$\delta^{-1} > \max_{i=1,2} \|\kappa_i(\mathbf{x})\|_{L^\infty(\Gamma)}. \tag{5.26}$$

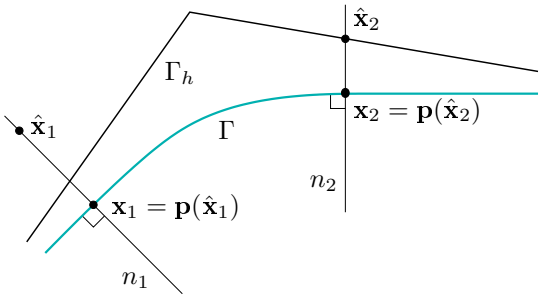

Figure 5.5.: Example for projection $\mathbf{p}$ and construction of extension operators. n_1 and n_2 are straight lines perpendicular to Γ. For v defined on Γ we have $v_\Gamma^e \equiv v(\mathbf{x}_1)$ on n_1. For v_h defined on Γ_h we have $v_{\Gamma_h}^e \equiv v_h(\hat{\mathbf{x}}_2)$ on n_2.

Assumption 5.14
In the remainder of the section we assume that (5.5), (5.6), (5.7)and (5.26) hold. ◇

We present two lemmas from [DD07]. Proofs are elementary and can be found in [DD07].

Lemma 5.15
For the projection operator $\mathbf{P}$ and the Hessian $\mathbf{H}$ the relation

$$\mathbf{P}(\mathbf{x})\mathbf{H}(\mathbf{x}) = \mathbf{H}(\mathbf{x})\mathbf{P}(\mathbf{x}) = \mathbf{H}(\mathbf{x}) \quad \text{for all } \mathbf{x} \in U$$

holds. For v defined on Γ and sufficiently smooth the following holds:

$$\nabla_{\Gamma_h} v_\Gamma^e(\mathbf{x}) = \mathbf{P}_h(\mathbf{x})\big(\mathbf{I} - d(\mathbf{x})\mathbf{H}(\mathbf{x})\big)\mathbf{P}(\mathbf{x})\nabla_\Gamma v(\mathbf{p}(\mathbf{x})) \quad \text{a.e. on } \Gamma_h. \tag{5.27}$$

Proof. Given in Section 2.3 in [DD07]. □

In (5.27) (and also below) we have results "a.e. on Γ_h" because quantities (derivatives, $\mathbf{P}_h$, etc.) are not well-defined on the edges of the triangulation Γ_h.

Lemma 5.16
For $\mathbf{x} \in \Gamma_h$ (not on an edge) define

$$\mu(\mathbf{x}) = \big[\Pi_{i=1}^2(1 - d(\mathbf{x})\kappa_i(\mathbf{x}))\big]\mathbf{n}(\mathbf{x})^T\mathbf{n}_h(\mathbf{x}), \tag{5.28}$$

$$\mathbf{A}(\mathbf{x}) = \frac{1}{\mu(\mathbf{x})}\mathbf{P}(\mathbf{x})\big[\mathbf{I} - d(\mathbf{x})\mathbf{H}(\mathbf{x})\big]\mathbf{P}_h(\mathbf{x})\big[\mathbf{I} - d(\mathbf{x})\mathbf{H}(\mathbf{x})\big]\mathbf{P}(\mathbf{x}). \tag{5.29}$$

Let $\mathbf{A}^e_{\Gamma_h}$ be the extension of $\mathbf{A}$ as in (5.25). The following identity holds for functions v and ψ that are defined on Γ_h and sufficiently smooth:

$$\int_{\Gamma_h} \nabla_{\Gamma_h} v \cdot \nabla_{\Gamma_h} \psi \, ds = \int_{\Gamma} \mathbf{A}^e_{\Gamma_h} \nabla_\Gamma v^e_{\Gamma_h} \cdot \nabla_\Gamma \psi^e_{\Gamma_h} \, ds. \tag{5.30}$$

Proof. Given in Section 2.3 in [DD07]. □

Due to the assumptions in (5.6) and (5.26) we have $\operatorname{ess\,inf}_{\mathbf{x}\in\Gamma_h} \mu(\mathbf{x}) > 0$ and thus $\mathbf{A}(\mathbf{x})$ is well-defined.

5.3.3. Discretization error analysis

We are interested in the difference between the terms

$$\tau \int_\Gamma \nabla_\Gamma \operatorname{id}_\Gamma \cdot \nabla_\Gamma \mathbf{v}_h \, ds \quad \text{and} \quad \tau \int_{\Gamma_h} \nabla_{\Gamma_h} \operatorname{id}_{\Gamma_h} \cdot \nabla_{\Gamma_h} \mathbf{v}_h \, ds \quad \text{for } \mathbf{v}_h \in \mathbf{V}_h.$$

Since $\nabla_\Gamma \operatorname{id}_\Gamma \cdot \nabla_\Gamma \mathbf{v}_h = \sum_{i=1}^3 \nabla_\Gamma (\operatorname{id}_\Gamma)_i \cdot \nabla_\Gamma (\mathbf{v}_h)_i$ we consider only one term in this sum, say the i-th. We write id_Γ and v for the *scalar* functions $(\operatorname{id}_\Gamma)_i$ and $(\mathbf{v}_h)_i$, respectively. We write $\operatorname{id}_{\Gamma_h}$ for $(\operatorname{id}_{\Gamma_h})_i$. Note that

$$\nabla_\Gamma \operatorname{id}_\Gamma = \mathbf{P} \nabla \operatorname{id}_\Gamma = \mathbf{P} e_i, \quad \nabla_{\Gamma_h} \operatorname{id}_{\Gamma_h} = \mathbf{P}_h \nabla \operatorname{id}_{\Gamma_h} = \mathbf{P}_h e_i,$$

with e_i the i-th basis vector in $\mathbb{R}^3$. We introduce scalar versions of the functionals f_Γ and f_{Γ_h} defined in (5.24) and (5.22) (without loss of generality we can take $\tau := 1$):

$$g(v) := \int_\Gamma \nabla_\Gamma \operatorname{id}_\Gamma \cdot \nabla_\Gamma v \, ds, \quad g_h(v) := \int_{\Gamma_h} \nabla_{\Gamma_h} \operatorname{id}_{\Gamma_h} \cdot \nabla_{\Gamma_h} v \, ds.$$

As noted in Remark 5.12, g is a bounded linear functional on $H^1(U)$. To guarantee that g_h and the extension operator in (5.25) are well-defined we assume $v \in H^1(\Gamma_h) \cap C(\Gamma_h)$. Therefore, in the analysis in this section we use the subspace W of $H^1(U)$ consisting of functions whose restriction to Γ_h belongs to $H^1(\Gamma_h) \cap C(\Gamma_h)$.

Remark 5.17
If we use a Hood-Taylor pair $\mathbf{V}_h \times Q_h$ in the discretization of the Navier-Stokes equations, then the i-th component $v \in V_h$ of $\mathbf{v}_h \in \mathbf{V}_h = (V_h)^3$ is continuous and piecewise polynomial (on the tetrahedral triangulation $\mathcal{S}$). Thus $v \in W$ holds. ◇

In this section we first derive, for $v \in W$, a bound for $|g(v) - g_h(v)|$ in terms of $\|v\|_{1,U} := \|v\|_{H^1(U)}$ and $\|\nabla_{\Gamma_h} v\|_{L^2(\Gamma_h)}$. This bound is given in Corollary 5.18. Using this bound we then derive a bound for

$$\sup_{v \in V_h} \frac{|g(v) - g_h(v)|}{\|v\|_1},$$

cf. Theorem 5.22. This immediately implies a bound for the approximation error as in (5.23), cf. Corollary 5.23.

The analysis is based on the following splitting:

$$\begin{aligned}
& g(v) - g_h(v) \\
&= \int_\Gamma \nabla_\Gamma \mathrm{id}_\Gamma \cdot \nabla_\Gamma v \, ds - \int_{\Gamma_h} \nabla_{\Gamma_h} \mathrm{id}_\Gamma^e \cdot \nabla_{\Gamma_h} v \, ds \\
&\quad + \int_{\Gamma_h} \nabla_{\Gamma_h} (\mathrm{id}_\Gamma^e - \mathrm{id}_{\Gamma_h}) \cdot \nabla_{\Gamma_h} v \, ds \\
&= \int_\Gamma \nabla_\Gamma \mathrm{id}_\Gamma \cdot \nabla_\Gamma v \, ds - \int_\Gamma \mathbf{A}_{\Gamma_h}^e \nabla_\Gamma \mathrm{id}_\Gamma \cdot \nabla_\Gamma v_{\Gamma_h}^e \, ds \qquad \text{(cf. (5.30))} \\
&\quad + \int_{\Gamma_h} \nabla_{\Gamma_h} (\mathrm{id}_\Gamma^e - \mathrm{id}_{\Gamma_h}) \cdot \nabla_{\Gamma_h} v \, ds \\
&= \int_\Gamma \nabla_\Gamma \mathrm{id}_\Gamma \cdot \nabla_\Gamma (v - v_{\Gamma_h}^e) \, ds + \int_\Gamma (\mathbf{I} - \mathbf{A}_{\Gamma_h}^e) \nabla_\Gamma \mathrm{id}_\Gamma \cdot \nabla_\Gamma v_{\Gamma_h}^e \, ds \\
&\quad + \int_{\Gamma_h} \nabla_{\Gamma_h} (\mathrm{id}_\Gamma^e - \mathrm{id}_{\Gamma_h}) \cdot \nabla_{\Gamma_h} v \, ds. \qquad (5.31)
\end{aligned}$$

In the corollary below we derive bounds for the three terms in (5.31). Note that the first two terms do *not* involve id_{Γ_h}.

Corollary 5.18
The three terms in (5.31) can be bounded by

$$\left| \int_\Gamma \nabla_\Gamma \mathrm{id}_\Gamma \cdot \nabla_\Gamma (v - v_{\Gamma_h}^e) \, ds \right| \leq c \, h_\Gamma \, \|v\|_{1,U}, \qquad (5.32)$$

$$\left| \int_\Gamma (\mathbf{I} - \mathbf{A}_{\Gamma_h}^e) \nabla_\Gamma \mathrm{id}_\Gamma \cdot \nabla_\Gamma v_{\Gamma_h}^e \, ds \right| \leq c \, h_\Gamma^2 \, \|\nabla_{\Gamma_h} v\|_{L^2(\Gamma_h)}, \qquad (5.33)$$

$$\left| \int_{\Gamma_h} \nabla_{\Gamma_h} (\mathrm{id}_\Gamma^e - \mathrm{id}_{\Gamma_h}) \cdot \nabla_{\Gamma_h} v \, ds \right| \leq c \, h_\Gamma \, \|\nabla_{\Gamma_h} v\|_{L^2(\Gamma_h)}, \qquad (5.34)$$

and thus

$$|g(v) - g_h(v)| \leq c \, h_\Gamma \, \|v\|_{1,U} + c \, h_\Gamma^2 \, \|\nabla_{\Gamma_h} v\|_{L^2(\Gamma_h)} + c \, h_\Gamma \, \|\nabla_{\Gamma_h} v\|_{L^2(\Gamma_h)}$$

holds for all $v \in W$.

Proof. (5.32)–(5.34) are proved in Lemma 4.1–4.3 in [GR07b]. These bounds together with the splitting (5.31) yield the result. □

In view of Corollary 5.18 and the error measure in (5.23) we want to derive a bound for $\|\nabla_{\Gamma_h} v\|_{L^2(\Gamma_h)}$ in terms of $\|v\|_1$ for v from the scalar finite element space V_h. An obvious approach is to apply an inverse inequality combined with a trace theorem, resulting in:

$$\|\nabla_{\Gamma_h} v\|_{L^2(\Gamma_h)} \leq c\, h_{\min}^{-1} \|v\|_{L^2(\Gamma_h)} \leq c\, h_{\min}^{-1} \|v\|_1 \quad \text{for all } v \in V_h. \tag{5.35}$$

This, however, is too crude (cf. the bound in Corollary 5.18). In order to be able to derive a better bound than the one in (5.35) we have to introduce some further assumptions related to the family of triangulations $\{\Gamma_h\}_{h_\Gamma > 0}$. We assume that to each triangulation $\Gamma_h = \cup_{F \in \mathcal{F}_h} F$ there can be associated a set of tetrahedra $\mathcal{S}_h^\Gamma$ with the following properties:

$$\text{For each } F \in \mathcal{F}_h \text{ there is a corresponding } S_F \in \mathcal{S}_h^\Gamma \text{ with } F \subset S_F. \tag{5.36}$$

$$\text{For } F_1, F_2 \in \mathcal{F}_h \text{ with } F_1 \neq F_2 \text{ we have } \operatorname{meas}_3(S_{F_1} \cap S_{F_2}) = 0. \tag{5.37}$$

$$\text{The family } \{\mathcal{S}_h^\Gamma\}_{h_\Gamma > 0} \text{ is shape-regular.} \tag{5.38}$$

$$c_0 h_\Gamma \leq \operatorname{diam}(S_F) \leq c h_\Gamma \quad \text{for all } F \in \mathcal{F}_h, \quad \text{with } c_0 > 0 \text{ (quasi-uniformity).} \tag{5.39}$$

$$\text{For each } S_F \in \mathcal{S}_h^\Gamma \text{ there is a tetrahedron } S \in \mathcal{S} \text{ such that } S_F \subset S. \tag{5.40}$$

Recall that $\mathcal{S}$ is the (fixed) tetrahedral triangulation that is used in the finite element discretization of the Navier-Stokes problem in (2.21)–(2.22). Note that the set of tetrahedra $\mathcal{S}_h^\Gamma$ has to be defined only close to the approximate interface Γ_h and that this set not necessarily forms a regular tetrahedral triangulation of Ω. Furthermore, it is *not* assumed that the family $\{\Gamma_h\}_{h_\Gamma > 0}$ is shape-regular or quasi-uniform.

Remark 5.19
Consider the construction of $\{\Gamma_h\}_{h_\Gamma > 0}$ as in Section 5.1.2. The approximate interface Γ_h is the zero level of the function $I(d_h)$, which is continuous piecewise linear on the tetrahedral triangulation $\mathcal{T}'$:

$$\Gamma_h = \bigcup_{F \in \mathcal{F}} F,$$

where each F is a triangle or a quadrilateral. To each F there can be associated a tetrahedron $S_F \in \mathcal{T}'$ such that $F \subset S_F$. Recalling the definition (5.10) of

an interface segment, we note that in fact $F = \Gamma_T$ for $T = S_F \in \mathcal{T}'$. If F is a quadrilateral then we can subdivide F and S_F in two disjoint triangles F_1, F_2 and two disjoint tetrahedra S_{F_1}, S_{F_2}, respectively, such that $F_i \subset S_{F_i} \subset S_F$ for $i = 1, 2$. One can check that this construction results in a family $\{\mathcal{S}_h^\Gamma\}_{h_\Gamma>0}$ that satisfies the conditions (5.36)-(5.40). ◇

In the following lemma we consider a standard affine mapping between a tetrahedron $S_F \in \mathcal{S}_h^\Gamma$ and the reference unit tetrahedron and apply it to the triangle $F \subset S_F$.

Lemma 5.20
Assume that the family $\{\Gamma_h\}_{h_\Gamma>0}$ is such that for the associated family of sets of tetrahedra $\{\mathcal{S}_h^\Gamma\}_{h_\Gamma>0}$ the conditions (5.36)-(5.40) are satisfied. Take $F \in \mathcal{F}_h$ and the corresponding $S_F \in \mathcal{S}_h^\Gamma$. Let $\hat{S}$ be the reference unit tetrahedron and $\Phi(\mathbf{x}) = \mathbf{J}\mathbf{x} + \mathbf{b}$ be an affine mapping such that $\Phi(\hat{S}) = S_F$. Define $\hat{F} := \Phi^{-1}(F)$. The following holds:

$$\|\mathbf{J}\|^2 \frac{\mathrm{meas}_3(\hat{S})}{\mathrm{meas}_3(S_F)} \leq c\, h_\Gamma^{-1}, \tag{5.41}$$

$$\|\mathbf{J}^{-1}\|^2 \frac{\mathrm{meas}_2(F)}{\mathrm{meas}_2(\hat{F})} \leq c, \tag{5.42}$$

with constants c independent of F and h_Γ.

Proof. Let $\rho(S_F)$ be the diameter of the maximal ball contained in S_F and similarly for $\rho(\hat{S})$. From standard finite element theory we have

$$\|\mathbf{J}\| \leq \frac{\mathrm{diam}(S_F)}{\rho(\hat{S})}, \quad \|\mathbf{J}^{-1}\| \leq \frac{\mathrm{diam}(\hat{S})}{\rho(S_F)}.$$

Using (5.38) and (5.39) we then get

$$\|\mathbf{J}\|^2 \frac{\mathrm{meas}_3(\hat{S})}{\mathrm{meas}_3(S_F)} \leq c \frac{\mathrm{diam}(S_F)^2}{\mathrm{meas}_3(S_F)} \leq c\ \mathrm{diam}(S_F)^{-1} \leq c\, h_\Gamma^{-1},$$

and thus the result in (5.41) holds.

The vertices of $\hat{F} = \Phi^{-1}(F)$ are denoted by $\hat{V}_i$, $i = 1, 2, 3$. Let $\hat{V}_1\hat{V}_2$ be a longest edge of $\hat{F}$ and $\hat{M}$ the point on this edge such that $\hat{M}\hat{V}_3$ is perpendicular to $\hat{V}_1\hat{V}_2$. Define $V_i := \Phi(\hat{V}_i)$, $i = 1, 2, 3$, and $M := \Phi(\hat{M})$. Then V_i, $i = 1, 2, 3$,

are the vertices of F and M lies on the edge V_1V_2. We then have

$$\begin{aligned}\operatorname{meas}_2(\hat{F}) &= \frac{1}{2}\|\hat{V}_1-\hat{V}_2\|\|\hat{V}_3-\hat{M}\| = \frac{1}{2}\|\mathbf{J}^{-1}(V_1-V_2)\|\|\mathbf{J}^{-1}(V_3-M)\| \\ &\geq \frac{1}{2}\|\mathbf{J}\|^{-2}\|V_1-V_2\|\|V_3-M\| \geq c\,\frac{\rho(\hat{S})^2}{\operatorname{diam}(S_F)^2}\operatorname{meas}_2(F),\end{aligned}$$

with a constant $c>0$. Thus we obtain

$$\|\mathbf{J}^{-1}\|^2\frac{\operatorname{meas}_2(F)}{\operatorname{meas}_2(\hat{F})} \leq c\,\frac{\operatorname{diam}(\hat{S})^2}{\rho(S_F)^2}\frac{\operatorname{diam}(S_F)^2}{\rho(\hat{S})^2} \leq c,$$

which completes the proof. □

Theorem 5.21
Assume that the family $\{\Gamma_h\}_{h_\Gamma>0}$ is such that for the associated family of sets of tetrahedra $\{\mathcal{S}_h^\Gamma\}_{h_\Gamma>0}$ the conditions (5.36)-(5.40) are satisfied. Then the following holds:

$$\|\nabla_{\Gamma_h} v\|_{L^2(\Gamma_h)} \leq c\,h_\Gamma^{-\frac{1}{2}}\|v\|_1 \quad \text{for all } v\in V_h.$$

Proof. Note that

$$\|\nabla_{\Gamma_h} v\|^2_{L^2(\Gamma_h)} = \sum_{F\in\mathcal{F}_h}\|\nabla_F v\|^2_{L^2(F)}.$$

Take $F\in\mathcal{F}_h$ and let S_F be the associated tetrahedron as explained above. Let $\hat{S}$ be the reference unit tetrahedron and $\Phi:\hat{S}\to S_T$ as in Lemma 5.20. Define $\hat{v}:=v\circ\Phi$. Using standard transformation rules and Lemma 5.20 we get

$$\begin{aligned}\|\nabla_F v\|^2_{L^2(F)} = \|\mathbf{P}_h\nabla v\|^2_{L^2(F)} &\leq \|\nabla v\|^2_{L^2(F)} = \sum_{|\alpha|=1}\|\partial^\alpha v\|^2_{L^2(F)} \\ &\leq c\,\|\mathbf{J}^{-1}\|^2\sum_{|\alpha|=1}\|(\partial^\alpha\hat{v})\circ\Phi^{-1}\|^2_{L^2(F)} \\ &\leq c\,\|\mathbf{J}^{-1}\|^2\frac{\operatorname{meas}_2(F)}{\operatorname{meas}_2(\hat{F})}\sum_{|\alpha|=1}\|\partial^\alpha\hat{v}\|^2_{L^2(\hat{F})} \leq c\sum_{|\alpha|=1}\|\partial^\alpha\hat{v}\|^2_{L^2(\hat{F})} \\ &\leq c\sum_{|\alpha|=1}\max_{\mathbf{x}\in\hat{F}}\left|\partial^\alpha\hat{v}(\mathbf{x})\right|^2 \leq c\sum_{|\alpha|=1}\max_{\mathbf{x}\in\hat{S}}\left|\partial^\alpha\hat{v}(\mathbf{x})\right|^2,\end{aligned}$$

with a constant c independent of F. From (5.40) it follows that $\hat{v}$ is a polynomial on $\hat{S}$ of maximal degree k, where k depends only on the choice of the

finite element space $\mathbf{V}_h$. On $\mathcal{P}_k^* := \{ p \in \mathcal{P}_k : p(0) = 0 \}$ we have, due to equivalence of norms:

$$\sum_{|\alpha|=1} \max_{\mathbf{x}\in\hat{S}} \left|\partial^\alpha \hat{v}(\mathbf{x})\right|^2 \leq c \sum_{|\alpha|=1} \|\partial^\alpha \hat{v}\|^2_{L^2(\hat{S})} \quad \text{for all } \hat{v} \in \mathcal{P}_k^*.$$

Because, for $\hat{v} \in \mathcal{P}_k$ and $|\alpha| = 1$, $\partial^\alpha \hat{v}$ is independent of $\hat{v}(0)$, the same inequality holds for all $\hat{v} \in \mathcal{P}_k$. Thus we get

$$\begin{aligned}\|\nabla_F v\|^2_{L^2(F)} &\leq c \sum_{|\alpha|=1} \|\partial^\alpha \hat{v}\|^2_{L^2(\hat{S})} \leq c\,\|\mathbf{J}\|^2 \sum_{|\alpha|=1} \|(\partial^\alpha v)\circ\Phi\|^2_{L^2(\hat{S})} \\ &= c\,\|\mathbf{J}\|^2 \frac{\mathrm{meas}_3(\hat{S})}{\mathrm{meas}_3(S_F)} \sum_{|\alpha|=1} \|\partial^\alpha v\|^2_{L^2(S_F)} \leq c\,h_\Gamma^{-1} \|\nabla v\|^2_{L^2(S_F)},\end{aligned}$$

with a constant c independent of F and h. Using (5.37) we finally obtain

$$\begin{aligned}\|\nabla_{\Gamma_h} v\|^2_{L^2(\Gamma_h)} &\leq c\,h_\Gamma^{-1} \sum_{F\in\mathcal{F}_h} \|\nabla v\|^2_{L^2(S_F)} \\ &\leq c\,h_\Gamma^{-1} \int_\Omega (\nabla v)^2\, d\mathbf{x} \leq c\,h_\Gamma^{-1} \|v\|_1^2,\end{aligned}$$

which proves the result. □

We now present the main result of this section.

Theorem 5.22
Let the assumptions be as in Theorem 5.21. The following holds:

$$\sup_{v\in V_h} \frac{|g(v) - g_h(v)|}{\|v\|_1} \leq c\sqrt{h_\Gamma}.$$

Proof. Combine the result in Corollary 5.18 with the one in Theorem 5.21. □

As a direct consequence we obtain:

Corollary 5.23
Let the assumptions be as in Theorem 5.21. For f_Γ and f_{Γ_h} as defined in Section 5.3.1 the following holds:

$$\sup_{\mathbf{v}\in\mathbf{V}_h} \frac{|f_\Gamma(\mathbf{v}_h) - f_{\Gamma_h}(\mathbf{v}_h)|}{\|\mathbf{v}_h\|_1} \leq \tau c\sqrt{h_\Gamma}.$$

Proof. Note that

$$f_\Gamma(\mathbf{v}_h) - f_{\Gamma_h}(\mathbf{v}_h) = -\tau \sum_{i=1}^{3} \Big(\int_\Gamma \nabla_\Gamma(\mathrm{id}_\Gamma)_i \cdot \nabla_\Gamma(\mathbf{v}_h)_i \, ds - \int_{\Gamma_h} \nabla_{\Gamma_h}(\mathrm{id}_{\Gamma_h})_i \cdot \nabla_{\Gamma_h}(\mathbf{v}_h)_i \, ds \Big),$$

and use the result in Theorem 5.22. □

An upper bound $\mathcal{O}(\sqrt{h_\Gamma})$ as in Corollary 5.23 for the error in the approximation of the localized force term may seem rather pessimistic, because Γ_h is an $\mathcal{O}(h_\Gamma^2)$ accurate approximation of Γ. Numerical experiments in Section 10.3 and results in [GMT07], however, indicate that the bound is sharp.

5.3.4. Improved Laplace-Beltrami discretization

In this section we show how the approximation of the localized force term can be improved, resulting in an improved error bound of the form $\mathcal{O}(h_\Gamma)$ (instead of $\mathcal{O}(\sqrt{h_\Gamma})$ in Corrolary 5.23).

From Corollary 5.18 and Theorem 5.21 we see that the $\sqrt{h_\Gamma}$ behavior is caused by the estimate in (5.34):

$$\left| \int_{\Gamma_h} \nabla_{\Gamma_h}(\mathrm{id}_\Gamma^e - \mathrm{id}_{\Gamma_h}) \cdot \nabla_{\Gamma_h} v \, ds \right| \leq c \, h_\Gamma \, \|\nabla_{\Gamma_h} v\|_{L^2(\Gamma_h)}. \tag{5.43}$$

The term $\nabla_{\Gamma_h} \mathrm{id}_{\Gamma_h}$ that is used in $g_h(v)$ occurs in (5.43) but not in the other two terms of the splitting, cf. (5.32), (5.33). We consider

$$\tilde{g}_h(v) = \int_{\Gamma_h} m_h \cdot \nabla_{\Gamma_h} v \, ds$$

and try to find a function $m_h = m_h(\mathbf{x})$ such that $\tilde{g}_h(v)$ remains easily computable and the bound in (5.43) is improved if we use m_h instead of $\nabla_{\Gamma_h} \mathrm{id}_{\Gamma_h}$. The latter condition is trivially satisfied for $m_h = \nabla_{\Gamma_h} \mathrm{id}_\Gamma^e$ (leading to a bound 0 in (5.43)). This choice, however, does not satisfy the first condition, because Γ is not known. We now discuss another possibility, that is used in the experiments in Section 10.3.

Due to $|d(\mathbf{x})| \leq c h_\Gamma^2$ we get from Lemma 5.15, for $\mathbf{x} \in \Gamma_h$:

$$\nabla_{\Gamma_h} \mathrm{id}_\Gamma^e(\mathbf{x}) = \mathbf{P}_h(\mathbf{x}) \mathbf{P}(\mathbf{x}) \nabla_\Gamma \mathrm{id}_\Gamma(\mathbf{p}(\mathbf{x})) + \mathcal{O}(h_\Gamma^2) = \mathbf{P}_h(\mathbf{x}) \mathbf{P}(\mathbf{x}) e_i + \mathcal{O}(h_\Gamma^2).$$

In the construction of the interface Γ_h, cf. Section 5.1.2, we have a piecewise *quadratic* function $d_h \approx d$ available. Define

$$\tilde{\mathbf{n}}_h(\mathbf{x}) := \frac{\nabla d_h(\mathbf{x})}{\|\nabla d_h(\mathbf{x})\|}, \qquad \tilde{\mathbf{P}}_h(\mathbf{x}) := \mathbf{I} - \tilde{\mathbf{n}}_h(\mathbf{x})\tilde{\mathbf{n}}_h(\mathbf{x})^T, \quad \mathbf{x} \in \Gamma_h.$$

Thus an obvious modification is based on the choice $m_h(\mathbf{x}) = \mathbf{P}_h(\mathbf{x})\tilde{\mathbf{P}}_h(\mathbf{x})e_i$, i. e.,

$$\tilde{g}_h(v) := \int_{\Gamma_h} \mathbf{P}_h(\mathbf{x})\tilde{\mathbf{P}}_h(\mathbf{x})e_i \cdot \nabla_{\Gamma_h} v \, ds = \int_{\Gamma_h} \tilde{\mathbf{P}}_h(\mathbf{x})e_i \cdot \nabla_{\Gamma_h} v \, ds. \tag{5.44}$$

In this approach the approximate interface Γ_h is *not* changed (piecewise planar). For piecewise quadratics d_h and v, the function $\nabla_{\Gamma_h} v = \mathbf{P}_h \nabla v$ is piecewise linear and $\tilde{\mathbf{P}}_h e_i$ is piecewise (very) smooth on the segments of Γ_h. Hence, the functional in (5.44) can be evaluated easily.

Under reasonable assumptions the modified functional indeed yields a better error bound:

Lemma 5.24
Assume that there exists $p > 0$ such that

$$\|\nabla d_h(\mathbf{x}) - \nabla d(\mathbf{x})\| \le c\, h_\Gamma^p, \quad \text{for } \mathbf{x} \in \Gamma_h. \tag{5.45}$$

Then for all $v \in W$ the following holds:

$$\left| \int_{\Gamma_h} \big(\nabla_{\Gamma_h} \mathrm{id}_\Gamma^e - \mathbf{P}_h \tilde{\mathbf{P}}_h e_i\big) \cdot \nabla_{\Gamma_h} v \, ds \right| \le c\, h_\Gamma^{\min\{p,2\}} \, \|\nabla_{\Gamma_h} v\|_{L^2(\Gamma_h)}.$$

Proof. From Lemma 5.15 we get for $\mathbf{x} \in \Gamma_h$ (not on an edge),

$$\begin{aligned} \nabla_{\Gamma_h} \mathrm{id}_\Gamma^e(\mathbf{x}) &= \mathbf{P}_h(\mathbf{x})\big(\mathbf{I} - d(\mathbf{x})\mathbf{H}(\mathbf{x})\big)\mathbf{P}(\mathbf{x})\nabla_\Gamma \mathrm{id}_\Gamma(\mathbf{p}(\mathbf{x})) \\ &= \mathbf{P}_h(\mathbf{x})\big(\mathbf{I} - d(\mathbf{x})\mathbf{H}(\mathbf{x})\big)\mathbf{P}(\mathbf{x})e_i. \end{aligned}$$

We also have $\nabla_{\Gamma_h} \mathrm{id}_{\Gamma_h} = \mathbf{P}_h \nabla \mathrm{id}_{\Gamma_h} = \mathbf{P}_h e_i$. Hence,

$$\begin{aligned} &\left| \int_{\Gamma_h} \nabla_{\Gamma_h} (\mathrm{id}_\Gamma^e - \mathrm{id}_{\Gamma_h}) \cdot \nabla_{\Gamma_h} v \, ds \right| && (5.46) \\ &= \left| \int_{\Gamma_h} \big(\mathbf{P}_h(\mathbf{I} - d\mathbf{H})\mathbf{P}e_i - \mathbf{P}_h e_i\big) \cdot \nabla_{\Gamma_h} v \, ds \right| \\ &\le c \operatorname{ess\,sup}_{\mathbf{x}\in\Gamma_h} \|\mathbf{P}_h(\mathbf{x})\big(\mathbf{I} - d(\mathbf{x})\mathbf{H}(\mathbf{x})\big)\mathbf{P}(\mathbf{x}) - \mathbf{P}_h(\mathbf{x})\| \, \|\nabla_{\Gamma_h} v\|_{L^2(\Gamma_h)} \\ &\le c \operatorname{ess\,sup}_{\mathbf{x}\in\Gamma_h} \big(\|\mathbf{P}_h(\mathbf{x})\mathbf{P}(\mathbf{x}) - \mathbf{P}_h(\mathbf{x})\tilde{\mathbf{P}}_h(\mathbf{x})\| && (5.47) \\ &\qquad + |d(\mathbf{x})| \, \|\mathbf{P}_h(\mathbf{x})\mathbf{H}(\mathbf{x})\mathbf{P}(\mathbf{x})\|\big) \, \|\nabla_{\Gamma_h} v\|_{L^2(\Gamma_h)}. && (5.48) \end{aligned}$$

We first derive a bound for (5.47). Using $\|\nabla d\| = 1$ it follows that $\|\nabla d_h\| = 1 + \mathcal{O}(h_\Gamma^p)$ holds. We drop $\mathbf{x}$ in the notation and using the assumption (5.45) we obtain

$$\begin{aligned}
\|\mathbf{P}_h\mathbf{P} - \mathbf{P}_h\tilde{\mathbf{P}}_h\| &= \|\mathbf{P}_h(\mathbf{P} - \tilde{\mathbf{P}}_h)\| \leq \|\mathbf{n}\mathbf{n}^T - \tilde{\mathbf{n}}_h\tilde{\mathbf{n}}_h^T\| \\
&\leq \|(\mathbf{n} - \tilde{\mathbf{n}}_h)\mathbf{n}^T\| + \|\tilde{\mathbf{n}}_h(\mathbf{n} - \tilde{\mathbf{n}}_h)^T\| = 2\|\mathbf{n} - \tilde{\mathbf{n}}_h\| \\
&= 2\left\|\nabla d - \frac{\nabla d_h}{\|\nabla d_h\|}\right\| \\
&\leq 2\left|1 - \|\nabla d_h\|^{-1}\right| \|\nabla d_h\| + 2\|\nabla d - \nabla d_h\| \leq c\, h^p.
\end{aligned}$$

We now turn to (5.48). Note that due to (5.5) $|d(\mathbf{x})| \leq c\, h_\Gamma^2$ for $\mathbf{x} \in \Gamma_h$, and

$$\operatorname{ess\,sup}_{\mathbf{x}\in\Gamma_h} \|\mathbf{P}_h(\mathbf{x})\mathbf{H}(\mathbf{x})\mathbf{P}(\mathbf{x})\| \leq \operatorname{ess\,sup}_{\mathbf{x}\in\Gamma_h} \|\mathbf{H}(\mathbf{x})\| \leq c,$$

hence yielding a bound $|d(\mathbf{x})|\|\mathbf{P}_h(\mathbf{x})\mathbf{H}(\mathbf{x})\mathbf{P}(\mathbf{x})\| \leq c\, h_\Gamma^2$ for the term in (5.48). Combined with the inequality $\|\mathbf{P}_h\mathbf{P} - \mathbf{P}_h\tilde{\mathbf{P}}_h\| \leq c\, h_\Gamma^p$ for the term in (5.47) this proves the result. □

If we assume that the condition in (5.45) is satisfied for $p = 2$, which is reasonable for a piecewise quadratic approximation d_h of d, we get the following improvement due to the modified functional $\tilde{g}_h$, cf. Corollary 5.18:

$$|g(v) - \tilde{g}_h(v)| \leq c\, h_\Gamma \|v\|_{1,U} + c\, h_\Gamma^2 \|\nabla_{\Gamma_h} v\|_{L^2(\Gamma_h)} \quad \text{for all } v \in W.$$

Combining this with the result in Theorem 5.21 yields (under the assumption as in Theorem 5.21):

$$|g(v) - \tilde{g}_h(v)| \leq c\, h_\Gamma \|v\|_{1,U} + c\, h_\Gamma^{3/2} \|v\|_1 \quad \text{for all } v \in V_h.$$

Hence, using this modified functional $\tilde{g}_h$ we have a $\mathcal{O}(h_\Gamma)$ error bound. We therefore define the *improved Laplace-Beltrami discretization* by

$$\tilde{f}_{\Gamma_h}(\mathbf{v}) := -\tau \int_{\Gamma_h} \tilde{\mathbf{P}}_h \nabla \operatorname{id}_{\Gamma_h} \cdot \nabla_{\Gamma_h} \mathbf{v}\, ds \;=\; -\tau \sum_{i=1}^{3} \tilde{g}_{h,i}(v_i) \tag{5.49}$$

for all $\mathbf{v} \in \mathbf{V}_h$.

This significant improvement ($\mathcal{O}(h_\Gamma)$ compared to the $\mathcal{O}(\sqrt{h_\Gamma})$ error bound for the functional f_{Γ_h}) is confirmed by the numerical experiments in Section 10.3.

5.4. Finite element space for the discontinuous pressure

After the analysis and improvement of the discretization of the localized surface force term f_Γ given in the previous section, we now turn to the approximation of the pressure which is discontinuous across the interface Γ if surface tension forces are present. We show that standard finite element spaces have only poor approximation properties for such functions with a jump across Γ and introduce a new extended finite element space which is more appropriate for this task. Most of the results presented in this section are from [GR07a].

5.4.1. Approximation error for standard FE spaces

In this section we consider the approximation error

$$\inf_{q_h \in Q_h} \|q_h - p^*\|_{L^2}$$

for a few standard finite element spaces Q_h and explain why in general for a function p^* that is discontinuous across Γ_h one can expect no better bound for this approximation error than $c\sqrt{h}$. This serves as a motivation for an improved pressure finite element space as presented in Section 5.4.2. To explain the effect underlying the $\sqrt{h}$ behavior of the error bound we analyze a concrete two-dimensional example as illustrated in Figure 5.6. We take $\Omega = (0,1)^2 \subset \mathbb{R}^2$ and define

$$\Omega_1 := \{\, (x,y) \in \Omega : \, x \leq 1 - y \,\}, \qquad \Omega_2 := \Omega \setminus \overline{\Omega}_1.$$

The interface Γ separating both subdomains from each other is given by

$$\Gamma = \{\, (x,y) \in \Omega : \, y = 1 - x \,\}.$$

A family of triangulations $\{\mathcal{T}_h\}_{h_\Gamma > 0}$ is constructed as follows. The starting triangulation T_0 consists of two triangles, namely the ones with vertices $\{(0,0),(0,1),(1,1)\}$ and $\{(0,0),(1,0),(1,1)\}$. Then a global regular refinement strategy (connecting the midpoints of edges) is applied repeatedly. This results in a nested sequence of triangulations T_{h_k}, $k = 1, 2, \ldots$, with mesh size $h_k = 2^{-k}$. In Figure 5.6 the triangulation $\mathcal{T}_{h_2}$ is shown. The set of triangles that contains the interface is given by (with $h := h_k$)

$$\mathcal{T}_h^\Gamma := \{\, T \in \mathcal{T}_h : \, \mathrm{meas}_1(T \cap \Gamma) > 0 \,\}.$$

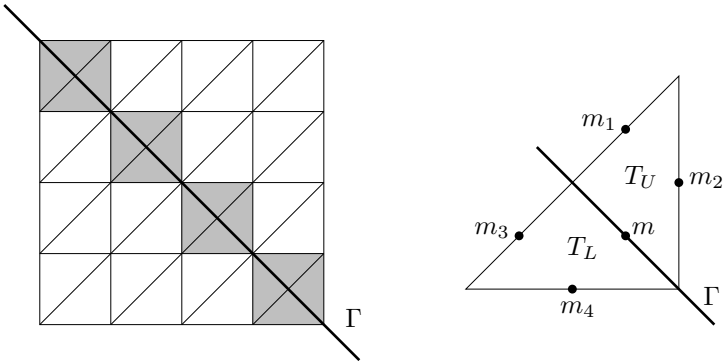

Figure 5.6.: Triangulation T_{h_2} and a triangle $T \in \mathcal{T}_{h_k}^{\Gamma}$.

In Figure 5.6 the elements in $\mathcal{T}_{h_2}^{\Gamma}$ are colored gray.

For $h = h_k$ we consider the finite element spaces

$$Q_h^0 := \{\, p : \Omega \to \mathbb{R} : \; p_{|T} \in \mathcal{P}_0 \quad \text{for all } T \in \mathcal{T}_h \,\} \qquad \text{(piecewise constants)},$$

$$Q_h^{1,\mathrm{disc}} := \{\, p : \Omega \to \mathbb{R} : \; p_{|T} \in \mathcal{P}_1 \quad \text{for all } T \in \mathcal{T}_h \,\} \qquad \text{(piecewise linears, discontinuous)},$$

$$Q_h^1 := \{\, p \in C(\Omega) : \; p_{|T} \in \mathcal{P}_1 \quad \text{for all } T \in \mathcal{T}_h \,\} \qquad \text{(piecewise linears, continuous)}.$$

Note that

$$Q_h^j \subset Q_h^{1,\mathrm{disc}} \quad \text{for } j = 0, 1. \tag{5.50}$$

We take p^* as follows: $p^*(x,y) = c_p > 0$ for all $(x,y) \in \Omega_1$, $p(x,y) = 0$ for all $(x,y) \in \Omega_2$. We study $\inf_{q_h \in Q_h} \|q_h - p^*\|_{L^2}$ for $Q_h \in \{Q_h^0, Q_h^{1,\mathrm{disc}}, Q_h^1\}$. For $Q_h = Q_h^{1,\mathrm{disc}}$ the identity

$$\inf_{q_h \in Q_h^{1,\mathrm{disc}}} \|q_h - p^*\|_{L^2}^2 = \sum_{T \in \mathcal{T}_h^{\Gamma}} \min_{q \in \mathcal{P}_1} \|q - p^*\|_{L^2(T)}^2$$

holds. Take $T \in \mathcal{T}_h^{\Gamma}$. Using a quadrature rule on triangles that is exact for all

polynomials of degree two we get, cf. Figure 5.6,

$$\begin{aligned}\min_{q\in\mathcal{P}_1}\|q-p^*\|^2_{L^2(T)} &= \min_{q\in\mathcal{P}_1}\Big(\int_{T_L}(q-c_p)^2\,dx\,dy+\int_{T_U}q^2\,dx\,dy\Big)\\ &= \frac{h^2}{12}\min_{q\in\mathcal{P}_1}\big((q(m_3)-c_p)^2+(q(m_4)-c_p)^2+(q(m)-c_p)^2\\ &\qquad +q(m_1)^2+q(m_2)^2+q(m)^2\big)\\ &\geq \frac{h^2}{12}\min_{q\in\mathcal{P}_1}\big((q(m)-c_p)^2+q(m)^2\big)=\frac{1}{24}c_p^2h^2.\end{aligned}$$

Thus we have

$$\inf_{q_h\in Q_h^{1,\mathrm{disc}}}\|q_h-p^*\|_{L^2}\geq\Big(\sum_{T\in\mathcal{T}_h^\Gamma}\frac{1}{24}c_p^2h^2\Big)^{\frac12}=\big(\frac{2}{h}\frac{1}{24}c_p^2h^2\big)^{\frac12}=\frac{1}{2\sqrt{3}}c_p\sqrt{h}.$$

Due to (5.50) this yields

$$\inf_{q_h\in Q_h}\|q_h-p^*\|_{L^2}\geq\frac{1}{2\sqrt{3}}c_p\sqrt{h}\quad\text{for } Q_h\in\{Q_h^0,Q_h^{1,\mathrm{disc}},Q_h^1\}. \tag{5.51}$$

To derive an upper bound for the approximation error we choose a suitable $q_h\in Q_h$. First consider $Q_h=Q_h^0$ and take $q_h^0\in Q_h^0$ as follows: $(q_h^0)_{|T}=c_p$ for all T with $\mathrm{meas}_1(T\cap\Omega_1)>0$, $q_h^0=0$ otherwise. With this choice we get

$$\|q_h^0-p^*\|_{L^2}=\Big(\sum_{T\in\mathcal{T}_h^\Gamma}\|q_h^0-p^*\|^2_{L^2(T)}\Big)^{\frac12}=\big(\sum_{T\in\mathcal{T}_h^\Gamma}c_p^2\frac14h^2\big)^{\frac12}=\frac{1}{\sqrt{2}}c_p\sqrt{h}. \tag{5.52}$$

For $Q_h=Q_h^1$ we take $q_h^1:=I_h(p^*)$, where I_h is the nodal interpolation operator (note: $p^*=c_p$ on Γ). Elementary computations yield

$$\|q_h^1-p^*\|_{L^2}=\big(\frac{1}{12}c_p^2h\big)^{\frac12}=\frac{1}{2\sqrt{3}}c_p\sqrt{h}. \tag{5.53}$$

Combination of (5.50), (5.51), (5.52) and (5.53) yields

$$\frac{1}{2\sqrt{3}}c_p\sqrt{h}\leq\inf_{q_h\in Q_h}\|q_h-p^*\|_{L^2}\leq\frac{1}{\sqrt{2}}c_p\sqrt{h}\quad\text{for } Q_h\in\{Q_h^0,Q_h^{1,\mathrm{disc}},Q_h^1\}. \tag{5.54}$$

Note that this approximation error result does not change if we apply only *local* refinement close to the interface and then replace h by h_Γ, where the latter denotes the mesh size of the triangles in $\mathcal{T}_h^\Gamma$.

