Discrete Projection Methods for Time-Dependent Viscous Incompressible Fluid Flow Simulations in Arbitrary Domains

Inaugural-Dissertation

zur

Erlangung des Doktorgrades der

Mathematisch-Naturwissenschaftlichen Fakultät

der Heinrich-Heine-Universität Düsseldorf

vorgelegt von

Jörg-Matthias Sautter

aus Tübingen

Düsseldorf, Juni 2004

Gedruckt mit der Genehmigung der Mathematisch-Naturwissenschaftlichen Fakultät der Heinrich-Heine-Universität Düsseldorf

Referentin: Professor Dr. Marlis Hochbruck
Korreferent: Professor Dr. Kristian Witsch
Tag der mündlichen Prüfung: 25. Juni 2004

Discrete Projection Methods for Time-Dependent Viscous Incompressible Fluid Flow Simulations in Arbitrary Domains

by

Jörg-Matthias Sautter

A thesis submitted to the
Department of Mathematics and Sciences
of the University of Düsseldorf, Germany
Directed by
Professor Dr. Marlis Hochbruck

Düsseldorf, June 2004

Acknowledgements

Since this is the part of my thesis which is most likely to be read by non-mathemeticians and mathematicians alike, I would like to take the opportunity to say a few words about those people who have somehow helped me in their very own way during my studies and while I was working on this thesis.

I would like to thank my advisor, Professor Dr. Marlis Hochbruck[1], for her continuous openness, her trust, the opportunities to participate in and develop different projects, the wonderful cooperation in many different aspects, and especially for the large amount of freedom and autonomy which was always given to me for my work. And I would like to thank Professor Dr. Kristian Witsch[1] for being on my committee and for reading this thesis.

In addition, my sincere thanks goes to Professor Dr. Donald St. Mary[2], Professor Dr. Nathaniel Whitaker[2], and Professor Dr. Steve F. McCormick[3], who conveyed to me the joy of mathematics through their own enthusiasm. It would never have come to this dissertation without them. I would also like to thank Marion Frey, Damaris Schneider, and especially Jacquelyn R. Deal for reading parts of this thesis and polishing my English.

A special thanks goes to my colleagues, especially to Berthold Nöckel, for continually helping me solve various intricate Linux problems, for endless discussions about math (anything from junior-high-level math to open mathematical questions), for countless coffee breaks (which, of course, weren't always actual breaks, but continuations of these math discussions), and for conversations about the world in general.

The wonderful years I spent at UMass Amherst and at CU Boulder have, in retrospect, been the most important to me and I am not sure whether I would have succeeded in writing this thesis without having had these incredible experiences. I have ever since been grateful for the hospitality I received from all the faculty and my friends there. As mentioned above, the joy of mathematics conveyed to me especially by Nate and Steve motivated me tremendously for and during all the years that followed. I should not forget to mention and thank Kris Gaida, Jacquelyn R. Deal, and Dr. Thaddeus Pace who, with their friendship,

[1] University of Düsseldorf
[2] University of Massachusetts at Amherst
[3] University of Colorado at Boulder

made my time in Amherst and Boulder unforgettable. It was Jackie who opened my eyes by showing me how all this math stuff comes across to non-mathematicians. I also owe thanks to my colleagues Paul Mullowney, Ben Bergen, and Dr. Luke Olson for digging me out of an avalanche in Rocky Mountain National Park in 2000.

As already mentioned above, I am very grateful to Marlis for the large amount of freedom and autonomy which was always given to me for my work. It was extraordinarily interesting and at the same time fun to explore the unbelievably huge field of fluid dynamics and its computational aspects. Following the footprints of others or just concentrating on one single aspect would have been a more direct way, but certainly not the way I wanted to go.

At the same time, Marlis gave me the opportunity to participate in several projects which diversified, alternated, and supplemented the work on my thesis. The projects on how chemical ingredients of hair dye affect hair, modeling the process of a detergent dissolving in a liquid, or the project on computer tomography showed me that the gap between industry and universities can be bridged. An issue sadly neglected at German universities, but not so much in the United States.

Altogether, I can say that working on this thesis has not only been a wonderful experience for me but also a lot of fun; interrupted, however, by numerous doubtful moments wondering how to make everything work. In 2003, I met an American at Camp I near the Polish glacier of Aconcagua. He was just weeks away from getting his PhD in civil engineering and couldn't stand it anymore, took a long break, and left for Argentina. Spending the evening with him and his friends up there, I wondered what the last weeks or months before having finished this thesis would feel like to me. I am glad that everything worked out very well and the progress after long evenings, late nights, and extensive weekends of work mostly gave me new enthusiasm and energy for the next stages that were ahead of me.

After having finished this work, I am in the fortunate position that I can finally show my true friend Alex what triangles are good for. With a smile on his face, he has always asked "Wia goat's Deine Dreieckla?"[4] and I'm not sure whether my answers on why triangles can somehow be important could ever convince him. Thank you for the truly long friendship, Alex! I am also grateful to my true friend Rainer and his wife Eve for their countless invitations and their wonderful hospitality on my rides from Düsseldorf to Pfullingen and back. How can I ever properly express my gratitude?

Finally, I apologize to all those I also should have mentioned explicitly but forgot to do so. And last, but not least, I would like to thank my parents for their support during my studies, and especially my brother, Björn, for his true friendship!

[4] "How are your triangles doing?" ☺

Contents

Introduction

The subject of the flow of fluids, and particularly of water, fascinates everybody. We can all remember, as children, playing in the bathtub or in mud puddles with the strange stuff. As we get older, we watch streams and waterfalls [...] and we are fascinated by this substance which seems almost alive relative to solids. The behaviour of fluids is in many ways very unexpected and interesting. [...] The efforts of a child trying to dam a small stream flowing in the street and his surprise at the strange way the water works its way out has its analog in our attempts over the years to understand the flow of fluids.