If instead of piecewise constants or piecewise linears we consider polynomials of higher degree, the approximation error still behaves like $\sqrt{h}$.

Similar examples, which have a $\sqrt{h}$ approximation error behavior, can be constructed using these finite element spaces on tetrahedral triangulations in 3D.

5.4.2. Extended finite element space

The analysis in the previous Section 5.2, which is confirmed by numerical experiments in Section 10.4, leads to the conclusion that there is a need for an improved finite element space for the pressure. In this section we present such a space which is based on an idea presented in [MDB99, BMUP01]. In that paper a so-called extended finite element space (XFEM) is introduced in the context of crack formations in structure mechanics which has good approximation properties for interface type of problems.

Here we apply the XFEM method to two-phase flow problems by constructing an extended pressure finite element space Q_h^Γ. In this section we explain the method and discuss some implementation issues. In Section 10.4 results of numerical experiments with this method are presented.

For $k \geq 1$ fixed we introduce the standard finite element space

$$Q_h = Q_h^k = \{\, q \in C(\Omega) \cap L_0^2(\Omega) : \, q|_T \in \mathcal{P}_k \quad \text{for all } T \in \mathcal{T} \,\}.$$

For $k = 1$, for example, this is the standard finite element space of continuous piecewise linear functions. We define the index set $\mathcal{J} = \{1, \ldots, n\}$, where $n = \dim Q_h$ is the number of degrees of freedom. Let $\mathcal{B} := \{q_j\}_{j \in \mathcal{J}}$ be the nodal basis of Q_h, i. e. $q_j(\mathbf{x}_i) = \delta_{i,j}$ for $i, j \in \mathcal{J}$ where $\mathbf{x}_i \in \mathbb{R}^3$ denotes the spatial coordinate of the i-th degree of freedom.

The idea of the XFEM method is to enrich the original finite element space Q_h by additional basis functions q_j^X for $j \in \mathcal{J}'$ where $\mathcal{J}' \subset \mathcal{J}$ is a given index set. An additional basis function q_j^X is constructed by multiplying the original nodal basis function q_j by a so called enrichment function Φ_j:

$$q_j^X(\mathbf{x}) := q_j(\mathbf{x})\, \Phi_j(\mathbf{x}). \tag{5.55}$$

This enrichment yields the extended finite element space

$$Q_h^X := \operatorname{span}\big(\mathcal{B} \cup \{q_j^X\}_{j \in \mathcal{J}'}\big).$$

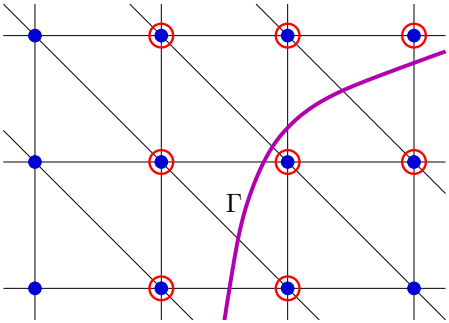

Figure 5.7.: Enrichment by additional basis functions for P_1 finite elements in a 2D example. Dots represent degrees of freedom of original basis functions, circles indicate where additional basis functions are added in the vicinity of the interface Γ.

This idea was introduced in [MDB99] and further developed in [BMUP01] for different kinds of discontinuities (kinks, jumps), which may also intersect or branch. The choice of the enrichment function depends on the type of discontinuity. For representing jumps the Heaviside function is proposed to construct appropriate enrichment functions. Basis functions with kinks can be obtained by using the distance function as enrichment function [MCCR03].

In our case the finite element space Q_h^1 is enriched by discontinuous basis functions q_j^X for $j \in \mathcal{J}' = \mathcal{J}_\Gamma := \{\, j \in \mathcal{J} : \operatorname{meas}_2(\Gamma \cap \operatorname{supp} q_j) > 0 \,\}$, as discontinuities only occur at the interface. This situation is illustrated in Figure 5.7 for a 2D example.

Let $d : \Omega \to \mathbb{R}$ be the signed distance function (or an approximation to it) with d negative in Ω_1 and positive in Ω_2. For example the level set function φ could be used for d. Then by means of the Heaviside function H we define

$$H_\Gamma(\mathbf{x}) := H(d(\mathbf{x})) = \begin{cases} 0 & \mathbf{x} \in \Omega_1, \\ 1 & \mathbf{x} \in \Omega_2. \end{cases}$$

As we are interested in functions with a jump across the interface we define the enrichment function

$$\Phi_j^H(\mathbf{x}) := H_\Gamma(\mathbf{x}) - H_\Gamma(\mathbf{x}_j) \tag{5.56}$$

and a corresponding function $q_j^X := q_j \cdot \Phi_j^H$, $j \in \mathcal{J}'$. The second term in the definition of Φ_j^H is constant and may be omitted (as it doesn't introduce new

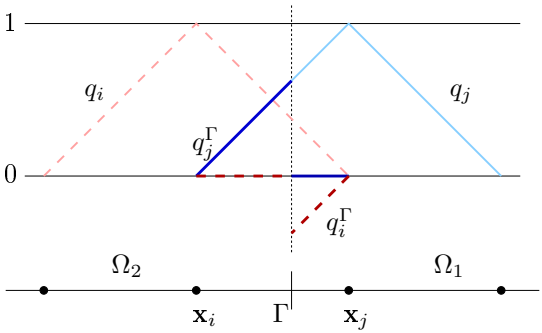

Figure 5.8.: Extended finite element basis functions q_i, q_i^Γ (dashed) and q_j, q_j^Γ (solid) for 1D case.

functions in the function space), but ensures the nice property $q_j^X(\mathbf{x}_i) = 0$ for all i, i. e., q_j^X vanishes in all degrees of freedom. As a consequence, we have

$$\operatorname{supp} q_j^X \subset \left(\operatorname{supp} q_j \cap \Omega^\Gamma\right), \tag{5.57}$$

where

$$\Omega^\Gamma := \bigcup_{T \in \mathcal{T}_h^\Gamma} T \tag{5.58}$$

is the union of all tetrahedra intersected by Γ,

$$\mathcal{T}_h^\Gamma := \{\, T \in \mathcal{T}_h : \operatorname{meas}_2(T \cap \Gamma) > 0 \,\}. \tag{5.59}$$

Thus $q_j^X \equiv 0$ in all T with $T \notin \mathcal{T}_h^\Gamma$.

In the following we will use the notation $q_j^\Gamma := q_j\, \Phi_j^H$ and

$$\mathcal{B}^\Gamma := \mathcal{B} \cup \{\, q_j^\Gamma : j \in \mathcal{J}_\Gamma \,\}, \tag{5.60}$$

$$Q_h^\Gamma := \operatorname{span}\left(\mathcal{B}^\Gamma\right) \tag{5.61}$$

to emphasize that the extended finite element space Q_h^Γ depends on the location of the interface Γ. In particular the dimension of Q_h^Γ may change if the interface is moved. The shape of the extended basis functions for the 1D case is sketched in Figure 5.8.

Remark 5.25
Note that Q_h^Γ can also be characterized by the following property: $q \in Q_h^\Gamma$ if and only if there exist functions $q_1, q_2 \in Q_h$ such that $q|_{\Omega_i} = q_i|_{\Omega_i}$, $i = 1, 2$. $\diamond$

The following result from [Reu08] shows that the extended finite element space Q_h^Γ offers the following optimal approximation property.

Theorem 5.26
For an integer $k \geq 0$ we define the space

$$H^k(\Omega_1 \cup \Omega_2) := \{\, p \in L_2(\Omega) : \, p_{|\Omega_i} \in H^k(\Omega_i),\, i = 1, 2 \,\}$$

with the norm $\|p\|^2_{k,\Omega_1\cup\Omega_2} := \|p\|^2_{k,\Omega_1} + \|p\|^2_{k,\Omega_2}$. Then for integer l, m with $0 \leq l < m \leq 2$ the following holds:

$$\inf_{q\in Q_h^\Gamma} \|p - q\|_{l,\Omega_1\cup\Omega_2} \leq c\, h^{m-l} \|p\|_{m,\Omega_1\cup\Omega_2} \tag{5.62}$$

for all $p \in H^m(\Omega_1 \cup \Omega_2)$.

Proof. Given in [Reu08]. □

Hence, for pressure solutions p with $p_{|\Omega_i} \in H^1(\Omega_i)$, $i = 1, 2$, (cf. Remark 5.1) we have

$$\inf_{q_h\in Q_h^\Gamma} \|q_h - p\|_{L^2} \leq ch.$$

This yields the desired $\mathcal{O}(h)$ bound, cf. Section 5.2.

In [BMUP01] the XFEM is applied to a few problems from linear elasticity demonstrating the ability of the method to capture jumps and kinks. These discontinuities also branch or intersect in some of the examples, in this case more elaborate constructions of the enrichment functions are used.

In [CB03] the XFEM is also applied to a two-phase flow problem. In that paper discontinuous material properties ρ and μ, but *no surface tension forces* were taken into account. Thus there is no jump in pressure, but the solution exhibits kinks at the interface. For the pressure and the level set function standard finite element spaces are used. The velocity field is discretized with an extended finite element space enriched by $\mathbf{v}_j^X(\mathbf{x}) = \mathbf{v}_j(\mathbf{x})\,|d(\mathbf{x})|$ to capture the kinks at the interface. The location of the interface is captured by a level set approach. The construction of the enrichment function is thus based on the level set function φ.

A similar idea of basis enrichment in the context of two-phase flow simulations is also suggested in [MCN03]. The pressure space is augmented by additional discontinuous basis functions $\tilde{q}_j^\Gamma(\mathbf{x}) = q_j(\mathbf{x})\,\tilde{\Phi}^H(\mathbf{x})$, $j \in \mathcal{J}_\Gamma$, where the enrichment function $\tilde{\Phi}^H$ is given by

$$\tilde{\Phi}^H|_{\Omega^\Gamma\cap\Omega_1} \equiv 1, \qquad \tilde{\Phi}^H|_{\Omega^\Gamma\cap\Omega_2} \equiv -1, \qquad \tilde{\Phi}^H|_{\Omega\setminus\Omega^\Gamma} \equiv 0.$$

For the velocity space additional basis functions $\mathbf{v}_T(\mathbf{x}) = \hat{\mathbf{v}}_T(\mathbf{x})\,|d(\mathbf{x})|$ for all $T \in \mathcal{T}_h^\Gamma$ are added where $\hat{\mathbf{v}}_T$ is the bubble function on T.

Remark 5.27
We comment on two related approaches that are known in the literature. In the papers [HH02, HH04, HLPS05] of HANSBO a discontinuous finite element space Q_h^{disc} is introduced and applied to a scalar elliptic interface problem. Boundary conditions at the interior interface are imposed in a weak sense by using a penalty method.

For the construction of Q_h^{disc} the standard finite element space Q_h^1 is modified by replacing each of the basis functions $q_j, j \in \mathcal{J}_\Gamma$, by the two functions

$$q_j^{\Gamma,i}(\mathbf{x}) = \begin{cases} q_j(\mathbf{x}) & \mathbf{x} \in \Omega_i, \\ 0 & \mathbf{x} \notin \Omega_i, \end{cases} \qquad i = 1, 2.$$

This yields the same finite element space as the XFEM approach applied to the case of a jump at the interface, i. e., $Q_h^{\text{disc}} = Q_h^\Gamma$, cf. Remark 5.25. For other kinds of discontinuities the approaches are essentially different, e. g., when the solution has a kink at the interface. While in the XFEM approach a different extended finite element space $\tilde{Q}_h^\Gamma$ is constructed by adding special basis functions suited to represent such a kink at the interface, in the approach of HANSBO the same finite element space Q_h^{disc} as given above is used, but the penalty term is changed to enforce continuity (in a weak sense) of the solution at the interface. In [HLPS05] an error analysis is given for solutions with kinks, where second order convergence in L^2 is shown for the modified P_1 elements on a non-degenerate triangulation. An a-posteriori error estimator for this type of finite elements is derived, too.

Another approach can be found in [MM00]. Here the standard finite element space Q_h^1 is extended by discontinuous basis functions q_T^Γ for $T \in \mathcal{T}_h^\Gamma$, which are defined by

$$q_T^\Gamma(\mathbf{x}) := \begin{cases} H_\Gamma(\mathbf{x}) - \sum_j H_\Gamma(\mathbf{x}_j) \cdot q_j(\mathbf{x}) & \text{for } \mathbf{x} \in T, \\ 0 & \text{otherwise.} \end{cases}$$

This introduces $|\mathcal{T}_h^\Gamma|$ new degrees of freedom, which influence the height of the jump in the corresponding elements. q_T^Γ is not only discontinuous across Γ but also across element boundaries (edges in 2D, faces in 3D) that intersect Γ where p^* is known to be continuous. Due to this disadvantage we did not consider this method for the approximation of discontinuous pressure in two-phase flows. ◇

5.4.3. Challenges related to XFEM

The results for the Stokes test cases presented in Section 10.4 are quite satisfactory. Nevertheless, in the application of the XFEM method to two-phase flow problems there are some hidden pitfalls. We mention a few challenges related to stability issues and to the application of XFEM to non-stationary Navier-Stokes two-phase flow problems.

As Q_h^Γ depends on the location of the interface Γ the space Q_h^Γ changes if the interface is moved. Thus the discretization of $b(\cdot,\cdot)$ has to be updated each time when the level set function (or VOF indicator function) has changed. In a Navier-Stokes code solving non-stationary two-phase flow problems this is nothing special since mass and stiffness matrices containing discontinuous material properties like density and viscosity have to be updated as well, cf. Section 6.2. What is special about the extended pressure finite element space is the fact that the dimension of Q_h^Γ may vary, i. e., some extended pressure unknowns may appear or disappear when the interface is moving. This has to be taken into account by a suitable interpolation procedure for the extended pressure unknowns.

Regarding stability, one has to treat carefully the situation where some extended basis functions q_j^Γ have only a "small" support, because then the resulting system matrices are ill-conditioned. As a consequence, the convergence rate of the iterative solvers can decrease significantly or solvers may even break down. One obvious possibility to deal with this stability problem is to skip the extended basis functions with relatively "small" contributions. What is meant by "small" will be specified in the following paragraphs.

A suitable strategy on how to decide which extended basis functions are to be skipped should fulfill the following two properties. On the one hand one wants to obtain a (more) stable basis of the extended pressure finite element space, on the other hand it should maintain the desired $\mathcal{O}(h)$ discretization error behavior. Such a strategy is described in [Reu08]. Let $\tilde{c} > 0, \alpha > 0$ be given parameters. For $j \in \mathcal{J}_\Gamma$ we consider the following condition for the corresponding extended basis function q_j^Γ:

$$\|q_j^\Gamma\|_{l,T} \leq \tilde{c} h_T^\alpha \|q_j\|_{l,T} \qquad \text{for all } T \in \mathcal{T}. \tag{5.63}$$

Here $l \in \{0,1\}$ is the degree of the Sobolev norm used for measuring the approximation error, cf. Theorem 5.26. We introduce the *reduced index set* $\tilde{\mathcal{J}}_\Gamma \subset \mathcal{J}_\Gamma$ by

$$\tilde{\mathcal{J}}_\Gamma := \{\, j \in \mathcal{J}_\Gamma : \text{(5.63) does not hold for } q_j^\Gamma \,\}$$

and the *reduced basis* $\tilde{\mathcal{B}}^\Gamma$ and *reduced extended finite element space* $\tilde{Q}_h^\Gamma$,

$$\tilde{\mathcal{B}}^\Gamma := \mathcal{B} \cup \{\, q_j^\Gamma : j \in \tilde{\mathcal{J}}_\Gamma \,\}, \tag{5.64}$$

$$\tilde{Q}_h^\Gamma := \operatorname{span}\left(\tilde{\mathcal{B}}^\Gamma\right). \tag{5.65}$$

In other words, all extended basis functions q_j^Γ are skipped, for which (5.63) holds. Then the optimal approximation property (5.62) given in Theorem 5.26 also holds for the reduced space $\tilde{Q}_h^\Gamma$ when choosing $\alpha = m$, cf. [Reu08].

Another important issue from the practical point of view is the design of efficient and robust solvers for the resulting discrete problems which have to be adapted to the extended pressure finite element space. These are topics of current research. Some comments on preconditioning the Schur complement in the context of XFEM are given in Remark 7.7. The idea is based on the fact, that the spectral condition number of $D_M^{-1}M$ is bounded uniformly w.r.t. h_Γ, cf. [Reu08]. Here M is the mass matrix corresponding to the basis $\mathcal{B}^\Gamma$ and D_M is its diagonal. Hence, the L_2-stability of $\mathcal{B}^\Gamma$ can be achieved by a simple diagonal scaling. The same holds for the reduced basis $\tilde{\mathcal{B}}^\Gamma$.

Theoretical issues like LBB-stability of the $\mathbf{V}_h - Q_h^\Gamma$ finite element pair are left for future research.

5.4.4. Implementation issues

Let Γ_h be a piecewise planar approximation of the interface Γ as described in Section 5.1. For practical reasons we do not consider Q_h^Γ but the space $Q_h^{\Gamma_h}$ which is much easier to construct. Here $Q_h^{\Gamma_h}$ is the extended pressure finite element space described above but with Γ replaced by its approximation Γ_h. We thus consider the finite element discretization (5.11) for the choice $Q_h = Q_h^{\Gamma_h}$. As the velocity space $\mathbf{V}_h$ is unchanged most of the terms are discretized as before. Only the evaluation of $b(\cdot,\cdot)$ requires further explanation.

For a basis function $\mathbf{v}_i \in \mathbf{V}_h$ and $j \in \mathcal{J}_\Gamma$ the evaluation of

$$b(\mathbf{v}_i, q_j^{\Gamma_h}) = -\sum_{T' \in \mathcal{T}_h'} \int_{T'} q_j^{\Gamma_h} \operatorname{div} v_i \, d\mathbf{x}$$

requires the computation of integrals with discontinuous integrands, as the extended pressure basis function $q_j^{\Gamma_h}$ has a jump across the interface. We sum over $T' \in \mathcal{T}_h'$ (and not $T \in \mathcal{T}_h$) because Γ_h is defined as in (5.9). Let

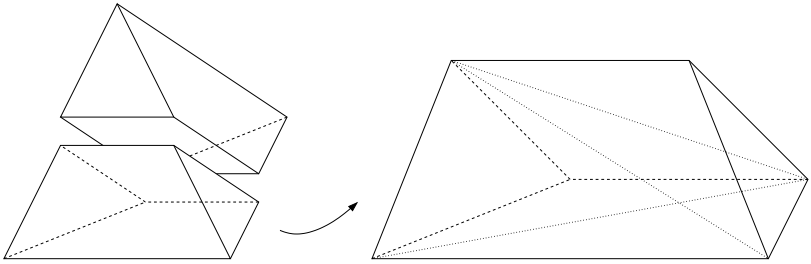

Figure 5.9.: Left: Parts of tetrahedron T' are non-tetrahedral, iff cutting face $T' \cap \Gamma_h$ is a quadrilateral. Right: Triangulation of the lower part into three tetrahedra.

$T \in \mathcal{T}_h$ be a tetrahedron with $T \cap \operatorname{supp} q_j^{\Gamma_h} \neq \emptyset$ and $T' \in \mathcal{T}_h'$ with $T' \subset T$ a child tetrahedron created by regular refinement of T. Due to (5.57) we have $T \in \mathcal{T}_h^\Gamma$, and define

$$T_i := T \cap \Omega_{i,h}, \quad T_i' := T' \cap \Omega_{i,h}, \quad i = 1, 2.$$

Using the definition of $q_j^{\Gamma_h}$, cf. (5.55), (5.56), we get

$$\begin{aligned} \int_{T'} q_j^{\Gamma_h} \operatorname{div} v_i \, d\mathbf{x} &= \int_{T_2'} q_j \operatorname{div} \mathbf{v}_i \, d\mathbf{x} - H_\Gamma(\mathbf{x}_j) \int_{T'} q_j \operatorname{div} \mathbf{v}_i \, d\mathbf{x} \\ &= \begin{cases} \int_{T_2'} q_j \operatorname{div} \mathbf{v}_i \, d\mathbf{x} & \text{if } \mathbf{x}_j \in \Omega_1, \\ -\int_{T_1'} q_j \operatorname{div} \mathbf{v}_i \, d\mathbf{x} & \text{if } \mathbf{x}_j \in \Omega_2. \end{cases} \end{aligned} \tag{5.66}$$

The integrands in the right hand side of (5.66) are continuous and the subdomains T_1', T_2' are polyhedral since by construction Γ_h consists of piecewise planar segments (cf. Section 5.1). For the computation of the integral over T_i' we distinguish two cases. The face $T' \cap \Gamma_h$ is either a triangle or a quadrilateral. In the first case one of the sets T_1', T_2' is tetrahedral, without loss of generality let T_1' be tetrahedral. Then integration over T_2' can be computed by

$$\int_{T_2'} G(\mathbf{x}) \, d\mathbf{x} = \int_{T'} G(\mathbf{x}) \, d\mathbf{x} - \int_{T_1'} G(\mathbf{x}) \, d\mathbf{x}.$$

In the second case both T_1', T_2' are non-tetrahedral, but can each be subdivided into three sub-tetrahedra, cf. Figure 5.9. In all cases the integration over T_i' can be reduced to integration on tetrahedra, for which standard quadrature rules can be applied.

6. Time discretization and coupling

In Section 6.1 we discuss several time discretization schemes for the non-stationary Navier-Stokes equations. The time discretization of the scalar level set equation is carried out in a similar way. After the time discretization a *coupled* system of quasi-stationary level set and Navier-Stokes equations is obtained. The treatment of the coupling is explained in Section 6.2.

6.1. Time discretization

In the following we consider the case that the time discretization is applied after the spatial discretization. This approach is called the 'Method of Lines' as opposed to the 'Rothe Method', where the time discretization is followed by a spatial discretization.

After the finite element discretization described in Chapter 4 we obtain the differential-algebraic equation (DAE)

$$\begin{pmatrix} M(t)\,\underline{\mathbf{u}}'(t) \\ 0 \end{pmatrix} + \begin{pmatrix} T(\underline{\mathbf{u}}(t),t) & B^T \\ B & 0 \end{pmatrix} \begin{pmatrix} \underline{\mathbf{u}}(t) \\ \underline{p}(t) \end{pmatrix} = \begin{pmatrix} \underline{\mathbf{b}}(t) \\ \underline{c}(t) \end{pmatrix}, \qquad t \in [t_0, t_f], \quad (6.1)$$

$$\text{initial condition} \quad \underline{\mathbf{u}}(t_0) = \underline{\mathbf{u}}^0, \quad (6.2)$$

with $T(\mathbf{u}(t),t) = A(t) + N(\underline{\mathbf{u}}(t),t)$, cf. (4.13).

Remark 6.1
The DAE (6.1) can also be seen as an *ordinary differential equation* (ODE) for functions $\underline{\mathbf{u}}(t) : [t_0, t_f] \to \mathbb{R}^{N_{\mathbf{V}_h}}$ which satisfy the discrete incompressibility constraint

$$B\,\underline{\mathbf{u}}(t) = \underline{c}(t) \qquad \text{for all } t \in [t_0, t_f]. \quad (6.3)$$

Note that (6.3) means that $J_{\mathbf{V}_h^D}(\underline{\mathbf{u}}(t), t) \in \mathbf{V}_h^{\mathrm{div}}$ (cf. (4.8)) for all $t \in [t_0, t_f]$, where

$$\mathbf{V}_h^{\mathrm{div}} := \{\, \mathbf{u}_h \in \mathbf{V}_h^D : b(\mathbf{u}_h, q_h) = (\operatorname{div} \mathbf{u}_h, q_h)_0 = 0 \quad \text{for all } q_h \in Q_h \,\}.$$

Hence, the DAE (6.1) is equivalent to the ODE

$$\begin{aligned} &\text{Find } \mathbf{u}_h(t) : [t_0, t_f] \to \mathbf{V}_h^{\mathrm{div}} \text{ such that for (almost every) } t \in [t_0, t_f] \\ &\quad m(\mathbf{u}_h'(t), \mathbf{v}_h) + n(\mathbf{u}_h(t); \mathbf{u}_h(t), \mathbf{v}_h) + a(\mathbf{u}_h(t), \mathbf{v}_h) = f(\mathbf{v}_h) \\ &\quad \text{holds for all } \mathbf{v}_h \in \mathbf{V}_h^{\mathrm{div}} \qquad \text{and} \qquad \mathbf{u}_h(t_0) = \mathbf{u}^0 \in \mathbf{V}_h^{\mathrm{div}}. \end{aligned} \tag{6.4}$$

In this sense the pressure $\underline{p} \in \mathbb{R}^{N_{Q_h}}$ in (6.1) can be interpreted as the Lagrange multiplier associated to the incompressibility constraint $B\,\underline{\mathbf{u}} = \underline{c}$. ◇

We now turn to the time discretization. Let

$$t_0 < t_1 < \ldots < t_{n_t} = t_f$$

be a discretization of the time interval $[t_0, t_f]$. For $0 \le i < n_t$ the size of the i-th time step is given by $k_i = t_{i+1} - t_i$. The approximation of $\underline{\mathbf{u}}(t_i), \underline{p}(t_i)$ is denoted by $\underline{\mathbf{u}}^i, \underline{p}^i$, respectively, in the following.

6.1.1. Time discretization for 1D model problem

Before discussing time discretization schemes for the DAE (6.1) arising from the 3D Navier-Stokes problem we first consider a simple 1D diffusion problem with time dependent coefficients.

Example 6.2
For $t \in [0,1]$, $x \in [0,1]$ we define the piecewise constant function

$$d(x,t) = \begin{cases} d_L & \text{if } x \le \gamma(t), \\ d_R & \text{if } x > \gamma(t), \end{cases}$$

for $\gamma(t) = \frac{1}{2} + \frac{1}{4}t$, $d_L = 1$, $d_R = 3$, and state the following non-stationary 1D diffusion problem: Find $u = u(x,t)$ such that

$$\begin{aligned} &d(x,t)\, u_t(x,t) = \varepsilon\, u_{xx}(x,t) \qquad \text{for } x \in [0,1], t \in [0,1], \\ &u|_{x=0} = u|_{x=1} = 0, \\ &u|_{t=0} = u^0(x) \end{aligned} \tag{6.5}$$

with $\varepsilon > 0$. The continuous initial value u^0 is given by

$$u^0(x) = \begin{cases} x & \text{for } x \le \frac{1}{2}, \\ (x-1)(-8x+3) & \text{for } x > \frac{1}{2}. \end{cases}$$

For the spatial discretization the interval $[0,1]$ is subdivided into $n+1$ equidistant intervals with length $h = (n+1)^{-1}$ and end points $x_i = ih$, $i = 0, 1, \ldots, n+1$. For $t \in [0,1]$ we collect the unknown values $u(x_i, t)$, $i = 1, \ldots, n$ in the vector $\underline{u}(t) \in \mathbb{R}^n$. If we use a finite difference discretization we end up with the ODE system

$$\hat{M}(t)\,\underline{u}'(t) = A\,\underline{u}(t), \qquad t \in [0,1], \tag{6.6}$$

where $\hat{M}(t)$ is a diagonal matrix with $\hat{M}_{ii}(t) = d(x_i, t)$ and the stiffness matrix

$$A = \frac{\varepsilon}{h^2} \begin{pmatrix} -2 & 1 & & \\ 1 & \ddots & \ddots & \\ & \ddots & \ddots & 1 \\ & & 1 & -2 \end{pmatrix}.$$

A finite element discretization with P_1 FE and nodal basis functions $\{v_i(x)\}_{i=1}^n$ yields the ODE system

$$M(t)\,\underline{u}'(t) = A\,\underline{u}(t), \qquad t \in [0,1], \tag{6.7}$$

where $M(t)$ is the mass matrix with entries

$$M_{ij}(t) = h^{-1} \int_0^1 d(x,t) v_i(x) v_j(x)\, dx$$

and A the stiffness matrix given above.

Note that (6.6), (6.7) can be written in the general form

$$\underline{u}'(t) = R(t)\,\underline{u}(t), \qquad t \in [0,1], \tag{6.8}$$

with $R(t) = \hat{M}(t)^{-1} A$ for the finite difference case (6.6) and $R(t) = M(t)^{-1} A$ for the finite element case (6.7). ◇

Remark 6.3 (Smoothness of $\hat{M}(t)$ and $M(t)$)
A simple computation shows that

$$M'_{ij}(t) = \frac{d_L - d_R}{h}\, v_i(\gamma(t))\, v_j(\gamma(t))\, \gamma'(t).$$

Hence, the matrix entries of $M(t)$ are C^1 w.r.t. time t. Furthermore, for a t with $\gamma(t) \neq x_i$, $i = 1, \ldots, n$, the matrix entries of $M(t)$ are of the same regularity as $\gamma(t)$ (C^∞ for Example 6.2). We emphasize that in contrast to that the diagonal entries of $\hat{M}(t)$ are discontinuous w.r.t. time t. To be more precise, $\hat{M}_{ii}(t)$ is piecewise constant w.r.t. t with a jump at $t = t_{\text{disc}}$ where $\gamma(t_{\text{disc}}) = x_i$, $i = 1, \ldots, n$. This will influence the convergence order of the time discretization scheme, see below. ◇

For Example 6.2 we study the following two time discretization schemes. Here $k := t_{n+1} - t_n$ denotes the length of the time step and the parameter θ controls the implicitness of the scheme. For a shorter notation we introduce $\theta' = 1 - \theta$.

$$\frac{\underline{u}^{n+1} - \underline{u}^n}{k} = \theta\, R(t_{n+1})\, \underline{u}^{n+1} + \theta'\, R(t_n)\, \underline{u}^n \tag{6.9}$$

is the well-known *theta-scheme*. Special cases are $\theta = 0$ (the explicit Euler scheme), $\theta = 1$ (the implicit Euler scheme) and $\theta = 1/2$ (the Crank-Nicholson scheme). The implicit Euler scheme is strongly A-stable, but only of first order. On the other hand the Crank-Nicholson scheme has second order accuracy for smooth $\underline{u}(t)$, but is not *strongly* A-stable, which may lead to instabilities in certain situations, cf. [Ran04].

$$\frac{u^{n+1} - u^n}{k} = \theta\, R(t_n)\, u^{n+1} + \theta'\, R(t_n)\, u^n \tag{6.10}$$

is a variant of the former scheme, where we applied one step of the theta-scheme to the linearized ODE problem $u'(t) = R(t_n)\, u(t)$. We will refer to that scheme as the *linearized theta-scheme*. The following lemma shows that this scheme is convergent.

Lemma 6.4
Let $y(t), \tilde{y}(t)$ be solutions of the ODE's

$$y'(t) = f(t, y(t)), \qquad \tilde{y}'(t) = \tilde{f}(t, \tilde{y}(t))$$

with $y(t_n) = \tilde{y}(t_n)$. If $f(t_n, y(t_n)) = \tilde{f}(t_n, y(t_n))$ and $f, \tilde{f}$ are C^1 functions w.r.t. t and y, then

$$y(t_{n+1}) - \tilde{y}(t_{n+1}) = \mathcal{O}(k^2).$$

Proof. By Taylor expansion there exists $\tau \in (t_n, t_{n+1})$ with

$$\begin{aligned} y(t_{n+1}) &= y(t_n) + k y'(t_n) + \frac{k^2}{2} y''(\tau) \\ &= y(t_n) + k f(t_n, y(t_n)) + \frac{k^2}{2} (f_t + f_y f)(\tau) \end{aligned}$$

and $\tilde{\tau} \in (t_n, t_{n+1})$ with

$$\begin{aligned}\tilde{y}(t_{n+1}) &= \tilde{y}(t_n) + k\tilde{y}'(t_n) + \frac{k^2}{2}\tilde{y}''(\tilde{\tau}) \\ &= y(t_n) + kf(t_n, y(t_n)) + \frac{k^2}{2}(\tilde{f}_t + \tilde{f}_y\tilde{f})(\tilde{\tau}).\end{aligned}$$ □

As a consequence, the linearized theta-scheme is convergent of order one for all $\theta \in [0, 1]$.

Remark 6.5 (Warning)
Starting with the formulation (6.6) or (6.7), one might think that

$$\frac{M(t_{n+1})\, u^{n+1} - M(t_n)\, u^n}{k} = \theta\, A\, u^{n+1} + \theta'\, A\, u^n$$

also seems to be a reasonable time discretization scheme. But one can easily show that in case of a time-dependent matrix $M(t)$ this scheme does in general *not converge* to the right solution and should therefore not be used. ◇

Using a *fixed* spatial discretization with finite differences for $n = 30$ unknowns and the choice $\varepsilon = 10^{-3}$ we implemented both time discretization schemes (6.9) and (6.10) in MATLAB [Mat]. Applying the theta-scheme with $\theta = \frac{1}{2}$ for a small time step size of $k = 10^{-4}$ we computed a reference solution $\underline{u}_1^* := \underline{u}^{10^4}$ approximating the solution $\underline{u}(t)$ of the ODE system (6.8) for the final time $t = 1$. For different numbers of time steps $n_t \leq 1000$ with step size $k = n_t^{-1}$, we computed the approximations $\underline{u}_1(k) := \underline{u}^{n_t}$ of the solution $\underline{u}(1)$ and compared them with the reference solution,

$$e(k) = |u_1(k) - u_1^*|.$$

The error $e(k)$ as a function of the step length k for $\theta = 1/2$ is shown in Figure 6.1 on the left.

We repeated this procedure for the finite element discretization on the same spatial grid. The error behavior is shown in Figure 6.1 on the right. Note that the reference solutions u_1^* for the finite difference and finite element case (slightly) differ from each other as the ODE systems (6.6) and (6.7) are different as well.

Second order convergence can only be observed for the Crank-Nicholson time discretization scheme (theta-scheme with $\theta = 1/2$) combined with the finite element discretization. In all other cases only first order convergence is achieved.

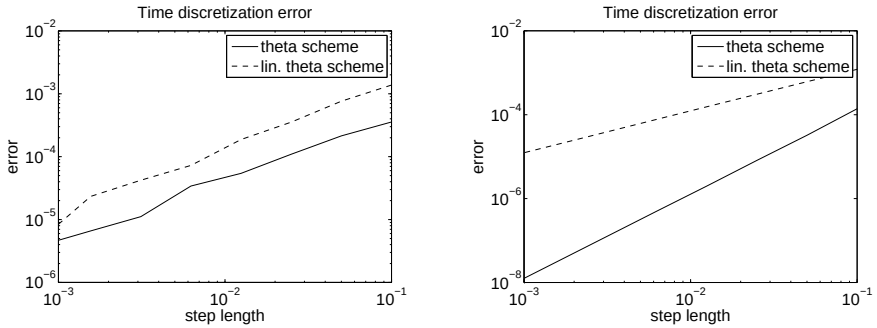

Figure 6.1.: Time discretization error $e(k)$ as function of step size k for theta-scheme and linearized theta-scheme, $\theta = 1/2$. Fixed spatial discretization with finite differences (on the left) or finite elements (on the right).

We emphasize that the Crank-Nicholson scheme yields only first order convergence when combined with the finite difference discretization. Obviously, this is due to the different regularities of $\hat{M}(t)$ and $M(t)$ as functions of t, cf. Remark 6.3. We will analyze this in the following paragraphs.

For that we further simplify (6.8) and study the following scalar ODE

$$\begin{aligned} u'(t) &= r(t)\, u(t) \qquad \text{for } t \in [t_0, t_f], \\ u(t_0) &= u_0, \end{aligned} \tag{6.11}$$

with time-dependent coefficient $r(t) > 0$. The solution of the ODE is given by

$$u(t) = C + e^{t \int_{t_0}^{t} r(\tau)\, d\tau}$$

where $C = u_0 - 1$.

We assume that there is a $t_\Gamma \in [t_0, t_f]$ such that $r(t)$ can be written as

$$r(t) = \begin{cases} f_L(t) & \text{if } t \in [t_0, t_\Gamma], \\ f_R(t) & \text{if } t \in (t_\Gamma, t_f], \end{cases}$$

with C^2 functions $f_L : [t_0, t_\Gamma] \to \mathbb{R}$, $f_R : [t_\Gamma, t_f] \to \mathbb{R}$.

Lemma 6.6
Let $t_n \to t_{n+1}$ be the time step with $t_\Gamma \in [t_n, t_{n+1}]$. Starting from $u^n = u(t_n)$ one time step of the theta-scheme and the linearized theta-scheme is applied yielding u_i^{n+1}, $i = 1, 2$, respectively. Then for the local truncation errors $\tau_i(\theta) := u(t_{n+1}) - u_i^{n+1}$ the following holds.

- For general f_L, f_R we have

$$\tau_i(\theta) = \mathcal{O}(k), \qquad i = 1, 2.$$

- If $f_L(t_\Gamma) = f_R(t_\Gamma)$, i. e., $r(t)$ is continuous on $[t_0, t_f]$, then

$$\tau_i(\theta) = \mathcal{O}(k^2), \qquad i = 1, 2.$$

- If $f_L(t_\Gamma) = f_R(t_\Gamma)$ and $f_L'(t_\Gamma) = f_R'(t_\Gamma)$, i. e., $r(t)$ is a C^1 function on $[t_0, t_f]$, then

$$\tau_1(\theta) = C\left(\theta - \frac{1}{2}\right) k^2 + \mathcal{O}(k^3) \qquad \text{and} \qquad \tau_2(\theta) = \mathcal{O}(k^2).$$

Proof. Without loss of generality we take $t_\Gamma = 0$, $t_n = -\lambda k$, $t_{n+1} = (1-\lambda)k$ with $0 \le \lambda \le 1$. A Taylor expansion of τ_i around $t = t_\Gamma$ then yields the results. □

6.1.2. One-step theta-scheme

In the following we will derive the theta-scheme and linearized theta-scheme for the non-stationary Navier-Stokes equations. Based on these schemes the fractional-step scheme can easily be constructed, cf. Section 6.1.3. For ease of presentation we assume homogeneous Dirichlet boundary conditions, i. e., $\underline{c}(t) = 0$, and that the operator $B(t)$ does *not* depend on t. In the general case the time discretization is slightly changed, see Remarks 6.9 and 6.10 below.