Richard P. Feynman[5]

Figure 1: A moving "obstacle" within incompressible fluid flow or an example of (real) fluid-structure interaction (cf. chapters 4 and 5)

[5] [33] on page 40-1

The Navier-Stokes equations for incompressible isothermal flow of a Newtonian fluid are at the core of fluid dynamics. In their original form, derived by Navier in 1822 and rederived by Stokes in 1845, they read

$$\frac{\partial u}{\partial t} + (u \cdot \nabla)u = \nu \Delta u - \nabla p \tag{1a}$$

$$\nabla \cdot u = 0 \tag{1b}$$

in Ω for $t \in [0, t_{\text{final}}]$ where u is the fluid velocity, p the normalized pressure, and ν the kinematic viscosity (see section 1.1.2). Provided that correct initial data and true physical boundary conditions are given, they describe a wide range of hydrodynamical fluid flow problems accurately as long as the continuum assumption is satisfied, which is the case in all relevant simulations occurring in practice. In addition, the Navier-Stokes equations (1) also play a major role in aerodynamics. This is due to the fact that the flow of a compressible fluid exhibits incompressible behavior at Mach numbers below approximately $M = 0.3$ (cf. [54]) and is therefore governed by (1) to a very high degree of accuracy. In other words, we can have (nearly) incompressible flow of a compressible fluid. This explains the importance of the incompressible Navier-Stokes equations in aerodynamics and, of course, also in hydrodynamics.

Since analytical solutions are only available in very special cases, numerical algorithms are essential for computing fluid flow simulations of complex, real-life applications with sufficient accuracy and efficiency. This is known as computational fluid dynamics (CFD), which belongs to the field of scientific computing.

Computational fluid dynamics numerically models a wide variety of fluid flow phenomena. Simulation of fluid flows is of particular interest in mechanical engineering where numerical simulations replace costly and time-consuming experiments with prototypes. In some cases, simulation is the only way to obtain insight into a system's behavior, either because experiments are impossible, as is the case for climate modeling, or because measurement instruments cannot be installed, as could be the case with a combustion chamber. Computational fluid dynamics simulation has therefore become an important tool in the design and development of a wide range of products.

Scientific computing is an interdisciplinary field for solving real-life problems from science and technology on a computer. It is the interface between (real-life) applications, computer science, and mathematics (cf. figure 2 and [106]). The notion *numerical simulation* plays a decisive role in scientific computing. Following the methodology of scientific computing, obtaining the numerical solution for the simulation of a real-life problem can be divided into four stages. First, a precise formulation of the problem in terms of equations, boundary conditions, initial condition and solution domain is needed. Next, the domain is usually divided up into elements or cells to form a discrete grid on which the numerical scheme operates. The third stage is the actual solution procedure, and in the final step,

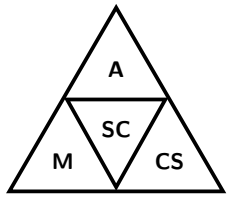


Figure 2: **S**cientific **C**omputing and its connections to **A**pplications, **C**omputer **S**cience, and **M**athematics

any post-processing, such as visualizing the results, is performed.

In order to solve the Navier-Stokes equations numerically, they must be discretized in space and time. Historically speaking, this was done for the first time in the 1950s with the finite difference method. The implementation is rather straightforward, especially for simple geometries. However, for more complicated geometries, the implementation becomes unwieldy and tedious very quickly. A decade later, another method for the discretization in space became popular. The finite element method is by far more suited for complex geometries than the finite difference method. It requires much more complex data structures, however, and higher order approximations are more difficult to achieve. Another method with similar geometrical flexibility as the finite element method is the finite volume method. The most often stated advantage of the latter is that linear conservation laws implied by the governing partial differential equations are always and inherently satisfied locally (at control level) and thus globally (via summation). On the other hand, properties of the original differential operator do not, in general, transfer to the finite volume discretized operator. The pros and cons concerning the two methods are discussed in [48], for example. Last but not least, spectral methods became popular in the 1970s. Due to global basis functions, these methods are suitable for high accuracy computations. In contrast, spectral methods are generally not practical for complex geometries and are inferior to the methods mentioned before.

From the large variety of existing time discretization schemes, fully coupled implicit methods, continuous projection methods, and their discrete counterparts are three classes of most common solution schemes. All have their pros and cons. The coupled approach yields the best stability behavior but also entails the largest numerical effort. Projection methods decouple velocity and pressure and reduce the problem to a sequence of simpler problems.

There are many publications on how to solve the Navier-Stokes equations in rather special cases (e.g., see [19, 59, 60, 65, 74, 87, 121], just to name a few). However, there is currently no software package available that can handle any kind of fluid flow problem in arbitrary domains. The difficulty arises from the huge field of different flow phenomena

(such as, e.g., dependence on the Reynolds number) and the numerical problems arising from simulations in complex domains, e.g. transient domains with coupled boundary conditions. This work is an attempt to clarify the issue how to simulate a large class of physical fluid flow problems and how to employ the appropriate mathematical tools.

This thesis presents new numerical simulation techniques for incompressible fluid flow problems offering advantages for obtaining solutions to a wide variety of time-dependent two- or three-dimensional viscous incompressible fluid dynamics problems. The techniques are tailored to solve

$$\begin{aligned} \varrho\left(\frac{\partial u}{\partial t}+(u\cdot\nabla)u\right) &= \nabla\cdot\left(-pI_{d\times d}+\eta(\nabla u+(\nabla u)^T)\right)+b \qquad && \text{in } \Omega \text{ for } t\in[0,t_{\text{final}}] \\ \nabla\cdot u &= 0 && \text{in } \Omega \text{ for } t\in[0,t_{\text{final}}] \end{aligned}$$

for a given body force b, viscosity η, prescribed boundary values on $\partial\Omega$, and initial data.

In particular, the presented techniques allow, in contrast to other existing solvers, simulations of Newtonian and non-Newtonian incompressible fluid flow problems with or without pressure modes in stationary and transient domains, possibly coupled with other physical effects such as fluid-structure interaction (see figure 1 for a real fluid-structure interaction example). This thesis describes the basic methodology, finite element spatial discretization on possibly time-dependent meshes that are moved with the fluid, new time integration schemes, and numerical comparisons of their efficiencies and applicability. In addition, illustrations are included from a number of representative calculations. The projection schemes may be regarded as extensions of the discrete versions (cf. [120, 121]) of the classical projection schemes of Chorin (cf. [14]) and Van Kan (cf. [73]) and variants of those. It is known (see for example [120]) that the projection approach requires only moderately smaller time steps than fully implicitly coupled schemes, but that the work to obtain comparative results with discrete projection methods as solvers is much lower.

The main contributions of this work are the clarification and unification of different formulations of the governing equations and their different weak forms and corresponding physically relevant boundary conditions, their applicability, the implementation of the different forms via a Lagrange multiplier ansatz, the extension of classical projection methods to discrete multistep projection methods and discrete one-step projection methods, the efficient solution of nonlinear and linear subproblems in the projection methods with algebraic multigrid, and the extension of the simulation techniques to free boundary and fluid-structure interaction problems. In particular, the efficiency of the methods or their inherent subproblems and their extensibility to multiphysics problems, such as fluid-structure interaction or free boundary problems, is emphasized rather than theoretical properties of the methods. This work is to be understood as a contribution and an attempt towards a solver for fluid flow problems in non-trivial domains for a much larger class of fluid flow problems compared to existing solvers. The programming has been done in Matlab® 5.3 and 6.5 and some routines from Femlab® 2.3 have been used.

This thesis is organized as follows: The physics of fluids and appropriate corresponding mathematical formulations are presented in chapter 1. It is described how the mathematical modeling is strongly influenced by the underlying physical principles. Since any modeling error spoils the results of the numerical simulation, it is particularly emphasized how to obtain adequate mathematical models for the simulation of the physical phenomena and explained which model should be used in certain situations. In chapter 2, the spatial discretization of the incompressible Navier-Stokes equations in arbitrary domains via the finite element method with Lagrange multipliers is described. In chapter 3, solution strategies for the time discretization are presented. The discrete projection methods are based on multistep methods and one-step methods and can be applied to problems with or without pressure modes. The efficient solution of the nonlinear and linear subproblems occurring in the velocity and pressure-correction step with algebraic multigrid is described. In chapter 4, the simulation techniques are extended to transient domains coupled with other physical phenomena, such as fluid-structure interaction problems and free boundary problems. Finally, chapter 5 shows the applicability and performance of the methods or their inherent subproblems in two- and three dimensional simulations.

Chapter 1

The Continuum Equations

1.1 Physics of fluids

Fluid mechanics is the branch of physical sciences concerned with how fluids behave at rest or in motion. Fluid mechanics examines the behavior of liquids, gases, and – in the case of very high energies – also plasmas. We have to understand fluid mechanics if we want to model the red spot on Jupiter, measure the velocity in a tornado, design an airfoil, or predict the behavior of subatomic particles in a betatron, to name but a few examples (see also [47, 85]). Fluid mechanics plays a significant role in literally dozens of fields within science and engineering: in meteorology, oceanography, and astronomy; for aerodynamics propulsion, and combustion; for biofluids, in acoustics, and in particle physics.

The study of fluid mechanics is subdivided into statics and dynamics which in turn are divided into incompressible and compressible flow (cf. table 1.1). Incompressible and compressible flow are divided into real and ideal. Real is divided into laminar and turbulent. And so on.

	hydrostatics	aerostatics	hydrodynamics	aerodynamics
pressure	✓	✓	✓	✓
density		✓		✓
velocity			✓	✓
examples	resting liquids	resting gas	moving liquid	moving gas

Table 1.1: The fields of fluid mechanics

The foundational axioms of fluid dynamics are the conservation laws, specifically, conservation of mass, conservation of linear momentum, conservation of angular momentum, conservation of energy, and the second law of thermodynamics (see also [103] and references therein for details).

The central equations for fluid dynamics are the Navier-Stokes equations, which are nonlinear partial differential equations describing fluid flow. In general, there are no closed-form solutions to the Navier-Stokes equations. The equations can be simplified in a number of ways. All simplifications make the equations easier in some sense, although not easy to solve in most cases. Some of them allow appropriate fluid dynamics problems to be solved in closed form. In order not to obtain only qualitative but also quantitative results, computational fluid dynamics is the essential tool.

1.1.1 Classification of fluid flow

Fluids are a subset of the phases of matter and encompass liquids and gases and in the case of very high energies also plasmas. They share the properties of not resisting deformation and the ability to flow. In other words, the main property that distinguishes a fluid from a solid is that a fluid cannot maintain shear stress for any length of time. If shear is applied to a fluid, it will move under the shear.

This thesis considers fluids – be it liquid or gas – which are single phase (cf. [85]), isothermal, purely viscous (cf. [54]), incompressible, and homogeneous, i.e. they exhibit constant density in space as well as in time due to their incompressibility. Each separate classification is important. This includes the large classes of Newtonian and non-Newtonian fluids. The concepts of viscosity and incompressibility which play a central role in this work will be explained briefly in the following. Furthermore, we suppose that the fluid may be regarded as a "continuum" of matter (see also [4, 23, 53, 103]). The dynamics of such a fluid, i.e. the fluid flow, is supposed to be laminar and instationary and is subject of the numerical simulation techniques in the upcoming chapters. Of course, fluid flows whose properties match those mentioned above sufficiently well may also be simulated with these numerical techniques.

Purely viscous fluids

Viscosity is perhaps the single most important property of fluid dynamics (see also [47]). This property of a fluid manifests as a resistance to flow and is measured by the tangential force per unit area, τ (a.k.a. shear stress), of either of two horizontal planes at distance δ apart, one of the planes moving with velocity v_δ relative to the other, the space between being occupied by the flowing fluid (cf. figure 1.1). This is due to the fact that the velocity of a fluid at a surface always equals the surface's velocity (cf. [33]).