The DAE system (6.1) is rewritten in the form

$$\begin{aligned} \underline{\mathbf{u}}'(t) + M(t)^{-1} B^T \underline{p}(t) &= M(t)^{-1}\, g(\underline{\mathbf{u}}(t), t), \\ B\,\underline{\mathbf{u}}(t) &= 0, \end{aligned} \tag{6.12}$$

where

$$g(\underline{\mathbf{u}}(t), t) := \underline{\mathbf{b}}(t) - T(\underline{\mathbf{u}}(t), t)\,\underline{\mathbf{u}}(t).$$

Similarly to Remark 6.1 we first eliminate the incompressibility constraint $B\underline{\mathbf{u}}(t) = 0$ and the corresponding Lagrange multiplier $\underline{p}(t)$ to replace the DAE system by an equivalent ODE system. This can be achieved by applying the $M(t)$-orthogonal projection $P(t)$ on $\ker B$, i. e., $P(t)$ is orthogonal w.r.t. the scalar product $\langle \cdot\,, \cdot \rangle_{M(t)} := \langle M(t)\,\cdot\,, \cdot \rangle$,

$$P(t) = I - M(t)^{-1} B^T (B M(t)^{-1} B^T)^{-1} B.$$

Note that $P(t)\,\underline{v} = \underline{v}$ for all $\underline{v} \in \ker B$ and $P(t)M(t)^{-1}B^T = 0$. Thus the solution $\underline{\mathbf{u}}(t) \in \ker B$ of (6.12) satisfies

$$\underline{\mathbf{u}}'(t) = P(t)M(t)^{-1}\, g(\underline{\mathbf{u}}(t), t) \qquad =: f(t, \underline{\mathbf{u}}(t)). \tag{6.13}$$

For a given velocity $\underline{\mathbf{u}}(t)$ the corresponding pressure $\underline{p}(t)$ is defined by the equation

$$BM(t)^{-1}B^T\underline{p}(t) = BM(t)^{-1}\, g(\underline{\mathbf{u}}(t), t). \tag{6.14}$$

Theta-scheme

We first derive the theta-scheme for the non-stationary Navier-Stokes equations. Applying the theta-scheme to the ODE system (6.13) yields

$$\frac{\underline{\mathbf{u}}^{n+1} - \underline{\mathbf{u}}^n}{k} = \theta\, P_{n+1}M_{n+1}^{-1}\, g(\underline{\mathbf{u}}^{n+1}, t_{n+1}) + \theta'\, P_nM_n^{-1}\, g(\underline{\mathbf{u}}^n, t_n). \tag{6.15}$$

Due to $P_{n+1}P_n = P_n$, the sequence $\{\underline{\mathbf{u}}^n\}_{n\geq 0}$ is also a solution of

$$\frac{\underline{\mathbf{u}}^{n+1} - \underline{\mathbf{u}}^n}{k} + M_{n+1}^{-1}B^T\tilde{\underline{p}} = \theta\, M_{n+1}^{-1}\, g(\underline{\mathbf{u}}^{n+1}, t_{n+1}) + \theta'\, P_nM_n^{-1}\, g(\underline{\mathbf{u}}^n, t_n), \tag{6.16}$$

$$B\,\underline{\mathbf{u}}^{n+1} = 0, \tag{6.17}$$

which can be seen by applying P_{n+1} to (6.16). Using (6.14) a simple calculation shows

$$P_nM_n^{-1}\, g(\underline{\mathbf{u}}^n, t_n) = M_n^{-1}\, g(\underline{\mathbf{u}}^n, t_n) - M_n^{-1}B^T\underline{p}^n.$$

Substituting this expression in (6.16) and multiplying by B from the left we obtain

$$\begin{aligned} BM_{n+1}^{-1}B^T\tilde{\underline{p}} &= \theta\, BM_{n+1}^{-1}\, g(\underline{\mathbf{u}}^{n+1}, t_{n+1}) \\ &\quad + \theta'\, BM_n^{-1}\left(g(\underline{\mathbf{u}}^n, t_n) - B^T\underline{p}^n\right) \\ &= \theta\, BM_{n+1}^{-1}B^T\underline{p}^{n+1}. \end{aligned}$$

This gives rise to the choice $\tilde{\underline{p}} = \theta\,\underline{p}^{n+1}$. Summarizing, the theta-scheme for the Navier-Stokes equations is given by

$$\begin{aligned} &[k^{-1}M_{n+1} + \theta\, T_{n+1}(\underline{\mathbf{u}}^{n+1})]\,\underline{\mathbf{u}}^{n+1} + \theta\, B^T\underline{p}^{n+1} \\ &\quad = \theta\,\underline{\mathbf{b}}^{n+1} + M_{n+1}\left(\frac{1}{k}\underline{\mathbf{u}}^n + \theta'\, M_n^{-1}\left(g(\underline{\mathbf{u}}^n, t_n) - B^T\underline{p}^n\right)\right), \end{aligned} \tag{6.18}$$

$$B\,\underline{\mathbf{u}}^{n+1} = 0. \tag{6.19}$$

Each time step of the (linearized) one-step theta-scheme requires the solution of a generalized Navier-Stokes problem. We should mention that whenever $\theta \neq 1$ we need an initial value $\underline{p}^0$ for the pressure which is obtained from the equation

$$BM(t_0)^{-1}B^T\underline{p}^0 = BM(t_0)^{-1}\, g(\underline{\mathbf{u}}^0, t_0).$$

The solution of a linear system with M_n on the right-hand side of (6.18) in each time step can be avoided by introducing an additional variable

$$\underline{\mathbf{z}}^n := M_n^{-1}\left(g(\underline{\mathbf{u}}^n, t_n) - B^T\underline{p}^n\right) \in \ker B.$$

After an initial computation of $\underline{\mathbf{z}}^0$ by solving the linear system $M_0\,\underline{\mathbf{z}}^0 = g(\underline{\mathbf{u}}^0, t_0) - B^T\underline{p}^0$, the variable can be updated in each time step by the simple recurrence formula

$$\theta\,\underline{\mathbf{z}}^{n+1} = \frac{\underline{\mathbf{u}}^{n+1} - \underline{\mathbf{u}}^n}{k} - \theta'\,\underline{\mathbf{z}}^n.$$

This leads to the following scheme,

$$\begin{aligned}
&[k^{-1}M_{n+1} + \theta\, T_{n+1}(\underline{\mathbf{u}}^{n+1})]\,\underline{\mathbf{u}}^{n+1} + \theta\, B^T\underline{p}^{n+1} \\
&\qquad = \theta\,\underline{\mathbf{b}}^{n+1} + M_{n+1}\left(\frac{1}{k}\underline{\mathbf{u}}^n + \theta'\,\mathbf{z}^n\right),
\end{aligned} \tag{6.20}$$

$$B\,\underline{\mathbf{u}}^{n+1} = 0, \tag{6.21}$$

$$\underline{\mathbf{z}}^{n+1} = \frac{\underline{\mathbf{u}}^{n+1} - \underline{\mathbf{u}}^n}{\theta k} - \frac{\theta'}{\theta}\,\underline{\mathbf{z}}^n. \tag{6.22}$$

Linearized theta-scheme

We now derive the linearized theta-scheme for the Navier-Stokes equations. The ODE system (6.13) is modified in the following way.

$$\underline{\mathbf{u}}'(t) = P(t_n)M(t_n)^{-1}\left[\underline{\mathbf{b}}(t) - T(\underline{\mathbf{u}}(t), t_n)\,\underline{\mathbf{u}}(t)\right] \qquad =: f_n(t, \underline{\mathbf{u}}(t)). \tag{6.23}$$

We emphasize that the compatibility condition $f_n(t_n, \underline{\mathbf{u}}(t_n)) = f(t_n, \underline{\mathbf{u}}(t_n))$ holds and thus Lemma 6.4 can be applied. Note that also other choices for such compatible $f_n(t, \underline{\mathbf{u}}(t))$ are possible. Application of the theta-scheme to (6.23) yields

$$\frac{\underline{\mathbf{u}}^{n+1} - \underline{\mathbf{u}}^n}{k} = P_n M_n^{-1}\left(\underline{\mathbf{b}}_\theta^{n+1} - \theta\, T_n(\underline{\mathbf{u}}^{n+1})\underline{\mathbf{u}}^{n+1} - \theta'\, T_n(\underline{\mathbf{u}}^n)\,\underline{\mathbf{u}}^n\right), \tag{6.24}$$

where we used the notation $C_n := C(t_n)$ for a time dependent operator $C = C(t)$ and the notation $\underline{c}_\theta^{n+1} := \theta\,\underline{c}^{n+1} + \theta'\,\underline{c}^n$ for vectors $\underline{c}^n, \underline{c}^{n+1}$. The sequence $\{\underline{\mathbf{u}}^n\}_{n\geq 0}$ is also a solution of

$$M_n \frac{\underline{\mathbf{u}}^{n+1} - \underline{\mathbf{u}}^n}{k} + \theta\, T_n(\underline{\mathbf{u}}^{n+1})\,\underline{\mathbf{u}}^{n+1} + \theta'\, T_n(\underline{\mathbf{u}}^n)\,\underline{\mathbf{u}}^n + B^T\underline{\tilde{p}} = \underline{\mathbf{b}}_\theta^{n+1}, \qquad (6.25)$$

$$B\,\underline{\mathbf{u}}^{n+1} = 0. \qquad (6.26)$$

Note that multiplying (6.25) with BM_n^{-1} from the left yields

$$BM_n^{-1}B^T\underline{\tilde{p}} = BM_n^{-1}\left(\underline{\mathbf{b}}_\theta^{n+1} - \theta\, T_n(\underline{\mathbf{u}}^{n+1})\,\underline{\mathbf{u}}^{n+1} - \theta'\, T_n(\underline{\mathbf{u}}^n)\,\underline{\mathbf{u}}^n\right).$$

Hence, for the choice $\underline{\tilde{p}} = \underline{p}_\theta^{n+1}$ the pressure variable satisfies the pressure equation $BM_n^{-1}B^T\underline{p}(t) = BM_n^{-1}\,g(\mathbf{u}(t), t_n)$ which is (6.14) linearized at $t = t_n$.

Summarizing, the linearized theta-scheme is as follows:

$$[k^{-1}M_n + \theta\, T_n(\underline{\mathbf{u}}^{n+1})]\,\underline{\mathbf{u}}^{n+1} + \theta\, B^T\underline{p}^{n+1} \qquad (6.27)$$
$$= \frac{1}{k}M_n\,\underline{\mathbf{u}}^n + \theta\,\underline{\mathbf{b}}^{n+1} + \theta'\,\left(\underline{\mathbf{b}}^n - T_n(\underline{\mathbf{u}}^n)\,\underline{\mathbf{u}}^n - B^T\underline{p}^n\right),$$

$$B\,\underline{\mathbf{u}}^{n+1} = 0. \qquad (6.28)$$

Remark 6.7
An alternative linearized scheme is the following, cf. [QV94],

$$M_n\frac{\underline{\mathbf{u}}_\theta^{n+1} - \underline{\mathbf{u}}^n}{\theta k} + T_n(\underline{\mathbf{u}}_\theta^{n+1})\,\underline{\mathbf{u}}_\theta^{n+1} + B^T\underline{p}_\theta^{n+1} = \underline{\mathbf{b}}_\theta^{n+1},$$

$$B\,\underline{\mathbf{u}}_\theta^{n+1} = \begin{cases} \theta'\,B\underline{\mathbf{u}}^0 & n = 0, \\ 0 & n \geq 1, \end{cases}$$

which is solved for the unknown quantities $\underline{\mathbf{u}}_\theta^{n+1}, p_\theta^{n+1}$. After that we set

$$\underline{\mathbf{u}}^{n+1} = \theta^{-1}(\underline{\mathbf{u}}^{n+1} - \theta'\,\underline{\mathbf{u}}^n), \qquad \underline{p}^{n+1} = \theta^{-1}(\underline{p}^{n+1} - \theta'\,\underline{p}^n). \qquad \diamond$$

Some Remarks

Remark 6.8 (Order of convergence for two-phase Stokes flow)
Numerical experiments of a two-phase Stokes flow problem (rising bubble problem, cf. also Section 11.2) have been conducted in [Ess08] to examine the convergence order of the theta-scheme and the linearized theta-scheme for the parameter choices $\theta = 0.5$ (Crank-Nicholson scheme) and $\theta = 1$ (implicit Euler scheme), respectively. Similar to the 1D experiments presented in Section 6.1.1, a reference solution $\mathbf{u}^*(t)$, $t \in [t_0, t_f]$, was computed applying a very small time step size k_{ref}. For these time discretization schemes solutions $\mathbf{u}_k(t)$, $t \in [t_0, t_f]$, were computed for different time step sizes $k > k_{\text{ref}}$ and afterwards the corresponding errors

$$e(k) := \|\mathbf{u}_k(t_f) - \mathbf{u}^*(t_f)\|_{L_2(\Omega)}$$

were calculated. The numerical results show second order convergence for the theta-scheme with $\theta = 0.5$ and first order convergence in the remaining three cases. Thus the expected theoretical convergence order was confirmed by these experiments. ◇

Up to now we assumed that the incompressibility constraint has the form $B\,\underline{\mathbf{u}}(t) = 0$. In the following we treat the more general case $B(t)\,\underline{\mathbf{u}}(t) = \underline{c}(t)$.

Remark 6.9 (Extension to non-homogeneous right-hand side)
A non-homogeneous right-hand side $\underline{c}(t) \neq 0$ is caused by non-homogeneous Dirichlet boundary conditions $\mathbf{u}|_{\Sigma_D} = \mathbf{u}_D$ for the velocity in the finite element discretization of the incompressibility constraint $\operatorname{div} \mathbf{u} = 0$. In order to explain the time discretization in this case, for a moment we introduce additional unknowns for the degrees of freedom located at the Dirichlet boundary Σ_D and collect these in the vector $\underline{\mathbf{u}}_D$. Doing so we obtain a finite element discretization

$$\begin{pmatrix} \tilde{M}(t)\,\underline{\tilde{\mathbf{u}}}'(t) \\ 0 \end{pmatrix} + \begin{pmatrix} \tilde{T}(\underline{\tilde{\mathbf{u}}}(t), t) & \tilde{B}^T \\ \tilde{B} & 0 \end{pmatrix} \begin{pmatrix} \underline{\tilde{\mathbf{u}}}(t) \\ \underline{p}(t) \end{pmatrix} = \begin{pmatrix} \underline{\tilde{\mathbf{b}}}(t) \\ 0 \end{pmatrix}, \qquad t \in [t_0, t_f],$$

with an augmented vector

$$\underline{\tilde{\mathbf{u}}}(t) = \begin{pmatrix} \underline{\mathbf{u}}(t) \\ \underline{\mathbf{u}}_D(t) \end{pmatrix}$$

and augmented matrices $\tilde{M}, \tilde{T}, \tilde{B}$. This system has the form of (6.1), but with $c(t) = 0$. Thus the derivation of the (linearized) one-step theta-scheme can be carried out as presented in this section. A subsequent elimination of the

unknowns $\underline{\mathbf{u}}_D$ (which can be substituted directly due to the known Dirichlet boundary condition) leads to the schemes (6.27) and (6.20) with modified right-hand sides $\underline{\mathbf{b}}^n$, $\underline{\mathbf{b}}^{n+1}$ and (6.28), (6.21) replaced by

$$B\,\underline{\mathbf{u}}^{n+1} = \underline{c}^{n+1}. \qquad \diamond$$

Remark 6.10 (Extension to time dependent operator $B(t)$)
We now assume that the operator $B(t)$ is time dependent, which is in general the case for an extended finite element discretization of the pressure due to the dynamics of the interface Γ, cf. Section 5.4. In the derivation of the one-step theta-scheme we used that $P_{n+1}P_n = P_n$, which is no longer true for a time dependent operator $B(t)$. Hence, a different approach has to be chosen which will not be explained here, but can be found in [RFG$^+$]. The same time stepping theme as in (6.20)–(6.22) is obtained, but with a different initial pressure $\underline{p}^0$ given by

$$B_0 M_0^{-1} B_0^T\, \underline{p}^0 = B_0 M_0^{-1}\, g(\underline{\mathbf{u}}^0, t_0) + \frac{dB}{dt}(t_0)\,\underline{\mathbf{u}}^0. \tag{6.29}$$

Note that the derivation of the *linearized* one-step theta-scheme remains unchanged for B depending on t, since here the projection $P(t)$ is applied only for a fixed time $t = t_n$, cf. (6.23). The initial pressure $\underline{p}^0$ should also be computed according to (6.29). $\diamond$

Time discretization of the level set equation

We now discuss the time discretization of the level set equation. After the finite element discretization of the level set equation we obtain the following system of ordinary differential equations,

$$E(\underline{\mathbf{u}}(t))\,\underline{\varphi}'(t) + H(\underline{\mathbf{u}}(t))\,\underline{\varphi}(t) = 0, \tag{6.30}$$

$$\underline{\varphi}(t_0) = \underline{\varphi}^0. \tag{6.31}$$

The application of the one-step theta-scheme to (6.30) yields

$$\left[\frac{1}{k}E(\underline{\mathbf{u}}^{n+1}) + \theta\, H(\underline{\mathbf{u}}^{n+1})\right] \underline{\varphi}^{n+1} = E(\underline{\mathbf{u}}^{n+1}) \left[\frac{1}{k}\underline{\varphi}^n - \theta'\, E(\underline{\mathbf{u}}^n)^{-1} H(\underline{\mathbf{u}}^n)\,\underline{\varphi}^n\right].$$

Together with the one-step theta-scheme for the Navier-Stokes problem (6.20)–(6.22) this leads to a coupled system which has to be solved for the unknown

quantities $(\underline{\mathbf{u}}^{n+1}, \underline{p}^{n+1}, \underline{\varphi}^{n+1})$ in each time step. This issue is further discussed in Section 6.2.

The application of the *linearized* theta-scheme to (6.30) leads to

$$\left[\frac{1}{k}E(\underline{\mathbf{u}}^n) + \theta H(\underline{\mathbf{u}}^n)\right] \underline{\varphi}^{n+1} = \left[\frac{1}{k}E(\underline{\mathbf{u}}^n) - \theta' H(\underline{\mathbf{u}}^n)\right] \underline{\varphi}^n.$$

Note that here only $\underline{\mathbf{u}}^n$ (and not $\underline{\mathbf{u}}^{n+1}$) has to be known to obtain the level set approximation $\underline{\varphi}^{n+1}$ at the new time $t = t_{n+1}$. The implications for the overall time stepping scheme (level set and Navier-Stokes equations) are described in Section 6.2.

6.1.3. Fractional-step scheme

For the fractional-step scheme the time step $t_n \to t_{n+1}$ with time step size $k = t_{n+1} - t_n$ is subdivided into three macro time steps

$$t_n \;\to\; t_{n+\Theta} \;\to\; t_{n+\hat{\Theta}} \;\to\; t_{n+1},$$

where

$$t_{n+\Theta} = t_n + \Theta k, \qquad t_{n+\hat{\Theta}} = t_{n+1} - \Theta k,$$

with macro time step sizes Θk, $(1-2\Theta)k$, Θk, respectively. Here $0 < \Theta < 1/2$ defines the portion of the first and third macro step in relation to the time step size k. The portion of the second macro step is denoted by $\Theta' := 1 - 2\Theta$.

The fractional-step scheme is based on a splitting $T = T_{(1)} + T_{(2)}$ of the operator T in (6.1), where in the first and third macro step the operator $T_{(1)}$ is treated implicitly while $T_{(2)}$ is treated explicitly, and vice-versa for the second macro step. A popular variant is based on the splitting

$$T_{(1)} = \alpha T, \qquad T_{(2)} = (1-\alpha)T$$

with $0 < \alpha < 1$, cf. [Tur99, Ran04]. This means that in each macro time step the theta-scheme is applied, where $\theta = \alpha$ is chosen for the first and third macro time step and $\theta = 1 - \alpha$ for the second macro time step. Hence, the application of the fractional-step scheme requires the solution of three generalized Navier-Stokes problems per time step. The level set is updated by applying the corresponding theta-scheme in each macro time step.

In the case of one-phase flow, for $\Theta = 1 - \sqrt{2}/2 = 0.292893\ldots$ and $1/2 < \alpha \leq 1$ this scheme has second order accuracy and is strongly A-stable, cf.

[BGP87]. Due to this fact we choose $\Theta = 1 - \sqrt{2}/2$ and discuss the choice of the parameter α in the following.

For a shorter notation we introduce $\alpha' = 1 - \alpha$. One often chooses $\alpha = \frac{1-2\theta}{1-\theta} = 0.585786\ldots$ as the operators in all three macro steps then have the same structure (up to a constant factor) due to $\alpha\theta = \alpha'\theta'$. This can be exploited in the construction of the system matrices, if M and A are time independent (which is usually the case for one-phase flow problems). For two phase problem this is no longer the case because of the non-stationary interface $\Gamma = \Gamma(t)$ and thus these matrices have to be rebuild in each macro step, anyway. However, there are still some advantages as for this choice of α the structure of the Schur complement preconditioners stays the same in each macro step, cf. Section 7.2.3. Thus we chose α as indicated above.

A linearized variant of the fractional step scheme can similarly be derived by applying the linearized theta-scheme in each macro time step. However, for this linearized variant we cannot expect better than first order convergence.

6.1.4. Fractional-step scheme with operator splitting

The fractional-step scheme was first proposed by Glowinski et al. [BGP87] in form of an operator splitting approach, where the two main challenges of the Navier-Stokes equations, incompressibility and nonlinearity, are decoupled from each other. It is based on the splitting $T_{(1)} = \alpha A$ and $T_{(2)} = (1-\alpha)A + N$ where the incompressibility constraint is omitted in the second macro step. The linearized fractional-step operator splitting scheme is as follows:

$$\begin{cases} \left[\frac{1}{\Theta k} M + \alpha A\right]_n \underline{\mathbf{u}}^{n+\Theta} + B^T \underline{p}^{n+\Theta} = \left[\frac{1}{\Theta k} M - \alpha' A - N\right]_n \underline{\mathbf{u}}^n + \underline{\mathbf{b}}^n \\ B\,\underline{\mathbf{u}}^{n+\Theta} = \underline{c}^n \\ \left[\frac{1}{\Theta k} E + \alpha H\right]_n \underline{\varphi}^{n+\Theta} = \left[\frac{1}{\Theta k} E - \alpha' H\right]_n \underline{\varphi}^n \end{cases} \tag{6.32}$$

$$\begin{cases} \left[\frac{1}{\Theta' k}M + \alpha' A + N\right]_{n+\Theta} \underline{\mathbf{u}}^{n+\hat{\Theta}} = \left[\frac{1}{\Theta' k}M - \alpha A\right]_{n+\Theta} \underline{\mathbf{u}}^{n+\Theta} \\ \qquad\qquad - B^T \underline{p}^{n+\Theta} + \underline{\mathbf{b}}^{n+\hat{\Theta}} \\ \left[\frac{1}{\Theta' k}E + \alpha' H\right]_{n+\Theta} \underline{\varphi}^{n+\hat{\Theta}} = \left[\frac{1}{\Theta' k}E - \alpha H\right]_{n+\Theta} \underline{\varphi}^{n+\Theta} \end{cases} \tag{6.33}$$

$$\begin{cases} \left[\frac{1}{\Theta k}M + \alpha A\right]_{n+\hat{\Theta}} \underline{\mathbf{u}}^{n+1} + B^T \underline{p}^{n+1} = \left[\frac{1}{\Theta k}M - \alpha' A - N\right]_{n+\hat{\Theta}} \underline{\mathbf{u}}^{n+\hat{\Theta}} \\ \qquad\qquad + \underline{\mathbf{b}}^{n+\hat{\Theta}} \\ B\,\underline{\mathbf{u}}^{n+1} = \underline{c}^{n+1} \\ \left[\frac{1}{\Theta k}E + \alpha H\right]_{n+\hat{\Theta}} \underline{\varphi}^{n+1} = \left[\frac{1}{\Theta k}E - \alpha' H\right]_{n+\hat{\Theta}} \underline{\varphi}^{n+\hat{\Theta}} \end{cases} \tag{6.34}$$

This scheme will be referred to as FS-OS scheme in the remainder of the text. Performing one time step of the FS-OS scheme requires the solution of an Oseen problem in the first and third macro step respectively, and one nonlinear Burgers-type problem in the second macro step. This has the advantage that the nonlinearity and the incompressibility constraint can be treated separately. Thus one can reuse solution techniques developed for linear Stokes problems. The level set variable $\underline{\varphi}$ is updated by applying the corresponding linearized theta-scheme in each macro time step.

At least for one-phase flow problems the FS-OS scheme is strongly A-stable, but has only first order accuracy. For further analysis of the scheme we refer to [KR94]. In [Bän98] the combination of the linearized FS-OS scheme with the implicit treatment of the surface force f_Γ is described and analyzed. This topic is discussed in Section 6.1.5 below.

6.1.5. Implicit treatment of the CSF term

The time discretization of the surface tension force term f_Γ deserves special attention, as its *explicit* treatment will lead to a capillary time step restriction

$$\Delta t < \sqrt{\frac{(\rho_1+\rho_2)h_\Gamma^3}{4\pi\tau}} \quad \sim \quad h^{3/2}\,\tau^{-1/2}.$$

A derivation of this bound can be found in [BKZ92]. To overcome this problem a semi-implicit treatment of f_Γ is suggested in [Bän98] which will be briefly described in the following. Recall from Section 5.3 that f_Γ can be expressed by means of the Laplace-Beltrami operator Δ_Γ,

$$f_\Gamma(\mathbf{v}) = \tau \int_\Gamma (\Delta_\Gamma \,\mathrm{id}_\Gamma)\cdot \mathbf{v}\, ds = -\tau \int_\Gamma \nabla_\Gamma \,\mathrm{id}_\Gamma \cdot \nabla_\Gamma \mathbf{v}\, ds, \qquad \mathbf{v}\in \mathbf{V}_h.$$

The new interface position Γ^{n+1} for $t = t_{n+1}$ can be expressed in terms of the old interface position Γ^n by means of the identity operator id_Γ,

$$\mathrm{id}_{\Gamma^{n+1}} = \mathrm{id}_{\Gamma^n} + k_{n+1}\,\mathbf{u}^{n+1} + \mathcal{O}(k_{n+1}^2), \tag{6.35}$$

where $k_{n+1} = t_{n+1} - t_n$ denotes the corresponding time step size. This gives rise to the formulation

$$\begin{aligned} f_{\Gamma^{n+1}}(\mathbf{v}) &= -\tau \int_{\Gamma^{n+1}} \nabla_\Gamma \,\mathrm{id}_{\Gamma^{n+1}} \cdot \nabla_\Gamma \mathbf{v}\, ds \qquad \mathbf{v}\in \mathbf{V}_h. \\ &\approx -\tau \int_{\Gamma^n} \nabla_\Gamma \,\mathrm{id}_{\Gamma^n} \cdot \nabla_\Gamma \mathbf{v}\, ds - k_{n+1}\,\tau \int_{\Gamma^n} \nabla_\Gamma \mathbf{u}^{n+1}\cdot \nabla_\Gamma \mathbf{v}\, ds, \end{aligned} \tag{6.36}$$

which is a *semi*-implicit discretization as the integration is performed on Γ^n instead of Γ^{n+1}. It is *implicit* in the sense that the second integral in (6.36) defines a bilinear form $c_\Gamma(\cdot,\cdot)_n$ on $\mathbf{V}\times\mathbf{V}$,

$$c_\Gamma(\mathbf{u},\mathbf{v})_n := \tau \int_{\Gamma^n} \nabla_\Gamma \mathbf{u}\cdot \nabla_\Gamma \mathbf{v}\, ds, \qquad \mathbf{u},\mathbf{v}\in \mathbf{V},$$

which contributes to the velocity operator on the left-hand side of the momentum equation (2.21). This interface diffusion term $c_\Gamma(\cdot,\cdot)_n$ is an elliptic operator and thus has a stabilizing effect. For the finite element discretization we consider the bilinear form

$$c_{\Gamma_h}(\mathbf{u}_h,\mathbf{v}_h)_n := \tau \int_{\Gamma_h^n} \nabla_{\Gamma_h} \mathbf{u}_h \cdot \nabla_{\Gamma_h} \mathbf{v}_h\, ds, \qquad \text{for all } \mathbf{u}_h,\mathbf{v}_h\in \mathbf{V}_h.$$

The corresponding matrix $C_n^{\Gamma_h} \in \mathbb{R}^{N_{\mathbf{V}_h} \times N_{\mathbf{V}_h}}$ is defined by

$$\langle C_n^{\Gamma_h}\, \underline{\mathbf{u}},\, \underline{\mathbf{v}} \rangle := c_{\Gamma_h}\big(J_{\mathbf{V}_h}(\underline{\mathbf{u}}),\, J_{\mathbf{V}_h}(\underline{\mathbf{v}})\big)_n$$

for all $\underline{\mathbf{u}}, \underline{\mathbf{v}} \in \mathbb{R}^{N_{\mathbf{V}_h}}$.

Applying this technique exemplarily to the linearized one-step theta-scheme results in the following scheme,

$$\begin{aligned} &[k^{-1} M_n + \theta\, T_n(\underline{\mathbf{u}}^{n+1}) + k\, C_n^{\Gamma_h}]\, \underline{\mathbf{u}}^{n+1} + \theta\, B^T \underline{p}^{n+1} \\ &\qquad = \frac{1}{k} M_n\, \underline{\mathbf{u}}^n + \underline{\mathbf{b}}_{\Gamma_h}^n + \theta\, \hat{\underline{\mathbf{b}}}^{n+1} + \theta' \left(\hat{\underline{\mathbf{b}}}^n - T_n(\underline{\mathbf{u}}^n)\, \underline{\mathbf{u}}^n - B^T \underline{p}^n\right), \end{aligned} \tag{6.37}$$

$$B\, \underline{\mathbf{u}}^{n+1} = 0. \tag{6.38}$$

Here $\underline{\mathbf{b}}_{\Gamma_h}$ denotes the vector representation of the surface force term, i. e., $\underline{\mathbf{b}}_{\Gamma_h}|_i = f_{\Gamma_h}(\mathbf{v}_i)$, $i = 1, \ldots, N_{\mathbf{V}_h}$, and $\hat{\underline{\mathbf{b}}} = \underline{\mathbf{b}} - \underline{\mathbf{b}}_{\Gamma_h}$ the remaining part of the right-hand side.

In [Hys06] the explicit and semi-implicit treatment of the surface tension term are compared to each other in numerical experiments considering an oscillating and a rising bubble example. It can be seen that the explicit treatment leads to numerical oscillations when the time step size exceeds a certain bound. On the other hand the semi-implicit treatment yields more stable results and allows for larger time step sizes.

Remark 6.11
Applying the improved Laplace-Beltrami discretization $\tilde{f}_{\Gamma_h}$ of the surface force term described in Section 5.3.4 requires a slight modification of (6.37). In this case $\underline{\mathbf{b}}_{\Gamma_h}$ and $C_n^{\Gamma_h}$ have to be replaced by their modified counterparts $\tilde{\underline{\mathbf{b}}}_{\Gamma_h}$ and $\tilde{C}_n^{\Gamma_h}$ defined by $\tilde{\underline{\mathbf{b}}}_{\Gamma_h}|_i = \tilde{f}_{\Gamma_h}(\mathbf{v}_i)$, $i = 1, \ldots, N_{\mathbf{V}_h}$, and

$$\langle \tilde{C}_n^{\Gamma_h}\, \underline{\mathbf{u}},\, \underline{\mathbf{v}} \rangle := \tilde{c}_{\Gamma_h}\big(J_{\mathbf{V}_h}(\underline{\mathbf{u}}),\, J_{\mathbf{V}_h}(\underline{\mathbf{v}})\big)_n$$

for all $\underline{\mathbf{u}}, \underline{\mathbf{v}} \in \mathbb{R}^{N_{\mathbf{V}_h}}$, where

$$\tilde{c}_{\Gamma_h}(\mathbf{u}_h, \mathbf{v}_h)_n := \tau \int_{\Gamma_h^n} \tilde{\mathbf{P}}_h(\mathbf{x}) \nabla \mathbf{u}_h \cdot \nabla_{\Gamma_h} \mathbf{v}_h \, ds, \qquad \text{for all } \mathbf{u}_h, \mathbf{v}_h \in \mathbf{V}_h. \qquad \diamond$$

6.2. Coupling of level set and Navier-Stokes equations

We consider the spatially discretized coupled system of level set and Navier-Stokes equations:

$$E(\underline{\mathbf{u}}(t))\,\underline{\varphi}'(t) + H(\underline{\mathbf{u}}(t))\,\underline{\varphi}(t) = 0, \tag{6.39}$$

$$M(\underline{\varphi}(t))\,\underline{\mathbf{u}}'(t) + T(\underline{\mathbf{u}}(t), \underline{\varphi}(t))\,\underline{\mathbf{u}}(t) + B^T\,\underline{p}(t) = \underline{\mathbf{b}}(\underline{\varphi}(t)), \tag{6.40}$$

$$B\,\underline{\mathbf{u}}(t) = \underline{c}. \tag{6.41}$$

Let the quantities $\underline{\mathbf{u}}^{\text{old}}$, $\underline{p}^{\text{old}}$, $\underline{\varphi}^{\text{old}}$ from the old time step t^{old} be given. We are looking for the quantities $\underline{\mathbf{u}} = \underline{\mathbf{u}}^{\text{new}}$, $\underline{p} = \underline{p}^{\text{new}}$, $\underline{\varphi} = \underline{\varphi}^{\text{new}}$ which approximate the solutions at t^{new}. The time step size is denoted by $k = t^{\text{new}} - t^{\text{old}}$.

The application of the *linearized* variant of the one-step theta-scheme to the coupled system (6.39)–(6.41) yields

$$\left[\frac{1}{k}E(\underline{\mathbf{u}}^{\text{old}}) + \theta H(\underline{\mathbf{u}}^{\text{old}})\right]\,\underline{\varphi} = \left[\frac{1}{k}E(\underline{\mathbf{u}}^{\text{old}}) - \theta' H(\underline{\mathbf{u}}^{\text{old}})\right]\,\underline{\varphi}^{\text{old}} \tag{6.42}$$

$$\left[\frac{1}{k}M(\varphi^{\text{old}}) + \theta\,T(\underline{\mathbf{u}}, \underline{\varphi}^{\text{old}})\right]\,\underline{\mathbf{u}} + \theta\,B^T\underline{p} = \left[\frac{1}{k}M(\varphi^{\text{old}}) - \theta'\,T(\underline{\mathbf{u}}^{\text{old}}, \underline{\varphi}^{\text{old}})\right]\,\underline{\mathbf{u}}^{\text{old}} \tag{6.43}$$

$$- \theta'\,B^T\underline{p}^{\text{old}} + \theta\,\underline{\mathbf{b}}(\underline{\varphi}) + \theta'\,\underline{\mathbf{b}}(\underline{\varphi}^{\text{old}})$$

$$B\underline{\mathbf{u}} = \underline{c}. \tag{6.44}$$

Here $\theta \in [0,1]$ and $\theta + \theta' = 1$. In a first step the level set equation (6.42) can be solved for $\underline{\varphi}$ and in a second step the Navier-Stokes equations (6.43)–(6.44) can be solved for $\underline{\mathbf{u}}$, $\underline{p}$. Thus a *decoupling* of level set and Navier-Stokes equations is achieved. This is a very nice property in terms of computational efficiency, but comes for the price of being only first order accurate in time. Hence, a linearized time discretization is the method of choice, when the spatial discretization error dominates the temporal discretization error.

In contrast to that, when applying a non-linearized time discretization scheme to (6.39)–(6.41) we may gain second order convergence, but will end up with a *fully coupled system*. The strategy used for the coupling of the level set and

Navier-Stokes equations will be explained exemplary for the one-step theta-scheme, cf. Section 6.1.2. It can be applied to other time discretization schemes in a similar manner.

Applying the one-step theta-scheme to the level set equation (6.39) yields

$$\begin{aligned} 0 &= \left[\frac{1}{k}E(\underline{\mathbf{u}}) + \theta\, H(\underline{\mathbf{u}})\right] \underline{\varphi} - E(\underline{\mathbf{u}}) \left[\frac{1}{k}\underline{\varphi}^{\text{old}} - \theta'\, E(\underline{\mathbf{u}}^{\text{old}})^{-1} H(\underline{\mathbf{u}}^{\text{old}})\, \underline{\varphi}^{\text{old}}\right] \\ &=: LS(\underline{\varphi}; \underline{\mathbf{u}}). \end{aligned} \tag{6.45}$$

For the Navier-Stokes equations (6.40)–(6.41) we obtain

$$\begin{aligned} 0 &= \left[\frac{1}{k}M(\underline{\varphi}) + \theta\, T(\tilde{\underline{\mathbf{u}}}, \underline{\varphi})\right] \mathbf{u} + \theta\, B^T \underline{p} - \theta\, \underline{\mathbf{b}}(\underline{\varphi}) \\ &\quad - M(\underline{\varphi}) \left[\frac{1}{k}\underline{\mathbf{u}}^{\text{old}} + \theta'\, M(\underline{\varphi}^{\text{old}})^{-1} \left[\underline{\mathbf{b}}(\underline{\varphi}^{\text{old}}) - T(\underline{\mathbf{u}}^{\text{old}}, \underline{\varphi}^{\text{old}})\, \underline{\mathbf{u}}^{\text{old}} - B^T \underline{p}^{\text{old}}\right]\right] \\ &=: NS_1(\underline{\mathbf{u}}, \underline{p}; \tilde{\underline{\mathbf{u}}}, \underline{\varphi}), \end{aligned} \tag{6.46}$$

$$\begin{aligned} 0 &= B\, \underline{\mathbf{u}} - \underline{c} \\ &=: NS_2(\underline{\mathbf{u}}). \end{aligned} \tag{6.47}$$

Here we introduced the quantity $\tilde{\underline{\mathbf{u}}}$ for a more flexible notation. In the convergence history of the coupling strategy this quantity will tend to $\underline{\mathbf{u}}$ in order to fulfill the Navier-Stokes equations.

The time discretization of the coupled system then reads as follows: Given $\underline{\mathbf{u}}^{\text{old}}, \underline{p}^{\text{old}}, \underline{\varphi}^{\text{old}}$ from the old time step t^{old}, find $\underline{\mathbf{u}}^{\text{new}}, \underline{p}^{\text{new}}, \underline{\varphi}^{\text{new}}$ for the new time step $t^{\text{new}} = t^{\text{old}} + k$ such that

$$LS(\underline{\varphi}^{\text{new}}; \underline{\mathbf{u}}^{\text{new}}) = 0, \tag{6.48}$$

$$NS_1(\underline{\mathbf{u}}^{\text{new}}, \underline{p}^{\text{new}}; \underline{\mathbf{u}}^{\text{new}}, \underline{\varphi}^{\text{new}}) = 0, \tag{6.49}$$

$$NS_2(\underline{\mathbf{u}}^{\text{new}}) = 0. \tag{6.50}$$

This coupled system is solved by a fixed point approach in the following way.

Algorithm 6.12 (Coupling)
Set $\underline{\mathbf{u}}^0 := \underline{\mathbf{u}}^{\text{old}}$, $\underline{\varphi}^0 := \underline{\varphi}^{\text{old}}$. For $m = 0, 1, 2, \ldots$ proceed

1. Solve the level set equation

$$LS(\underline{\varphi}^{m+1}; \underline{\mathbf{u}}^m) = 0$$

for $\underline{\varphi}^{m+1}$.

2. Solve the Navier-Stokes equations

$$NS_1(\underline{\mathbf{u}}^{m+1}, \underline{p}^{m+1}; \underline{\mathbf{u}}^{m+1}, \underline{\varphi}^{m+1}) = 0, \qquad NS_2(\underline{\mathbf{u}}^{m+1}) = 0$$

for $\underline{\mathbf{u}}^{m+1}, \underline{p}^{m+1}$.

3. Set $m \leftarrow m+1$ and return to step 1.

For the solution of the sub problems in steps 1 and 2 iterative solvers are used, where the quantities $\underline{\varphi}^m$ resp. $\underline{\mathbf{u}}^m, \underline{p}^m$ from the last fixed point step are used as initial values for the solvers. If both solvers perform zero iterations, i. e., the stopping criteria of both solvers are already satisfied for the initial values and hence $\underline{\varphi}^{m+1} = \underline{\varphi}^m$, $\underline{\mathbf{u}}^{m+1} = \underline{\mathbf{u}}^m$, $\underline{p}^{m+1} = \underline{p}^m$, then the fixed point loop is stopped. We then set $\underline{\varphi}^{\text{new}} := \underline{\varphi}^{m+1}$, $\underline{\mathbf{u}}^{\text{new}} := \underline{\mathbf{u}}^{m+1}$ and $\underline{p}^{\text{new}} := \underline{p}^{m+1}$.

Remark 6.13
Assume that the stopping criteria of the iterative solvers for the level set and Navier-Stokes equations are chosen such that

$$\|LS(\underline{\varphi}^{m+1})\| \leq \varepsilon_{LS}, \quad \|NS_1(\underline{\mathbf{u}}^{m+1}, \underline{p}^{m+1})\| \leq \varepsilon_{NS_1}, \quad \|NS_2(\underline{\mathbf{u}}^{m+1})\| \leq \varepsilon_{NS_2}.$$

Then after convergence of the coupling loop we have

$$\|LS(\underline{\varphi}^{\text{new}})\| \leq \varepsilon_{LS}, \quad \|NS_1(\underline{\mathbf{u}}^{\text{new}}, \underline{p}^{\text{new}})\| \leq \varepsilon_{NS_1}, \quad \|NS_2(\underline{\mathbf{u}}^{\text{new}})\| \leq \varepsilon_{NS_2}.$$

Hence, the quantities $(\underline{\varphi}^{\text{new}}, \underline{\mathbf{u}}^{\text{new}}, \underline{p}^{\text{new}})$ fulfill equations (6.48)–(6.50) up to the tolerance used by the iterative solvers.

However, the choice of suitable tolerances $\varepsilon_{LS}, \varepsilon_{NS_1}, \varepsilon_{NS_2}$ for the iterative solvers is a delicate task. If the tolerances are chosen too restrictive, the coupling loop may not converge. On the other hand, choosing the tolerances too loose will lead to an inaccurate solution $(\underline{\varphi}^{\text{new}}, \underline{\mathbf{u}}^{\text{new}}, \underline{p}^{\text{new}})$. Another possibility is to choose different tolerance parameters ε^m in each iteration of the coupling loop, for example $\varepsilon_{LS}^0 = \delta \|LS(\underline{\varphi}^m)\|$ and

$$\varepsilon_{LS}^m = \max(\delta\, \varepsilon_{LS}^{m-1}, \frac{\varepsilon_{LS}}{2}) \quad \text{for } m = 1, 2, \ldots,$$

with $0 < \delta < 1$, for instance $\delta = 0.1$. A systematic approach for taking appropriate tolerance parameters ε is not available, yet, and is left as a topic for future research. ◇

One can think of other variants for the fixed point strategy. For example, we can replace the solution of the Navier-Stokes system in step 2 by the Oseen problem

$$NS_1(\underline{\mathbf{u}}^{m+1}, \underline{p}^{m+1}; \underline{\mathbf{u}}^{m}, \underline{\varphi}^{m+1}) = 0, \qquad NS_2(\underline{\mathbf{u}}^{m+1}) = 0,$$

which is solved for $\underline{\mathbf{u}}^{m+1}, \underline{p}^{m+1}$. This will in general require more fixed point steps than the first variant, but less solution effort in each step.

Another possibility is to interchange the order of solution of the sub problems, i. e., first solve the Navier-Stokes problem and after that the level set problem:

Algorithm 6.14 (Coupling — reverse order)
Set $\underline{\mathbf{u}}^0 := \underline{\mathbf{u}}^{\text{old}}$, $\underline{\varphi}^0 := \underline{\varphi}^{\text{old}}$. For $m = 0, 1, 2, \ldots$ proceed

1. Solve the Navier-Stokes equations

   $$NS_1(\underline{\mathbf{u}}^{m+1}, \underline{p}^{m+1}; \underline{\mathbf{u}}^{m+1}, \underline{\varphi}^{m} = 0), \qquad NS_2(\underline{\mathbf{u}}^{m+1}) = 0$$

   for $\underline{\mathbf{u}}^{m+1}, \underline{p}^{m+1}$.

2. Solve the level set equation

   $$LS(\underline{\varphi}^{m+1}; \underline{\mathbf{u}}^{m+1}) = 0$$

   for $\underline{\varphi}^{m+1}$.

3. Set $m \leftarrow m + 1$ and return to step 1.

For our experience this solution order is more expensive in terms of computational time. As the solution of the Navier-Stokes problem is much harder than the solution of the level set problem, one should solve the level set equation in advance to get a better initial approximation of the interface Γ^{new} for the time-consuming solution of the Navier-Stokes problem.

7. Iterative solvers

In the following we describe how the discrete problems arising in each iteration of the coupling loop are solved, cf. Algorithm 6.12. For the level set equation we use a Krylov subspace method for non-symmetric systems (due to the convection term $\mathbf{u} \cdot \nabla \varphi$), e. g., GMRES or BiCGStab [Saa03], which are preconditioned by SSOR. The solution of the discrete Navier-Stokes system is more involved and will be explained further in the next sections. For ease of notation we write $\mathbf{u}$, p, φ instead of $\underline{\mathbf{u}}$, $\underline{p}$, $\underline{\varphi}$, dropping the '$\underline{\ }$', in the remainder of the chapter.

7.1. Navier-Stokes solvers

We consider the discrete Navier-Stokes problem

$$K(\mathbf{u}) \begin{pmatrix} \mathbf{u} \\ p \end{pmatrix} = \begin{pmatrix} \mathbf{b} \\ c \end{pmatrix}, \tag{7.1}$$

where

$$K(\mathbf{u}) := \begin{pmatrix} T(\mathbf{u}) & B^T \\ B & 0 \end{pmatrix}.$$

Note that this notation can be used both for stationary and non-stationary problems. In the case of a stationary problem we have $T(\mathbf{u}) = A + N(\mathbf{u})$. For the non-stationary case, after applying some time discretization scheme from Chapter 6, $T(\mathbf{u})$ is a linear combination of the mass matrix M and the convective-diffusive part $A + N(\mathbf{u})$ and $\mathbf{b}, c$ denote the corresponding right-hand sides.

The nonlinearity $T(\mathbf{u})$ is treated by a fixed point approach employing defect correction in each iteration cycle:

Algorithm 7.1 (Fixed point defect correction)
Let initial values $\mathbf{u}^0, p^0$ be given. For $m = 0, 1, 2, \ldots$ repeat

1. Compute the residual vector

$$\begin{pmatrix} \mathbf{r}^m \\ s^m \end{pmatrix} = K(\mathbf{u}^m) \begin{pmatrix} \mathbf{u}^m \\ p^m \end{pmatrix} - \begin{pmatrix} \mathbf{b} \\ c \end{pmatrix}.$$

2. Solve the linear Stokes or Oseen problem

$$K(\mathbf{u}^m) \begin{pmatrix} \Delta\mathbf{u}^m \\ \Delta p^m \end{pmatrix} = \begin{pmatrix} \mathbf{r}^m \\ s^m \end{pmatrix}. \tag{7.2}$$

   yielding the update $(\Delta\mathbf{u}^m, \Delta p^m)^T$.

3. Defect correction: Obtain the new iterates

$$\begin{pmatrix} \mathbf{u}^{m+1} \\ p^{m+1} \end{pmatrix} = \begin{pmatrix} \mathbf{u}^m \\ p^m \end{pmatrix} - \omega_m \begin{pmatrix} \Delta\mathbf{u}^m \\ \Delta p^m \end{pmatrix} \tag{7.3}$$

   with some step length ω_m.

Note that by far most of the computational work is done in the second step (7.2).

The step length ω_m in (7.3) can either be taken fixed as $\omega_m \equiv 1$ or $\omega_m \equiv 0.9$ (applying some damping) for all m, or may be adjusted by some step length control in each fixed point iteration. One strategy for step length control is discussed in the following.

The optimal step length ω_{opt} is given by the one dimensional optimization problem

$$\omega_{opt} = \arg\min_{\omega} \left\| K(\mathbf{u}^m - \omega\Delta\mathbf{u}^m) \begin{pmatrix} \mathbf{u}^m - \omega\Delta\mathbf{u}^m \\ p^m - \omega\Delta p^m \end{pmatrix} - \begin{pmatrix} \mathbf{b} \\ c \end{pmatrix} \right\|. \tag{7.4}$$

Here $\|\cdot\|$ denotes the Euclidean norm. This formula is also known as *line search* in the optimization community, cf. [NW99]. However, this approach is not feasible as the optimization problem in (7.4) is nonlinear in ω and would require, if solved iteratively, the repeated discretization of the nonlinear part N in each optimization step, which is expensive in terms of computational time. Therefore we modify the problem in (7.4) slightly to obtain a simpler one:

$$\tilde{\omega}_{opt} = \arg\min_{\omega} \left\| \tilde{K} \begin{pmatrix} \mathbf{u}^m - \omega\Delta\mathbf{u}^m \\ p^m - \omega\Delta p^m \end{pmatrix} - \begin{pmatrix} \mathbf{b} \\ c \end{pmatrix} \right\| \tag{7.5}$$

with $\tilde{K} := K(\mathbf{u}^m - \omega_{m-1}\Delta\mathbf{u}^m)$ using the step length ω_{m-1} from the last iteration (and $\omega_{-1} := 1$ at the beginning). Note that this optimization problem is linear in ω. Applying the necessary optimality condition, the solution of (7.5) is given by

$$\tilde{\omega}_{opt} = \frac{\left\langle \tilde{K}\begin{pmatrix}\Delta\mathbf{u}^m\\ \Delta p^m\end{pmatrix},\ \tilde{K}\begin{pmatrix}\mathbf{u}^m\\ p^m\end{pmatrix} - \begin{pmatrix}\mathbf{b}\\ c\end{pmatrix}\right\rangle}{\left\|\tilde{K}\begin{pmatrix}\Delta\mathbf{u}^m\\ \Delta p^m\end{pmatrix}\right\|^2}, \tag{7.6}$$

where $\langle\cdot,\cdot\rangle$ denotes the Euclidean scalar product. Note that the evaluation of (7.6) only requires the one-time construction of $\tilde{K} = K(\mathbf{u}^m - \omega_{m-1}\Delta\mathbf{u}^m)$ and the computation of some matrix-vector multiplications and scalar products. This is only little effort compared to the most time consuming part of Algorithm 7.1, the solution of the Stokes/Oseen problem (7.2) in the second step. Algorithm 7.1 with the choice $\omega_m = \tilde{\omega}_{opt}$ from (7.6) is called *adaptive fixed point defect correction method* in [Tur99].

We experienced that the step length control given by (7.6) is more robust than using a fixed step length ω_m. In most of the iterations, ω_m given by (7.6) is almost 1, indicating that the linear part of $K(\mathbf{u})$ is dominant in those cases.

7.2. Oseen solvers

In this section the iterative solution of the discrete Stokes or Oseen problem

$$\hat{K}\begin{pmatrix}\mathbf{u}\\ p\end{pmatrix} = \begin{pmatrix}\mathbf{b}\\ c\end{pmatrix} \tag{7.7}$$

is considered. Here

$$\hat{K} = \begin{pmatrix}\hat{T} & B^T\\ B & 0\end{pmatrix}$$

is a block matrix with saddle point structure for some (constant) regular matrix $\hat{T}$, hence (7.7) constitutes a linear problem. E.g., for $\hat{T} = A$ we have the stationary Stokes problem and for $\hat{T} = M + \gamma(A + N(\hat{\mathbf{u}}))$ an Oseen problem. The latter kind of problems arises, e. g., within the fixed point defect correction, cf. Equation (7.2) in Algorithm 7.1, where $\hat{K} = K(\hat{\mathbf{u}})$ for some fixed $\hat{\mathbf{u}}$.

For ease of notation we drop the '$\hat{}$' in the notation, i. e., we simply write K, T instead of $\hat{K}$, $\hat{T}$ in the remainder of this section.

In [PRR05] three iterative solvers for the solution of discrete *Stokes* problems are investigated, namely preconditioned CG, MINRES and an inexact Uzawa method. Two of them (CG and MINRES) exploit the symmetry of the Stokes equations. As we are interested in the application to Navier-Stokes problems which are non-symmetric due to the convection term $\mathbf{u} \cdot \nabla \mathbf{u}$, we concentrate on iterative solvers for non-symmetric Oseen problems.

Iterative Oseen solvers considered in this thesis for the solution of (7.7) can be divided into two classes.

- The first class, so called *Uzawa methods*, exploit the saddle point structure of K and are based on its Schur complement factorization, cf. Subsection 7.2.1.
- The second class consists of iterative solvers for general non-symmetric systems, e. g., GMRES or GCR, which are directly applied to the block matrix K without exploiting the saddle point structure of the problem. In the following we call them *general Krylov subspace methods*.

7.2.1. Uzawa type methods

Uzawa methods are related to the Schur complement factorization of K,

$$K = \begin{pmatrix} T & 0 \\ B & -I \end{pmatrix} \cdot \begin{pmatrix} I & T^{-1}B^T \\ 0 & S \end{pmatrix} \tag{7.8}$$

with the Schur complement matrix

$$S := B\,T^{-1}B^T.$$

Throughout this section we assume that T is symmetric positive definite.

Remark 7.2 (Uzawa methods for Oseen problems)
The construction of Uzawa methods and theoretical results in that context always assume that T is symmetric positive definite (i. e., the Stokes case), yielding a symmetric positive semi-definite Schur complement S. For small Reynolds numbers, however, the methods can also be applied to Oseen problems and turned out to be successful solution methods in practice, even though T is not s.p.d. anymore. ◇

The Schur complement factorization (7.8) can also be regarded as block LU factorization of K. The corresponding block forward-backward substitutions lead to the following algorithm:

Algorithm 7.3 (Schur complement method)

1. Solve $T\,\mathbf{v} = \mathbf{b}$.
2. Solve $S\,p = B\,\mathbf{v} - c$.
3. Solve $T\,\mathbf{u} = \mathbf{b} - B^T p$.

In the first and third step linear systems for the matrix T have to be solved. Depending on the properties of T, suitable (preconditioned) Krylov subspace methods or multigrid methods can be applied. The solution of the pressure system in the second step deserves further explanation. We use an iterative Krylov subspace method, where in each iteration matrix-vector multiplications $s = Sq$ have to be computed. Because the definition of S involves T^{-1}, the computation of s requires the solution of a linear system for T: solve $T\,\mathbf{r} = B^T q$, then $s = B\,\mathbf{r}$. The solution $\mathbf{r}$ has to be determined iteratively with high accuracy, otherwise the outer Krylov solver for the pressure system will diverge. Typically the solution of the inner T-system demands three orders of magnitude higher accuracy than of the outer pressure iteration. This high computational costs make the Schur complement method unattractive in practice.

We therefore use a variant of this approach, in which the linear systems for T (and S) have to be solved with less accuracy, which explains the name *inexact* Uzawa method. Here T^{-1} is replaced by the application of a symmetric positive definite preconditioner Q_T^{-1} for T, leading to the inexact Schur complement $\hat{S} := B\,Q_T^{-1} B^T$.

Instead of solving the pressure system $\hat{S}\,q = w$ with high accuracy, an approximate inverse of $\hat{S}$ is applied, namely $\tilde{q} = \Psi(w) \approx \hat{S}^{-1} w$ with the property

$$\|\Psi(w) - \tilde{q}\|_{\hat{S}} \leq \delta\,\|\tilde{q}\|_{\hat{S}} \quad \text{for all } w \in Q$$

for some $\delta < 1$. For the realization of Ψ we use a suitable Krylov subspace method with initial vector equal to zero, e. g., CG in case of Stokes or GMRES in case of Oseen problems. The Krylov subspace method is preconditioned with an preconditioner Q_S for the Schur complement. The design of preconditioners Q_T and Q_S is discussed in Section 7.2.3.

The inexact Uzawa method is based on iterative defect correction of the Oseen equation (7.7):

$$\begin{pmatrix} \mathbf{u}^{m+1} \\ p^{m+1} \end{pmatrix} = \begin{pmatrix} \mathbf{u}^{m} \\ p^{m} \end{pmatrix} + \begin{pmatrix} \mathbf{d}_1^{m} \\ d_2^{m} \end{pmatrix}$$

where the defect $(\mathbf{d}_1^m, d_2^m)^T$ is given by the solution of

$$K \begin{pmatrix} \mathbf{d}_1^m \\ d_2^m \end{pmatrix} = \begin{pmatrix} \mathbf{b} \\ c \end{pmatrix} - K \begin{pmatrix} \mathbf{u}^m \\ p^m \end{pmatrix} =: \begin{pmatrix} \mathbf{r}_1^m \\ r_2^m \end{pmatrix}. \tag{7.9}$$

Applying forward-backward substitution (i. e., Algorithm 7.3) to (7.9) the defect can be computed by first solving $\mathbf{v}^m = T^{-1}\mathbf{r}_1^m$ and then

$$\begin{aligned} d_2^m &= S^{-1}(B\,\mathbf{v}^m - r_2^m), \\ \mathbf{d}_1^m &= \mathbf{v}^m - T^{-1}B^T d_2^m. \end{aligned} \tag{7.10}$$

For the inexact Uzawa method we now replace T^{-1} by Q_T^{-1} and S^{-1} by the application of Ψ as explained above. Introducing $\mathbf{w}^m := \mathbf{u}^m + \mathbf{v}^m$ the following algorithm is obtained:

Algorithm 7.4 (Inexact Uzawa method)
Let $\mathbf{u}^0, p^0$ be given. Compute the residual $\mathbf{r}_1^0 = \mathbf{b} - T\mathbf{u}^0 - B^T p^0$.

For $m = 0, 1, 2, \ldots$ iterate:

1. Compute auxiliary vector
$$\mathbf{w}^m := \mathbf{u}^m + Q_T^{-1}\mathbf{r}_1^m,$$
2. Pressure defect:
$$d_2^m := \Psi(B\,\mathbf{w}^m - c),$$
3. Pressure update:
$$p^{m+1} := p^m + d_2^m,$$
4. Velocity update:
$$\mathbf{u}^{m+1} := \mathbf{w}^m - Q_T^{-1}B^T d_2^m,$$
5. Residual update:
$$\mathbf{r}_1^{m+1} := \mathbf{r}^m - T\,(\mathbf{u}^{m+1} - \mathbf{u}^m) - B^T d_2^m.$$

For the case of T being a symmetric positive definite matrix (i. e., we consider a generalized Stokes problem), in [PRR05] a more detailed inspection and a rigorous analysis of the inexact Uzawa method is given. It is shown, that the

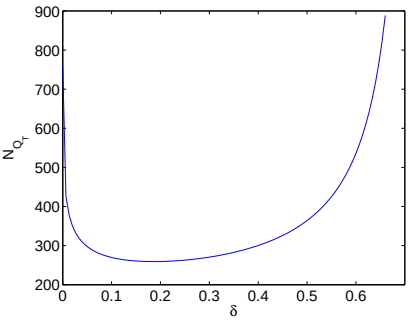

Figure 7.1.: Number of Q_T^{-1} evaluations for the inexact Uzawa algorithm as a function of δ.

error reduction of one iteration of Algorithm 7.4 in a suitable norm can be bounded by

$$2\mu_T + \delta\,(1 + \mu_T) := g(\mu_T, \delta),$$

where $\mu_T := \|I - Q_T^{-\frac{1}{2}}\, T\, Q_T^{-\frac{1}{2}}\|$ is the contraction number of the preconditioner Q_T. Hence, for an efficient solution the parameter δ (i. e., the accuracy of Ψ) should be chosen dependent on the quality of the preconditioner Q_T.

Remark 7.5 (Choice of δ)
We consider the Stokes case, i. e., T is symmetric positive definite. If one uses one multigrid V-cycle as preconditioner Q_T^{-1}, then the contraction number is typically about $\mu_T = 0.1$. For this case it suffices to choose $\delta < \frac{0.8}{1.1} = 0.\overline{72}$ to get a convergent method because then $g(0.1, \delta) < 1$. That means that the pressure system has to be solved only with low accuracy. But what is the optimal choice of δ in terms of computational effort? The arithmetic costs are dominated by the application of Q_T^{-1}, hence δ should be chosen such that the number N_{Q_T} of evaluations of Q_T^{-1} is small. The plot in Figure 7.1 shows a typical behavior of N_{Q_T} as a function of δ for the example $\mu_T = 0.1$. It turns out that low arithmetic costs can be achieved for a rather broad range of δ (roughly $\delta \in [0.1, 0.4]$) and that δ is not very sensitive to this quantity N_{Q_T} within this range. Choosing δ very small is inefficient, as this would require many matrix-multiplication with $\hat{S}$ involving the evaluation of Q_T^{-1}. The same holds for δ close to 0.7 where slow convergence is observed in practice, since the method diverges for $\delta \geq 0.\overline{72}$. ◇

7.2.2. General Krylov type methods

Another approach to solve the Oseen equation (7.7) is to apply a suitable iterative solver directly to the matrix K, disregarding the property that K is a block matrix with saddle point structure. If T is symmetric (i. e., the Stokes case), then K is also symmetric and thus the preconditioned MINRES algorithm is a suitable Krylov solver for this case. For the general Oseen case, T and K are non-symmetric, thus preconditioned GMRES, BiCGSTAB or GCR (cf. [Saa03]) are methods of choice. In all cases K is preconditioned by the diagonal block preconditioner

$$Q_K^{-1} = \begin{pmatrix} Q_T^{-1} & 0 \\ 0 & Q_S^{-1} \end{pmatrix}, \tag{7.11}$$

where Q_T^{-1}, Q_S^{-1} are preconditioners for the upper left block T and the Schur complement S, respectively. The design of Q_T^{-1} and Q_S^{-1} is discussed in Section 7.2.3. Note that the Schur complement matrix of $\tilde{K} := Q_K^{-1}K$ is given by

$$\tilde{S} = (Q_S^{-1}B)(Q_T^{-1}T)^{-1}(Q_T^{-1}B^T) = Q_S^{-1}S.$$

Most of the standard Krylov subspace methods assume that the preconditioner Q_K^{-1} is *linear* and *constant* in each step. But if one uses for Q_T^{-1}, e. g., the application of some GMRES iterations, both aforementioned assumptions on Q_K^{-1} are not fulfilled anymore. In this case one should use so called *flexible* Krylov methods, which do not have these restrictions concerning the preconditioner. Examples of such methods are GCR or flexible GMRES.

7.2.3. Preconditioning

In this section we describe the design of preconditioners for the upper left block T of K and the Schur complement S.

Preconditioning of T

Consider the linear system of equations $T\,\underline{\mathbf{x}} = \underline{\mathbf{b}}$ and denote by

$$\underline{\mathbf{y}} = Q_T^{-1}\underline{\mathbf{b}}$$

the application of the corresponding preconditioner. If T is the discretization of a diffusion-dominated differential operator, then performing one step of a

standard multigrid solver (V-cycle with Jacobi or Gauss-Seidel smoother) is an efficient preconditioner Q_T^{-1}. Here the hierarchical structure of the triangulations can be exploited, cf. Definition 3.5.

For small time steps, due to the time discretization the matrix T is dominated by $\frac{1}{k}M$. Since systems involving the mass matrix are relatively easy to solve due to its bounded condition number, applying a multigrid method is often not worth the effort. In this case, for the preconditioner Q_T^{-1} we usually apply one step of a damped Jacobi iteration

$$\underline{\mathbf{x}}^{(m+1)} = \underline{\mathbf{x}}^{(m)} + \omega[\mathrm{diag}(T)]^{-1}(\underline{\mathbf{b}} - T\,\underline{\mathbf{x}}^{(m)})$$

or one symmetric successive over-relaxation (SSOR) step. With the choice $\underline{\mathbf{x}}^{(0)} = 0$ as initial guess the damped Jacobi preconditioner simplifies to

$$\underline{\mathbf{y}}_i = \frac{\omega}{T_{ii}}\underline{\mathbf{b}}_i, \qquad i = 1, \ldots, N_{\mathbf{V}_h} \tag{7.12}$$

and the SSOR preconditioner is given by the following algorithm.

Algorithm 7.6 (SSOR preconditioner)

1. Compute auxiliary vector $\underline{\mathbf{w}}$ by damped Gauss-Seidel iteration for initial guess $\underline{\mathbf{x}}^{(0)} = 0$,

$$\underline{\mathbf{w}}_i = \frac{\omega}{T_{ii}}\left(b_i - \sum_{j=1}^{i-1} T_{ij}\,\underline{\mathbf{w}}_j\right), \qquad i = 1, \ldots, N_{\mathbf{V}_h}.$$

2. Compute $\underline{\mathbf{y}}$ by damped backwards Gauss-Seidel iteration

$$\underline{\mathbf{y}}_i = (2-\omega)\,\underline{\mathbf{w}}_i - \frac{\omega}{T_{ii}}\sum_{j=i+1}^{N_{\mathbf{V}_h}} T_{ij}\,\underline{\mathbf{y}}_j, \qquad i = N_{\mathbf{V}_h}, N_{\mathbf{V}_h}-1, \ldots, 1.$$

Preconditioning of the Schur complement S

In the following we restrict ourselves to the Stokes case, i. e., T is symmetric positive definite and hence the Schur complement S is symmetric positive semi-definite. For a discussion of preconditioners for the more general Oseen case we refer to Remark 7.8.

Since S is not explicitly available as $N_{Q_h} \times N_{Q_h}$ matrix, only by its (usually approximative) application to some vector, the design of preconditioners for

the Schur complement is different from the techniques for the design of Q_T^{-1} presented in the foregoing section. For example, we cannot apply SSOR to S, as this would require matrix entries of S which are not available.

Therefore we seek for matrices $G \in \mathbb{R}^{N_{Q_h} \times N_{Q_h}}$ which are spectrally equivalent to S, i. e.,

$$\gamma_S \langle G\, q,\ q\rangle \leq \langle S\, q,\ q\rangle \leq \Gamma_S \langle G\, q,\ q\rangle \qquad \text{for all } q \in \mathbb{R}^{N_{Q_h}} / \ker S \tag{7.13}$$

with constants $\Gamma_S, \gamma_S > 0$. These constants should be independent of the grid size h, time step size $k = \Delta t$ and the ratios $\frac{\mu_1}{\mu_2}$, $\frac{\rho_1}{\rho_2}$ of dynamic viscosity and density of the two phases, respectively.

We first consider the stationary Stokes case, i. e., $T = A$. Let $M_\mu \in \mathbb{R}^{N_{Q_h}}$ be defined by

$$\langle M_\mu\, p,\ q\rangle := (p,\ q)_\mu := (\mu^{-1}\, J_{Q_h}\, p,\ J_{Q_h} q)_0 \tag{7.14}$$

for $p, q \in \mathbb{R}^{N_{Q_h}}$ and $(\cdot,\cdot)_0$ the usual L_2 scalar product. In other words, M_μ is the pressure mass matrix with respect to the scaled L_2 scalar product $(\cdot,\cdot)_\mu$. In [OR06] it is shown that $G = M_\mu$ fulfills (7.13) with constants independent of h and μ.

For the non-stationary Stokes problem we assume $T = \xi M + A$ with $\xi > 0$, which is the outcome of some time discretization scheme as described in Chapter 6. We define the scaled pressure stiffness matrix $A_\rho \in \mathbb{R}^{N_{Q_h} \times N_{Q_h}}$ by

$$\langle A_\rho\, p,\ q\rangle := (\rho^{-1}\, \nabla J_{Q_h} p,\ \nabla J_{Q_h} q)_0. \tag{7.15}$$

Then we take

$$\tilde{G}^{-1} = M_\mu^{-1} + \xi A_\rho^{-1} \tag{7.16}$$

as Schur complement preconditioner for the non-stationary Stokes problem. In [OPR06] it is shown that this preconditioner $\tilde{G}$ fulfills the property

$$\langle S\, q,\ q\rangle \leq C \langle \tilde{G}\, q,\ q\rangle \qquad \text{for all } q \in \mathbb{R}^{N_{Q_h}} / \ker S$$

with a constant C independent of ξ, μ and ρ. The other bound $c\langle \tilde{G}\, q,\ q\rangle \leq \langle S\, q,\ q\rangle$ has not been proved yet. This is because of missing regularity results for the generalized Stokes interface problem.

However, numerical results obtained by the application of the Uzawa method (cf. Algorithm 7.3) in [OPR06] indicate, that the number of PCG iterations with the Schur complement S and preconditioner $\tilde{G}$ are almost constant with

respect to h, k, and moderate ratios $\frac{\mu_1}{\mu_2}$ and $\frac{\rho_1}{\rho_2}$. Thus the preconditioner $\tilde{G}$ turns out to be robust for the generalized Stokes interface problem within certain parameter ranges, even though a theoretical robustness result is missing.

Remark 7.7 (Preconditioning for extended pressure space)
If extended finite elements are used for the pressure space (cf. Section 5.4), some modifications have to be considered for the preconditioning of the Schur complement operator.

- Extended basis functions with very small support may occur which will blow up the condition number of the pressure mass matrix M_μ. We experienced that a simple diagonal scaling does a good job. The condition number of $D^{-1}M_\mu$ with $D = \mathrm{diag}(M_\mu)$ is bounded independently of h (cf. [Reu08]) and is rather low.
- The definition of A_ρ makes no sense for extended basis functions as functions with jumps are not weakly differentiable. Thus A_ρ is replaced by the operator
$$\tilde{A}_\rho := BQ_M^{-1}B^T,$$
where Q_M^{-1} is a preconditioner for M, for example the inverse of the diagonal of M. In practice we use $D^{-1}\tilde{A}_\rho$ with $D = \mathrm{diag}(M_\mu)$ to account for the different scaling of the extended basis functions. ◇

Remark 7.8 (Preconditioners for Oseen case)
Unfortunately, the preconditioners for the generalized Stokes interface problem presented in this section turned out to be unsatisfactory when applied to the Oseen problem in some cases, even for relatively small Reynolds numbers. For some Oseen test cases we experienced that it was even better to use no Schur complement preconditioning at all. Hence, an extension of the preconditioning techniques to the Oseen case, which compared to the Stokes case involves an additional convective term $\mathbf{w} \cdot \nabla \mathbf{u}$, is of great interest.

There are some ideas in the literature which are based on the following observation: If there was a matrix $T_p \in \mathbb{R}^{N_{Q_h} \times N_{Q_h}}$ with the commutation property $B^T T_p^{-1} = T^{-1} B^T$, then the Schur complement would be given by $S = BT^{-1}B^T = BB^T T_p^{-1}$ and thus

$$S^{-1} = T_p (BB^T)^{-1}.$$

In general it is not possible to find such a matrix T_p, but there are ways to construct matrices $\hat{T}_p \in \mathbb{R}^{N_{Q_h} \times N_{Q_h}}$ for which

$$Q_S^{-1} = \hat{T}_p (BB^T)^{-1} \tag{7.17}$$

turns out to be a reasonable approximation of the inverse Schur complement. In [KLW02] $\hat{T}_p$ is obtained by the discretization of a pressure convection-diffusion operator. This is motivated by the fact that the commutator

$$\left(\frac{1}{k}\rho I - \mu\Delta + \rho(\mathbf{w}\cdot\nabla)\right)_{\mathbf{V}} \nabla - \nabla \left(\frac{1}{k}\rho I - \mu\Delta + \rho(\mathbf{w}\cdot\nabla)\right)_{Q}$$

is zero inside Ω for $\mathbf{w}$ constant and expected to be small for smooth $\mathbf{w}$.

A more algebraic approach is taken in [EHS+06] where the discrete commutator

$$C := TB^T - B^T\hat{T}_p$$

is considered. Note that

$$B^T\hat{T}_p^{-1} - T^{-1}B^T = T^{-1}C\hat{T}_p^{-1} =: \tilde{C} \tag{7.18}$$

is the commutator used in the derivation of (7.17). The idea is to construct $\hat{T}_p$ in such a way that it fulfills the minimal commutator property

$$\|C\|_F \to \min. \tag{7.19}$$

Here $\|\cdot\|_F$ is the Frobenius norm. Note that due to (7.18) this is equivalent to the minimization problem $\|\tilde{C}\|_{\tilde{F}} \to \min$ in the norm $\|\cdot\|_{\tilde{F}}$ defined by $\|X\|_{\tilde{F}} := \|TX\hat{T}_p\|_F$ for all $X \in \mathbb{R}^{N_{\mathbf{V}_h}\times N_{Q_h}}$. An equivalent formulation of (7.19) are the normal equations

$$(BB^T)[\hat{T}_p]_j = B\,[TB^T]_j \qquad \text{for all } 1 \le j \le N_{Q_h}.$$

The solution is given by $\hat{T}_p = (BB^T)^{-1}BTB^T$ and due to (7.17) the Schur complement preconditioner has the form

$$Q_S^{-1} = (BB^T)^{-1}BTB^T(BB^T)^{-1}. \tag{7.20}$$

A comparison and critical review of the two preconditioners from [KLW02, EHS+06] can be found in [OV07] where both are applied to a few numerical 2D and 3D test cases. For the case of a circulating flow field $\mathbf{w}$ and a small kinematic viscosity coefficient $\nu = \frac{\mu}{\rho}$ none of the presented preconditioners provides satisfactory convergence results. The design of more appropriate preconditioners for the Oseen case is currently a field of active research. ◇

7.3. Some practical remarks

In our software toolbox DROPS we implemented a set of different iterative solvers to be able to compare them with regard to their efficiency for the solution of two-phase flow problems. For an overview of solvers and preconditioners available in DROPS we refer to Section 9.1.7. The ones that are mostly used are the following:

- For the solution of the level set equation we use a GMRES solver which is preconditioned by SSOR (Algorithm 7.6).
- For the linearization of the Navier-Stokes problem the fixed point defect correction (Algorithm 7.1) is used.
- For the solution of Stokes or Oseen problems we mostly use the inexact Uzawa method (Algorithm 7.4) or the GCR method. As Schur complement preconditioner we often use the method given in (7.16) (and the variant described in Remark 7.7 for the XFEM case, respectively) or the minimal commutator preconditioner, cf. Remark 7.8.
- For the solution of systems with the matrix T we usually use appropriate Krylov subspace methods preconditioned by SSOR or Jacobi. In the case that T is symmetric positive definite we use the CG method. For non-symmetric T we use the GMRES or BiCGSTAB method.

The solvers are nested on a hierarchy of levels, cf. Figure 7.2, for example the Navier-Stokes fixed point loop requires the Oseen solver which requires a Krylov subspace method for systems with the matrix T involving an SSOR preconditioner. In our implementation we used a template mechanism to enable the plug-in of different solvers in an easy way, cf. Section 9.1.7 for more details.

Each level of the solver hierarchy introduces new parameters which have to be set. This huge set of parameters $\delta_1, \ldots, \delta_m$ gives rise to the problem of how to choose them appropriately to get a convergent and efficient overall method. Up to now this choice depends more or less on the experience of the user. This undesirable procedure should be improved in future by studying dependencies between different parameters. This should lead to a strategy with a reduced number $m' < m$ of user-chosen parameters $\delta_1, \ldots, \delta_{m'}$, which then automatically induce the values of the remaining parameters.

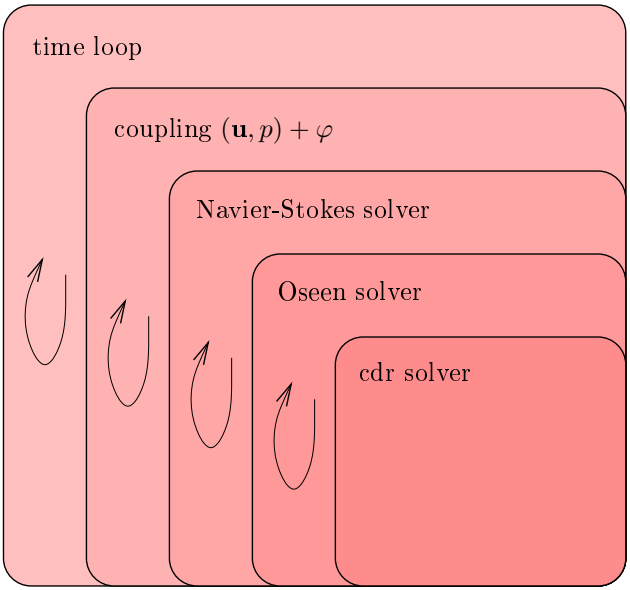

Figure 7.2.: The nestedness of time loop, coupling loop and iterative solvers shows the complexity of two-phase flow simulations.

8. Maintenance of the level set function

8.1. Reparametrization

During the evolution of the level set function φ, which is driven by the velocity field $\mathbf{u}$, the property of φ being close to a (signed) distance function is lost. This affects the refinement of the interfacial region and the treatment of the discontinuous material properties if represented by a smoothed jump (cf. Section 4.1.2). Moreover, the advection of φ becomes less accurate in regions where φ is very steep and the problem of finding the zero level set of φ becomes ill-conditioned in regions where φ is very flat. Therefore, a reparametrization technique is used to reestablish the signed distance function property. Important issues related to this reparametrization of φ are the following:

1. The zero level of φ should be preserved.
2. The norm of the gradient of φ should be close to one: $\|\nabla\varphi\| \approx 1$.
3. The reparametrization can be used to smooth φ (close to the interface) and thus stabilize the evolution of the level set function.

Different reparametrization techniques are known in the literature, cf. [Set96b, Set99, HT05]. The most often used method is based on a pseudo time stepping scheme for the Eikonal equation

$$\|\nabla\psi\| = 1.$$

Let φ_h be a given approximation of the level set function, and consider the following first order partial differential equation for $\psi = \psi(x,\tau)$:

$$\begin{aligned} \frac{\partial\psi}{\partial\tau} &= S_\alpha(\varphi_h)(1 - \|\nabla\psi\|), \qquad \tau \geq 0,\ x \in \Omega \\ \psi(x,0) &= \varphi_h, \end{aligned} \tag{8.1}$$

with

$$S_\alpha(\zeta) = \frac{\zeta}{\sqrt{\zeta^2 + \alpha^2}}, \qquad \zeta \in \mathbb{R},$$

where α is a regularization parameter ($0 < \alpha \ll 1$). The function S_α is a smoothed sign function. It keeps the zero level invariant (due to $S_\alpha(0) = 0$) and guarantees that the solution converges for $\tau \to \infty$ to a solution of the Eikonal equation. Thus, for sufficiently large $\tau_f > 0$ one can use the function $\psi(\cdot, \tau_f)$ as a reparametrization of φ_h.

The equation (8.1) can be reformulated in the more convenient form

$$\frac{\partial \psi}{\partial \tau} + \mathbf{w}(\psi) \cdot \nabla \psi = S_\alpha(\varphi_h) \quad \text{with } \mathbf{w}(\psi) := S_\alpha(\varphi_h) \frac{\nabla \psi}{\|\nabla \psi\|}. \tag{8.2}$$

The equation (8.2) can be solved numerically and then yields a reparametrization of φ_h. To stabilize the evolution a diffusion term can be added to the equation. For a further discussion of this reparametrization method we refer to the literature [SSO94, SF99, TE00]. We implemented such a method, but encountered the following two difficulties with this approach. Firstly, the algorithm is difficult to control because several parameters have to be chosen: the regularization parameter α, the diffusion parameter, the size of the considered time interval τ_f, the time step in the evolution. Secondly, and more important, in our simulations the zero level was changed too much. This is due to the fact that the invariance of the zero level only applies to the continuous case, but does not hold true for the discrete solution ψ_h.

We then considered alternative reparametrization methods. A simple variant of the Fast Marching method (cf. [KS98, Set96a]) turned out to perform much better in our numerical simulations. In [HT05] a survey and comparison of different reparametrization methods is given, where also the Fast Marching method is deemed the most accurate and efficient one. The algorithm is described in the following.

Let there be given a continuous piecewise quadratic function $\varphi_h \in V_h$ corresponding to the triangulation $\mathcal{T}_h$. We introduce some notation. The regular refinement of $\mathcal{T}_h$ is denoted by $\mathcal{T}_h' := \{ T' \in \mathcal{K}(T) : T \in \mathcal{T}_h \}$. The collection of all vertices in $\mathcal{T}_h'$ is denoted by $\mathcal{V}$. Note that φ_h is uniquely determined by its values on $\mathcal{V}$. For $T \in \mathcal{T}_h'$, $\mathcal{V}(T)$ is the set of the four vertices of T. Furthermore, for $v \in \mathcal{V}$, $\mathcal{T}(v)$ is the set of all tetrahedra which have v as a vertex:

$$\mathcal{T}(v) = \{ T \in \mathcal{T}_h' : v \in \mathcal{V}(T) \}.$$

Finally, for $v \in \mathcal{V}$, $\mathcal{N}(v)$ is the collection of all neighboring vertices of v (i. e., for each $w \in \mathcal{N}(v)$ there is an edge in $\mathcal{T}_h'$ connecting v and w):

$$\mathcal{N}(v) := \left(\bigcup_{T \in \mathcal{T}(v)} \mathcal{V}(T) \right) \setminus \{v\}.$$

We define

$$\mathcal{T}_\Gamma := \{\, T \in \mathcal{T}_h' : \operatorname{meas}_2(T \cap \Gamma) > 0 \,\}$$

the collection of tetrahedra which intersect the interface. Let Γ_h be the discrete approximation of the interface as defined in (5.9). Remind that the interface approximation Γ_h consists of planar segments Γ_T,

$$\begin{cases} \Gamma_T := \Gamma_h \cap T \quad \text{is a planar segment, for all } T \in \mathcal{T}_\Gamma, \\ \text{and } \Gamma_h = \bigcup_{T \in \mathcal{T}_\Gamma} \Gamma_T. \end{cases} \tag{8.3}$$

Remember that the planar segment Γ_T in (8.3) is either a triangle or a quadrilateral.

The algorithm splits up into two phases: The *initialization phase*, where only the values on vertices close to the interface are changed, and the *extension phase*, where the information is propagated from the interface to the vertices in the far field.

We first explain the initialization phase of the reparametrization algorithm. We define the set of vertices corresponding to $\mathcal{T}_\Gamma$:

$$\mathcal{V}_\Gamma := \{\, v \in \mathcal{V}(T) : T \in \mathcal{T}_\Gamma \,\}. \tag{8.4}$$

For each $v \in \mathcal{V}_\Gamma$ we define a discrete (approximate) distance function $d(v)$ as follows. For $v \in \mathcal{V}_\Gamma$ and $T \in \mathcal{T}(v) \cap \mathcal{T}_\Gamma$ let Γ_T be the plane segment as in (8.3), with vertices denoted by $Q_1, \ldots, Q_m$, where $m = 3$ or 4. Let W be the plane in $\mathbb{R}^3$ which contains the planar segment Γ_T and $P_W : \mathbb{R}^3 \to W$ the orthogonal projection on W. For $v \in \mathcal{V}_\Gamma$ and $T \in \mathcal{T}(v) \cap \mathcal{T}_\Gamma$ we define

$$d_T(v) := \begin{cases} \|v - P_W v\| & \text{if } P_W v \in T, \\ \min_{1 \le j \le m} \|v - Q_j\| & \text{otherwise,} \end{cases}$$

cf. Figure 8.1 as an illustration. The quantity $d_T(v)$ is a measure for the distance between v and Γ_T. Note that if $P_W v \in T$ holds, then $d_T(v)$ is

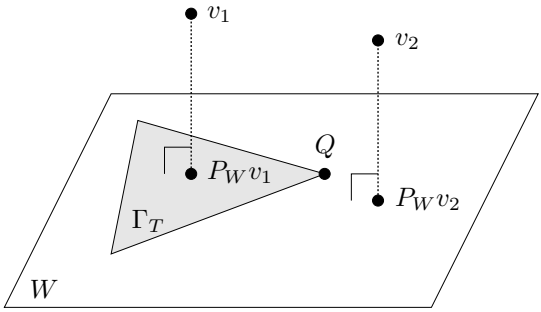

Figure 8.1.: Evaluation of d_T in the initialization phase for two vertices v_1, v_2 by orthogonal projection on W: here $d_T(v_1) = \|v_1 - P_W v_1\|$ and $d_T(v_2) = \|v_2 - Q\|$.

precisely this distance. Since Γ_h consists of piecewise planar segments Γ_T for $T \in \mathcal{T}_\Gamma$ we define $d(v)$ as an approximate distance between v and Γ_h by

$$d(v) := \min_{T \in \mathcal{T}(v) \cap \mathcal{T}_\Gamma} d_T(v) \quad \text{for } v \in \mathcal{V}_\Gamma. \tag{8.5}$$

After this initialization phase the grid function $\{ \big(v,\ d(v)\big) : v \in \mathcal{V}_\Gamma \}$ is an approximate distance function from the interface Γ_h for the vertices $v \in \mathcal{V}_\Gamma$.

The second phase of the reparametrization algorithm consists of a loop in which the approximate distance function d is extended to neighbor vertices of $\mathcal{V}_\Gamma$ and then to neighbors of neighbors, etc. To explain this more precisely we introduce two sets of vertices.

The first set $\hat{\mathcal{V}} \subset \mathcal{V}$ comprises the vertices where the values of the distance function $d : \mathcal{V} \to \mathbb{R}$ have already been computed. Right after the initialization phase we thus have $\hat{\mathcal{V}} = \mathcal{V}_\Gamma$. We call $\hat{\mathcal{V}}$ the *finalized set*.

The second one is the set of so-called *active* vertices $\mathcal{A} \subset \mathcal{V} \setminus \hat{\mathcal{V}}$, which consists of vertices $v \notin \hat{\mathcal{V}}$ that have a neighboring vertex in $\hat{\mathcal{V}}$:

$$\mathcal{A} := \{ v \in \mathcal{V} \setminus \hat{\mathcal{V}} : \mathcal{N}(v) \cap \hat{\mathcal{V}} \neq \emptyset \}. \tag{8.6}$$

$\mathcal{A}$ is called *active set*. So after the initialization phase, the initial active set $\mathcal{A}_0$ is given by

$$\mathcal{A}_0 := \{ v \in \mathcal{V} \setminus \mathcal{V}_\Gamma : \mathcal{N}(v) \cap \mathcal{V}_\Gamma \neq \emptyset \}. \tag{8.7}$$

For $v \in \mathcal{A}$ we define an approximate distance function in a similar way as in the initialization phase. Since its values may change if the finalized and active set are updated, we denote it by $\tilde{d} : \mathcal{A} \to \mathbb{R}$ to emphasize its tentative

character in contrast to d, which will be the final outcome of the algorithm. The construction of $\tilde{d}$ is described in the following.

Take $v \in \mathcal{A}$ and $T \in \mathcal{T}(v)$ with $\mathcal{V}(T) \cap \hat{\mathcal{V}} \neq \emptyset$. Note that such a T exists if $\mathcal{A}$ is nonempty. There are three possible cases, namely $|\mathcal{V}(T) \cap \hat{\mathcal{V}}| \in \{1, 2, 3\}$.

- If $|\mathcal{V}(T) \cap \hat{\mathcal{V}}| = 1$, say $\mathcal{V}(T) \cap \hat{\mathcal{V}} = \{w\}$, we define

$$d_T(v) := d(w) + \|v - w\|.$$

- For the other two cases, i. e., $\mathcal{V}(T) \cap \hat{\mathcal{V}} = \{w_i\}_{1 \leq i \leq m}$ with $m = 2$ or $m = 3$, we use an orthogonal projection as in the initialization phase. Let W be the line (plane) in $\mathbb{R}^3$ through the points w_1, w_2 $(, w_3)$ and $P_W : \mathbb{R}^3 \to W$ the orthogonal projection on W. We define

$$d_T(v) := \begin{cases} d(P_W v) + \|v - P_W v\| & \text{if } P_W v \in T, \\ \min_{1 \leq j \leq m} \left[d(w_j) + \|v - w_j\| \right] & \text{otherwise.} \end{cases} \tag{8.8}$$

  The value $d(P_W v)$ in (8.8) is determined by linear interpolation of the *known* values $d(w_j)$, $1 \leq j \leq m$. This is well-defined as $w_j \in \hat{\mathcal{V}}$ for $1 \leq j \leq m$ and d is already defined on $\hat{\mathcal{V}}$. Note that $P_W v \in T$ is satisfied if all faces of T are acute triangles.