For Newtonian fluids, the shear stress τ is proportionally related to the rate of change of the shear strain $\frac{\partial u_1}{\partial x_2}$, i.e.

$$\tau = \frac{F}{A} = \eta \frac{v_\delta}{\delta} = \eta \frac{\partial u_1}{\partial x_2} \tag{1.1}$$

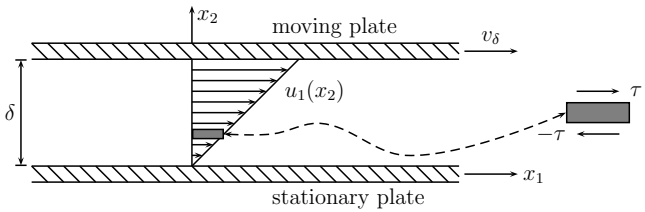


Figure 1.1: Viscous drag between two parallel plates (plate viscometer)

where η is called the coefficient of dynamic viscosity, or more commonly, the viscosity. It is constant for a fixed temperature and pressure and is independent of the velocity or the rate of strain. Viscosity diminishes as temperature rises, often by about 2% per degree Celsius; it also increases with an increase in pressure (cf. [47, 54]). The unit of the viscosity is the Pascal second. The kinematic viscosity is defined as

$$\nu = \frac{\eta}{\varrho}. \tag{1.2}$$

Fluids for which the linear, instantaneous relation (1.1) between the stress and the rates of strain (or the velocity gradients) does not exist are referred to as non-Newtonian fluids. These fluids may have extremely varied properties and the study of their characteristic response to a given stress is subject of the science of rheology (cf. [54]). Non-Newtonian fluids are divided into three main categories, namely purely viscous fluids, time-dependent fluids, and viscoelastic fluids which again all divide into several subcategories (see [54, 47] for example). Purely viscous fluids can be modeled by (1.1) where η depends, e.g. on the velocity or the rate of strain (see also chapter 5 for an example).

Incompressible fluid flow

Incompressible fluids are fluids whose density is independent of the pressure. Whether to use compressible or incompressible fluid dynamics depends on the Mach number of the problem. As a rough guide, compressible effects can be ignored at Mach numbers below approximately $M = 0.3$. Nearly all problems involving liquids are in this regime and modeled as incompressible (cf. [54]). Even gases are often modeled as incompressible, although gases and liquids have quite different characteristics from a thermodynamic point of view. This is justified due to the fact that the pattern of the flow of air can be similar to that of water (see also section 1.2). However, *incompressible fluid* is a thermodynamic term whereas *incompressible flow* is a fluid mechanical term. We can have (nearly) incompressible flow of a compressible fluid. This explains the importance of the incompressible Navier-Stokes equations.

In the remainder of this work the focus point will be on the flow of a fluid which possesses inertia and viscosity but which is effectively incompressible. This program might appear to be modest, but it lies at the center of fluid mechanics (see also [49] and references therein).

1.1.2 Fluid dynamics: Then and now

The first mathematical description of the motion of an ideal fluid was formulated by Euler in 1755 (cf. [29]) as a statement of Newton's second law of motion applied to a fluid moving under an internal force known as the pressure gradient. The Euler equations are important theoretically, but omit the effects of friction. To incorporate friction, in 1822 Navier derived (cf. [86]) the equations (1) of motion for a viscous fluid in which he included the effects of attraction and repulsion between neighboring molecules while at the same time assuming the continuum hypothesis. For Navier, the parameter ν was simply a function of the molecular spacing to which he attached no particular physical significance (cf. [11]). Finally, in 1845, Stokes rederived the viscous equations (1) and, unlike Navier, made it clear that the parameter ν has an important physical meaning: namely the viscosity, known today as kinematic viscosity, not to be confused with dynamic viscosity.

Although the Navier-Stokes equations have been known for a long time, the issue of existence and uniqueness of solutions is still an open problem and the concept of "blow-up" for the Navier-Stokes equations has recently received considerable publicity in the context of one of the million dollar prize problems offered by the Clay Mathematics Institute (see also [11, 31] and section 1.2.2).

In the early simulation methods for viscous incompressible flow, vorticity and stream function were the calculated variables, and in the late sixties, simulations in terms of primitive variables, i.e. velocity and pressure, began. Pioneering work in this direction was performed by Harlow and Welch [58] who introduced the staggered grid. Projection methods were introduced through Chorin [14] and Temam [112, 113] within the finite difference regime. In the late seventies, spectral methods became popular. They are suitable for high accuracy computations, but only in very simple domains. In the eighties, considerable interest was evinced for the techniques for handling flows in arbitrary shaped geometries. On the one hand, methods for transforming complex geometries into simple ones have been proposed (a discussion can be found, e.g., in [114]) and on the other hand, automatic grid generation for arbitrary domains has been a major research area since. This was the launch pad for finite element or finite volume based algorithms for the simulation of realistic flow problems. It is nowadays a research area evoking more interest than ever (see also chapters 2 and 3).

1.2 The Navier-Stokes equations

This section is meant to be a very brief introduction to the Navier-Stokes equations for single phase, isothermal, purely viscous, and homogeneous fluids. It is by no means complete. A rigorous derivation of the Navier-Stokes equations can be found in [53, 103] and for the physics of hydrodynamics see [54], for example.

As mentioned in the previous section, fluids are substances which cannot resist gravitational forces at rest, such as gases, liquids, honey, and even glaciers or glass, just to name a few. The goal is to describe the motion of a fluid in a two- or three-dimensional region $\bar{\Omega} = \Omega \cup \partial\Omega$ within some time interval $[0, t_{\text{final}}]$. Let $x \in \bar{\Omega}$ be a point and consider a *particle of fluid*[1] moving through x at time t. When the model of particles of fluid is applicable, the fluid can be treated as a continuous medium. Now, imagine a particle in the fluid. Think about a particle of dust suspended in the fluid. This particle traverses a well-defined trajectory (cf. figure 1.2). Following the motion of this particle corresponds to a Lagrangian point of view. In the following, an Eulerian point of view will be taken. Then, the motion of the fluid is completely determined by the quantities velocity, pressure, and density,

$$\begin{aligned} u : \Omega \times [0, t_{\text{final}}] &\rightarrow \mathbb{R}^d \\ p : \Omega \times [0, t_{\text{final}}] &\rightarrow \mathbb{R} \\ \varrho : \Omega \times [0, t_{\text{final}}] &\rightarrow \mathbb{R} \end{aligned}$$

if we will assume that the temperature of the fluid is constant in space and time, i.e. the fluid is isothermal.