The tentative approximate distance function $\tilde{d} : \mathcal{A} \to \mathbb{R}$ at active vertices $v \in \mathcal{A}$ is defined by

$$\tilde{d}(v) := \min\{\, d_T(v) : T \in \mathcal{T}(v) \text{ with } \mathcal{V}(T) \cap \hat{\mathcal{V}} \neq \emptyset \,\} \tag{8.9}$$

The complete reparametrization method is as follows:

Algorithm 8.1 (Fast Marching method)

1. Initialization: construct $\mathcal{V}_\Gamma$ and compute $d(\mathcal{V}_\Gamma)$ as in (8.4), (8.5).
2. Construct initial active set $\mathcal{A}_0$ and compute $\tilde{d}(\mathcal{A}_0)$ as in (8.7), (8.9).
3. Initialize finalized set $\hat{\mathcal{V}} := \mathcal{V}_\Gamma$ and active set $\mathcal{A} := \mathcal{A}_0$.
4. While $\mathcal{A} \neq \emptyset$, repeat the following steps:
   a) Determine $v_{\min} \in \mathcal{A}$ such that $\tilde{d}(v_{\min}) = \min_{v \in \mathcal{A}} \tilde{d}(v)$.
   b) Update finalized set $\hat{\mathcal{V}} := \hat{\mathcal{V}} \cup \{v_{\min}\}$ and define $d(v_{\min}) := \tilde{d}(v_{\min})$.
   c) Update active set $\mathcal{A} := (\mathcal{A} \cup \tilde{\mathcal{N}}) \setminus \{v_{\min}\}$ where $\tilde{\mathcal{N}} := \mathcal{N}(v_{\min}) \setminus \hat{\mathcal{V}}$.

d) (Re)compute $\tilde{d}(v)$ for $v \in \tilde{\mathcal{N}}$.

5. For all $v \in \mathcal{V}$, set $d(v) := \operatorname{sign}(\varphi_h(v)) \cdot d(v)$.

After this reparametrization we have $\hat{\mathcal{V}} = \mathcal{V}$ and a grid function $d(v)$, $v \in \mathcal{V}$, which uniquely determines a continuous piecewise quadratic function $\tilde{\varphi}_h \in V_h$ on the triangulation $\mathcal{T}_h$. This $\tilde{\varphi}_h$ is the reparametrization of φ_h. For $\tilde{\varphi}_h$ one can construct an approximate zero level set $\tilde{\Gamma}_h$ as described in Section 5.1. The reparametrization procedure guarantees $\tilde{\Gamma}_h \subset \bigcup_{T \in \mathcal{T}_\Gamma} T$. However, in general we have $\tilde{\Gamma}_h \neq \Gamma_h$, i. e., the discrete zero level set may be slightly changed. Since $\tilde{\varphi}_h$ is close to a signed distance function, the variations in $\nabla\tilde{\varphi}_h$ are usually smaller than the variations in $\nabla\varphi_h$. Due to this property, the reparametrization method has a stabilizing effect.

Remark 8.2 (Complexity)
The number of arithmetic operations for the initialization phase (steps 1–3 in Algorithm 8.1) is $\mathcal{O}(|\mathcal{V}_\Gamma| + |\mathcal{A}_0|)$. For the extension phase (steps 4–5 in Algorithm 8.1) the complexity is governed by step 4.a). The search for $v_{\min} \in \mathcal{A}$ has linear complexity in our implementation, but could be implemented more efficiently to gain $\mathcal{O}(\log|\mathcal{A}|)$ complexity which is the optimal one. As the steps in 4.a)–4.b) are repeated $N_\mathcal{V} := |\mathcal{V} \setminus \mathcal{V}_\Gamma|$ times, the overall complexity of the extension phase is $\mathcal{O}(N_\mathcal{V}^2)$ in our implementation and could be improved to be $\mathcal{O}(N_\mathcal{V} \log N_\mathcal{V})$. Although this bound $\mathcal{O}(N_\mathcal{V}^2)$ indicates suboptimal complexity (compared to $\mathcal{O}(N_\mathcal{V} \log N_\mathcal{V})$), in our simulations the time needed for the reparametrization is negligible compared to the computing times for discretization and iterative solution of the Navier-Stokes equations. ◇

8.2. Conservation of mass

The temporal and spatial discretization of the level set equation does not conserve mass. The same holds for the reparametrization of the level set function, cf. [Hup06] where this topic is investigated further. This loss of mass is reduced if the grid is refined. Such finer grids, however, result in higher computational costs. Therefore we introduce another strategy to compensate for the mass loss.

After each time step, we shift the interface in normal direction such that the volume of Ω_1 at current time is the same as at time $t = t_0$. To realize this we exploit the fact that the level set function is close to a signed distance function. In order to shift the interface over a distance δ in outward normal direction, we only have to subtract δ from the level set function.

Let $V(\varphi) := \text{meas}_3\{\, \mathbf{x} \in \Omega : \varphi(x) < 0 \,\}$ denote the volume of Ω_1 corresponding to a level set function φ and let φ_h be the discrete level set function at a given time. We have to find $d \in \mathbb{R}$ such that

$$V(\varphi_h - \delta) - \text{meas}_3(\Omega_1(0)) = 0$$

holds. In order to keep the number of evaluations of V low, we use a method with a high rate of convergence, namely the Anderson-Björck method [AB73], to solve this equation. We then set $\varphi_h^{\text{new}} := \varphi_h - \delta$ and discard φ_h.

Note that this strategy only works if Ω_1 consists of a single component. If there are multiple components, mass must be preserved for each of them. In this case the algorithm can be modified to shift φ_h only locally. Discontinuities that may occur in the level set function can be removed by a subsequent reparametrization step. In the case of topology changes more elaborate techniques have to be applied which are based on *local* mass conservation. An example is the paper [PSVW05], where the level set method is combined with VOF techniques to improve local mass conservation.

Finally note that the shifting of the level set function to obtain a better mass conservation introduces a new source of discretization errors.

9. Software package DROPS

The software package DROPS is developed within the framework of the Collaborative Research Center SFB 540 [SFB] where the goal of the involved mathematical projects (B4 and C7) is two-fold: on the one hand we want to develop and improve numerical methods for the simulation of two-phase flow problems and on the other hand the aim is to simulate realistic two-phase systems which are of interest for the project partners from the engineering department.

The development of DROPS is mainly conducted by the Chair of Numerical Mathematics, RWTH Aachen University, Germany. Due to the complexity of two-phase flow problems we need the ability to perform parallel computations. In a tight cooperation the parallelization of DROPS is realized at the Chair of Scientific Computing, RWTH Aachen University. The code is developed by a couple of people, where the current core development team consists of three persons, Jörg Grande, Oliver Fortmeier and the author of this thesis.

The DROPS code is written in C++. Especially the implementation of the iterative solvers heavily uses the object-oriented and template programming features of C++[1]. Some further information including a gallery of simulation examples can be found on the DROPS website [DRO].

Section 9.1 describes some fundamental concepts and the most important classes of DROPS. In Section 9.2 we give a brief introduction to the parallel version.

9.1. Fundamental concepts and data structures

In this section important data structures and algorithms implemented in DROPS are presented. Figure 9.1 gives an overview of the main components

[1] Thus our code is also used by some compiler manufacturers as a benchmark test for their C++ compilers (e. g., SUN, Microsoft).

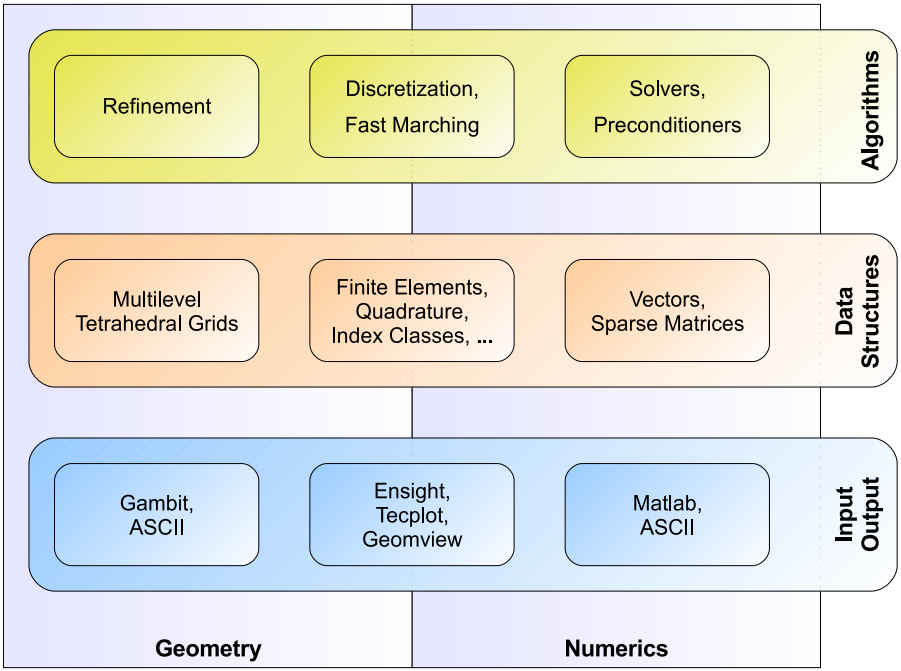

Figure 9.1.: Overview of modules and structure of DROPS.

of the software. The different modules are arranged in the diagram such that they obey two levels of structuring, namely in vertical and horizontal direction.

The vertical structure of the figure distinguishes between input and output routines, data structures and algorithms. While the different methods to bring input in and get output from the DROPS kernel are described in Section 9.1.8, we decided to present algorithms in conjunction with related data structures. This corresponds to the object-oriented perspective of C++ classes, where data structures (as data members) and functionality (as member functions) are combined with each other.

In a horizontal structure Figure 9.1 classifies the different modules into the categories 'geometry'and 'numerics', emphasizing the fact that we tried to decouple geometrical data such as the grid from numerical data such as vectors and matrices. Some tasks, however, require geometrical as well as numerical information and are therefore located in the middle column. One example are the discretization routines for setting up stiffness matrices, where in a loop over all tetrahedra the corresponding matrix entries are accumulated. The geometrical and numerical data structures are described in Sections 9.1.1 and 9.1.2, respectively.

9.1.1. Geometrical objects: multilevel triangulation and simplices

In this section we discuss the data structures that represent geometrical objects such as vertices, edges, faces, tetrahedra, the boundary and the multilevel grid. The corresponding data structures are called `VertexCL`, `EdgeCL`, `FaceCL`, `TetraCL`, `BoundaryCL` and `MultiGridCL`, respectively. Note that all C++ classes in DROPS have a suffix `CL` to distinguish data type identifiers from object identifiers.

Boundary and boundary segments

We assume that the boundary $\Sigma = \partial\Omega$ is partitioned into elementary boundary segments Σ_j, $j = 0, \ldots, N_\Sigma - 1$. Note that here we used a C style numbering starting with zero. To give an example, if Ω is a cube, then Σ can be partitioned into $N_\Sigma = 6$ boundary segments $\Sigma_0, \ldots, \Sigma_5$, cf. Figure 9.2. Each boundary segment is represented by a `BndSegCL` object. Up to now DROPS can only handle boundary segments which are piecewise planar. The class `BoundaryCL` contains an array of all `BndSegCL` objects.

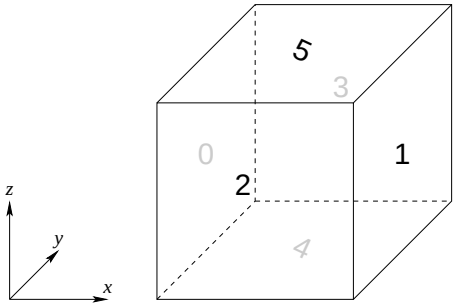

Figure 9.2.: A cube and its 6 boundary segments $\Sigma_0, \ldots, \Sigma_5$.

Simplices

In the following we describe the representation of the simplices.

VertexCL. Each vertex V stores its coordinates $\mathbf{x}_V \in \mathbb{R}^3$ as a `Point3DCL` object. If V is located on the boundary Σ, it stores a list of `BndPointCL` objects, each containing the index j of the boundary segment Σ_j with $\mathbf{x}_V \in \Sigma_j$ and the 2D coordinate in the local reference frame. Note that V may be located on multiple boundary segments. For the example in Figure 9.2 a vertex may be located on up to 3 boundary segments.

EdgeCL. Each edge E is linked to the two vertices V_1, V_2 which are connected by E. If the edge is further refined into two sub-edges E_1, E_2, then there is also a link to the midpoint vertex V_m. Note that $E_1 = V_1V_m$ and $E_2 = V_mV_2$. If E is located on the boundary, then it stores the indices j of the boundary segments with $\{\mathbf{x}_{V_1}, \mathbf{x}_{V_2}\} \subset \Sigma_j$. Note that an edge can be located on at most 2 boundary segments.

FaceCL. Each face F is linked to its neighboring tetrahedra. For a boundary face the index j of the corresponding unique boundary segment Σ_j is stored. Note that a face F may possess up to 4 neighboring tetrahedra. This is the case if F is an inner face connecting two tetrahedra T_1 and T_2 which are irregularly refined such that F is not subdivided by the corresponding green refinement rule. Then there are two green children $T_1' \in \mathcal{K}(T_1)$ and $T_2' \in \mathcal{K}(T_2)$ also sharing F as a common face.

TetraCL. Each tetrahedron T is linked to its 4 vertices, 6 edges and 4 faces. If $\ell(T) > 0$, i. e., T is not stored in the initial triangulation $\mathcal{T}_0$, then T is linked to its parent tetrahedron. If T is refined, then it is also

linked to its children $T' \in \mathcal{K}(T)$. T stores the integer values $\mathsf{mark}(T)$ (the refinement mark) and $\mathsf{status}(T)$ (the actual refinement rule), cf. Section 3.2.3.

Furthermore, each simplex class contains an `UnknownHandleCL` object which stores the indices of unknowns belonging to this simplex, cf. Section 9.1.3.

Multilevel triangulation

The class `MultiGridCL` represents a multilevel triangulation $\mathcal{M} = (\mathcal{T}_0, \ldots, \mathcal{T}_J)$, cf. Definition 3.5. The data structure is based on the corresponding hierarchical decomposition $\mathcal{H} = (\mathcal{G}_0, \ldots, \mathcal{G}_J)$, cf. Definition 3.7. That means that the tetrahedra are stored in $J+1$ lists, each one for the hierarchical surplus $\mathcal{G}_j$ of a different level. The vertices, edges and faces are stored in a similar manner, where the level of such a sub-simplex S is defined as

$$\ell(S) := \min\{\, \ell(T) : T \in \mathcal{H} \text{ contains } S \text{ as sub-simplex} \,\}.$$

Furthermore, `MultiGridCL` contains a `BoundaryCL` object storing all boundary segments.

The `MultiGridCL` constructor takes a `MGBuilderCL` object as input argument which creates the initial triangulation $\mathcal{T}_0$. `MGBuilderCL` serves as an abstract base class from which specific classes can be derived. For instance, the derived class `BrickBuilderCL` can be used to generate an initial triangulation of a cuboid-shaped domain.

The member function `Refine()` calls the refinement algorithm (cf. Algorithm 3.11) described in Section 3.2.3. It expects that the tetrahedra $T \in \mathcal{T}_J$ in the input multilevel triangulation are marked for refinement or for coarsening. This can be achieved by calling the member functions `SetRegRefMark()` or `SetRemoveMark()` of the corresponding `TetraCL` objects.

There are different kinds of iterators to access the simplices in the multilevel triangulation. The `MultiGridCL` member functions `GetTriangTetraBegin(L)` and `GetTriangTetraEnd(L)` return iterators to cycle through all tetrahedra

$$T \in \mathcal{T}_{\mathtt{L}}$$

of a certain triangulation. Similarly the member functions `GetAllTetraBegin(L)` and `GetAllTetraEnd(L)` can be used to iterate over all

$$T \in \bigcup_{j=0}^{\mathtt{L}} \mathcal{G}_j,$$

where the level $\ell(T)$ of the iterated tetrahedra T is increasing from 0 to `L`. Similar iterators exist for vertices, edges and faces as well.

The iterators are implemented such that a corresponding `for` loop can be executed by multiple OpenMP threads in parallel [Ope]. This allows for faster computations on shared memory machines. As the importance and availability of multi-core architectures is growing nowadays and will grow further in the future, this is a relevant advantage regarding computational efficiency.

9.1.2. Numerical objects: vectors and sparse matrices

Vectors

In DROPS there are two different type of vectors: `SVectorCL` for short vectors with a handful of entries and `VectorCL` for vectors with a large number of entries. Throughout this chapter we assume that indices always start with the number zero (C style numbering).

`SVectorCL<RowsN>` is a template class with template parameter `RowsN` for vectors $\mathbf{x} \in \mathbb{R}^{\texttt{RowsN}}$ with a fixed dimension `RowsN`. It is mostly used for storing coordinates. For this purpose we defined the typedefs `Point2DCL`, `Point3DCL` and `BaryCoordCL` which are identical to `SVectorCL<2>`, `SVectorCL<3>` and `SVectorCL<4>`, respectively.

The data type `VectorCL` is used for storing vectors $\underline{x} \in \mathbb{R}^N$ where N is large and may differ from object to object. The type is defined as a `typedef` for `VectorBaseCL<double>`. `VectorBaseCL<RealT>` is a template class for vectors with entry type `RealT` and is an ancestor of `std::valarray<RealT>`. Thus `VectorCL` derives the benefits of the efficient expression template mechanisms available for arithmetical operations involving `valarray` objects. By setting a debug flag `DebugNumericC` range checking and other debug features can be enabled which are switched off by default due to performance reasons.

Matrices

There are two different types of matrices in DROPS, `SMatrixCL` for small matrices and `MatrixCL` for large sparse matrices.

The template class `SMatrixCL<RowsN,ColsN>` is used for small matrices $M \in \mathbb{R}^{\texttt{RowsN}\times\texttt{ColsN}}$ with fixed dimensions.

Sparse matrices are stored in objects of the type `SparseMatBaseCL<RealT>` where `RealT` indicates the type of the entries. For convenience, we introduced a typedef `MatrixCL` for `SparseMatBaseCL<double>`.

We use the *compressed row storage* format (CSR) which is described in the following. For a sparse matrix with m rows and N non-zero entries, `SparseMatBaseCL` contains a vector `RowBegin` with $m + 1$ integer entries, a vector `ColIndex` with N integer entries and a vector `Val` with N entries of type `RealT`. For a row `i` the indices from `RowBegin[i-1]` to `RowBegin[i]-1` indicate the range in `Val` where the values of the non-zero entries are stored. The column indices of the corresponding values are stored in `ColIndex`.

As it is a tedious task to compute the sparsity pattern stored in `RowBegin` and `ColIndex`, we use an intermediate storage format called `SparseMatBuilderCL<RealT>` when setting up a new sparse matrix M. The `SparseMatBuilderCL` first collects and accumulates all entries in a `std::map` based data structure. After that a call of the member function `Build()` automatically creates the corresponding `SparseMatBaseCL` object M and deletes the `map`s afterwards.

As `map`s are often too memory consuming we use them only for initializing M. When updating M in subsequent steps the sparsity pattern is reused by default, i. e., access to `SparseMatBuilderCL` entries directly returns the corresponding `SparseMatBaseCL` entries in `Val`. If the sparsity pattern should not be reused (for example when the extended pressure space Q_h^Γ changed because the interface Γ has moved) all matrix entries should be deleted by a call to the member function `clear()` to force a complete initialization of the matrix.

9.1.3. The link between grid and unknowns: indices

As mentioned before we decided to decouple the geometrical data (grid) from the numerical data (matrices, vectors). This is advantageous, because then the iterative solvers only have to deal with matrices and vectors but not with the grid. As a matrix-vector multiplication does not require a loop over all grid entities this substantially saves computational time. But for the interpretation of a solution vector $\underline{\mathbf{u}}$ it is necessary to know which vector entries are associated with a certain vertex V, for example. Here the concept of indices comes into play.

Index descriptions and numberings

For each finite element type used in a solution strategy there exists an associated index. An index $\mathcal{J}$ is described by an `IdxDescCL` object. It contains the number of degrees of freedom (DoF) for each simplex type, n_V, n_E, n_F, n_T, and the overall number of unknowns, $N_{\mathcal{J}}$. To give an example, a P_1-index has $n_V = 1$ DoF per vertex (and $n_E = n_F = n_T = 0$), an index for vector-valued P_2-FE has $n_V = n_E = 3$ DoFs for each vertex and edge (and $n_F = n_T = 0$).

As a next step we have to create a *numbering* of all degrees of freedom which belong to the index $\mathcal{J}$, where degrees of freedom on Dirichlet boundaries are omitted. This is done by a function `CreateNumbering(...)`, which is usually a member function of the applied problem class (cf. Section 9.1.4). By this we also obtain the overall number of unknowns, $N_{\mathcal{J}}$, which is equal to the dimension of the vectors associated with $\mathcal{J}$. Thus at the end `CreateNumbering(...)` sets the value $N_{\mathcal{J}}$ in the corresponding `IdxDescCL` object.

The numbering is stored by `UnknownHandleCL` objects contained in the corresponding `VertexCL`, `EdgeCL`, `FaceCL` and `TetraCL` objects. Note that for a single simplex maybe multiple such numbers have to be stored, namely one for each index or, in other words, one for each finite element type.

For an extended finite element space a call to `UpdateXNumbering(...)` augments the usual numbering, also called *base numbering*, by a numbering for the extended degrees of freedom. These numbers are not stored in the `UnknownHandleCL` objects, but in a separate `ExtendedIdxCL` object. It contains a vector `xidx` $\in \mathbb{N}^{N_{\mathcal{J}}}$ where the entry `xidx[j]` either stores the number of the extended DoF belonging to the base DoF `j` or it contains a flag that the DoF `j` is not extended. Note that `UpdateXNumbering(...)` has to be called each time the interface has moved to account for the changed extended DoFs.

Vector and matrix descriptions

A `VecDescCL` object contains a vector `Data` of type `VectorCL` and a pointer `RowIdx` to the associated index of type `IdxDescCL`. Calling the member function `SetIdx(idx)` sets the pointer and resizes the vector to the right dimension. Similarly, a `MatDescCL` object contains a sparse matrix `Data` and pointers `RowIdx` and `ColIdx` to the associated row and column indices, respectively. A call of the member function `SetIdx(ridx,cidx)` sets the pointers and deletes all matrix entries. The right dimension of the matrix are set later by `SparseMatBuilderCL`, cf. Section 9.1.2.

9.1.4. Problem classes

There are several problem classes in DROPS representing different types of partial differential equations. So far we have problem classes for the Poisson, Stokes and Navier-Stokes problem (one-phase), the level-set equation and the two-phase Stokes and Navier-Stokes problem. For example the class for the two-phase Stokes problem is called `InstatStokes2PhaseP2P1CL`. All problem classes are derived from a common base class `ProblemCL` which contains three objects constituting a problem:

- the domain Ω, given by a multilevel triangulation (`MultiGridCL`),
- the boundary conditions and boundary values, given by a `BndDataT` object,
- the coefficients and right hand-side of the partial differential equation, given by a `CoeffT` object.

`BndDataT` and `CoeffT` are template parameters of the template class `ProblemCL` as their specific structure may vary among different problem types. Their meaning is discussed in the subsequent sections.

A specific problem class usually contains the index descriptions of the applied finite element types and several matrix and vector descriptions. Among the member functions there are `CreateNumbering(...)` procedures for the indices (cf. Section 9.1.3) and different `Setup...(...)` routines to compute the matrices and the right-hand side vectors constituting the finite element discretization.

Boundary data

The boundary data are described by a `BndDataCL<BndValT>` object. It contains an array of `BndSegDataCL<BndValT>` objects, one for each boundary segment Σ_j, cf. Section 9.1.1. Each `BndSegDataCL` object stores the boundary condition and a function pointer for evaluating the corresponding boundary values of type `BndValT`. The choice of the template parameter `BndValT` depends on whether the boundary condition applies to a scalar (`double`) or vector-valued (`Point3DCL`) quantity. The prescribed boundary condition of type `BndCondT` can be one of

- `DirBC`, `Dir0BC` for non-homogeneous and homogeneous Dirichlet boundary conditions, respectively,

- `NatBC`, `Nat0BC` for non-homogeneous and homogeneous natural boundary conditions, respectively,
- `Per1BC`, `Per2BC` for periodic boundary conditions denoting corresponding boundaries.

`WallBC` and `OutflowBC` are alias names for `Dir0BC` and `Nat0BC`, respectively.

Coefficients

As an example to describe the classes representing the coefficients of a specific partial differential equation we consider a scalar convection-diffusion problem for the unknown function $u = u(\mathbf{x}, t)$,

$$u_t + \mathbf{v}(\mathbf{x}, t) \cdot \nabla u - \operatorname{div}(a(\mathbf{x}, t)\nabla u) = f(\mathbf{x}, t) \qquad \text{in } \Omega \times [t_0, t_f].$$

This type of problem is represented by the problem class `InstatPoissonP1CL`. The corresponding `PoissonCoeffCL` contains the functions $\mathbf{v}(\mathbf{x}, t)$, $a(\mathbf{x}, t)$ and $f(\mathbf{x}, t)$ as static member functions which have to be implemented by the user.

For the two-phase flow problem (2.13) the corresponding coefficient class stores quantities such as densities ρ_i and dynamic viscosities μ_i of the phases Ω_i, $i = 1, 2$, the surface tension coefficient τ and the vector of gravitational acceleration $\mathbf{g}$.

9.1.5. Useful tools for discretization

In the discretization procedures `Setup...(...)` of the problem classes several sparse matrices representing the discrete differential operators and vectors for the right-hand side have to be constructed. This is done by iterating over all tetrahedra $T \in \mathcal{T}_h$, where for a single tetrahedron T contributions to the matrix and vector entries are computed. These contributions are integrals over T and the integrands are functions which can be defined locally on T, e. g., basis functions or gradients of basis functions.

Grid functions

For representing the integrands and computing the integrals over T we use `LocalP1CL` and `LocalP2CL` objects (for linear and quadratic functions, respectively) and quadrature rules `Quad2CL`, `Quad5CL` (exact for polynomials up

to degree 2 or 5, respectively). All these classes have a template parameter `ValT` for the function values and are derived from a common base class `GridFunctionCL<ValT,PointsN>`. This class stores `PointsN` values of type `ValT` which are associated to distinct nodes in a tetrahedron described by barycentric coordinates (`BaryCoordCL`, cf. Section 9.1.2). For a `LocalP1CL` object these nodes are the 4 vertices of the tetrahedron, for a `LocalP2CL` object the 6 midpoints of the edges are added, cf. Figure 4.1. For the `Quad...CL` objects the nodes are defined by the quadrature points of the corresponding quadrature rule

Arithmetic operations such as `+`, `-`, `*`, `/` for `GridFunctionCL` objects are defined pointwise. In the same way functions can be applied to `GridFunctionCL` objects using the member function `apply(...)`. Due to inheritance all this functionality is also provided for the derived `LocalP...CL` and `Quad...CL` classes. This is very useful when creating complex integrands like $(\mathbf{u} \cdot \nabla v_j)\, v_i$.

Several variants of `assign(...)` member functions enable the initialization of the `LocalP...CL` and `Quad...CL` objects. Additionally, `LocalP...CL` objects can be evaluated in an arbitrary point $\mathbf{x} \in T$ given by its barycentric coordinates. The `Quad...CL` objects have a member function `quad(...)` which applies the quadrature rule and returns the result of the numerical integration.

Local numberings

A `LocalNumbCL` object is initialized with an index description of index $\mathcal{J}$, the corresponding boundary data object and a tetrahedron T. It collects the numbering of the local degrees of freedom of T according to the index $\mathcal{J}$, cf. Section 9.1.3. If a degree of freedom is on a boundary it also provides the associated boundary condition and the number j of the corresponding boundary segment Σ_j. Up to now `LocalNumbCL` can only be used for P_2 finite elements.

Integration over interface patches or parts of a tetrahedron

An `InterfacePatchCL` object is initialized by a tetrahedron T and the level set function φ_h given by an P_2-FE `VecDescCL` object. It extracts the `LocalP2CL` object corresponding to φ_h, decides whether $\Gamma_h \cap T \neq$ and provides information about the sign ($\in \{+, -, 0\}$) of each degree of freedom.

The member function `ComputeForChild(i)` computes the planar interface patch $\Gamma_{T'} = \Gamma_h \cap T'$ for the `i`th regular child $T' \in \mathcal{K}(T)$, `i` $= 0, \ldots, 7$. $\Gamma_{T'}$

is represented by the coordinates of its vertices, which are given in terms of barycentric coordinates with respect to the parent T. Note that for the computation of the patches the regular refinement of T is not really constructed in the sense that geometrical data structures are changed.

After calling `ComputeCutForChild(i)` the member function `quad(...)` can be used to compute the integral over the cut part $T' \cap \Omega_1$ or $T' \cap \Omega_2$, where the integrand is an arbitrary quadratic function f given by a `LocalP2CL` object. The additional member function `quadBothParts(...)` provides the integrals over both cut parts $T' \cap \Omega_i$, $i = 1, 2$.

9.1.6. Time discretization and coupling

For the one-phase Stokes and Navier-Stokes problem the one-step theta-scheme (cf. Section 6.1) is represented by the classes `InstatStokesThetaSchemeCL` and `InstatNavStokesThetaSchemeCL`, respectively. Both classes have a template parameter `SolverT` for the type of the solver used in each time step. The computation of one time step is performed by the member function `DoStep(...)`.

For the two-phase Stokes and Navier-Stokes problem we have to consider a coupled system for velocity $\mathbf{u}$, pressure p and level set function φ, cf. Section 6.2. During the implementation it turned out that the coupling and time discretization should be combined in one class and cannot be decoupled in separate classes as they are closely connected to each other. However, the different coupling classes all have a similar structure, thus we decided to derive them from a base class `TimeDisc2PhaseCL` which stores common data members and defines a common abstract interface by means of virtual member functions such as `DoStep(...)`.

For the time discretization of the two-phase Navier-Stokes problem we implemented the following classes:

- `ThetaScheme2PhaseCL`: coupled one-step theta-scheme as given in Algorithm 6.12,
- `LinThetaScheme2PhaseCL`: linearized one-step theta-scheme, cf. (6.42)–(6.44),
- `FracStep2PhaseCL`: coupled fractional-step scheme, applies the `ThetaScheme2PhaseCL` for each macro time step, cf. Section 6.1.3,

- `OpSplitting2PhaseCL`: coupled fractional-step scheme with operator splitting, cf. Section 6.1.4.

All these coupling classes have a template parameter `SolverT` controlling the type of the iterative solver used in each time step. For the time discretization of two-phase Stokes problems we decided to apply the corresponding classes for the two-phase Navier-Stokes problems rather than create Stokes specific time discretization classes. This avoids code duplication and enhances code maintainability.

9.1.7. Iterative solvers and preconditioners

For the implementation of iterative solvers we tried to use a software design that accounts for the nested hierarchy of the solution methods, cf. Figure 7.2. For example, the Navier-Stokes fixed point loop requires an Oseen solver which applies a Krylov subspace method involving some preconditioner. Bearing this in mind we use a template mechanism to specify the inner solution components as template parameters. On the one hand this enables an easy plug-in of different solution components to test and compare reasonable combinations of solvers available from the DROPS solver toolbox. On the other hand this technique assures efficient code since the compiler can perform full code optimization for the template specialization which is known at the moment of compilation.

Example 9.1
As an illustrative example for the template plug-in mechanism we give a piece of code for the definition of a Stokes solver:

```
// preconditioner for upper left block preconditioner
typedef SSORPcCL ULPcPcT;
ULPcPcT ULPcPc(...);

// preconditioner for upper left block
typedef PCGSolverCL<ULPcPcT> ULSolverT;
ULSolverT ULsolver( ULPcPc, ...);
typedef SolverAsPreCL<ULSolverT> ULPcT;
ULPcT ULPc( ULsolver);

// Schur complement preconditioner
typedef ISPreCL SchurPcT;
SchurPcT SchurPc( ...);

// Stokes solver
```

```
typedef InexactUzawaCL<ULPcT, SchurPcT> StokesSolverT;
StokesSolverT StokesSolver( ULPc, SchurPc, ...);
```

Hence, the object `StokesSolver` represents an inexact Uzawa method. For Q_T we chose some iterations of an SSOR-preconditioned CG method (`ULPc`) applied to the upper left block of the saddle point matrix. The Schur complement preconditioner Q_S is given by `SchurPc`. ◇

We emphasize that there is a conceptual difference between solver objects and preconditioner objects. Solver classes are derived from a common base class `SolverBaseCL` storing the tolerance and the maximum number of iterations, i. e., the stopping criterion, as well as the norm of the residual and number of iterations used after the last execution of the solver. Each solver class comprises a member function `Solve(...)` calling the routine of the iterative solver for a given initial guess. In contrast, each preconditioner class contains the analogon `Apply(...)` calling the preconditioner for the initial guess 0.

In the following we list most of the solvers and preconditioners available from the DROPS solver toolbox.

Solvers

Navier-Stokes solvers, cf. Section 7.1.

- `FixedPtDefectCorrCL`: Algorithm 7.1 with step length $\omega_m = 1$,
- `AdaptFixedPtDefectCorrCL`: Algorithm 7.1 with step length ω_m as in (7.6).

Both are template classes where the template parameter `SolverT` determines the type of the Oseen solver. The latter is applied to solve the linearized problems inside the fixed point loop.

Oseen solvers, cf. Section 7.2.

- `SchurSolverCL`: Algorithm 7.3,
- `PSchurSolverCL`: Algorithm 7.3 with Schur complement preconditioning,
- `UzawaCL`: a variant of the Uzawa algorithm described in [BPV97],
- `InexactUzawaCL`: Algorithm 7.4,
- `PMResSPCL`: preconditioned MINRES solver for the Stokes problem.

Some of the classes provide template parameters `ULPcT`, `SchurPcT` to determine the type of the preconditioners Q_T, Q_S, for the upper left block T of K and the Schur complement S, respectively.

For the application of a general Krylov subspace method to the saddle point matrix K one can use the class `BlockMatrixSolverCL<SolverT>` where the template parameter `SolverT` specifies the type of the Krylov solver.

Krylov subspace methods

- `CGSolverCL`, `PCGSolverCL`: CG method and preconditioned variant,
- `MResSolverCL`, `PMResSolverCL`: MINRES method and preconditioned variant,
- `GMResSolverCL`, `GMResRSolverCL`: GMRES method and GMRES-Recursive method with left or right preconditioning,
- `BiCGStabSolverCL`: preconditioned BiCGSTAB method,
- `GCRSolverCL`: preconditioned GCR method.

The classes representing preconditioned Krylov subspace methods have a template parameter `PcT` designating the type of the preconditioner.

Multigrid method

The `MGSolverBaseCL` represents a multigrid solver (V-cycle) with a fixed number of smoothing steps. There are two template parameters `SmootherT` and `SolverT` which control the type of the smoother and the coarse grid solver, respectively. The multigrid method is special in the sense that it requires a hierarchy of linear systems

$$A_\ell \underline{x}_\ell = \underline{b}_\ell, \qquad \ell = 0, 1, \ldots, L$$

and prolongations

$$\mathcal{P}_\ell : \mathbf{V}_h^{\ell-1} \to \mathbf{V}_h^\ell, \qquad \ell = 1, \ldots, L,$$

to interpolate from level $\ell - 1$ to the finer level ℓ. Due to the nestedness of the multilevel triangulation $\mathcal{M}$ the hierarchy of finite element spaces is nested, i. e., $\mathbf{V}_h^{\ell-1} \subset \mathbf{V}_h^\ell$, hence the prolongations are defined in the canonical way. For each level the corresponding stiffness and prolongation matrices A_ℓ, P_ℓ and right-hand side vector b_ℓ are stored in a `MGLevelDataCL` object. The corresponding restriction matrices are given by $R_\ell := P_\ell^T$ and don't have to be stored separately. The hierarchy of matrices and vectors is represented by the data structure `MGDataCL` which is simply a list of `MGLevelDataCL` objects.

Preconditioners

The DROPS solver toolbox comprises the preconditioner classes given in the following lists. For a discussion of the preconditioners we refer to Section 7.2.3.

Matrix-based preconditioners

- `JACPcCL`: one step of the Jacobi preconditioner,
- `GSPcCL`, `SGSPcCL`: one step of the Gauss-Seidel or symmetric Gauss-Seidel preconditioner,
- `SSORPcCL`, `MultiSSORPcCL`: one or multiple steps of the SSOR preconditioner,
- `DummyPcCL`: no preconditioning

For most of these preconditioners there exists a variant which can be used as smoother for the multigrid solver.

The wrapper class `SolverAsPreCL` enables the use of a solver object as a preconditioner. That means that the `Apply(...)` member function of the wrapper class calls the `Solve(...)` member function of the solver class with initial guess zero. This mechanism is used in Example 9.1 in the definition of the preconditioner for the upper left block, `ULPc`, which wraps the solver object `ULsolver`.

Schur complement preconditioners Q_S

- `ISPreCL`: the Schur complement preconditioner (7.16) where M_μ^{-1} and A_ρ^{-1} are replaced by one step of the SSOR preconditioner applied to the corresponding pressure matrices,
- `ISNonlinearPreCL`: the same Schur complement preconditioner, but with M_μ^{-1} and A_ρ^{-1} replaced by some iterations of a Krylov subspace method which can be chosen by means of a template argument,
- `ISBBTPreCL`: the variant of the Schur complement preconditioner (7.16) described in Remark 7.7, usually applied in case of an extended pressure space,
- `MinCommPreCL`: the minimal commutator preconditioner (7.20) for Oseen problems described in Remark 7.8.

The `DiagBlockPreCL` is used in conjunction with solvers of type `BlockMatrixSolverCL`. It combines a preconditioner Q_T for the upper left block with a

preconditioner Q_S for the Schur complement yielding the diagonal block preconditioner Q_K defined in (7.11).

9.1.8. Input and output

In this section we describe input and output interfaces for different types of data.

Numerical data

Vectors and sparse matrices can be saved to and restored from files by using the input and output stream operators, `>>` and `<<`, implemented for `VectorCL` and `MatrixCL` objects. The matrix format can be read by MATLAB [Mat] which is very useful, e. g., for computing condition numbers or the spectrum of a matrix.

Geometrical data

The initial triangulation $\mathcal{T}_0$ can be read from a mesh file generated with the mesh generator GAMBIT [Gam]. To construct the corresponding multilevel triangulation a `ReadMeshBuilderCL` object containing the mesh file name is passed to the constructor of the `MultiGridCL` object. Here the concept of the `MGBuilderCL` class is applied, cf. Section 9.1.1, from which `ReadMeshBuilderCL` is derived. Other input file formats can be implemented by adding further ancestors of `MGBuilderCL`.

For the input and output of a *hierarchy* of triangulations $\mathcal{M} = (\mathcal{T}_0, \ldots, \mathcal{T}_J)$ we use a self-defined file format. For saving a `MultiGridCL` object representing a multilevel triangulation we use a software technique called serialization. For this reason the class representing this task is called `MGSerializationCL`. The deserialization is done by the class `FileBuilderCL`, which is an ancestor of `MGBuilderCL` and is passed to the constructor of `MultiGridCL`. It reads the files written out before by a `MGSerializationCL` object and recreates the corresponding `MultiGridCL` object.

In this way, a cancelled simulation run can be restarted from the last time step where a serialized multilevel triangulation was saved to the file system. In a first step the geometrical data is deserialized from the file system using the class `FileBuilderCL`. After that the vectors representing the numerical

solutions are restored by means of the class `ReadEnsightP2SolCL`, see the subsequent section.

Visualization

For 3D visualization purposes we mainly use the software package Ensight [Ens]. The class `EnsightP2SolOutCL` writes out the geometrical information (tetrahedra and coordinates of the vertices) and the numerical solutions ($\mathbf{u}_h$, p, φ evaluated in all P_2 degrees of freedom) using a specific Ensight file format. This format can also be read by other visualization packages such as ParaView [Para].

The class `ReadEnsightP2SolCL` restores the vectors $\underline{\mathbf{u}}$, $\underline{p}$ and $\underline{\varphi}$ from the files written out by the class `EnsightP2SolOutCL`. However, this only works properly if the multilevel triangulations at the time of storing and restoring are the same.

There are interfaces to some other visualization tools as well.

- `GeomMGOutCL`, `GeomSolOutCL` for visualization of geometry and numerical solution with Geomview [Geo],
- `TecPlotSolOutCL`, `TecPlot2DSolOutCL` for visualization of geometry and numerical solution (in 3D or on a 2D cut plane, respectively) with TecPlot [Tec],
- `MapleMGOutCL`, `MapleSolOutCL` for visualization of geometry and numerical solution with Maple [Map].

9.2. Parallelization

For the simulation of two-phase flow problems the computational complexity is very high and thus the use of parallel machines is of great importance. In this section we will only consider a parallelization for *distributed memory machines* by means of a *message passing interface* (MPI [Mes94, MPI]). *Shared memory parallelization* by means of OpenMP [Ope] has also been applied to some parts of DROPS, cf. [TSaM+05] for a description of the parallelized routines and some benchmark computations. Both parallelization concepts can be combined when using multi-core processors which are connected by a high-speed network. For the parallelization of DROPS we pursue such a

hybrid parallelization approach due to the growing importance of multi-core architectures.

In Section 9.2.1 we present a data distribution format for the geometrical data and, based on this, we also derive a distribution format for the numerical data. In Definition 9.2 below the geometrical data distribution format will be made mathematically precise by a formal specification of a so-called *admissible hierarchical decomposition*. This data distribution format is such that the following holds:

1. Let $T \in \mathcal{G}_k$ be an element from the hierarchical surplus on level k, cf. Definition 3.7. Then T is stored on one processor, say p, as a so-called master element. In certain cases (explained below) a ghost copy of T is stored on *one* other processor, say q.

2. The children of T (if they exist) are all stored as masters either on processor p or, if T has a ghost copy, on processor q. For $T \in \mathcal{G}_k$, $k > 0$, the parent of T or a ghost copy of it is stored on the same processor p where T is stored as master.

For the multilevel refinement algorithm a crucial point is that for a tetrahedron T one needs information about all children of T, cf. Section 3.2.3. Due to property 2 this information is available on the local processor (p or q) *without communication*. The first property shows that in a certain sense the overlap of tetrahedra is small.

In a parallel run of a simulation the computational load has to be distributed uniformly among the processors. So in practice an adaptive finite element solver has to be combined with dynamic load balancing and data migration between the processors. This is the topic of Section 9.2.2.

The main results concerning the admissible hierarchical decomposition, the parallel multilevel refinement method and the load balancing strategy can be summarized as follows:

- An admissible hierarchical decomposition has the desirable properties 1 (small storage overhead) and 2 (data locality) from above. This result is given in Section 9.2.1.

- The application of the parallel refinement algorithm to an admissible hierarchical decomposition is well-defined and results in an admissible hierarchical decomposition. This is proved in [GR05].

- Given an admissible hierarchical decomposition one can apply a suitable load balancing and data migration algorithm such that after data

migration one still has an admissible hierarchical decomposition. We comment on this in Section 9.2.2.

9.2.1. Data distribution

Distribution of geometrical data: admissable hierarchical decomposition

Let the sequence $\mathcal{M} = (\mathcal{T}_0, \ldots, \mathcal{T}_J)$ of triangulations be a multilevel triangulation and $\mathcal{H} = (\mathcal{G}_0, \ldots, \mathcal{G}_J)$ the corresponding hierarchical decomposition. In this section we introduce a particular format for the distribution of the tetrahedra in $\mathcal{H}$ among processors on a parallel machine. We assume that the processors are numbered by $1, \ldots, P$.