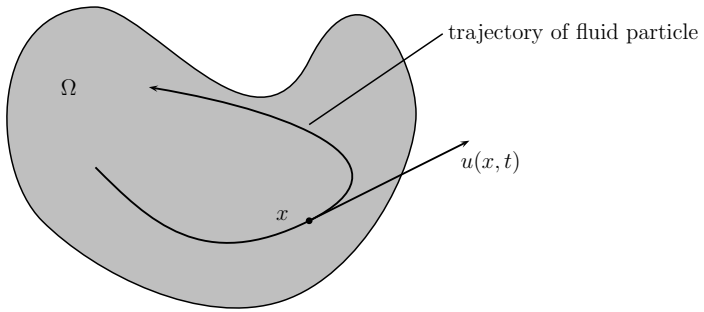


Figure 1.2: Fluid particle moving in a region $\bar{\Omega}$

[1] A particle of fluid must not be confused with the molecule (or atom) that makes up the fluid.

Conservation of momentum

Let $W \subset \Omega$ and let $W_t = \{x(y,t) \,|\, y \in W\}$ be the volume transported by the flow. Linear momentum $P_{W_t}(t)$ of W_t at time t and the force $F_{W_t}(t)$ acting upon W_t are defined as

$$P_W(t) = \int_{W_t} \varrho(x,t)u(x,t)\,dV \quad \text{and} \quad F_W(t) = \int_{\partial W_t} s(n,x,t)\,dA + \int_{W_t} b\,dV$$

where $s = s(n(x,t),x,t)$ is a surface force density and

$$b : \Omega \times [0, t_{\text{final}}] \to \mathbb{R}^d$$

is a given function called the body force density representing body forces, such as gravity acting upon the fluid, for example. It can be shown that the principle of linear momentum under weak assumptions requires the surface force density to be simple. The stress principle is put to use through Cauchy's fundamental lemma: there is a tensor $\sigma(x,t)$, called the Cauchy stress tensor, so that

$$s(n,x,t) = \sigma(x,t)n.$$

That is, the traction s does not depend arbitrarily on the normal n; it is in fact a linear homogeneous function of it (cf. [103, 117]). It is shown in [103] that applying Newton's law

$$\dot{P}_{W_t}(t) = F_{W_t}(t)$$

yields

$$\int_{W_t} \varrho \left(\frac{\partial u}{\partial t} + (u \cdot \nabla)u \right) dV = \int_{W_t} \nabla \cdot \sigma(x,t)\,dV + \int_{W_t} b\,dV \qquad \forall W_t \subset \Omega$$

and hence

$$\varrho \left(\frac{\partial u}{\partial t} + (u \cdot \nabla)u \right) = \nabla \cdot \sigma + b$$

with

$$\sigma = -pI_{d\times d} + \sigma_{\text{viscous}}.$$

The viscous part of the stress tensor is determined by two constants η and λ which are called the first and second coefficients of viscosity, respectively.

$$\sigma_{\text{viscous}} = \eta(\nabla u + (\nabla u)^T) + \lambda(\nabla \cdot u)I_{d\times d}. \tag{1.3}$$

Both can be measured in an experiment. Roughly speaking, the first coefficient of viscosity, a.k.a. dynamic viscosity, expresses what is generally understood by viscosity, i.e. the slow movement of a fluid due to internal friction, while the second coefficient of viscosity is related to the speed of sound in the fluid and thus related to the incompressibility of the fluid (see also [54]). Therefore, the equation for conservation for momentum is

$$\varrho \left(\frac{\partial u}{\partial t} + (u \cdot \nabla)u \right) = \nabla \cdot (-pI_{d\times d} + \eta(\nabla u + (\nabla u)^T) + \nabla(\lambda(\nabla \cdot u)) + b. \tag{1.4}$$

Conservation of mass

Let $W \subset \Omega$ and assume that for every t the fluid has a well-defined mass density distribution $\varrho(x,t) : f(\bar{\Omega}) \to \mathbb{R}_+$ for any deformation f of $\bar{\Omega}$. Then the mass of fluid in W at time t is given by

$$m(W_t) = \int_{W_t} \varrho(x,t)\,dV.$$

The assumption that ϱ exists is a *continuum assumption*. It does not hold any longer if one considers the molecular structure of matter. However, for most *macroscopic phenomena* occurring in nature, it is believed that this assumption is extremely accurate. It is shown in [103] that conservation of mass, i.e.

$$\frac{d}{dt} m(W_t) = 0$$

for all $W \subset \Omega$ and for all $t \in [0, t_{\text{final}}]$, is equivalent to local conservation of mass,

$$\frac{\partial \varrho}{\partial t} + \nabla \cdot (\varrho u) = 0. \tag{1.5}$$

This is called the differential form of the law of conservation of mass. It is known as the equation of continuity[2].

Compressible Navier-Stokes equations

(1.4) and (1.5) resemble $d+1$ equations for $d+2$ unknowns u, p, and ϱ. The last equation required is the so-called equation of state, $p = \varphi(\varrho)$, which relates pressure and density. While conservation of momentum and conservation of mass can be derived mathematically from physical principles, the equation of state must be determined experimentally. Equations (1.4) and (1.5) and the equation of state yield the Navier-Stokes equations for an isothermal compressible fluid,

$$\begin{aligned} \varrho \left(\frac{\partial u}{\partial t} + (u \cdot \nabla)u \right) &= \nabla \cdot (-pI_{d\times d} + \eta(\nabla u + (\nabla u)^T) + \nabla(\lambda(\nabla \cdot u)) + b \\ \frac{\partial \varrho}{\partial t} + \nabla \cdot (\varrho u) &= 0 \\ p &= \varphi(\varrho). \end{aligned}$$

[2] Why is (1.5) called the continuity equation? We quote [4], 'It has been called the equation of continuity for many years, although not for any evident reason', and then from [90], 'The equation [...] has been called the continuity equation to emphasize that the continuum assumptions (the assumption that density and velocity may be defined at every point in space) are prerequisite.' But he goes on to say, properly, 'The continuum assumption is, of course, a foundation for all the basic laws.' Thus, we side with [4]. See also [48, 103].

Incompressible Navier-Stokes equations

For incompressible homogeneous fluids, i.e. $\varrho = \varrho_0 =$ constant in space and time, the equation of continuity (1.5) simplifies to $\nabla \cdot u = 0$ and hence the viscous part of the stress tensor simplifies to $\sigma_{\text{viscous}} = \eta(\nabla u + (\nabla u)^T)$. This yields the incompressible Navier-Stokes equations in total stress form[3],

$$\varrho\left(\frac{\partial u}{\partial t} + (u \cdot \nabla)u\right) = \nabla \cdot \sigma + b = \nabla \cdot (-pI_{d\times d} + \eta(\nabla u + (\nabla u)^T) + b \qquad \text{in } \Omega \tag{1.6a}$$

$$\nabla \cdot u = 0 \qquad \text{in } \Omega. \tag{1.6b}$$

What makes this form of the incompressible Navier-Stokes equations so attractive is the fact that boundary conditions at an outflow or free boundary can be implemented as natural boundary conditions. It should be noted that the total stress form also allows for varying viscosity, i.e. it is also suited to model flow of non-Newtonian purely viscous fluids.

Except for the essential non-linearity of the advection term $(u \cdot \nabla)u$ and the implicit presence of the velocity-pressure coupling through $\nabla \cdot u = 0$, the Navier-Stokes equations share many of the features of advection-diffusion equations (cf. [49]). While the additional term ∇p appears to be a relevant physical quantity – a force per unit volume, i.e. acceleration in Newton's second law – it is actually that plus a bit more; p also plays the role of a Lagrange multiplier for the incompressibility constraint by adjusting itself instantaneously in time (related to the infinite speed of sound in an incompressible medium) and everywhere in space so that $\nabla \cdot u = 0$ everywhere (including the boundary) and for all time.

Actually, it is the combination of nonlinearity and the pressure-velocity coupling that makes the Navier-Stokes equations difficult (if not impossible, in general) to solve. If either is absent, the equations are much simpler and are known to have solutions (cf. [38, 39, 40]) – the limiting cases being Stokes flow and the so-called Burgers equation, respectively.

Alternative forms

There are many equivalent ways to represent both the viscous terms and the advection terms in the Navier-Stokes equations, many of which are based on the ever-present constraint equation $\nabla \cdot u = 0$. These alternative representations, all of which are equivalent in the continuum, lead to semi-discrete (continuous time, discrete space) equations that are generally not equivalent – and which sometimes offer advantages over the conventional form of the momentum equation presented above. The above form (1.6) is not the most common form, although it is the most general form. We will briefly derive some alternative forms of the incompressible Navier-Stokes equations, including their most common form.

[3] a.k.a. stress-divergence form

For a Newtonian fluid, i.e. a fluid of which the viscosity is constant, the viscosity term can be reduced in the following way.

Proposition 1.1 *Let $u := \mathbb{R}^d \to \mathbb{R}^d$ for $d = 2$ or $d = 3$. The divergence of the viscous stress tensor* (1.3) *simplifies to*

$$\nabla \cdot \sigma_{\text{viscous}} = \nabla \cdot (\eta(\nabla u + (\nabla u)^T)) \tag{1.7a}$$

$$= \eta(\Delta u + \nabla(\nabla \cdot u)) \tag{1.7b}$$

$$= \eta \Delta u \tag{1.7c}$$

if the viscosity η is constant in space and $\nabla \cdot u = 0$.

The proof is just elementary calculus. Applying proposition 1.1 to (1.6) and dividing by ϱ yields the most common form of all velocity-pressure forms,

$$\begin{aligned} \frac{\partial u}{\partial t} + (u \cdot \nabla)u &= -\nabla p' + \nu \Delta u + b' \\ \nabla \cdot u &= 0 \end{aligned} \tag{1.8}$$

with the normalized pressure $p' = \frac{p}{\varrho}$, the kinematic viscosity ν as defined in (1.2), and the normalized body force density $b' = \frac{b}{\varrho}$. For the sake of simple notation, p is typically used instead of p' and b instead of b'. In contrast to the total stress form, the form (1.8) is not suitable for non-Newtonian fluid flow and boundary conditions at an outflow or free boundary are no longer natural boundary conditions. In fact, it leads to non-physical outflow boundary conditions in conjunction with the finite element Galerkin method.

In contrast to many other publications (e.g. [8, 59, 65, 87, 120, 121]), we prefer (1.6) to the simpler conventional form (1.8). The reasons are – as mentioned above – related to the weak forms, natural boundary conditions, and non-Newtonian fluids. Only the total stress form leads to natural boundary conditions that represent true physical forces (see also [48]).

Besides modifying the viscous terms, also the advection term may be represented in an alternative way. The divergence form (of advection) is given by

$$\varrho\left(\frac{\partial u}{\partial t} + \nabla \cdot (uu^T)\right) = -\nabla p + \eta \Delta u$$

since $\nabla \cdot (vu^T) = (u \cdot \nabla)v$. If this divergence form is combined with the total stress form, the total divergence form

$$\varrho \frac{\partial u}{\partial t} = \nabla \cdot (\sigma - uu^T)$$

is obtained, which offers the following advantages when discretized via the Galerkin finite element method: it leads to the proper global momentum balance (cf. [48]), and its natural boundary conditions can be used to implement boundary conditions of total momentum flux, $n \cdot (\varrho uu^T - \sigma)$, if desired at a boundary.

Concomitant pressure field

Although initial conditions of the Navier-Stokes equations need only to be set for the velocity, some numerical methods do require an initial value for the pressure. Since every velocity field is accompanied by a pressure field, it is important to calculate the concomitant pressure field. Taking the divergence of (1.6a) yields the first form of the pressure Poisson equation,

$$\Delta p = \nabla \cdot \left(\nabla \cdot (\eta(\nabla u + (\nabla u)^T)) + b - (u \cdot \nabla)u - \frac{\partial u}{\partial t} \right). \tag{1.9}$$

Proposition 1.2 *Let $u := \mathbb{R}^d \to \mathbb{R}^d$ be sufficiently smooth and $d = 2$ or $d = 3$. Then*

$$\Delta u = \nabla(\nabla \cdot u) - \nabla \times \nabla \times u$$

and

$$\nabla \cdot (\nabla \times \nabla \times u) = 0$$

where the operators are defined as in appendix C.