For the set of elements in the hierarchical surplus on level k that are stored on processor p we introduce the notation

$$\mathcal{G}_k(p) := \{\, T \in \mathcal{G}_k : T \text{ is stored on processor } p \,\}$$

and we define

$$\mathcal{H}(p) := (\mathcal{G}_0(p), \ldots, \mathcal{G}_J(p)).$$

Note that in general $\mathcal{H}(p)$ is not a hierarchical decomposition (in the sense of Definition 3.7). The sequence

$$\tilde{\mathcal{H}} = (\mathcal{H}(1), \ldots, \mathcal{H}(P)) \tag{9.1}$$

is called a *distributed hierarchical decomposition* (corresponding to $\mathcal{H}$).

In general the intersection $\mathcal{G}_k(p) \cap \mathcal{G}_k(q)$, $p \neq q$, may be nonempty. Note that such an *overlapping distribution of the elements is necessary*, due to the fact that parents and children are linked by pointers. Consider, for example, the situation depicted in Figure 9.3 where a parent T and its child $T' \in \mathcal{K}(T)$ are stored on different processors, say 1 and 2. Since pointers from one local memory to another are not allowed in a distributed memory setting, we have to use a copy to realize this pointer. One could store a copy of T on processor 2 to represent the link between T and T' as a pointer on processor q. If one does not allow such ghost copies, all ancestors and descendants of a tetrahedron must be on the same processor. This would cause very coarse data granularity, poor load balancing and hence low parallel efficiency.

For each level k and processor p we introduce a set of *master elements*, $\mathsf{Ma}_k(p) \subset \mathcal{G}_k(p)$, and a set of *ghost elements*, $\mathsf{Gh}_k(p) \subset \mathcal{G}_k(p)$. In the formulation of the conditions below we use the two conventions $\mathcal{K}(T) := \emptyset$ if $\mathsf{status}(T) = \mathsf{NoRef}$ and $\mathsf{Ma}_{J+1}(p) := \emptyset$.

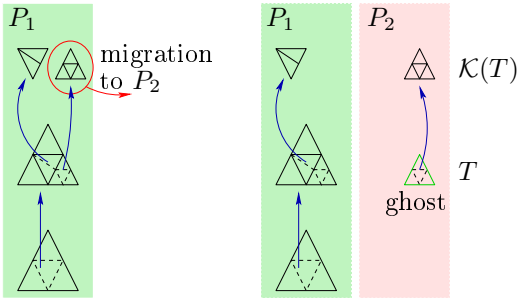

Figure 9.3.: Ghost elements are required to represent links between parents and their children as pointers across memory boundaries are not allowed for distributed memory machines. In the depicted example the parent T is stored on processor P_2 as a ghost to represent the link to its children $\mathcal{K}(T)$.

We now formalize the conditions on data distribution as follows.

Definition 9.2 (Admissible hierarchical decomposition)
The distributed hierarchical decomposition $\tilde{\mathcal{H}}$ is called an *admissible hierarchical decomposition* if for all $k = 1, \ldots, J$ the following conditions are fulfilled:

(A1) **Partitioning of $\mathcal{G}_k(p)$:** The sets of masters and ghosts form a disjoint partitioning of $\mathcal{G}_k(p)$:

$$\forall\, p \quad \mathsf{Ma}_k(p) \cup \mathsf{Gh}_k(p) = \mathcal{G}_k(p) \quad \text{and} \quad \mathsf{Ma}_k(p) \cap \mathsf{Gh}_k(p) = \emptyset$$

(A2) **Existence:** Every element from $\mathcal{G}_k$ is represented as a master element on level k:

$$\mathcal{G}_k = \bigcup_{p=1}^{P} \mathsf{Ma}_k(p)$$

(A3) **Uniqueness:** Every element from $\mathcal{G}_k$ is represented by at most one master element on level k:

$$\forall\, p_1, p_2 : \qquad \mathsf{Ma}_k(p_1) \cap \mathsf{Ma}_k(p_2) \neq \emptyset : p_1 = p_2$$

(A4) **Child–parent locality:** A child master element and its parent (as master or ghost) are stored on the same processor:

$$\forall\, p \quad \forall\, T \in \mathcal{G}_k \quad \forall\, T' \in \mathcal{K}(T) : \qquad T' \in \mathsf{Ma}_{k+1}(p) : T \in \mathcal{G}_k(p)$$

(A5) **Ghosts are parents:** Ghost elements always have children:

$$\forall p \quad \forall T \in \mathsf{Gh}_k(p): \qquad \mathcal{K}(T) \neq \emptyset$$

(A6) **Ghost–children locality:** A ghost element and its children are stored on the same processor:

$$\forall p \quad \forall T \in \mathsf{Gh}_k(p): \qquad \mathcal{K}(T) \subset \mathsf{Ma}_{k+1}(p) \qquad \diamond$$

Remark 9.3
Consider a consistent initial triangulation $\mathcal{T}_0 = \mathcal{G}_0$ with a non-overlapping distribution of the tetrahedra: $\mathcal{G}_0(p) \cap \mathcal{G}_0(q) = \emptyset$ for all $p \neq q$. In this case all tetrahedra can be stored as masters and there are no ghosts. Then the distributed hierarchical decomposition $\tilde{\mathcal{H}} = ((\mathcal{G}_0(1)), \ldots, (\mathcal{G}_0(P)))$ is obviously admissible. $\diamond$

Two elementary results are given in the following lemma.

Lemma 9.4
Let $\tilde{\mathcal{H}}$ as in (9.1) be a distributed hierarchical decomposition. The following holds:

1. If the conditions (A3), (A5) and (A6) are satisfied then for any element from $\mathcal{G}_k$ there is at most one corresponding ghost element:

$$\forall T \in \mathcal{G}_k \quad \forall p, q: \qquad T \in \mathsf{Gh}_k(p) \cap \mathsf{Gh}_k(q) : p = q$$

2. If the conditions (A1), (A2), (A3), (A4) and (A6) are satisfied then all children of a parent are stored as master elements on one processor:

$$\forall T \in \mathcal{G}_k \quad \exists p: \qquad \mathcal{K}(T) \subset \mathsf{Ma}_{k+1}(p)$$

Proof. Given in [GR05]. $\square$

In [GR05] a parallel version of Algorithm 3.11 is presented which is based on an admissible hierarchical decomposition and is suitable for distributed memory machines. In our implementation we use the DDD package [DDD] for the management of the distributed tetrahedra, faces, edges and vertices. For a given input-multilevel triangulation the parallel method *ParRefinement* produces the same output-multilevel triangulation as the serial method *SerRefinement*. In this sense the "computational part" of the algorithm is not changed. It is proved that the application of the parallel refinement algorithm to an admissible hierarchical decomposition is well-defined and results in an admissible hierarchical decomposition.

Remark 9.5
Let $T \in \mathsf{Ma}_k(p)$ be a parent master element. From the second result in Lemma 9.4 and (A4) it follows that either all children are masters on the same processor p as T, or they are masters on some other processor q. In the latter case, the element T has a corresponding ghost element on processor q. Due to this property, in the parallel refinement algorithm we use the strategy:

- If a parent tetrahedron T has a ghost copy then operations that involve children of T are performed on the processor on which the ghost and the children are stored.

From condition (A4) it follows that a child master element has its parent (as ghost or as master) on the same processor. Therefore we use the strategy:

- Operations that involve the parent of T are performed on the processor on which the master element of T and its parent are stored.

The first result in Lemma 9.4 shows that every $T \in \mathcal{H}$ has at most one ghost copy. Moreover, due to (A5) all leaves ($T \in \mathcal{T}_J$) have no ghost copies. In this sense the overlap of tetrahedra between the processors is small. ◇

The main differences of *ParRefinement* compared to the serial version *SerRefinement* (Algorithm 3.11) are the following:

- After the call of $\mathsf{DetermineMarks}(\mathcal{G}_k)$ in step (1) of phase I the edge refinement patterns have to be communicated to keep them consistent on all processors.
- If simplices are deleted, they have to be logged off from DDD.
- If new simplices are created, they have to be logged in to DDD. Additionally, simplices on processor boundaries have to be identified with each other.
- After phase II the maximum number of levels has to be determined and communicated among the processors.

Distribution of numerical data

Let $\underline{x} \in \mathbb{R}^N$ a vector and $A \in \mathbb{R}^{N \times N}$ a (sparse) matrix. The numbering $\mathcal{J} = \{1, \ldots, N\}$ is associated to certain degrees of freedom of the hierarchical decomposition $\mathcal{H}$. Based on the distributed hierarchical decomposition $\tilde{\mathcal{H}}$ we will define a corresponding distribution of the numerical data $\underline{x}$ and A. For this purpose we first introduce the notion of a domain decomposition.

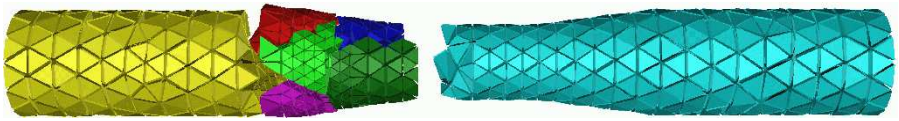

Figure 9.4.: Domain decomposition for $P = 8$ processors. Each color represents a different processor.

Definition 9.6 (Domain decomposition)
Let $\mathcal{H}$ be a hierarchical decomposition and $\tilde{\mathcal{H}}$ its admissible distribution among the processors. Due to the conditions (A2) and (A3) every tetrahedron $T \in \mathcal{H}$ can be assigned a unique processor on which T is stored as a master element. In other words, we have a well-defined function $\mathsf{master} : \mathcal{H} \to \{1, \ldots, P\}$ that is given by

$$\mathsf{master}(T) = p \quad \Leftrightarrow \quad T \in \mathsf{Ma}_{\ell(T)}(p).$$

Here $\ell(T)$ is the level of T, cf. Definition 3.7. For $0 \le j \le J$ and $1 \le p \le P$ we define

$$\mathcal{T}_j(p) := \{\, T \in \mathcal{T}_j : \mathsf{master}(T) = p \,\} \qquad \text{and} \qquad \Omega_j(p) := \bigcup_{T \in \mathcal{T}_j(p)} T.$$

Then for each $0 \le j \le J$ the sequence $(\mathcal{T}_j(1), \ldots, \mathcal{T}_j(P))$ is a partition of the triangulation $\mathcal{T}_j$ (due to (A2), (A3)) and is called the *domain decomposition of level* j corresponding to the admissible hierarchical decomposition $\tilde{\mathcal{H}}$. ◇

Figure 9.2.1 shows a domain decomposition for $P = 8$ processors.

A domain decomposition of level j automatically induces a distribution of the numerical data on level j. Without loss of generality we assume that the (global) numbering $\mathcal{J} = \{1, \ldots, N\}$ is associated with the finest level J. Let $\mathcal{J}(p) = \{1, \ldots, N_p\}$ be a (local) numbering of the degrees of freedom of the local triangulation $\mathcal{T}_J(p)$ on processor p, $1 \le p \le P$. Then the relation between a local number $i \in \mathcal{J}(p)$ and its global counterpart $j \in \mathcal{J}$ is given by the coincidence matrix $I_p \in \mathbb{R}^{N \times N_p}$,

$$(I_p)_{i,j} := \begin{cases} 1 & \text{if degree of freedom with global number } j \in \mathcal{J} \text{ exists} \\ & \text{on processor } p \text{ with local number } i \in \mathcal{J}(p), \\ 0 & \text{else.} \end{cases}$$

Degrees of freedom which are located on multiple processors form the so called *processor boundary*.

Definition 9.7 (Accumulated and distributed storage)
For a (global) vector $\underline{x} \in \mathbb{R}^N$ the sequence

$$\underline{x}_A = (I_1\underline{x}, \ldots, I_P\underline{x}) \in \mathbb{R}^{N_1} \times \ldots \times \mathbb{R}^{N_P}$$

is called the corresponding *accumulated vector*. That means that for unknowns on a processor boundary each adjacent processor stores the same global value.

The sequence $\underline{x}_D = (\underline{x}_1, \ldots, \underline{x}_P)$ of vectors $\underline{x}_p \in \mathbb{R}^{N_p}$ is called the *distributed vector* corresponding to $\underline{x}$, if

$$\underline{x} = \sum_{p=1}^{P} I_p^T \underline{x}_p.$$

In this case the global value of an unknown on a processor boundary is the sum of all local values stored on the adjacent processors. The same holds for entries of a *distributed matrix* $A_D = (A_1, \ldots, A_P)$ with

$$A = \sum_{p=1}^{P} I_p^T A_p I_p.$$

◇

Remark 9.8 (Computation of distributed stiffness matrix)
For a stiffness matrix $A \in \mathbb{R}^{N\times N}$ the local distributed matrix $A_p \in \mathbb{R}^{N_p \times N_p}$ coincides with the stiffness matrix corresponding to the subdomain $\Omega_J(p)$ with triangulation $\mathcal{T}_J(p)$. Thus the local matrices M_p can be set up independently by the different processors $p = 1, \ldots P$ without any communication. Furthermore, the parallelization of the `Setup` routines (cf. Section 9.1.4) is a trivial task. ◇

The conversion of a distributed into an accumulated vector is achieved by summing up the vector entries on processor boundaries which requires communication between adjacent processors. Obviously, the conversion in the other direction is not unique. For computing the matrix-vector multiplication $\underline{y} = A\,\underline{x}$ we use the accumulated storage $\underline{x}_A$ as input and obtain the result $\underline{y}_D$ in a distributed fashion:

$$A\,\underline{x} = \left(\sum_{p=1}^{P} I_p^T A_p I_p\right) \underline{x} = \sum_{p=1}^{P} I_p^T \underbrace{A_p(I_p\underline{x})}_{=:\underline{y}_p} = \underline{y}.$$

Hence, the computation of the matrix-vector multiplication does not require any communication. The scalar product of two vectors $\underline{x}, \underline{y}$ can be computed

efficiently if one of them is stored accumulated, for example $\underline{x}_A$, and the other one distributed, $\underline{y}_D$. Then the computation of

$$(\underline{x},\, \underline{y}) = \underline{x}^T \sum_{p=1}^{P} I_p^T \underline{y}_p = \sum_{p=1}^{P} (I_p \underline{x})^T \underline{y}_p = \sum_{p=1}^{P} (I_p \underline{x},\, \underline{y}_p)$$

only requires the global summation of P real numbers (obtained by a call to `MPI::AllReduce(...)`).

9.2.2. Distribution of work load

Considering the simulation of a rising bubble as an example, during an adaptive simulation run the multilevel triangulation $\mathcal{M}$ will change as the refinement zone is moving upwards following the bubble geometry. Hence, the distributed hierarchical decomposition $\tilde{\mathcal{H}}$ and the numerical data have to be redistributed from time to time to ensure a balance of the computational load. Otherwise the situation may occur that almost all unknowns are stored on one processor, say p, while the others only have to solve problems of small size. On the one hand this leads to an inefficient usage of the overall memory. On the other hand runtime scalability severely decreases since all processors have to wait at synchronization points such as `MPI::AllReduce(...)` until processor p has finished its work.

The challenge of the so-called *load balancing* is to find a mapping

$$m : \mathcal{T} \to \{1, \ldots, P\}$$

describing the distribution of the tetrahedra among the processors such that

a) the corresponding processor boundary is as small as possible *and*

b) all processors have almost the same number of tetrahedra.

This problem statement is equivalent to a *graph partitioning problem* which will be stated in Definition 9.10. For this reason, m is also called a *partitioning* of $\mathcal{T}$. We now introduce the notion of a weighted dual graph.

Definition 9.9 (Weighted dual graph)
For a triangulation $\mathcal{T}$ the corresponding *dual graph* $G(\mathcal{T}) = (V, E)$ is given by the node set $V = \mathcal{T}$ and the edge set $E \subset \mathcal{T} \times \mathcal{T}$, where $(T_1, T_2) \in E$ iff the tetrahedra T_1, T_2 share a common face.

By introducing *weight functions* $\alpha : V \to \mathbb{R}_+$ for nodes and $\beta : E \to \mathbb{R}_+$ for edges of the graph the computational load $\alpha(v_T)$ of the corresponding

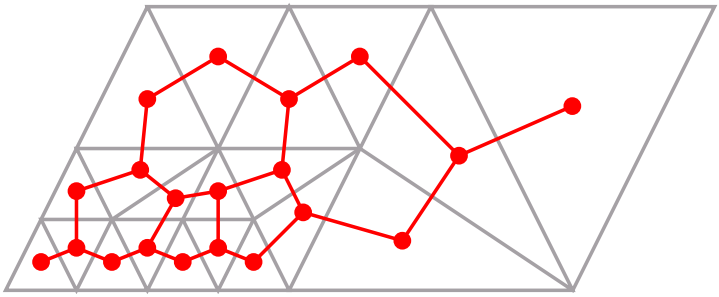

Figure 9.5.: Dual graph for 2D triangulation.

tetrahedron T and the amount of communication $\beta(e_F)$ for the corresponding face F can be described. $G_w(\mathcal{T}) = (V, E, \alpha, \beta)$ is called a *weighted dual graph*.◇

Figure 9.5 shows a 2D example for a dual graph. For a subset $\tilde{V} \subset V$ we define $\alpha(\tilde{V}) := \sum_{v \in \tilde{V}} \alpha(v)$ corresponding to the total load of $\tilde{V}$. For a given partitioning m the set

$$E_{\text{cut}}(m) := \{ (T_1, T_2) \in E : m(T_1) \neq m(T_2) \}$$

corresponds to the faces forming the processor boundary where communication takes place.

The graph partitioning problem is given by the following definition:

Definition 9.10 (Generalized graph partitioning problem)
For a constant $C > 1$ and a given weighted dual graph (V, E, α, β) find a partitioning $m : V \to \{1, \ldots, P\}$ such that

$$\text{cost}_{\text{comm}}(m) := \sum_{e \in E_{\text{cut}}(m)} \beta(e) \quad \to \quad \min$$

and

$$\alpha(V_p) \leq C \frac{\alpha(V)}{P}$$

with $V_p := m^{-1}(p)$. ◇

The graph partitioning problem belongs to the class of NP-hard problems, in this sense an *optimal* partitioning cannot be computed efficiently. Nevertheless, there are a couple of heuristic approaches with polynomial runtime yielding reasonable results. For a survey on this topic we refer to [Cha98]. We

use the package ParMETIS [Parb] which realizes a parallel multilevel graph partitioning algorithm described in [KK98].

Based on a partitioning m computed by a graph partitioning tool the tetrahedra and numerical data are rearranged among the processors. This phase is called *data migration*. To obtain again an admissible hierarchical decomposition after the migration phase we have to ensure that the properties (A1)–(A5) hold. In particular all children of a common parent have to stay together as masters on a single processor, cf. Lemma 9.4. Thus in the following we give a definition for a reduced dual graph, where the children of a common parent are represented by a single multi-node. For this purpose we introduce a map

$$P : \bigcup_{k=0}^{J} \mathcal{G}_k \to \bigcup_{k=0}^{J-1} \mathcal{G}_k$$

from a tetrahedron $T \in \mathcal{G}_k$ to its parent tetrahedron $P(T) \in \mathcal{G}_{k-1}$, $k = 1, \ldots, J$, with the convention $P(T) = T$ for all $T \in \mathcal{G}_0$. For $T \in \mathcal{T}$ we define the corresponding equivalence class

$$[T]_P := \{\, S \in \mathcal{T} : \; P(S) = P(T) \,\}.$$

Definition 9.11 (Reduced dual graph)
For a triangulation $\mathcal{T}$ let $G_w(\mathcal{T}) = (V, E, \alpha, \beta)$ be the corresponding weighted dual graph. The *reduced dual graph* $G'_w(\mathcal{T}) = (V', E', \alpha', \beta')$ is given by the reduced node set

$$V' := \{\, [T]_P : \, T \in \mathcal{T} \,\}$$

inducing the reduced edge set

$$E' := \{\, (v'_1, v'_2) : \quad \exists\, v_1 \in v'_1, v_2 \in v'_2 : \; (v_1, v_2) \in E \,\} \setminus \{\, (v', v') : \, v' \in V' \,\}.$$

The weight functions α', β' are given by

$$\alpha'(v') := \sum_{v \in v'} \alpha(v),$$

$$\beta'((v'_1, v'_2)) := \sum_{e \in E \cap (v'_1 \times v'_2)} \beta(e). \qquad \diamond$$

Figure 9.6 shows the reduced dual graph corresponding to the dual graph given in Figure 9.5. The tetrahedra forming a multi-node are surrounded by a bold frame. Note that the dual graph $G(\mathcal{T})$ in Figure 9.5 has 20 nodes and 24 edges whereas the reduced dual graph $G'(\mathcal{T})$ in Figure 9.6 has only 8 nodes and 9 edges.

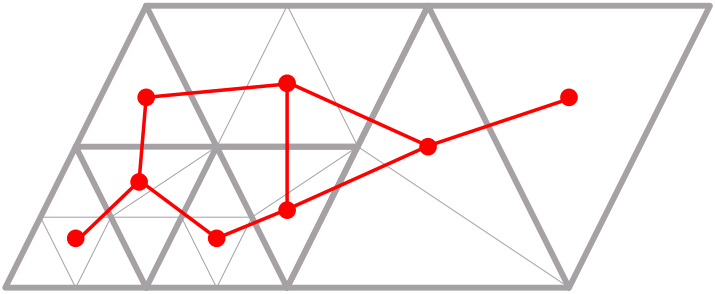

Figure 9.6.: Reduced dual graph for 2D triangulation.

After computing a load balancing partitioning $m' : V' \to \{1, \ldots, P\}$ of the reduced dual graph $G'_w(\mathcal{T})$, for the data migration we use an migration algorithm described in [Gro02]. The migration of the tetrahedra is carried out by means of the DDD package. After the migration for the new distributed hierarchical decomposition $\tilde{\mathcal{H}}$ the property

$$\mathsf{master}(T) = m'([T]_P)$$

holds. In [Gro02] it is shown that for an admissible input hierarchical decomposition the distributed hierarchical decomposition after the migration is again admissible.

Remark 9.12 (Migration of numerical data)
If a tetrahedron T is moved from one processor to another, also certain vector entries corresponding to the degrees of freedom on T have to be migrated. The valid migration of numerical data is a delicate task and will not be discussed in this thesis. ◇

9.2.3. Current status and outlook

The parallel refinement algorithm and load balancing strategy described in [Gro02] have been implemented in 2002 and were successfully applied on a parallel machine with up to 64 processors. This implementation has served as a starting point for a further parallelization of DROPS which began in 2005 and is currently conducted by our partners at the Chair of Scientific Computing, RWTH Aachen University. Since then, more and more parallel functionality has been added. At the current stage we are able to perform parallel simulations of two-phase flow problems on adaptive grids which are changing in time.

The next steps will be the improvement of the efficiency of the iterative solvers and the design of efficient parallel preconditioners. We also need to implement a parallel version of the fast marching algorithm, cf. Section 8.1, which is still missing. The parallelization of the multigrid solver will require a redesign of the load balancing strategy, since up to now we only consider the migration of the triangulation on the finest level J, but not of all triangulation levels. This will also have an impact on the definition of the weight functions α' and β'.

Part III.

Numerical results

10. Test cases

In this chapter we present several test cases. Some of them are designed to verify the functionality of several numerical components such as the interface capturing by the level set method (Section 10.1) and the reparametrization of the level set function by the Fast Marching method (Section 10.2). Other test cases are used to numerically measure the order of convergence for different discretizations of the surface tension force (Section 10.3) and different finite element spaces for the pressure (Section 10.4).

10.1. Advection of the interface

Consider the unit cube $\Omega = [0,1]^3$ and a ball

$$\Omega_1 = \{\, \mathbf{x} \in \mathbb{R} : \, \|\mathbf{x} - \mathbf{x}_M\| < 0.2 \,\}$$

inside with center $\mathbf{x}_M = (0.5,\ 0.25,\ 0.5)$. Take the fixed velocity field

$$\hat{\mathbf{u}}(\mathbf{x}) = c(\mathbf{y}) \cdot (\mathbf{y}_2,\ \ -\mathbf{y}_1,\ 0)$$

where $\mathbf{y} = \mathbf{x} - (0.5,\ 0.5,\ 0.5)$ and $c(\mathbf{y}) = 4\|\mathbf{y}\|(0.5 - \|\mathbf{y}\|)$. Hence, $\hat{\mathbf{u}}$ is a circular velocity field which vanishes at the boundary $\partial\Omega$, cf. Figure 10.1 for a plot of $\hat{\mathbf{u}}$. We consider the time interval $[t_0, t_f] = [0, 20]$ and define the velocity field

$$\mathbf{u}(\mathbf{x}, t) = \begin{cases} \hat{\mathbf{u}}(\mathbf{x}) & t \le 10, \\ -\hat{\mathbf{u}}(\mathbf{x}) & t > 10. \end{cases}$$

I. e., $\mathbf{u}$ changes its sign at the time moment $t = 10$. Note that for an interface $\Gamma \subset \Omega$ moving with velocity $\mathbf{u}(\mathbf{x}, t)$ we have

$$\Gamma(t_0 + t) = \Gamma(t_f - t) \qquad \text{for } t \in [t_0, \frac{t_0 + t_f}{2}] = [0, 10]. \tag{10.1}$$

For the initial value φ_0 of the level set function we use the signed distance function for the sphere $\Gamma = \partial\Omega_1$. As a test case for the advection of the

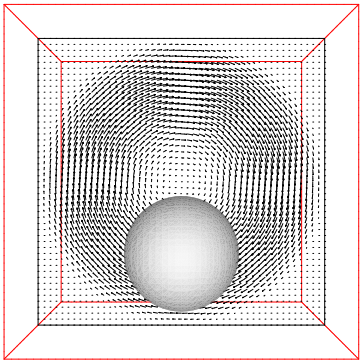

Figure 10.1.: Interface for $t = 0$. Also shown is the velocity field $\hat{\mathbf{u}}$ on the slice $z = 0$.

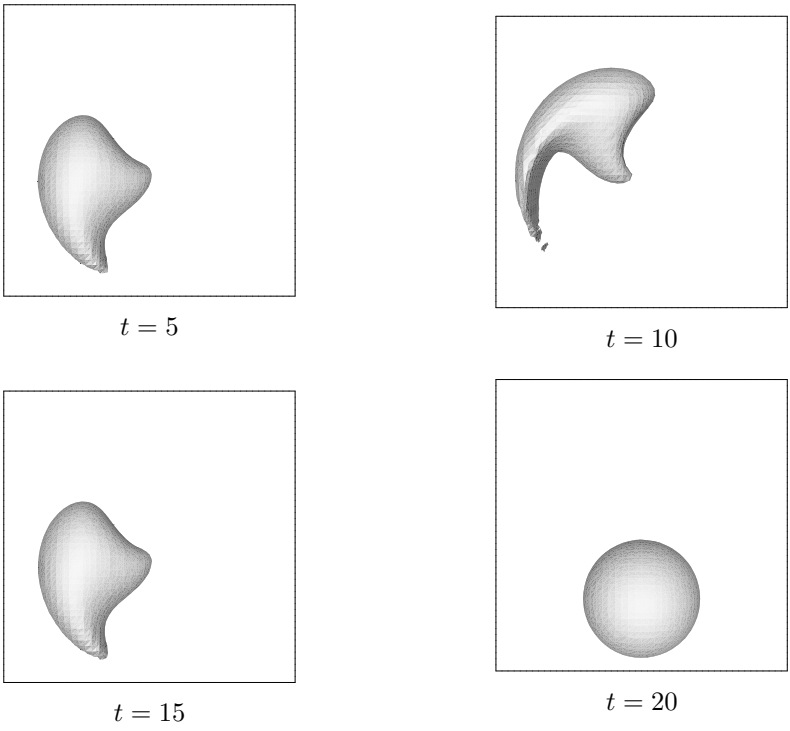

Figure 10.2.: Interface for different time steps.

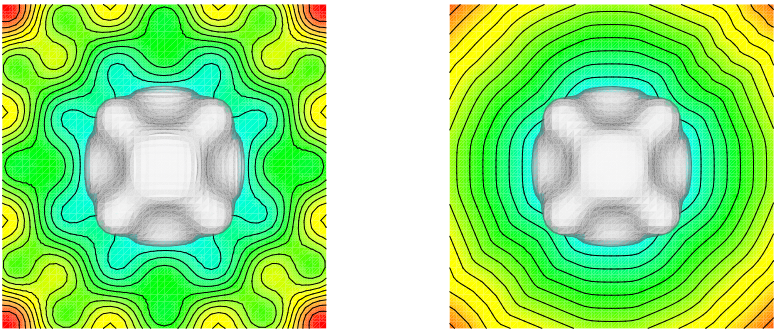

Figure 10.3.: Zero level and contour lines on slice $z = 0$ before (left) and after reparametrization (right).

interface we perform 200 time steps of size $dt = 0.1$ of the level set equation (2.25). The triangulation of Ω consists of $24 \times 24 \times 24$ subcubes each subdivided into six tetrahedra yielding 117649 unknowns for the level set function φ. The results for different time steps are shown in Figure 10.2. As one can see, the interfaces for $t \in \{0, 20\}$ and for $t \in \{5, 15\}$ are almost identical, which is reasonable regarding (10.1). However, we did not exactly measure the discrepancies.

10.2. Reparametrization

We consider the cubic domain $\Omega = [-1, 1]^3$ and the scalar function

$$\varphi(\mathbf{x}) = \big|\mathbf{x} \cdot \big(1 + 0.2\, g(\mathbf{x})\big)\big| - 0.5, \qquad \mathbf{x} \in \Omega,$$

where $g(\mathbf{x}) = \cos(10x)\cos(10y)\cos(10z)$ for $\mathbf{x} = (x, y, z) \in \Omega$. The zero level of φ and contour lines of φ on the slice $z = 0$ are shown in Figure 10.3 on the left. Apparently, φ is not a distance function as its contour lines are not equidistant.

For spatial discretization Ω is split into $24 \times 24 \times 24$ subcubes, where each of them is subdivided into six tetrahedra. The corresponding P_2 discretization of φ requires 117649 unknowns. Applying the Fast Marching method described in Section 8.1 we obtain the reparametrized function $\tilde{\varphi}$, which is an approximate distance function, cf. Figure 10.3 on the right. Comparing φ and $\tilde{\varphi}$, the

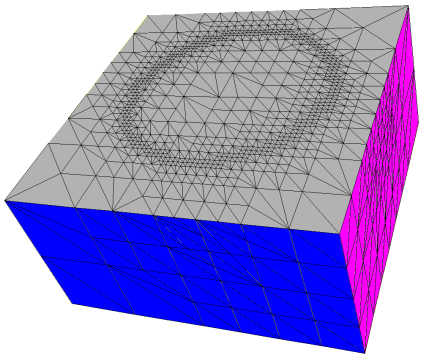

Figure 10.4.: Lower half part of the 4 times refined mesh $\mathcal{T}_4$.

zero level is only slightly changed. A quantitative comparison measuring the difference of the corresponding interfaces has not been performed, yet.

10.3. Approximation order of surface tension force discretization

In this section we present results of a numerical experiment which indicates that the $\mathcal{O}(\sqrt{h})$ bound in Corollary 5.23 is sharp. Furthermore, for the improved approximation described in Section 5.3.4 the $\mathcal{O}(h)$ bound will be confirmed numerically.

We consider the domain $\Omega := [-1,1]^3$ where the ball $\Omega_1 := \{\, \mathbf{x} \in \Omega : \|\mathbf{x}\| < R \,\}$ is located in the center of the domain. In our experiments we take $R = \frac{1}{2}$.

For the discretization a uniform tetrahedral mesh $\mathcal{T}_0$ is used where the vertices form a $6 \times 6 \times 6$ lattice, hence $h_0 = \frac{1}{5}$. This coarse mesh $\mathcal{T}_0$ is locally refined in the vicinity of $\Gamma = \partial\Omega_1$ using the adaptive refinement algorithm presented in Section 3.2.3. This repeated refinement process yields the gradually refined meshes $\mathcal{T}_1, \mathcal{T}_2, \ldots$ with *local* (i. e., close to the interface) mesh sizes $h_i = \frac{1}{5} \cdot 2^{-i}, i = 1, 2, \ldots$. Part of the tetrahedral triangulation $\mathcal{T}_4$ is shown in Figure 10.4. The corresponding finite element spaces $\mathbf{V}_i := \mathbf{V}_{h_i} = (V_{h_i})^3$ consist of vector functions where each component is a continuous piecewise quadratic function on $\mathcal{T}_i$.

The interface $\Gamma = \partial\Omega_1$ is a sphere and thus the curvature $\kappa = -\frac{2}{R}$ is constant.

If we discretize the flow problem using $\mathbf{V}_i$ as discrete velocity space, we have to approximate the surface tension force

$$f_\Gamma(\mathbf{v}) = \frac{2\tau}{R}\int_\Gamma \mathbf{n}_\Gamma \cdot \mathbf{v}\, ds = \tau \int_\Gamma \nabla_\Gamma \operatorname{id}_\Gamma \cdot \nabla_\Gamma \mathbf{v}\, ds, \qquad \mathbf{v} \in \mathbf{V}_i. \tag{10.2}$$

To simplify notation, we take a fixed $i \geq 0$ and the corresponding local mesh size parameter is denoted by $h = h_i$. For the approximation of the interface we use the approach described in Section 5.1.2.

The discrete approximation of the surface tension force is

$$f_{\Gamma_h}(\mathbf{v}) = \tau \int_{\Gamma_h} \nabla_{\Gamma_h} \operatorname{id}_{\Gamma_h} \cdot \nabla_{\Gamma_h} \mathbf{v}\, ds, \qquad \mathbf{v} \in \mathbf{V}_i.$$

We are interested in, cf. Corollary 5.23,

$$\|f_\Gamma - f_{\Gamma_h}\|_{\mathbf{V}_i'} := \sup_{\mathbf{v}\in\mathbf{V}_i} \frac{f_\Gamma(\mathbf{v}) - f_{\Gamma_h}(\mathbf{v})}{\|\mathbf{v}\|_1}. \tag{10.3}$$

The evaluation of $f_\Gamma(\mathbf{v})$, for $\mathbf{v} \in \mathbf{V}_i$, requires the computation of integrals on curved triangles or quadrilaterals $\Gamma \cap S$ where S is a tetrahedron from the mesh $\mathcal{T}_i$. We are not able to compute these exactly. Therefore, we introduce an artificial force term which, in this model problem with a known constant curvature, is computable and sufficiently close to f_Γ.

Lemma 10.1
For $\mathbf{v} \in \mathbf{V} = (H_0^1(\Omega))^3$ define

$$\hat{f}_{\Gamma_h}(\mathbf{v}) := \frac{2\tau}{R}\int_{\Gamma_h} \mathbf{n}_h \cdot \mathbf{v}\, ds,$$

where $\mathbf{n}_h$ is the piecewise constant outward unit normal on Γ_h. Then the following inequality holds:

$$\|f_\Gamma - \hat{f}_{\Gamma_h}\|_{\mathbf{V}'} \leq ch. \tag{10.4}$$

Proof. Let $\Omega_{1,h} \subset \Omega$ be the domain enclosed by Γ_h, i. e., $\partial\Omega_{1,h} = \Gamma_h$. We define $D_h^+ := \Omega_1 \setminus \Omega_{1,h}$, $D_h^- := \Omega_{1,h} \setminus \Omega_1$ and $D_h := D_h^+ \cup D_h^-$. Due to Stokes theorem, for $\mathbf{v} \in \mathbf{V}$ we have

$$|f_\Gamma(\mathbf{v}) - \hat{f}_{\Gamma_h}(\mathbf{v})| = \frac{2\tau}{R}\left|\int_{\Omega_1} \operatorname{div}\mathbf{v}\, d\mathbf{x} - \int_{\Omega_{1,h}} \operatorname{div}\mathbf{v}\, d\mathbf{x}\right| \tag{10.5}$$

$$= \frac{2\tau}{R}\left|\int_{D_h^+} \operatorname{div}\mathbf{v}\, d\mathbf{x} - \int_{D_h^-} \operatorname{div}\mathbf{v}\, d\mathbf{x}\right| \tag{10.6}$$

$$\leq \frac{2\tau}{R}\int_{D_h} |\operatorname{div}\mathbf{v}|\, d\mathbf{x}. \tag{10.7}$$

Using the Cauchy-Schwarz inequality, we get the estimate

$$|f_\Gamma(\mathbf{v}) - \hat{f}_{\Gamma_h}(\mathbf{v})| \le c\sqrt{\mathrm{meas}_3(D_h)}\,\|\mathbf{v}\|_1 \quad \text{for all} \quad \mathbf{v} \in \mathbf{V}.$$

which results in the upper bound

$$\|f_\Gamma - \hat{f}_{\Gamma_h}\|_{\mathbf{V}'} \le c\sqrt{\mathrm{meas}_3(D_h)}. \tag{10.8}$$

Note that for the piecewise planar approximation Γ_h of the interface Γ we have $\mathrm{meas}_3(D_h) = \mathcal{O}(h^2)$ and thus (10.4) holds. □

From Lemma 10.1 we obtain $\|f_\Gamma - \hat{f}_{\Gamma_h}\|_{\mathbf{V}_j'} \le c\,h$ with a constant c independent of j. Thus we have

$$\|\hat{f}_{\Gamma_h} - f_{\Gamma_h}\|_{\mathbf{V}_i'} - ch \le \|f_\Gamma - f_{\Gamma_h}\|_{\mathbf{V}_i'} \le \|\hat{f}_{\Gamma_h} - f_{\Gamma_h}\|_{\mathbf{V}_i'} + ch. \tag{10.9}$$

The term $\|\hat{f}_{\Gamma_h} - f_{\Gamma_h}\|_{\mathbf{V}_i'}$ can be evaluated as follows. Since Γ_h is piecewise planar and $\mathbf{v} \in \mathbf{V}_i$ is a piecewise quadratic function, for $\mathbf{v} \in \mathbf{V}_i$, both $\hat{f}_{\Gamma_h}(\mathbf{v})$ and $f_{\Gamma_h}(\mathbf{v})$ can be computed exactly (up to machine accuracy) using suitable quadrature rules.

For the evaluation of the dual norm $\|\cdot\|_{\mathbf{V}_i'}$ we proceed as follows. Let $\{\mathbf{v}_j\}_{j=1,\dots,n}$ ($n := \dim \mathbf{V}_i$) be the standard nodal basis in $\mathbf{V}_i$ and $J_{\mathbf{V}_i} : \mathbb{R}^n \to \mathbf{V}_i$ the isomorphism $J_{\mathbf{V}_i}\underline{x} = \sum_{k=1}^n x_k \mathbf{v}_k$. Let M_h be the mass matrix and A_h the stiffness matrix of the Laplacian:

$$\begin{aligned} (M_h)_{ij} &:= \int_\Omega \mathbf{v}_i \cdot \mathbf{v}_j \, d\mathbf{x}, \\ (A_h)_{ij} &:= \int_\Omega \nabla \mathbf{v}_i \cdot \nabla \mathbf{v}_j \, d\mathbf{x}. \end{aligned} \qquad 1 \le i, j \le n.$$

Define $C_h = A_h + M_h$. Note that for $\mathbf{v} = J_{\mathbf{V}_i}\underline{x} \in \mathbf{V}_i$ we have $\|\mathbf{v}\|_1^2 = \langle C_h\underline{x}, \underline{x}\rangle$. Take $e \in \mathbf{V}_i'$ and define $\underline{e} \in \mathbb{R}^n$ by $\underline{e}_j := e(\mathbf{v}_j)$, $j = 1, \dots, n$. Due to

$$\|e\|_{\mathbf{V}_i'} = \sup_{\mathbf{v}\in\mathbf{V}_i} \frac{|e(\mathbf{v})|}{\|\mathbf{v}\|_1} = \sup_{\underline{x}\in\mathbb{R}^n} \frac{|\sum_{j=1}^n x_j e(\mathbf{v}_j)|}{\sqrt{\langle C_h\underline{x}, \underline{x}\rangle}}$$

we obtain

$$\|e\|_{\mathbf{V}_i'} = \sup_{\underline{x}\in\mathbb{R}^n} \frac{\langle \underline{x}, \underline{e}\rangle}{\sqrt{\langle C_h\underline{x}, \underline{x}\rangle}} = \|C_h^{-1/2}\underline{e}\| = \sqrt{\langle C_h^{-1}\underline{e}, \underline{e}\rangle}. \tag{10.10}$$

Thus for the computation of $\|e\|_{\mathbf{V}_i'}$ we proceed in the following way:

i	$\|\hat{f}_{\Gamma_h} - f_{\Gamma_h}\|_{\mathbf{V}_i'}$	order	$\|\hat{f}_{\Gamma_h} - \tilde{f}_{\Gamma_h}\|_{\mathbf{V}_i'}$	order
0	1.79 E-1	–	1.32 E-1	–
1	1.40 E-1	0.35	4.43 E-2	1.57
2	1.03 E-1	0.45	1.46 E-2	1.61
3	7.22 E-2	0.51	5.06 E-3	1.52
4	5.02 E-2	0.53	1.78 E-3	1.51

Table 10.1.: Error norms and numerical order of convergence for different refinement levels.

1. Compute $\underline{e} = \big(e(\mathbf{v}_j)\big)_{j=1}^{n}$.
2. Solve the linear system $C_h\, \underline{z} = \underline{e}$ up to machine accuracy.
3. Compute $\|e\|_{\mathbf{V}_i'} = \sqrt{\langle \underline{z},\, \underline{e} \rangle}$.

We applied this strategy to $e := \hat{f}_{\Gamma_h} - f_{\Gamma_h}$. The results are given in the second column in Table 10.1. The numerical order of convergence in the third column of this table clearly indicates an $\mathcal{O}(\sqrt{h})$ behavior. Due to (10.9) this implies the same $\mathcal{O}(\sqrt{h})$ convergence behavior for $\|f_\Gamma - f_{\Gamma_h}\|_{\mathbf{V}_i'}$. This indicates that the $\mathcal{O}(\sqrt{h})$ bound in Corollary 5.23 is sharp.

The same procedure can be applied with f_{Γ_h} replaced by the modified (improved) approximate surface tension force

$$\tilde{f}_{\Gamma_h}(\mathbf{v}) = -\tau \sum_{i=1}^{3} \tilde{g}_{h,i}(v_i)$$

with $\tilde{g}_{h,i}$ as defined in (5.44). This yields the results in the fourth column in Table 10.1. For this modification the numerical order of convergence is significantly better, namely at least first order in h. From (10.9) it follows that for $\|f_\Gamma - \tilde{f}_{\Gamma_h}\|_{\mathbf{V}_i'}$ we can expect $\mathcal{O}(h^p)$ with $p \geq 1$.

Summarizing, we conclude that the results of these numerical experiments confirm the theoretical $\mathcal{O}(\sqrt{h})$ error bound derived in the analysis in Section 5.3.3 and show that the modified approximation indeed leads to (much) better results.