The proof is again elementary calculus. Note that only for $d = 3$ "$\nabla\times$" is the ordinary curl operator. Invoking propositions 1.1 and 1.2, and equation (1.6b) to (1.9) yields the simplified pressure Poisson equation

$$\Delta p = \nabla \cdot (b - (u \cdot \nabla)u).$$

The pressure Poisson equation which works best numerically, however, is that in which the seemingly zero viscous term is retained:

$$\Delta p = \nabla \cdot \big(\nabla \cdot (\eta(\nabla u + (\nabla u)^T)) + b - (u \cdot \nabla)u \big) \tag{1.10}$$

This equation is called the consistent pressure Poisson equation. Alternatively,

$$\Delta p = \nabla \cdot (\eta \Delta u + g - (u \cdot \nabla)u)$$

with the assumptions of proposition 1.1 and all the disadvantages mentioned for the most common velocity-pressure form.

Although there are no boundary conditions for the pressure in the Navier-Stokes equations, we now need boundary conditions for the pressure in order to solve the pressure Poisson equation. Multiplying the momentum equation (1.6a) by the unit normal vector and invoking proposition 1.1 yields

$$\frac{\partial p}{\partial n} = n \cdot \left(\eta \Delta u + b - \varrho \frac{\partial u}{\partial t} - \varrho (u \cdot \nabla)u \right) \tag{1.11}$$

if $n \cdot u$ is specified on the boundary, i.e. on Γ^D, (see also [48, 93]). Γ^D denotes that part of the boundary where Dirichlet boundary conditions are specified and Γ^N that part, where Neumann boundary conditions are given. If the normal velocity is not specified on the boundary, i.e. on Γ^N, but a traction consition is given (see section 1.2.1), the boundary condition for (1.10) is a Dirichlet condition,

$$p = \eta n \cdot (\nabla u + (\nabla u)^T)n - n \cdot f$$

on Γ^N. This implies that there is no hydrostatic pressure mode present (see also sections 1.2.1 and 3.1.2).

There are many other equivalent ways to represent the incompressible Navier-Stokes equations, just as there are other forms of writing the advection and diffusion term in terms of velocity and pressure. Additionaly, there are also other forms such as the velocity-vorticity formulation or the stream function-vorticity formulation, just to name two of them. For an exemplary overview see [48]. An evolution equation for the pressure may be obtained by differentiating the pressure Poisson equation in time and using the equation of motion to eliminate the time derivatives of the velocity in favor of the velocity and pressure. The result is a Poisson equation for $\frac{\partial p}{\partial t}$ to be solved subject to Neumann boundary conditions that arise by differentiating equation (1.11) in time and interchanging the order of the normal spatial derivative and temporal derivative on the left-hand side. According to [96], this method appears to be untested in practice.

Similar flows

It should be noted that for any (numerical) simulation, the dimension of the problem is crucial. Two flows are called similar if they have similar geometries (modulo some scaling) and the same Reynolds number,

$$Re := \frac{L_c U_c}{\nu} = \frac{\varrho L_c U_c}{\eta},$$

where L_c and U_c denote a characteristic length and velocity of the flow. Solutions of the Navier-Stokes equations do not only depend on initial and boundary values or the viscosity, but also on the Reynolds number. Although, say, air and water are fluids with different characteristics, their flows can be quite similar, if they have a similar Reynolds number.

For low Reynolds number flow, such as fluid flow in micro systems($L_c U_c$ small), a.k.a. microfluidics, or flow with high viscosity (ν large), such as in artery flow simulations, discrete Navier-Stokes equations become stiff and hence influence the performance of the solvers.

1.2.1 Initial boundary value problem

In order to compute the motion of an incompressible homogeneous fluid, the Navier-Stokes equations (1.6) need to be equipped with boundary conditions and an initial value. In the first part of this section, boundary conditions required for the upcoming simulations will be described. Requirements for initial values will be addressed in the second part.

Boundary conditions

The issue of boundary conditions for the Navier-Stokes equations is even larger than that of how to write the Navier-Stokes equations themselves. Therefore, only boundary conditions which will be required for numerical simulations in the following chapters are described.

For every component u_α of the velocity, the boundary of the domain is split into two parts, $\partial\Omega = \Gamma_\alpha^D \cup \Gamma_\alpha^N$, where Γ_α^D denotes that part of $\partial\Omega$ where u_α is constrained by Dirichlet boundary conditions and Γ_α^N denotes the part where u_α suffices a Neumann boundary condition. If $\Gamma_\alpha^D = \Gamma_\beta^D$ and $\Gamma_\alpha^N = \Gamma_\beta^N$ $\forall \alpha, \beta \in \{1, \dots, d\}$, we define $\Gamma^D := \Gamma_\alpha^D$ and $\Gamma^N := \Gamma_\alpha^N$.

Particularly important is the following partitioning of the boundary of the domain: Let Γ_{in} and Γ_{out} represent those parts of the boundary where fluid flows into the domain according to some prescribed velocity profile and where fluid flows out of the domain depending on some force acting upon the fluid at the outlet. And let Γ_{solid} and Γ_{fb} be those parts of the boundary where the fluid's relative velocity to the wall is zero and where the fluid's boundary is free, i.e. depending on a surface tension acting upon the fluid on this part of the boundary. We then have $\Gamma^N = \Gamma_{\text{out}} \cup \Gamma_{\text{fb}}$ and $\Gamma^D = \Gamma_{\text{in}} \cup \Gamma_{\text{solid}}$ and, of course, $\partial\Omega = \Gamma^N \cup \Gamma^D$. The boundary conditions on these parts of the domain are described in the following.