Results of numerical experiments for a Stokes two-phase flow problem using both f_{Γ_h} and $\tilde{f}_{\Gamma_h}$ are presented in Section 10.4.

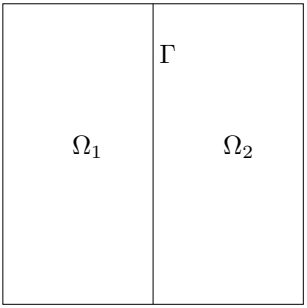

Figure 10.5.: 2D illustration of the phase distribution for test case A with $\Gamma = \Gamma_1$.

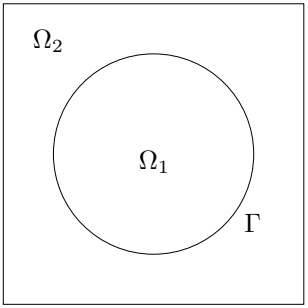

Figure 10.6.: 2D illustration of the phase distribution for test case B.

10.4. Pressure jump induced by surface tension

In this section we consider the following Stokes problem on the domain $\Omega = (-1,1)^3$ using the notation from Chapter 5,

$$\begin{aligned} a(\mathbf{u},\mathbf{v}) + b(\mathbf{v},p) &= f_{\mathrm{SF}}(\mathbf{v}) \quad \text{for all } \mathbf{v} \in \mathbf{V}, \\ b(\mathbf{u},q) &= 0 \quad \text{for all } q \in Q. \end{aligned} \tag{10.11}$$

Here $f_{\mathrm{SF}} \in \mathbf{V}'$ is a surface force term on the interface Γ which will be specified in the two test cases below. For simplicity we assume constant viscosity $\mu = 1$. The finite element discretization of (10.11) is as follows:

$$\begin{aligned} a(\mathbf{u}_h,\mathbf{v}_h) + b(\mathbf{v}_h,p_h) &= f_{\mathrm{SF},h}(\mathbf{v}_h) \quad \text{for all } \mathbf{v}_h \in \mathbf{V}_h, \\ b(\mathbf{u}_h,q_h) &= 0 \quad \text{for all } q_h \in Q_h, \end{aligned} \tag{10.12}$$

where $f_{\mathrm{SF},h} \in \mathbf{V}_h'$ is an approximation of f_{SF}. We choose a uniform initial triangulation $\mathcal{T}_0$ where the vertices form a $5\times5\times5$ lattice and apply an adaptive refinement algorithm presented in [GR05]. Local refinement of the coarse mesh $\mathcal{T}_0$ in the vicinity of Γ yields the gradually refined meshes $\mathcal{T}_1, \mathcal{T}_2, \mathcal{T}_3, \mathcal{T}_4$ with local mesh sizes $h_\Gamma = h_i = 2^{-i-1}, i = 0,\dots,4$ at the interface. For the discretization of $\mathbf{u}$ we choose the standard finite element space of piecewise quadratics:

$$\mathbf{V}_h := \{\, \mathbf{v} \in C(\Omega)^3 : \mathbf{v}|_T \in \mathcal{P}_2 \text{ for all } T \in \mathcal{T}_h, \quad \mathbf{v}_{|\partial\Omega} = 0 \,\}.$$

We compute the errors

$$e_{\mathbf{u}} := \mathbf{u}^* - \mathbf{u}_h \quad \text{and} \quad e_p := p^* - p_h$$

interface	# ref.	$\dim \mathbf{V}_h$	$\dim Q_h^1$	$\dim Q_h^{\Gamma_h}$	$\dim Q_h^0$
$\Gamma = \Gamma_1$	0	1029	125	150	384
	1	6801	455	536	1984
	2	31197	1657	1946	8384
	3	131433	6235	7324	33984
	4	537717	24093	28318	136384
$\Gamma = \Gamma_2$	0	1029	125	190	384
	1	7749	543	768	2304
	2	42633	2313	3146	11556
	3	200469	9607	12808	52088
	4	871881	39229	51774	221796

Table 10.2.: Dimensions of the finite element spaces for test case A.

for different choices of the pressure finite element space Q_h to compare the approximation properties of the different spaces. In our experiments we used piecewise constant or continuous piecewise linear elements, i. e., the spaces Q_h^0, Q_h^1 respectively, and the extended pressure space $Q_h^{\Gamma_h}$ introduced in Section 5.4.2.

10.4.1. Test case A: Pressure jump at a planar interface

This simple test case is designed to examine interpolation errors of finite element spaces for the approximation of discontinuous jumps of the pressure variable.

For f_{SF} we choose the artificial surface force $f_{\mathrm{SF}} = f_{\mathrm{ASF}}$ where

$$f_{\mathrm{ASF}}(\mathbf{v}) = \sigma \int_\Gamma \mathbf{v} \cdot \mathbf{n}\, ds, \qquad \mathbf{v} \in \mathbf{V}$$

and $\sigma > 0$ is a constant. Note that $f_{\mathrm{ASF}} \in \mathbf{V}'$. Then the unique solution of (10.11) is given by

$$u^* = 0, \qquad p^* = \begin{cases} C & \text{in } \Omega_1, \\ C + \sigma & \text{in } \Omega_2. \end{cases}$$

since $b(\mathbf{v}, p^* - C) = -\int_{\Omega_2} \sigma \operatorname{div} \mathbf{v}\, d\mathbf{x} = \int_\Gamma \sigma \mathbf{v}\mathbf{n}\, ds$ for arbitrary $\mathbf{v} \in \mathbf{V}$. Here C is a constant such that $\int_\Omega p^*\, d\mathbf{x} = 0$. In our calculations we used $\sigma = 1$.

We consider two different interfaces Γ_1 and Γ_2, which are both planes. Γ_1 is defined by

$$\Gamma_1 = \{ (x, y, z) \in \Omega : \ z = 0 \}.$$

In this case the two subdomains are given by $\Omega_1 := \{ (x, y, z) \in \Omega : \ z < 0 \}$ and $\Omega_2 := \Omega \setminus \overline{\Omega}_1$, cf. Figures 10.5 and 10.7. Interface Γ_2 is defined by

$$\Gamma_2 = \{ (x, y, z) \in \Omega : \ y + z = 1 \}$$

and the corresponding subdomains are $\Omega_1 := \{ (x, y, z) \in \Omega : \ y + z < 0 \}$ and $\Omega_2 := \Omega \setminus \overline{\Omega}_1$, cf. Figure 10.9. We emphasize that for both interfaces the interface approximation Γ_h is exact, i. e., $\Gamma_h = \Gamma$, allowing for an exact discretization of the interfacial force, i. e., $f_{\mathrm{ASF},h} = f_{\mathrm{ASF}}$.

Due to $\mathbf{g} = 0$, $\mathbf{u}^* \in \mathbf{V}_h$ and the fact that $\| f_{\mathrm{ASF},h} - f_{\mathrm{ASF}} \|_{\mathbf{V}_h'} = 0$ the error bound (5.12) simplifies to

$$\mu \| e_{\mathbf{u}} \|_1 + \| e_p \|_{L^2} \leq c \inf_{q_h \in Q_h} \| p^* - q_h \|_{L^2}. \tag{10.13}$$

Thus the errors in velocity and pressure are solely controlled by the approximation property of the finite element space Q_h.

The number of velocity and pressure unknowns for the grids $\mathcal{T}_0, \ldots, \mathcal{T}_4$ with different refinement levels are shown in Table 10.2. Note that $\dim Q_h^{\Gamma_h} > \dim Q_h^1$ due to the extended basis functions and that $\dim Q_h^0$ is even (much) larger.

Remark 10.2
Note that for $f_{\mathrm{SF}} = f_\Gamma$ the corresponding pressure solution would be $p^* = 0$ as the curvature of Γ vanishes. Therefore this would not be an interesting test case. ◇

Interface at $\Gamma = \Gamma_1$

For $\Gamma = \Gamma_1$, the interface Γ is located at the element boundaries of tetrahedra intersected by Γ, i. e., for each tetrahedron T intersecting Γ we have that $\Gamma \cap T$ is equal to a face of T.

In this special situation, the discontinuous pressure p^* can be represented exactly in the finite element space Q_h^0 of piecewise constants, thus the finite element solution $(\mathbf{u}_h, p_h) \in \mathbf{V}_h \times Q_h^0$ is equal to $(\mathbf{u}^*, p^*)$. This is confirmed by the numerical results: the exact solution $(\mathbf{u}^*, p^*)$ fulfills the discrete equations (up to round-off errors). The same holds for the extended finite element space $Q_h^{\Gamma_h}$.

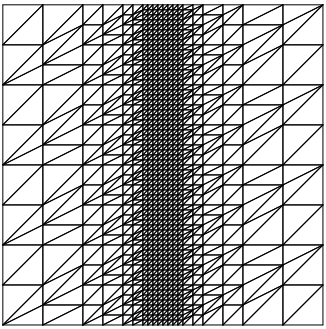

Figure 10.7.: Slice of grid $\mathcal{T}_h'$ at $x = 0$ after 3 refinements for $\Gamma = \Gamma_1$.

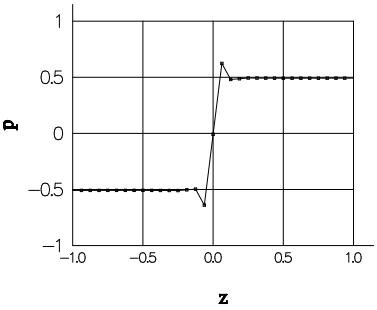

Figure 10.8.: 1D-profile of pressure jump at $x = y = 0$ for $p_h \in Q_h^1$. 3 refinements, $\Gamma = \Gamma_1$.

# ref.	$\|e_{\mathbf{u}}\|_{L^2}$	order	$\|e_{\mathbf{u}}\|_1$	order	$\|e_p\|_{L^2}$	order
0	4.26E-02	–	4.26E-01	–	5.32E-01	–
1	1.85E-02	1.2	3.41E-01	0.32	3.78E-01	0.49
2	7.09E-03	1.38	2.55E-01	0.42	2.68E-01	0.5
3	2.60E-03	1.45	1.85E-01	0.46	1.90E-01	0.5
4	9.37E-04	1.47	1.33E-01	0.48	1.34E-01	0.5

Table 10.3.: Errors and numerical order of convergence for the $P_2 - Q_h^1$ finite element pair, $\Gamma = \Gamma_1$.

For the P_1 finite elements we obviously have $p^* \notin Q_h^1$. The grid $\mathcal{T}_3$ after 3 times refinement and the corresponding pressure solution are shown in Figures 10.7 and 10.8. The error norms for different grid refinement levels are shown in Table 10.3. The L^2-error of the pressure shows a decay of $\mathcal{O}(h^{1/2})$. This confirms the theoretical results for the interpolation error $\min_{q \in Q_h^1} \|p^* - q_h\|_{L^2}$, cf. Section 5.4.1 and (10.13). The velocity error in the H^1-norm shows the same $\mathcal{O}(h^{1/2})$ behavior, whereas in the L^2-norm the error behaves like $\mathcal{O}(h^{3/2})$.

Interface at $\Gamma = \Gamma_2$

We now consider the case $\Gamma = \Gamma_2$. This problem corresponds to the 2D problem discussed in Section 5.4.1, cf. Figure 5.6. Γ is chosen such that $\Gamma \cap F \neq F$ for all faces of the triangulations $\mathcal{T}_0, \mathcal{T}_1, \mathcal{T}_2, \mathcal{T}_3$. As a consequence, $p^* \notin Q_h^0$ and $p^* \notin Q_h^1$, but $p^* \in Q_h^{\Gamma_h}$. We checked that the finite element

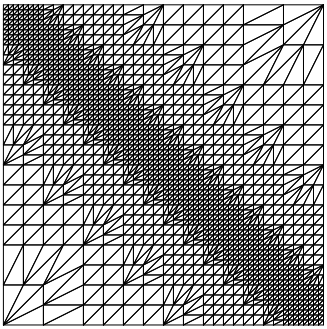

Figure 10.9.: Slice of grid at $x = 0$ after 3 refinements for $\Gamma = \Gamma_2$.

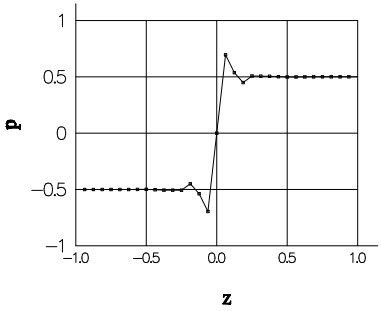

Figure 10.10.: 1D-profile of pressure jump at $x = y = 0$ for $p_h \in Q_h^1$. 3 refinements, $\Gamma = \Gamma_2$.

# ref.	$\|e_{\mathbf{u}}\|_{L^2}$	order	$\|e_{\mathbf{u}}\|_1$	order	$\|e_p\|_{L^2}$	order
0	2.53E-02	–	2.56E-01	–	5.44E-01	–
1	1.24E-02	1.02	2.25E-01	0.18	3.99E-01	0.45
2	5.03E-03	1.31	1.75E-01	0.36	2.88E-01	0.47
3	1.89E-03	1.41	1.29E-01	0.44	2.06E-01	0.48
4	6.88E-04	1.46	9.35E-02	0.47	1.46E-01	0.49

Table 10.4.: Errors and numerical order of convergence for the $P_2 - Q_h^1$ finite element pair, $\Gamma = \Gamma_2$.

solution $(\mathbf{u}_h, p_h) \in \mathbf{V}_h \times Q_h^{\Gamma_h}$ is in fact equal to $(\mathbf{u}^*, p^*)$.

Let us first discuss the results for P_1 finite elements. The grid $\mathcal{T}_3$ after 3 times refinement and the corresponding pressure solution for P_1 finite elements are shown in Figures 10.9 and 10.10 resp. The error norms for different grid refinement levels are shown in Table 10.4. The same convergence orders as for the case $\Gamma = \Gamma_1$ are obtained, cf. Table 10.3.

The results for the P_0 finite elements are shown in Table 10.5. Compared to P_1 finite elements, the errors are slightly larger but show similar convergence orders, i. e., $\mathcal{O}(h^{1/2})$ for the pressure L^2-error and velocity H^1-error as well as $\mathcal{O}(h^{3/2})$ for the L^2 velocity error.

# ref.	$\|e_{\mathbf{u}}\|_{L^2}$	order	$\|e_{\mathbf{u}}\|_1$	order	$\|e_p\|_{L^2}$	order
0	3.98E-02	–	3.49E-01	–	7.30E-01	–
1	1.64E-02	1.28	2.75E-01	0.35	4.89E-01	0.58
2	6.14E-03	1.41	2.04E-01	0.43	3.35E-01	0.54
3	2.22E-03	1.47	1.48E-01	0.46	2.34E-01	0.52
4	7.92E-04	1.49	1.06E-01	0.48	1.65E-01	0.51

Table 10.5.: Errors and numerical order of convergence for the $P_2 - Q_h^0$ finite element pair, $\Gamma = \Gamma_2$.

10.4.2. Test case B: Static bubble

In this test case (cf. Example 5.2) we consider a static bubble $\Omega_2 = \{ x \in \mathbb{R}^3 : \|x\| \leq r \}$ in the cube Ω with $r = 2/3$ (see Figure 10.6). We assume that surface tension is present, i. e., $f_{\mathrm{SF}} = f_\Gamma$ with $\tau = 1$. This problem has the unique solution

$$u^* = 0, \qquad p^* = \begin{cases} C & \text{in } \Omega_1, \\ C + \tau\kappa & \text{in } \Omega_2. \end{cases}$$

Since $\kappa = -2/r$, the pressure jump is equal to $[p^*]_\Gamma = 3$. A 2D variant of this test case is presented in [FCD+06, GMT07, Smo01].

Note that in this test case the errors in velocity and pressure are influenced by two error sources, namely the approximation error of the discontinuous pressure p^* in Q_h (as in test case A) and errors induced by the discretization of the surface force f_Γ, cf. (5.19).

The number of velocity and pressure unknowns for the grids $\mathcal{T}_0, \ldots, \mathcal{T}_4$ with different refinement levels are shown in Table 10.6. Note that $\dim Q_h^{\Gamma_h}$ is significantly larger than $\dim Q_h^1$, but that $\dim Q_h^{\Gamma_h} \ll \dim \mathbf{V}_h$.

#	test case B		
ref.	$\dim \mathbf{V}_h$	$\dim Q_h^1$	$\dim Q_h^{\Gamma_h}$
0	1029	125	176
1	5523	337	533
2	30297	1475	2295
3	139029	6127	9413
4	569787	24373	37355

Table 10.6.: Dimensions of the finite element spaces for test case B.

Remark 10.3
As Γ has constant curvature, for $\sigma = -\frac{2\tau}{r}$ the two considered surface forces coincide: $f_\Gamma = f_{\mathrm{ASF}}$. ◇

We consider test case B for two different approximations of the CSF term f_Γ, namely the "naive" Laplace-Beltrami discretization f_{Γ_h} as in (5.22) and the modified Laplace-Beltrami discretization $\tilde{f}_{\Gamma_h}$ as in (5.49). For the pressure space we choose $Q_h = Q_h^1$ and $Q_h = Q_h^{\Gamma_h}$. We did not consider the space Q_h^0 as it yields results comparable to those for Q_h^1. Table 10.7 shows the decay of the pressure L^2-norm for the four different experiments. We observe poor $\mathcal{O}(h^{1/2})$ convergence in the cases where $p_h \in Q_h^1$ or when the surface tension force f_Γ is discretized by f_{Γ_h}. For the L^2 and H^1-norm of the velocity error we observe convergence orders of $\mathcal{O}(h^{3/2})$ and $\mathcal{O}(h^{1/2})$, respectively, which is similar to the results in test case A.

We emphasize that only for the *combination* of the extended pressure finite element space $Q_h^{\Gamma_h}$ with the improved approximation $\tilde{f}_{\Gamma_h}$ we achieve $\mathcal{O}(h^\alpha)$ convergence with $\alpha \geq 1$ for the pressure L^2-error. The velocity error in the H^1-norm shows a similar behavior (at least first order convergence), in the L^2-norm we even have second order convergence, cf. Table 10.8.

For the improved Laplace-Beltrami discretization $\tilde{f}_{\Gamma_h}$ the corresponding pressure solutions $p_h \in Q_h^1$ and $p_h \in Q_h^{\Gamma_h}$ are shown in Figure 10.11.

#	$\|e_p\|_{L^2}$ for $p_h \in Q_h^1$				$\|e_p\|_{L^2}$ for $p_h \in Q_h^{\Gamma_h}$			
ref.	f_{Γ_h}	order	$\tilde{f}_{\Gamma_h}$	order	f_{Γ_h}	order	$\tilde{f}_{\Gamma_h}$	order
0	1.60E+0	–	1.60E+0	–	3.12E-1	–	1.64E-1	–
1	1.07E+0	0.57	1.07E+0	0.57	1.00E-1	1.64	4.97E-2	1.73
2	8.23E-1	0.38	8.23E-1	0.38	6.24E-2	0.68	1.66E-2	1.58
3	5.80E-1	0.51	5.80E-1	0.51	4.28E-2	0.54	7.16E-3	1.22
4	4.13E-1	0.49	4.13E-1	0.49	2.95E-2	0.54	2.83E-3	1.34

Table 10.7.: Pressure errors for the $P_2 - Q_h^1$ and $P_2 - Q_h^\Gamma$ finite element pair and different discretizations of f_Γ.

μ-dependence of the errors

We repeated the computations of $(\mathbf{u}_h, p_h) \in \mathbf{V}_h \times Q_h^{\Gamma_h}$ for the improved Laplace-Beltrami discretization $\tilde{f}_{\Gamma_h}$ on the fixed grid $\mathcal{T}_3$ varying the viscosity μ. The errors are given in Table 10.9. We clearly observe that the velocity

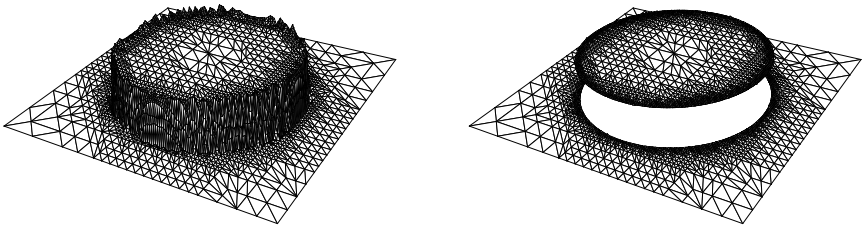

Figure 10.11.: Finite element pressure solution $p_h \in Q_h^1$ (on the left) and $p_h \in Q_h^{\Gamma_h}$ (on the right), visualized on slice of $\mathcal{T}_4'$ at $z = 0$.

# ref.	$\|e_{\mathbf{u}}\|_{L^2}$	order	$\|e_{\mathbf{u}}\|_1$	order
0	7.16E-03	–	1.10E-01	–
1	1.57E-03	2.19	4.26E-02	1.37
2	3.25E-04	2.28	1.70E-02	1.33
3	8.57E-05	1.92	7.43E-03	1.19
4	1.75E-05	2.29	2.40E-03	1.63

Table 10.8.: Errors and numerical order of convergence for the $P_2 - Q_h^\Gamma$ finite element pair and improved Laplace-Beltrami discretization $\tilde{f}_{\Gamma_h}$.

errors are proportional to μ^{-1} whereas the pressure error is independent of μ. This confirms the bound in (5.19).

μ	$\|e_{\mathbf{u}}\|_{L^2}$	$\|e_{\mathbf{u}}\|_1$	$\|e_p\|_{L^2}$
10	8.62E-06	7.51E-04	8.71E-03
1	8.57E-05	7.43E-03	7.16E-03
0.1	8.58E-04	7.44E-02	6.87E-03
0.01	8.57E-03	7.44E-01	6.88E-03
0.001	8.57E-02	7.43E+00	7.16E-03

Table 10.9.: Errors for the $P_2 - Q_h^\Gamma$ finite element pair and improved Laplace-Beltrami discretization $\tilde{f}_{\Gamma_h}$ on $\mathcal{T}_3$ for different viscosities μ.

11. Application examples

In the following sections we present some simulation results for real two-phase flow problems originating from droplet and falling film applications. Note that the results were obtained with the *serial* version of the DROPS code, as the parallel version does not provide the full functionality, yet. For example, the XFEM discretization of the pressure (cf. Section 5.4) is only implemented in the serial version and still has to be parallelized. Hence, the meshes used in the following examples are relatively coarse due to memory limitations or to keep computational times affordable. The parallelization of the whole DROPS code is a crucial task for the future as it will enable more levels of grid refinement to achieve more accurate solutions.

11.1. Levitated droplet in measuring cell

This experiment originates from an interdisciplinary research project [SFB] on the modeling of flow and mass transfer phenomena at the interface between a single droplet and the surrounding fluid. For NMR measurements of the velocity $\mathbf{u}$, cf. [AGHK$^+$05], a drop is levitated in a special device, which consists of a vertical glass tube with a narrowing in the middle. It is shown in horizontal position in Figure 11.1. A fluid flows from the top of the tube downwards. A drop which is lighter than the surrounding fluid is injected at the bottom of the tube and starts to rise upwards. At a certain point its buoyancy forces are balanced by the forces induced by the counterflow and the drop is levitated at a stable position. A photo of a levitated droplet is given in Figure 1.1. The aim of the following numerical simulation is to compute the equilibrium position and drop shape for two different two-phase systems:

- System A: silicon oil drop in heavy water (D_2O).
- System B: n-butanol drop in water.

The computational domain Ω and its triangulation are illustrated in Figure 11.1. It is $5 \cdot 10^{-2}\,m$ long and has a diameter of $7.2 \cdot 10^{-3}\,m$ at inlet

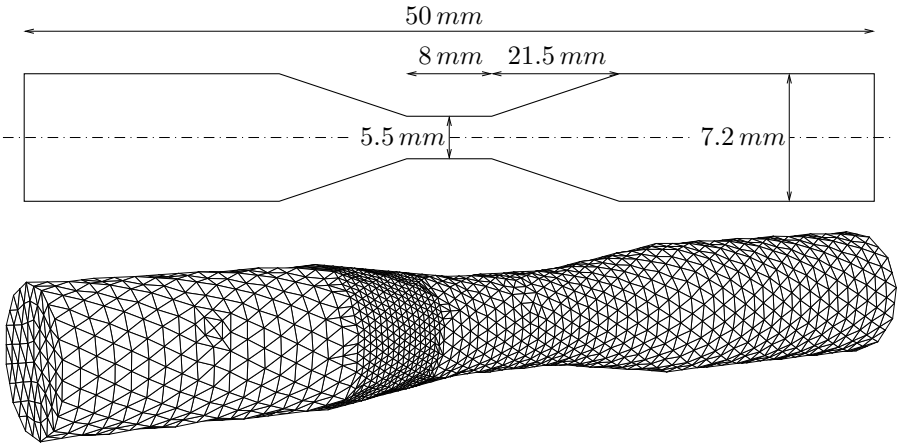

Figure 11.1.: 2D sketch (top) and 3D triangulation (bottom) of measuring cell.

and outlet and a diameter of $5.5 \cdot 10^{-3}\, m$ at the narrowest part of the tube, cf. the 2D sketch of the rotationally symmetric domain at the top of Figure 11.1. As an initial condition the drop is assumed to be spherical with a radius of r_d and is located $7 \cdot 10^{-3}\, m$ below the narrowest part. The initial triangulation $\mathcal{T}_0$ consisting of 4635 tetrahedra is successively refined in the vicinity of the drop. The finest triangulation $\mathcal{T}_2$ consists of 19254 tetrahedra for system A and 11712 tetrahedra for system B, hence, for system A roughly 75% and for system B roughly 60% of the tetrahedra are located in the refinement region.

The boundary conditions are chosen as follows:

- a prescribed parabolic inflow profile at the top of the tube (non-homogeneous Dirichlet boundary condition for $\mathbf{u}$) with maximum inflow velocity $\mathbf{u}_{in}$ in the middle of the inlet,
- an outflow boundary condition at the bottom of the tube (homogeneneous natural boundary condition) and
- no-slip boundary conditions at the remaining walls of the tube (homogeneous Dirichlet boundary condition for $\mathbf{u}$).

For the initial conditions we set φ_0 the signed distance function for the initial spherical drop and $\mathbf{u}_0$ the solution of the following stationary Stokes problem,

$$\begin{aligned} -\operatorname{div}(\mu(\varphi_0)\,\mathbf{D}(\mathbf{u})) + \nabla p &= \rho(\varphi_0)\,\mathbf{g}, \\ \operatorname{div}\mathbf{u} &= 0, \end{aligned} \qquad \text{in } \Omega. \tag{11.1}$$

quantity		System A		System B	
	(unit)	silicon oil	heavy water	n-butanol	water
ρ	(kg/m^3)	955	1107	845.4	986.5
μ	$(kg/m\,s)$	$2.6 \cdot 10^{-3}$	$1.2 \cdot 10^{-3}$	$3.281 \cdot 10^{-3}$	$1.388 \cdot 10^{-3}$
τ	(N/m)	$2 \cdot 10^{-3}$		$1.63 \cdot 10^{-3}$	
r_d	(m)	$1.75 \cdot 10^{-3}$		$1 \cdot 10^{-3}$	
$\mathbf{u}_{in}$	(m/s)	$25 \cdot 10^{-3}$		$35 \cdot 10^{-3}$	

Table 11.1.: Material properties of different liquids and experimental parameters used in the levitated bubble simulations.

System	ρ_2/ρ_1	μ_2/μ_1	Re	Eo
A	1.16	0.46	598	66.5
B	1.17	0.42	70.4	23.7

Table 11.2.: Characteristic dimensionless numbers of the drop problem for system A and B.

The material properties ρ, μ, τ together with the experimental parameters r_d (initial radius of droplet) and $\mathbf{u}_{in}$ (maximum inflow velocity) are given in Table 11.1. There are four dimensionless numbers characterizing the drop problem, namely the density ratio ρ_2/ρ_1 (index 1 denotes the droplet phase, index 2 the continuous phase), the viscosity ratio μ_2/μ_1, the Reynolds number Re and the Eötvös number Eo,

$$Re = (2r_d)^{3/2}\sqrt{g}\,\frac{\rho_2}{\mu_2}, \qquad Eo = 4\rho_2\, g\, \frac{r_d^2}{\tau},$$

with $g = 9.81\, m/s^2$ denoting the gravitational acceleration. They are given in Table 11.2 for system A and B, respectively. Sometimes the Morton number $M = Eo^3/Re^4$ is used instead of the Reynolds number.

The droplet shape at its equilibrium position and the corresponding stationary velocity field are shown in Figure 11.2 for system A and in Figure 11.4 for system B. For visualization purposes the velocity field is plotted on a 2D cartesian grid intersecting the unstructured tetrahedral grid. In Figure 11.3 the n-butanol droplet (system B) is shown at an intermediate stage where it is still rising upwards. Here a part of the unstructered grid is visualized which

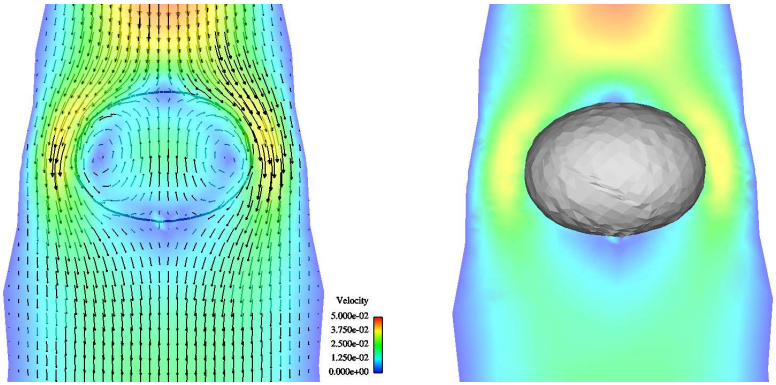

Figure 11.2.: Equilibrium position of silicon oil drop, visualized on slice. Velocity field (left) and shape of droplet (right).

is refined in the vicinity of the interface. Note that the grid resolution was chosen relatively coarse due to the non-parallel run of the simulation, as the XFEM discretization of the pressure is not available for the parallel version of the DROPS code, yet.

11.2. Rising bubble

In this numerical experiment we consider a single n-butanol droplet inside a cuboid tank $\Omega = [0, 20 \cdot 10^{-3}] \times [0, 30 \cdot 10^{-3}] \times [0, 20 \cdot 10^{-3}]\, m^3$ filled with water, cf. Figure 11.5. The material properties of this two-pase system can be found in Table 11.1. Initially at rest ($\mathbf{u}_0 = 0\, m/s$) the bubble starts to rise in y-direction due to buoyancy effects.

For the initial triangulation $\mathcal{T}_0$ the domain Ω is subdivided into $4 \times 10 \times 4$ sub-cubes each consisting of 6 tetrahedra. After that the grid is refined four times in the vicinity of the interface Γ. As time evolves the grid is adapted to the moving interface. Figure 11.6 shows the drop and a part of the adaptive mesh for two different time steps.

For a butanol droplet with radius $1\, mm$, in Figure 11.7 the y-coordinate of the droplet's barycenter $\overline{\mathbf{x}}_d$ is shown as a function of time, where

$$\overline{\mathbf{x}}_d(t) = \mathrm{meas}_3(\Omega_1(t))^{-1} \int_{\Omega_1(t)} \mathbf{x}\, d\mathbf{x}.$$

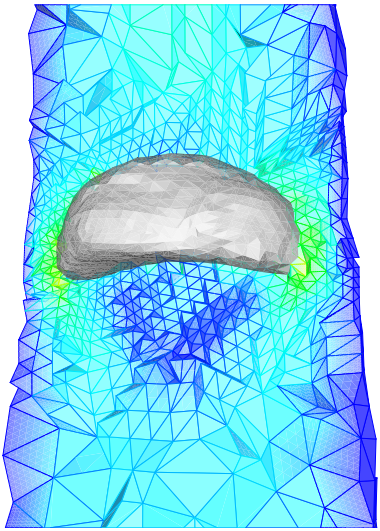

Figure 11.3.: Butanol drop rising in water.

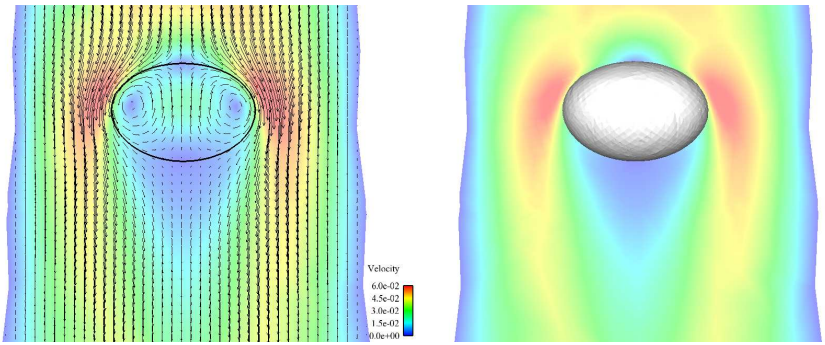

Figure 11.4.: Equilibrium position of n-butanol drop, visualized on slice. Velocity field (left) and shape of droplet (right).

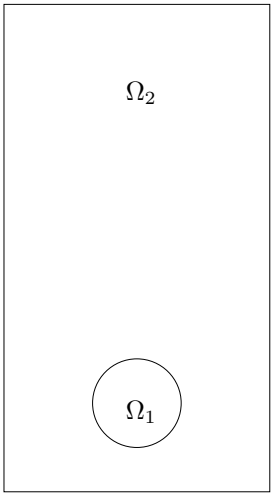

Figure 11.5.: 2D setup of the rising bubble example.

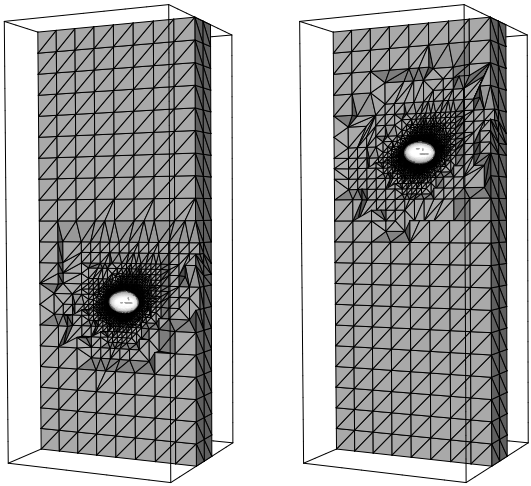

Figure 11.6.: Interface and part of grid for a rising bubble with radius $r_d = 1\,mm$ at the times $t = 0.2\,s$ (left) and $t = 0.4\,s$ (right).

The average velocity $\overline{\mathbf{u}}_d(t)$ of the drop is given by

$$\overline{\mathbf{u}}_d(t) = \mathrm{meas}_3(\Omega_1(t))^{-1} \int_{\Omega_1(t)} \mathbf{u}(\mathbf{x}, t)\, d\mathbf{x}.$$

Note that $\overline{\mathbf{x}}_d'(t) = \overline{\mathbf{u}}_d(t)$. Figure 11.8 shows the velocity in y-direction of a butanol droplet with radius $1\,mm$ as a function of time. After a certain time the bubble reaches a terminal rise velocity $u_r = \max_{t\in[t_0,t_f]} \|\overline{\mathbf{u}}(t)\|$. For the radius $r_d = 10^{-3}\,m$ we obtain $u_r = 53 \cdot 10^{-3}\,m/s$.

We computed the terminal rise velocities u_r of rising butanol droplets for different drop radii r_d, cf. Table 11.2. Note that for the larger droplets with $r_d \geq 1.5 \cdot 10^{-3}\,m$ a coarser mesh was used (3 times local refinement instead of 4 times as for the smaller droplets) because of memory limitations. The values are compared to model predictions where we applied an algebraic model of Henschke [Hen03] described in Remark 11.1 below. In Figure 11.9 the terminal rise velocity u_r is plotted versus the bubble radius r_d giving a comparison of model and simulation results. Note that the results agree very well for smaller droplets with radii $r_d \leq 1\,mm$. For the larger bubbles the relative deviations $\frac{|u_r^{\mathbf{DROPS}} - u_r^{\mathrm{model}}|}{|u_r^{\mathrm{model}}|}$ are up to 5%. We believe that the deviations for the

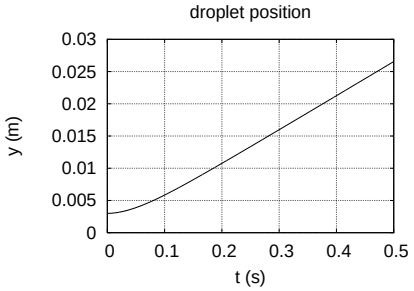

Figure 11.7.: Position y of barycenter of a rising butanol droplet with radius $1\,mm$ as a function of time t.

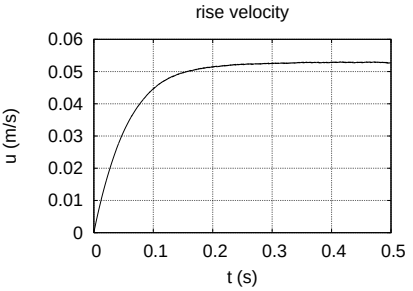

Figure 11.8.: Rise velocity u of a butanol droplet with radius $1\,mm$ as a function of time t.

r_d	(mm)	0.5	0.75	1	1.25	1.5	1.75	2
u_r^{DROPS}	(mm/s)	25.7	40.8	53.0	57.1	56.7	55.2	53.9
u_r^{model}	(mm/s)	25.5	40.3	53.7	60.0	57.5	55.6	55.8

Table 11.3.: Terminal rise velocity for different droplet radii r_d, obtained by DROPS simulation and predicted by algebraic model, cf. Remark 11.1.

larger bubbles are caused by the coarse grid resolution and that the results can be improved on a finer mesh as soon as the parallel version of DROPS is available.

Remark 11.1 (Algebraic model for terminal rise velocity)
In [Hen03] a model is derived for the terminal rise velocity u_r as a function of r_d. Using the notation

$$\|(x,y)\|_\alpha := (x^\alpha + y^\alpha)^{1/\alpha}$$

for $x, y \in \mathbb{R}$, $\alpha \geq 1$, the model is as follows,

$$u_r^{\text{model}} = \frac{u_{\text{ball}} u_{\text{o,c}}}{\|(u_{\text{ball}}, u_{\text{o,c}})\|_{\alpha_1}}, \tag{11.2}$$

where

$$u_{\text{o,c}} := \left\| \left(\sqrt{\frac{\alpha_2 \tau}{\rho_2 r_d}}, \sqrt{\frac{|\rho_2 - \rho_1| g\, r_d}{\rho_2}} \right) \right\|_8$$

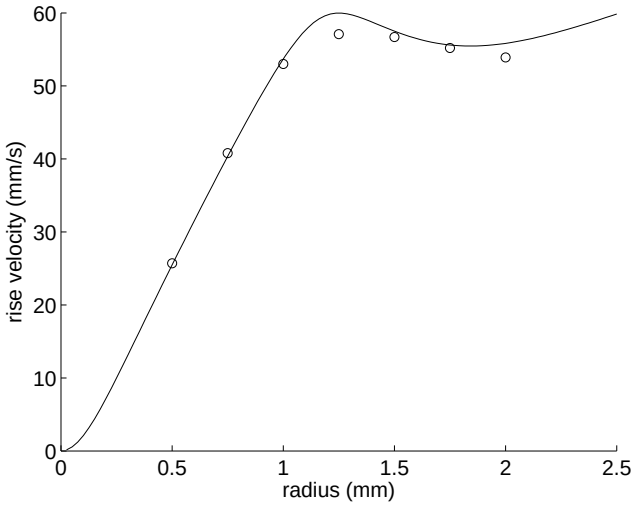

Figure 11.9.: Terminal rise velocities u_r for different droplet radii r_d. Algebraic model, cf. Remark 11.1, (solid line) vs. DROPS simulation results (circles).

is the terminal rise velocity of oscillating or cap-shaped droplets and α_1, α_2 are model parameters.

$$u_{\text{ball}} = \frac{Re_{\text{ball}}\,\mu_2}{2\rho_2 r_d}$$

denotes the terminal rise velocity of ball-shaped droplets. Here Re_{ball} is given by

$$\begin{aligned} Re_{\text{ball}} &= f\,Re_{\text{circ}} + (1-f)\,Re_{\text{rigid}}, \\ Re_{\text{circ}} &= \frac{Ar}{12(0.065Ar+1)^{1/6}}, \\ Re_{\text{rigid}} &= \sqrt{\frac{\frac{4}{3}Ar}{432Ar^{-1} + 20Ar^{-1/3} + 0.51\frac{Ar^{1/3}}{Ar^{1/3}+140}}}, \end{aligned}$$

where $Ar = \frac{g\,|\rho_2-\rho_1|\,\rho_2(2r_d)^3}{\mu_2^2}$ is the Archimedes number and $f = \frac{2\mu_2\lambda}{2\mu_2\lambda+3\mu_1}$ with $0 < \lambda = 1 - \frac{1}{1+(r_d/r_{\text{tr}})^{10}} < 1$. The parameter r_{tr} describes the transition regime from rigid to circulating droplets. The three model parameters $\alpha_1, \alpha_2, r_{\text{tr}}$ have been fitted to measurement data, yielding $\alpha_1 = 6.57$, $\alpha_2 = 2.89$ and $r_{tr} = 1.365 \cdot 10^{-3}\,m$ for the two-phase system n-butanol/water. ◇

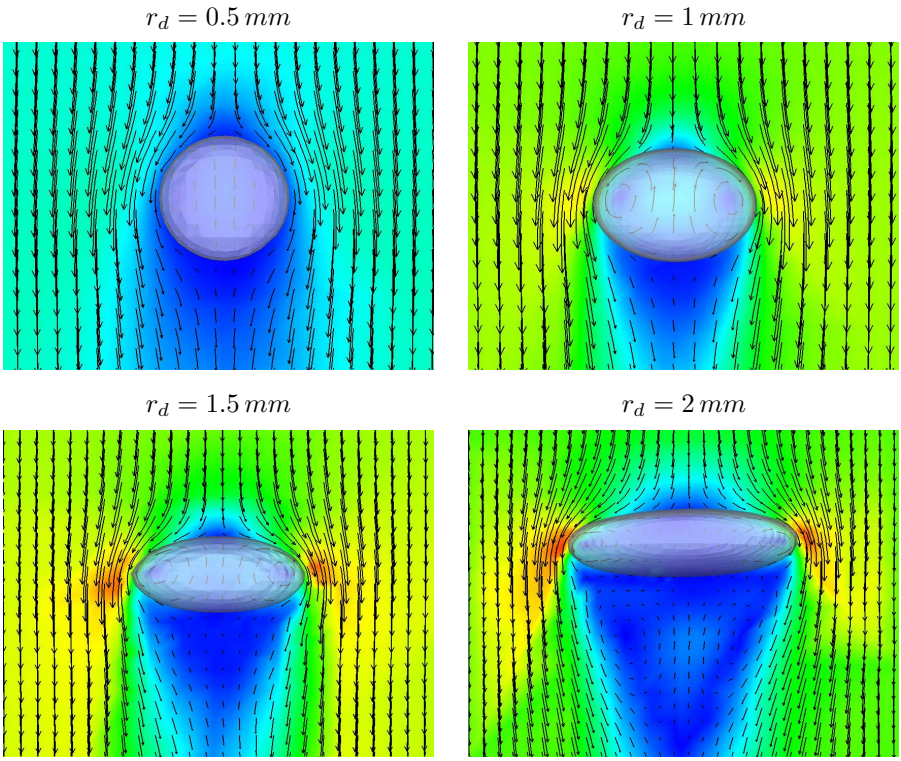

Figure 11.10.: Shape of n-butanol droplets for different radii r_d and velocity field $\mathbf{u} - \mathbf{u}_d$ visualized on slice.

The droplet shapes of rising butanol droplets for different radii r_d are shown in Figure 11.10. The droplet shape is almost spherical for $r_d = 0.5\,mm$ and becomes more and more flattened for larger radii. The corresponding velocity field $\mathbf{u} - \mathbf{u}_d$ (which is the velocity with respect to a reference frame moving with droplet speed $\mathbf{u}_d$) is visualized on a slice in the middle of the domain. Toroidal vortices can be found inside the droplets. For $r_d = 2\,mm$ we also observed a small vortex structure in the wake of the bubble.

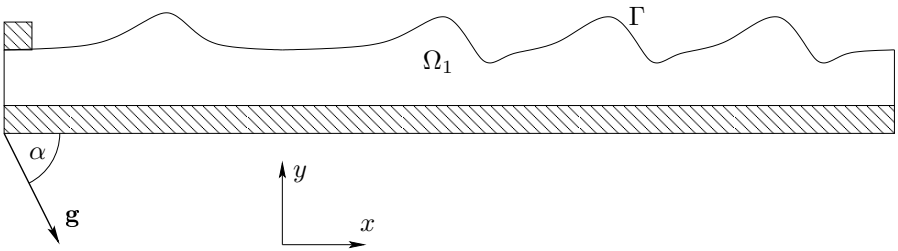

Figure 11.11.: 2D sketch of falling film.

11.3. Falling film

In this section we consider a falling film flow which is an example for a fluid/gas two-phase flow problem. Falling film flows are one of the research topics in the collaborative research center [SFB]. Due to their large interfacial area falling films are used in many chemical engineering applications, e. g., for heating and cooling devices, evaporation processes and as reactors for phase interface reactions.

The sketch of a falling film experiment is shown in Figure 11.11. It consists of an inclined plate with a rectangular inlet channel at the top. The fluid exits the inlet and develops a thin liquid film which is running down the plate. The interface between liquid and gaseous phase develops a wavy structure (even without external excitation) which enhances the heat and mass transport in the film. For falling film problems usually the coordinate system is chosen such that x is in flow direction, y is in normal direction to the plate and z denotes the transversal direction, cf. Figure 11.11.

For the numerical experiment we will not consider the whole range of the film experiment but only a wave in the region where the film profile becomes periodic. Hence, in the domain $\Omega = [0, L_x] \times [0, L_y] \times [0, L_z]$ we choose periodic boundary conditions in x and z direction and homogeneous Dirichlet boundary conditions for $y = 0$ and $y = L_y$. Let $\Omega_1(t)$ denote the liquid phase and $\Omega_2(t)$ the ambient gas phase. The initial conditions are chosen as follows. The initial local film thickness $\delta(x, z)$ is given by

$$\delta(x,z) = \delta_0 \left(1 + \omega \sin\left(2\pi \frac{x}{L_x}\right) \cos\left(2\pi \frac{z}{L_z}\right)\right)$$

with the average film thickness $\delta_0 > 0$ and the amplitude $0 < \omega < 1$. In our experiments we used $\delta_0 = 6.35 \cdot 10^{-4}\, m$ and $\omega = 0.5$. The initial value for the

quantity	(unit)	DMS-T05	air
ρ	(kg/m^3)	909.3	1.2
μ	$(kg/m\,s)$	$5.183 \cdot 10^{-3}$	$1.71 \cdot 10^{-5}$
τ	(N/m)	$2 \cdot 10^{-3}$	

Table 11.4.: Material properties used for the falling film problem.

level set function is given by $\varphi_0(x, y, z) = y - \delta(x, z)$. $\mathbf{u}_0$ is the solution of the stationary Stokes problem (11.1).

The size of the domain chosen in the numerical experiment is $L_x = 20.9 \cdot 10^{-3}\,m$ and $L_y = L_z = 4 \cdot 10^{-3}\,m$. The plate is inclined by the angle $\alpha = 0$ to the graviational acceleration vector, hence, the plate is assumed to be vertical. The material properties are given in Table 11.4.

For the initial triangulation $\mathcal{T}_0$ the domain is subdivided in $10 \times 6 \times 2$ cuboids each consisting of 6 tetrahedra. To refine the part of the domain where the fluid film is located we use the following strategy. For a tetrahedron T let $\mathbf{x}_T = (x_T, y_T, z_T) \in \mathbb{R}^3$ denote its barycenter. We mark all tetrahedra T with $y_T < 1.5 \cdot 10^{-3}\,m$ for refinement and apply the multilevel refinement algorithm to obtain $\mathcal{T}_1$. Repeating this one more time yields the triangulation $\mathcal{T}_2$ which then consists of 20 080 tetrahedra. We will use this static triangulation $\mathcal{T}_2$ for all time steps. An adaptive refinement in the vicinity of the interface Γ as for the rising bubble example was not possible due to the periodic boundary conditions. Up to now the refinement algorithm does not garantuee identical surface triangulations of corresponding periodic boundaries. This feature will be added in a future version of DROPS.

We emphasize that the falling film problem is very challenging from the numerical point of view due to the large jumps of the material properties in the liquid and gaseous phase as well as the large extent of the interface. Here we only give the results for the nonstationary *Stokes* film flow, i. e., we neglected the convective term $\mathbf{u} \cdot \nabla \mathbf{u}$ in the Navier-Stokes equations (2.14)–(2.15). This can be justified by a small Reynolds number $Re = 25.8$, where the Reynolds number for the film problem is defined by

$$Re = \frac{\overline{U}_{Nu}\,\delta_0\,\mu_1}{\rho_1} = \frac{\delta_0^3\,\rho_1^2\,\cos(\alpha) g}{3\mu_1^2}$$

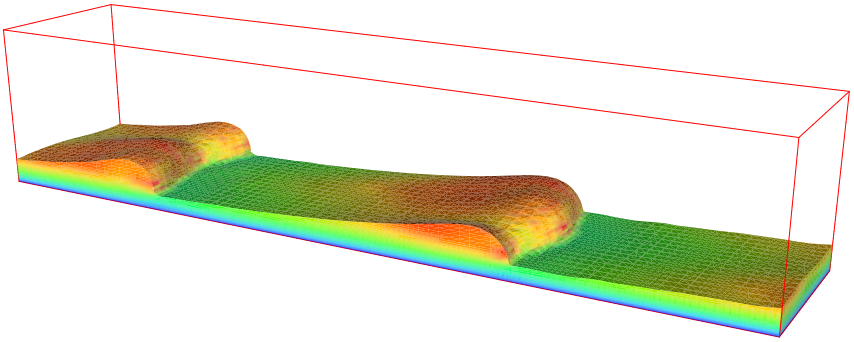

Figure 11.12.: Stokes flow of falling film for silicon oil DMS-T05 and air, incline angle 0°.

with the average Nusselt velocity $\overline{U}_{Nu}$,

$$\overline{U}_{Nu} = \frac{\cos(\alpha) g \, \rho_1 \, \delta_0}{2\mu_1}.$$

Figure 11.12 shows the falling film for the time $t = 0.2\,s$. The shape of the waves qualitatively looks similar to those depicted in Figure 1.2. A quantitative comparison with measurement data [LASLR05, SMDK06] or 2D film simulations [DLK07] from our project partners has to be accomplished in the future.

12. Summary and Outlook

We presented a numerical approach for solving three-dimensional incompressible two-phase flow problems. The governing equations are given by the continuum surface force (CSF) model, where the interface conditions are expressed by a localized surface force term. The interface is captured by a level set technique. For the spatial discretization a finite element method based on a tetrahedral multilevel triangulation is used. A multilevel refinement algorithm allows for local grid adaption. Applying a one-step theta-scheme for time discretization leads to a coupled system of level set and Navier-Stokes equations which is solved by a fixed point approach. This coupling can be avoided when applying a linearized variant of the one-step theta scheme. The nonlinearity of the Navier-Stokes problem is treated by a defect correction method. The resulting linear Oseen systems are solved by Uzawa-type methods or general Krylov methods which are applied to the block matrix, using special problem-adapted preconditioners for the Schur complement. From time to time the level set function has to be reparametrized by a fast marching method. Several numerical results demonstrate the capability of our approach to solve 3D incompressible two-phase flow problems for real two-phase systems originating from droplet and falling film applications.

On the basis of our experience in this field we come to the conclusion that *numerical methods originally designed for the simulation of incompressible one-phase flows are often not appropriate tools to solve incompressible two-phase flow problems.* In this context we mention numerical oscillations of the velocity at the interface, so-called *spurious currents*, which are reported by many other authors, e. g., [LNS+94, WKP99, FCD+06]. Hence, new methods and concepts have to be developed which address the special properties of two-phase systems such as discontinuous material properties, discontinuous pressure jump across the interface or the localized surface tension force term, just to mention a few. This thesis contributes to these topics by introducing and analyzing the following two new numerical methods:

- We developed an *improved Laplace-Beltrami discretization* $\tilde{f}_{\Gamma_h}$ of the localized surface tension force term f_Γ, cf. Section 5.3, which is superior compared to a standard Laplace-Beltrami discretization on a piecewise

planar interface approximation Γ_h. The improved discretization $\tilde{f}_{\Gamma_h}$ is of first order w. r. t. h in the dual norm $\|\cdot\|_{\mathbf{V}_h'}$, whereas the standard discretization f_{Γ_h} is only of order $\frac{1}{2}$. This has been shown by the theoretical analysis in Section 5.3.3 and was also observed in numerical experiments presented in Section 10.3.

- We introduced an *extended finite-element space* Q_h^Γ for the pressure, cf. Section 5.4, which is suitable for the approximation of functions which are smooth on $\Omega_1 \cup \Omega_2$ but discontinuous across Γ. For such functions the approximation error in $\|\cdot\|_{L_2(\Omega)}$ is $\mathcal{O}(\sqrt{h})$ for standard finite element spaces (conforming and non-conforming as well), cf. Section 5.4.1, whereas we achieve $\mathcal{O}(h^2)$ for the XFEM space Q_h^Γ, cf. Theorem 5.26. Combining pressure XFEM with the improved Laplace-Beltrami discretization for the surface tension force, numerical results for the standard test case of a static bubble show a substantial reduction of spurious currents compared to standard approaches, cf. Section 10.4.

In addition to these two methods, the main characteristics of our numerical strategy are the following:

- The level set method is applied for capturing the interface between the two phases, cf. Section 2.2.1.
- The spatial discretization is based on a hierarchy of grids which are constructed in such a way that they are consistent (i. e., no hanging nodes) and that the hierarchy of triangulations is stable, cf. Chapter 3. Local refinement and coarsening are easy to realize.
- For the discretization of level set and Navier-Stokes equations we use conforming P_2 finite elements for the velocity $\mathbf{u}$ and level set function φ, cf. Chapter 4, as well as extended finite elements for the pressure p, cf. Section 5.4.
- The one-step theta-scheme or a linearized variant of it is applied for time integration, cf. Section 6.1.2.
- In each time step the nonlinearity of the discrete Navier-Stokes problem is treated by a fixed point defect correction. The Oseen problems are solved by an inexact Uzawa method or Krylov subspace methods, where the Schur complement is preconditioned by special preconditioning techniques accounting for the piecewise constant material properties ρ and μ. All these issues are discussed in Chapter 7.
- The Fast Marching method is used for reparametrization of the level set

function φ, cf. Section 8.1.

- Most of the numerical components have been parallelized to enable the simulation of complex two-phase flow problems with sufficient resolution in affordable computational time, cf. Section 9.2. However, some important issues such as the XFEM discretization of the pressure are still missing in the parallel version of DROPS.

We emphasize that the combination of the level set method, finite element discretization, extended pressure finite element space, Laplace-Beltrami partial integration and multilevel local refinement is unique among all numerical strategies known to the author for the simulation of two-phase flow problems.

There are still many open questions and unresolved challenges left which should be addressed in the future. As an outlook we mention some of the topics, which are, in the opinion of the author of the thesis, among the most important to be considered.

New functionality

- Consider a variable surface tension coefficient $\tau = \tau(x,t)$. In this case the surface tension force term in weak formulation reads as follows,

$$f_\Gamma^{\text{var}}(\mathbf{v}) = \int_\Gamma \kappa \mathbf{n} \cdot (\tau \mathbf{v}) - \nabla_\Gamma \tau \cdot \mathbf{v} \, ds \qquad \text{for all } \mathbf{v} \in \mathbf{V}.$$

  As we saw in Section 5.3 an accurate discretization $f_{\Gamma_h}^{\text{var}} \in \mathbf{V}_h'$ of f_Γ^{var} is a delicate task. We should guarantee at least first order convergence w. r. t. the grid size h_Γ at the interface, i. e.,

$$\| f_{\Gamma_h}^{\text{var}} - f_\Gamma^{\text{var}} \|_{\mathbf{V}_h'} = \mathcal{O}(h_\Gamma^p)$$

  with $p \geq 1$. A method for the discretization of $f_{\Gamma_h}^{\text{var}}$, which extends the ideas used for a constant surface tension coefficient τ, has been implemented in DROPS. A systematic analysis of the quality of this approach has not been performed, yet.

- Include heat and mass transport in both phases. The additional partial differential equations are transient reaction-diffusion equations with piecewise constant coefficients due to the different material properties of the two phases. In the case of mass transport the concentration c has a jump at the interface. As was shown in Section 5.4.1, standard FEM techniques will not lead to satisfactory results. Here an XFEM approach combined with a Nitsche technique as described in [HH04] has an optimal approximation property. An alternative XFEM approach without

the need for a penalty term also shows this optimal order of convergence in numerical experiments, cf. [MCCR03]. We mention that we implemented a standard FEM discretization of a two-phase convection-diffusion equation in DROPS, but did not systematically analyze its accuracy, yet.

- Study Marangoni effects induced by a temperature-dependent or concentration-dependent surface tension coefficient τ.

- Include mass transport on the interface to model the contamination of the interface with surfactants. The higher the interface concentration of surfactants, the lower the surface tension. By this mechanism these impurities are made responsible for the so called *stagnant cap* of a droplet, i. e., a region where the interface gets rigid and inner circulations are dramatically slowed down compared to clean systems.

  For the discretization of the convection-diffusion equation on Γ an Eulerian finite element method described in [ORG08] will be applied. Here we got first preliminary results for a reaction-diffusion equation on a static sphere.

Improvement of numerical methods

- The design of better Schur complement preconditioners for the Oseen problem, especially for convection-dominated problems, is of great interest.

- Alternative iterative solvers such as multigrid methods applied to the Oseen problem [LR08] or the application of projection methods such as Chorin or SIMPLE [Ran04] should be compared to the methods described in Section 7.2 with respect to their efficiency.

- Other coupling strategies besides the one described in Algorithm 6.12 should be considered, for example, methods based on defect correction or Newton-type methods. A comparison with linearized time discretization schemes which avoid such a coupling will show the benefits and disadvantages of the different approaches.

- A suitable adaptive control of the time step size should be considered in the future.

- The stabilization of the finite element discretization of the Navier-Stokes equations by SDFEM [RST96, GLOS05] would enable the simulation of flow problems with higher Reynolds numbers.

- Up to now the level set function is used as an indicator for grid refinement which has proved to be satisfactory for the two-phase flow problems considered in this thesis. In the case of coupled two-phase momentum, heat and mass transport a more sophisticated control of grid adaption is demanded. Then error estimation techniques, e. g., in the spirit of [Ver96], have to be combined with appropriate strategies to decide which elements should be marked for refinement or coarsening.

Questions related to XFEM

- Is the finite element pair $\mathbf{V}_h \times Q_h^\Gamma$ LBB stable? If this is not the case, how can it be stabilized, e. g., by adding appropriate stabilization terms to the discretization?
- A good understanding of the time discretization in the case of a time dependent discrete divergence operator $B = B(t)$ is still lacking.
- The velocity $\mathbf{u} \in \mathbf{V}$ has a kink at the interface which is not optimally resolved by the standard piecewise quadratic finite elements currently applied for the discretization $\mathbf{u}_h \in \mathbf{V}_h$. Here an XFEM approach combined with a Nitsche technique as described in [HH04] or a modified XFEM abs-enrichment [MCCR03] would lead to an optimal approximation property. However, the technical difficulties arising from this method seem to overbalance its benefits.

Many of these issues are topics of current research and some of them will be implemented in the DROPS package in the near future. Our goal is to provide DROPS as an efficient and accurate 3D simulation tool for flow and transport processes in two-phase systems. The future perspectives are twofold in the following sense. From the mathematical point of view DROPS will serve as a framework to improve existing and develop new numerical methods for two-phase flow problems. From the application point of view DROPS will help users from the engineering community to solve their distinct real-life two-phase problems. Here both disciplines, numerical mathematics as well as engineering science, will benefit from each other. On the one hand the engineers will gain more insight from improved simulations, and on the other hand the mathematicians can learn from application examples, which numerical components should be further improved.

Bibliography

[AB73] N. Anderson and Å. Björck. A new high order method of regula falsi type for computing a root of an equation. *BIT*, 13:253–264, 1973.

[AGHK+05] A. Amar, E. Groß-Hardt, A. Khrapichev, S. Stapf, A. Pfennig, and B. Blümich. Visualizing flow vortices inside a single levitated drop. *J. Magn. Reson.*, 1:177, 2005.

[AMS01] S. N. Antontsev, A. M. Meirmanov, and V. A. Solonnikov. Smooth interface in a two-component Stokes flow. *Ann. Univ. Ferrara - Sez. VII - Sc. Mat.*, XLVII:269–284, 2001.

[AMY00] S. N. Antontsev, A. M. Meirmanov, and B. V. Yurinsky. A free-boundary problem for Stokes equations: classical solutions. *Interfaces and Free Boundaries*, 2:413–424, 2000.

[Bän98] E. Bänsch. *Numerical methods for the instationary Navier-Stokes equations with a free capillary surface.* Habilitation thesis, University of Freiburg, 1998.

[Bän01] E. Bänsch. Finite element discretization of the Navier-Stokes equations with a free capillary surface. *Numer. Math.*, 88:203–235, 2001.

[Bas96] P. Bastian. *Parallele adaptive Mehrgitterverfahren.* Teubner, Stuttgart, 1996.

[BBJ+97] P. Bastian, K. Birken, K. Johannsen, S. Lang, N. Neuß, H. Rentz-Reichert, and C. Wieners. UG – a flexible software toolbox for solving partial differential equations. *Comput. Vis. Sci.*, 1:27–40, 1997.

[BBJ+99] P. Bastian, K. Birken, K. Johannsen, S. Lang, V. Reichenberger, G. Wittum, and C. Wrobel. A parallel software-platform for solving problems of partial differential equations using unstructured grids and adaptive multigrid methods. In E. Krause and

W. Jäger, editors, *High Performance Computing in Science and Engineering*, pages 326–339. Springer, Berlin, 1999.

[Beh02] M. Behr. Free-surface flow simulations in the presence of inclined walls. *Comput. Methods Appl. Mech. Engrg.*, 191:5467–5483, 2002.

[Bey95] J. Bey. Tetrahedral grid refinement. *Computing*, 55:355–378, 1995.

[Bey98] J. Bey. *Finite-Volumen- und Mehrgitterverfahren für elliptische Randwertprobleme*. Advances in Numerical Methods. Teubner, Stuttgart, 1998.

[Bey00] J. Bey. Simplicial grid refinement: on Freudenthal's algorithm and the optimal number of congruence classes. *Numer. Math.*, 85:1–29, 2000.

[BGP87] M. O. Bristeau, R. Glowinski, and J. Periaux. Numerical methods for the Navier-Stokes equations. Application to the simulation of compressible and incompressible flows. *Computer Physics Report*, 6:73–188, 1987.

[BKW04] D. Bothe, M. Koebe, and H.-J. Warnecke. VOF-simulations of the rise behavior of single air bubbles with oxygen transfer to the ambient liquid. In F.-P. Schindler, editor, *Transport Phenomena with Moving Boundaries*, pages 134–146. VDI-Verlag, Düsseldorf, 2004.

[BKZ92] J. U. Brackbill, D. B. Kothe, and C. Zemach. A continuum method for modeling surface tension. *J. Comput. Phys.*, 100:335–354, 1992.

[BMUP01] T. Belytschko, N. Moës, S. Usui, and C. Parimi. Arbitrary discontinuities in finite elements. *Int. J. Num. Meth. Eng.*, 50:993–1013, 2001.

[BPV97] J. H. Bramble, J. E. Pasciak, and A. T. Vassilev. Analysis of the inexact Uzawa algorithm for saddle point problems. *SIAM Numer. Anal.*, 34:1072–1092, 1997.

[BSW83] R. E. Bank, A. H. Sherman, and A. Weiser. Refinement algorithms and data structures for regular local mesh refinement. In R. Stepleman, editor, *Scientific Computing*, pages 3–17. North-Holland, Amsterdam, 1983.

[CB03] J. Chessa and T. Belytschko. An extended finite element method for two-phase fluids. *ASME Journal of Applied Mechanics*, 70:10–17, 2003.

[Cha98] B. Chamberlain. Graph partitioning algorithms for distributing workloads of parallel computations. Technical Report UW-CSE-98-10-03, Department of Computer Science and Engineering, University of Washington, 1998.

[CHMO96] Y. C. Chang, T. Y. Hou, B. Merriman, and S. Osher. A level set formulation of Eulerian interface capturing methods for incompressible fluid flows. *J. Comput. Phys.*, 124:449–464, 1996.

[CMR08] G. Compère, E. Marchandise, and J.-F. Remacle. Transient adaptivity applied to two-phase incompressible flows. *J. Comput. Phys.*, 227(3):1923–1942, 2008.

[DD07] A. Demlow and G. Dziuk. An adaptive finite element method for the Laplace-Beltrami operator on implicitly defined surfaces. *SIAM J. Numer. Anal.*, 45:421–442, 2007.

[DDD] DDD, dynamic distributed data. Parallel environment for management and migration of distributed data. `http://sit.iwr.uni-heidelberg.de/~ddd/`.

[DFG+06] J. Du, B. Fix, J. Glimm, X. Jia, X. Li, Y. Li, and L. Wu. A simple package for front tracking. *J. Comput. Phys.*, 213(2):613–628, 2006.

[DLK07] G. F. Dietze, A. Leefken, and R. Kneer. Investigation of the back flow phenomenon in falling liquid films. *Journal of Fluid Mechanics*, 595:435–459, 2007.

[DRO] DROPS package for simulation of two-phase flows. `http://www.igpm.rwth-aachen.de/DROPS/`.

[Dzi91] G. Dziuk. An algorithm for evolutionary surfaces. *Numer. Math.*, 58:603–611, 1991.

[EHS+06] H. C. Elman, V. E. Howle, J. Shadid, R. Shuttleworth, and R. Tuminaro. Block preconditioners based on approximate commutators. *SIAM J. Sci. Comput.*, 27:1651–1668, 2006.

[Ens] Ensight, CEI Inc., 3D visualization tool. `http://www.ensight.com/`.

[Ess08] P. Esser. Iterative Löser für instationäre Stokes-Gleichungen zur Modellierung von Zwei-Phasen-Strömungen. Diploma thesis (in german), IGPM, RWTH Aachen University, 2008.

[FCD+06] M. M. Francois, S. J. Cummins, E. D. Dendy, D. B. Kothe, J. M. Sicilian, and M. W. Williams. A balanced-force algorithm for continuous and sharp interfacial surface tension models within a volume tracking framework. *J. Comput. Phys.*, 213(1):141–173, 2006.

[For07] O. Fortmeier. Loadbalancing of hierarchical tetrahedral grids using a graph approach, 7th December 2007. Presentation, Tag der Informatik 2007, Aachen.

[Fre42] H. Freudenthal. Simplizialzerlegungen von beschränkter Flachheit. *Annals of Mathematics*, 43:580–582, 1942.

[Gam] GAMBIT, ANSYS Inc., mesh generation tool. `http://www.fluent.com/`.

[Geo] Geomview, 3D viewing tool. `http://www.geomview.org/`.

[GGL+98] J. Glimm, J. W. Grove, X. L. Li, K.-M. Shyue, Y. Zeng, and Q. Zhang. Three-dimensional front tracking. *SIAM J. Sci. Comput.*, 19:703–727, 1998.

[GLOS05] T. Gelhard, G. Lube, M. A. Olshanskii, and J.-H. Starcke. Stabilized finite element schemes with LBB-stable elements for incompressible flows. *J. Comput. Appl. Math.*, 177(2):243–267, 2005.

[GMT07] S. Ganesan, G. Matthies, and L. Tobiska. On spurious velocities in incompressible flow problems with interfaces. *Comp. Meth. Appl. Mech. Engng.*, 196:1193–1202, 2007.

[GR05] S. Groß and A. Reusken. Parallel multilevel tetrahedral grid refinement. *SIAM J. Sci. Comput.*, 26(4):1261–1288, 2005.

[GR07a] S. Groß and A. Reusken. An extended pressure finite element space for two-phase incompressible flows with surface tension. *J. Comput. Phys.*, 224:40–58, 2007.

[GR07b] S. Groß and A. Reusken. Finite element discretization error analysis of a surface tension force in two-phase incompressible flows. *SIAM J. Numer. Anal.*, 45(4):1679–1700, 2007.

[Gro02] S. Groß. Parallelisierung eines adaptiven Verfahrens zur numerischen Lösung partieller Differentialgleichungen. Diploma thesis (in german), IGPM, RWTH Aachen University, 2002.

[GRR06] S. Groß, V. Reichelt, and A. Reusken. A finite element based level set method for two-phase incompressible flows. *Comp. Visual. Sci.*, 9(4):239–257, 2006.

[GSM+05] S. Groß, M. Soemers, A. Mhamdi, F. Al-Sibai, A. Reusken, W. Marquardt, and U. Renz. Identification of boundary heat fluxes in a falling film experiment using high resolution temperature measurements. *Int. J. Heat Mass Tran.*, 48:5549–5562, 2005.

[GT05] S. Ganesan and L. Tobiska. Finite element simulation of a droplet impinging a horizontal surface. In *Proceedings of ALGORITMY 2005*, pages 1–11, 2005.

[GW01] I. Ginzburg and G. Wittum. Two-phase flows on interface refined grids modeled with VOF, staggered finite volumes, and spline interpolants. *J. Comput. Phys.*, 166:302–335, 2001.

[Hen03] M. Henschke. *Auslegung pulsierter Siebboden-Extraktionskolonnen.* Habilitation thesis (in german), RWTH Aachen University, 2003.

[HH02] A. Hansbo and P. Hansbo. An unfitted finite element method, based on Nitsche's method, for elliptic interface problems. *Comput. Methods Appl. Mech. Engrg.*, 191(47–48):5537–5552, 2002.

[HH04] A. Hansbo and P. Hansbo. A finite element method for the simulation of strong and weak discontinuities in solid mechanics. *Comput. Methods Appl. Mech. Engrg.*, 193(33–35):3523–3540, 2004.

[HLPS05] P. Hansbo, C. Lovadina, I. Perugia, and G. Sangalli. A Lagrange multiplier method for the finite element solution of elliptic interface problems using non-matching meshes. *Numer. Math.*, 100(1):91–115, 2005.

[HN81] C. W. Hirt and B. D. Nichols. Volume of fluid (VOF) method for the dynamics of free boundaries. *J. Comput. Phys.*, 39:201–22, 1981.

[HT05] S. Hysing and S. Turek. The Eikonal equation: numerical ef-

ficiency vs. algorithmic complexity on quadrilateral grids. In *Proceedings of ALGORITMY 2005*, pages 22–31, 2005.

[Hup06] M. Huppertz. Reparametrization techniques for level set methods. Diploma thesis, IGPM, RWTH Aachen University, 2006.

[Hys06] S. Hysing. A new implicit surface tension implementation for interfacial flows. *Int. J. Num. Meth. Fluids*, 51(6):659–672, 2006.

[JT94] A. A. Johnson and T. E. Tezduyar. Mesh update strategies in parallel finite element computations of flow problems with moving boundaries and interfaces. *Comput. Methods Appl. Mech. Engrg.*, 119:73–94, 1994.

[KGM+08] M. Karalashvili, S. Groß, A. Mhamdi, A. Reusken, and W. Marquardt. Incremental identification of transport coefficients in convection-diffusion systems. *SIAM J. Sci. Comput.*, 2008. In press.

[KK98] G. Karypis and V. Kumar. A parallel algorithm for multilevel graph partitioning and sparse matrix ordering. *Journal of Parallel and Distributed Computing*, 48(1):71–95, 1998.

[KLW02] D. Kay, D. Loghin, and A. J. Wathen. A preconditioner for the steady state Navier-Stokes equations. *SIAM J. Sci. Comput.*, 24:237–256, 2002.

[KR94] P. Kloucek and F. Rys. Stability of the fractional step θ-scheme for the nonstationary Navier-Stokes equations. *SIAM J. Numer. Anal.*, 31:1312–1335, 1994.

[KS98] R. Kimmel and J. A. Sethian. Computing geodesic paths on manifolds. *Proc. Natl. Acad. Sci. USA*, 95(15):8431–8435, 1998.

[LASLR05] V.V. Lel, F. Al-Sibai, A. Leefken, and U. Renz. Local thickness and wave velocity measurements of wavy films. *Experiments in Fluids*, 39(5):856 – 864, 2005.

[LNS+94] B. Lafaurie, C. Nardone, R. Scardovelli, S. Zaleski, and G. Zanetti. Modelling merging and fragmentation in multiphase flows with SURFER. *J. Comput. Phys.*, 113(1):134–147, 1994.

[LR08] M. Larin and A. Reusken. A comparative study of efficient iterative solvers for generalized Stokes equations. *Numerical Linear Algebra with Applications*, 15:13–34, 2008.

[Mao07] D. Mao. Towards front-tracking based on conservation in two space dimensions II, tracking discontinuities in capturing fashion. *J. Comput. Phys.*, 226(2):1550–1588, 2007.

[Map] Maple, Waterloo Maple Inc., computer algebra system. `http://www.maplesoft.com/`.

[Mar05] W. Marquardt. Model-based experimental analysis of kinetic phenomena in multi-phase reactive systems. *Trans IChemE*, 83(A6):561–573, 2005.

[Mat] MATLAB, The MathWorks, numerical algebra system. `http://www.mathworks.com/`.

[MCCR03] N. Moës, M. Cloirec, P. Cartraud, and J.-F. Remacle. A computational approach to handle complex microstructure geometries. *Comput. Methods Appl. Mech. Engrg.*, 192:3163–3177, 2003.

[MCN03] P.D. Minev, T. Chen, and K. Nandakumar. A finite element technique for multifluid incompressible flow using Eulerian grids. *J. Comput. Phys.*, 187(1):255–273, 2003.

[MDB99] N. Moës, J. Dolbow, and T. Belytschko. A finite element method for crack growth without remeshing. *Int. J. Num. Meth. Eng.*, 46:131–150, 1999.

[Mes94] Message Passing Interface Forum. MPI: A Message-Passing Interface Standard. *International Journal of Supercomputer Applications*, 8(3/4):165–414, 1994.

[MGCR07] E. Marchandise, P. Geuzaine, N. Chevaugeon, and J.-F. Remacle. A stabilized finite element method using a discontinuous level set approach for the computation of bubble dynamics. *J. Comput. Phys.*, 225(1):949–974, 2007.

[MM00] J. Mosler and G. Meschke. 3D FE analysis of cracks by means of the strong discontinuity approach. In *Proceedings of European Congress on Computational Methods in Applied Sciences and Engineering, ECCOMAS*, Barcelona, 2000.

[MPI] Message passing interface (MPI) for distributed memory parallelization. `http://www.mpi-forum.org`.

[MR06] E. Marchandise and J.-F. Remacle. A stabilized finite element method using a discontinuous level set approach for solving two

phase incompressible flows. *J. Comput. Phys.*, 219(2):780–800, 2006.

[NW99] J. Nocedal and S. J. Wright. *Numerical Optimization*. Springer Series in Operations Research. Springer, Berlin, Heidelberg, New York, 1999.

[OF01] S. Osher and R. P. Fedkiw. Level set methods: An overview and some recent results. *J. Comput. Phys.*, 169:463–502, 2001.

[Ope] OpenMP, a specification for shared memory parallelization. `http://www.openmp.org`.

[OPR06] M. A. Olshanskii, J. Peters, and A. Reusken. Uniform preconditioners for a parameter dependent saddle point problem with application to generalized Stokes interface equations. *Numer. Math.*, 105(252):159–191, 2006.

[OR04] M. A. Olshanskii and A. Reusken. Grad-div stabilization for Stokes equations. *Math. Comp.*, 73:1699–1718, 2004.

[OR06] M. A. Olshanskii and A. Reusken. Analysis of a Stokes interface problem. *Numer. Math.*, 103:129–149, 2006.

[ORG08] M. A. Olshanskii, A. Reusken, and J. Grande. An Eulerian finite element method for elliptic equations on moving surfaces. *SIAM J. Numer. Anal.*, 2008. Submitted.

[OS88] S. Osher and J. A. Sethian. Fronts propagating with curvature dependent speed: algorithms based on Hamilton-Jacobi formulations. *J. Comput. Phys.*, 79:12–49, 1988.

[OV07] M. A. Olshanskii and Y. V. Vassilevski. Pressure Schur complement preconditioners for the discrete Oseen problem. *SIAM J. Sci. Comput.*, 29(6):2686–2704, 2007.

[Para] ParaView, parallel visualization application. `http://www.paraview.org`.

[Parb] ParMETIS, parallel graph partitioning package. `http://glaros.dtc.umn.edu/gkhome/views/metis`.

[Pri06] R. Prignitz. Numerische Behandlung der Levelsetgleichung für Zweiphasenströmung. Diploma thesis (in german), IGPM, RWTH Aachen University, 2006.

[PRR05] J. Peters, V. Reichelt, and A. Reusken. Fast iterative solvers for discrete Stokes equations. *SIAM J. Sci. Comput.*, 27(2):646–666, 2005.

[PS01] S. B. Pillapakkam and P. Singh. A level-set method for computing solutions to viscoelastic two-phase flow. *J. Comput. Phys.*, 174:552–578, 2001.

[PSVW05] S. P. van der Pijl, A. Segal, C. Vuik, and P. Wesseling. A mass-conserving level-set method for modelling of multi-phase flows. *Int. J. Num. Meth. Fluids*, 47:339–361, 2005.

[QV94] A. Quarteroni and A. Valli. *Numerical Approximation of Partial Differential Equations*, volume 23 of *Springer Series in Computational Mathematics*. Springer-Verlag, Berlin, Heidelberg, 1994.

[Ran04] R. Rannacher. Incompressible viscous flows. In E. Stein, R. de Borst, and T. J. R. Hughes, editors, *Encyclopedia of Computational Mechanics*, volume 3, chapter 6. Wiley, 2004.

[Reu08] A. Reusken. Analysis of an extended pressure finite element space for two-phase incompressible flows. *Comp. Vis. Sci.*, 2008. In press.

[RFG+] A. Reusken, O. Fortmeier, J. Grande, S. Groß, M. Larin, and H. Nguyen. *DROPS package, User's guide.* IGPM, RWTH Aachen University.

[RST96] H.-G. Roos, M. Stynes, and L. Tobiska. *Numerical Methods for Singularly Perturbed Differential Equations — Convection-Diffusion and Flow Problems*, volume 24 of *Springer Series in Computational Mathematics*. Springer-Verlag, Berlin, 1996.

[Rud97] M. Rudman. Volume-tracking methods for interfacial flow calculations. *Int. J. Num. Meth. Fluids*, 24:671–691, 1997.

[Saa03] Y. Saad. *Iterative Methods for Sparse Linear Systems.* SIAM, Philadelphia, PA, USA, 2003.

[Scr60] L. E. Scriven. Dynamics of a fluid interface. Equations of motion for Newtonian surface fluids. *Chem. Eng. Sci.*, 12:98–108, 1960.

[Set96a] J. A. Sethian. A fast marching level set method for monotonically advancing fronts. *Proc. Natl. Acad. Sci. USA*, 93:1591–1595, 1996.

[Set96b] J. A. Sethian. Theory, algorithms, and applications of level set methods for propagating interfaces. *Acta Numerica*, 5:309–395, 1996.

[Set99] J. A. Sethian. *Level Set Methods and Fast Marching Methods.* Cambridge University Press, 1999.

[SF99] M. Sussman and E. Fatemi. An efficient interface preserving level set re-distancing algorithm and its application to interfacial incompressible fluid flow. *SIAM J. Sci. Comput.*, 20(4):1165–1191, 1999.

[SFB] Collaborative research center SFB 540, RWTH Aachen University, funded by the German Research Foundation (DFG). `http://www.sfb540.rwth-aachen.de`.

[SMDK06] A. Schagen, M. Modigell, G. Dietze, and R. Kneer. Simultaneous measurement of local film thickness and temperature distribution in wavy liquid films using a luminescence technique. *Int. J. Heat Mass Tran.*, 49(25-26):5049–5061, 2006.

[Smo01] A. Smolianski. *Numerical Modeling of Two-Fluid Interfacial Flows.* Phd thesis, University of Jyvaskyla, 2001.

[Smo05] A. Smolianski. Finite-element/level-set/operator-splitting (FELSOS) approach for computing two-fluid unsteady flows with free moving interfaces. *Int. J. Num. Meth. Fluids*, 48(3):231–269, 2005.

[SSO94] M. Sussman, P. Smereka, and S. Osher. A level set approach for computing solutions to incompressible two-phase flow. *J. Comput. Phys.*, 114:146–159, 1994.

[Sus03] M. Sussman. A second order coupled level set and volume-of-fluid method for computing growth and collapse of vapor bubbles. *J. Comput. Phys.*, 187:110–136, 2003.

[TBE+01] G. Tryggvason, B. Bunner, A. Esmaeeli, D. Juric, N. Al-Rawahi, W. Tauber, J. Han, S. Nas, and Y.-J. Jan. A front-tracking method for the computations of multiphase flow. *J. Comput. Phys.*, 169:708–759, 2001.

[TBM92] T. E. Tezduyar, M. Behr, and S. Mittal. A new strategy for finite element calculations involving moving boundaries and interfaces — the deforming-spatial-domain/space-time procedure: II. computation of free-surface flows, two-liquid flows, and flows

with drifting cylinders. *Comput. Methods Appl. Mech. Engrg.*, 94:353–371, 1992.

[TE00] A.-K. Tornberg and B. Engquist. A finite element based level-set method for multiphase flow applications. *Comput. Vis. Sci.*, 3:93–101, 2000.

[Tec] Tecplot, Tecplot Inc., 2D/3D visualization tool. `http://www.tecplot.com/`.

[Tez07] T. E. Tezduyar. Finite elements in fluids: Stabilized formulations and moving boundaries and interfaces. *Computers & Fluids*, 36(2):191–206, 2007.

[Tor00] A.-K. Tornberg. *Interface tracking methods with application to multiphase flows.* Phd thesis, Royal Institute of Technology, Department of Numerical Analysis and Computing Science, Stockholm, 2000.

[Tor02] A.-K. Tornberg. Multi-dimensional quadrature of singular and discontinuous functions. *BIT*, 42(3):644–669, 2002.

[TSaM+05] Ch. Terboven, A. Spiegel, D. an Mey, S. Groß, and V. Reichelt. Experiences with the OpenMP parallelization of DROPS, a Navier-Stokes solver written in C++. In *Proceedings of the first international workshop on OpenMP (IWOMP 2005)*, 2005.

[Tur99] S. Turek. *Efficient Solvers for Incompressible Flow Problems: An Algorithmic and Computational Approach*, volume 6 of *Lecture Notes in Computational Science and Engineering.* Springer, Berlin, Heidelberg, 1999.

[UG] UG, software tool for the numerical solution of partial differential equations on unstructured meshes in 2D/3D. `http://sit.iwr.uni-heidelberg.de/~ug/`.

[UT92] S. O. Unverdi and G. Tryggvason. A front-tracking method for viscous, incompressible multi-fluid flows. *J. Comput. Phys.*, 100:25–37, 1992.

[Ver96] R. Verfürth. *A Review of A-Posteriori Error Estimation and Adaptive Mesh-Refinement Techniques.* Advances in Numerical Mathematics. Wiley-Teubner, New York, 1996.

[WKP99] M. W. Williams, D. B. Kothe, and E. G. Puckett. Convergence and accuracy of continuum surface tension models. In W. Shyy

and R. Narayanan, editors, *Fluid Dynamics at Interface*, pages 294–305. Cambridge University Press, Cambridge, 1999.

[ZALC05] X. Zheng, A. Anderson, J. Lowengrub, and V. Cristini. Adaptive unstructured volume remeshing II: Application to two- and three-dimensional level-set simulations of multiphase flow. *J. Comput. Phys.*, 208:191–220, 2005.

Curriculum Vitae

Persönliche Daten

Name	Sven Groß
Adresse	Harscampstr. 73 52062 Aachen
E-Mail	gross@igpm.rwth-aachen.de
Geburtsdatum/-ort	07.05.1976 in Georgsmarienhütte
Familienstand	verheiratet, 1 Kind
Staatsangehörigkeit	deutsch

Schulbildung

1982 – 1983	Gemeinschaftsgrundschule Tonbrennerstraße in Aachen
1983 – 1986	Gemeinschaftsgrundschule Saarstraße in Aachen
1986 – 1995	Viktoriagymnasium in Aachen
6/1995	Abitur

Zivildienst

8/1995 – 9/1996	Zivildienst als Integrationshelfer

Studium

10/1996 – 9/2002	Mathematikstudium mit Nebenfach Informatik an der RWTH Aachen University
WS 97/98 – SS 02	Tätigkeit als studentische Hilfskraft am Lehrstuhl D für Mathematik und am Institut für Geometrie und Praktische Mathematik, RWTH Aachen University
9/1998	Vordiplom mit Note „sehr gut“
9/2002	Diplom mit Auszeichnung Thema: Parallelisierung eines adaptiven Verfahrens zur numerischen Lösung partieller Differentialgleichungen

Beruflicher Werdegang

seit 10/2002	Wissenschaftlicher Mitarbeiter am Lehrstuhl für Numerische Mathematik an der RWTH Aachen University
seit 10/2002	Mitglied des Sonderforschungsbereiches SFB 540 an der RWTH Aachen University
seit 06/2007	assoziiertes Mitglied der Graduiertenschule AICES an der RWTH Aachen University