No-slip condition: The fact that fluid cannot penetrate into a solid requires that the components of the velocity normal to the boundary surface should be equal for the fluid and the solid. On the other hand, the viscosity stresses prevent any slipping of the fluid relative to the solid surface. Therefore, the correct boundary condition for solid boundaries is the no-slip condition:

$$u = u_{\text{D}} \qquad \text{on } \Gamma_{\text{solid}} \subset \Gamma^D.$$

If $\partial\Omega = \Gamma^D$, then it is also required that

$$\int_{\partial\Omega} n \cdot u_{\text{D}} = 0$$

since the fluid is incompressible, i.e.

$$0 = \int_\Omega \nabla \cdot u \, dV = \int_{\partial\Omega} n \cdot u_D \, dA$$

by the divergence theorem. In this case, p is determined only up to an arbitrary additive constant (called the hydrostatic pressure). Therefore, the restriction

$$\int_\Omega p \, dA = 0 \qquad \text{for } t \in [t_0, t_{\text{final}}]$$

is imposed.

Slip condition: For macroscale simulations, the fluid at a fluid-wall interface obeys a no-slip condition. It is known, however, that the classical no-slip condition is violated for some complex fluids including polymers, elastomers, and suspensions (cf. [42]). The slip boundary condition for a fluid is given by

$$\begin{aligned} u \cdot n &= 0 \\ \tau^T \left(\eta(\nabla u + (\nabla u)^T - p I_{d\times d}) \right) n &= 0. \end{aligned}$$

The first equation is a no-penetration condition and the second equation reflects the fact that there is no tangential force exerted upon the fluid by the wall, i.e. the fluid is allowed to slip along the wall. At boundaries where the fluid is subject to slip boundary conditions, no boundary layers emerge. However, slip boundary conditions are more often applied to fluid dynamics problems in literature than they actually should since simple Newtonian fluids obey a no-slip condition on a solid wall and therefore generate boundary layers. Nevertheless, slip conditions, although unphysical in many cases, are convenient since they allow for simpler grids due to the absence of boundary layers.

Inflow condition: In principle, these boundary conditions are also no-slip boundary conditions. In contrast to the no-penetration condition above, the normal part of the inflow velocity is non-zero, in general.

$$u = u_{\text{in}} \qquad \text{on } \Gamma_{\text{in}} \subset \Gamma^D$$

Traction condition: In addition to specified velocity, another boundary condition that is used in some branches of fluid mechanics is a force (per unit area) balance boundary condition. This is referred to as specified traction and reads

$$\sigma n = (\eta(\nabla u + (\nabla u)^T) - p I_{d\times d}) n = f_{\text{traction}} \qquad \text{on } \Gamma^N$$

where f_{traction} is the applied force (traction) on the boundary. This type of boundary condition is the reason why we will prefer (1.6) to the other forms of the Navier-Stokes equations described in this section. Only the total stress form leads to natural boundary conditions that represent true (physical) forces.

The two most important cases of the traction condition are outflow boundary conditions, $f_{\text{traction}} = f_{\text{out}}$, and free boundary conditions, $f_{\text{traction}} = f_{\text{st}}$. In the case of outflow boundary conditions, f_{traction} is simply the force acting upon the fluid at the outlet. In the case of a free boundary, the applied force on the boundary is the surface tension which is proportional to the curvature of the boundary, i.e.

$$f_{\text{traction}} = f_{\text{st}} = \kappa \left(\frac{1}{r_1} + \frac{1}{r_2} \right) n \qquad \text{on } \Gamma_f$$

where r_1 and r_2 are the boundary's radii of curvature (see also chapter 4). In two dimensions, i.e. $d = 2$, we set $r_2 = \infty$. κ is a constant depending on the fluid (see also [1, 50]).

There is an ongoing discussion in literature about the correct outflow boundary conditions for a simulation of fluid flow. The outflow boundary conditions above model true physical forces and are therefore correct. Nevertheless, it might be difficult or even impossible to find the true physical force at the outflow boundary. This is a crucial point regarding the quantitative accuracy of any numerical simulation in fluid dynamics involving an outflow boundary.

In addition to the boundary conditions mentioned above, there are many others. For other outflow boundary conditions see [19, 20, 120, 121], for example. Straight-out conditions, neutral conditions, etc., are not considered in this work.

Inflow and no-slip boundary conditions are also referred to as constraints, Dirichlet, or essential boundary conditions. All traction boundary conditions are implemented as generalized Neumann boundary conditions (see section 2.2.2).

Initial data

For the velocity-pressure formulation (1.6), initial data for the velocity is all that is needed. The initial velocity field u_0 is supposed to be divergence-free and comply with essential boundary conditions (cf. [48]), i.e.

$$\begin{aligned} \nabla \cdot u_0 &= 0 && \text{in } \Omega \\ n \cdot u_0 &= n \cdot u_D && \text{on } \Gamma^D. \end{aligned}$$

An initial boundary value problem

The following initial boundary value problem for incompressible flow is subject to the upcoming numerical techniques and simulations if not stated otherwise.

$$\varrho \left(\frac{\partial u}{\partial t} + (u \cdot \nabla) u \right) = \nabla \cdot \left(-p I_{d\times d} + \eta(\nabla u + (\nabla u)^T) \right) + b \quad \text{in } \Omega \text{ for } t \in [0, t_{\text{final}}] \tag{1.12a}$$

$$\nabla \cdot u = 0 \quad \text{in } \Omega \text{ for } t \in [0, t_{\text{final}}] \tag{1.12b}$$

$$u = u_{\text{in}} \quad \text{on } \Gamma_{\text{in}} \subset \Gamma^D \tag{1.12c}$$

$$u = u_D \quad \text{on } \Gamma_{\text{solid}} \subset \Gamma^D \tag{1.12d}$$

$$f_{\text{st}} = (\eta(\nabla u + (\nabla u)^T) - p I_{d\times d}) n \quad \text{on } \Gamma_{\text{fb}} \subset \Gamma^N \tag{1.12e}$$

$$f_{\text{out}} = (\eta(\nabla u + (\nabla u)^T) - p I_{d\times d}) n \quad \text{on } \Gamma_{\text{out}} \subset \Gamma^N \tag{1.12f}$$

$$u = u_0 \quad \text{in } \Omega \text{ at } t = 0 \tag{1.12g}$$

$$\nabla \cdot u_0 = 0 \quad \text{in } \Omega \tag{1.12h}$$

$$n \cdot u_0 = n \cdot u_D \quad \text{on } \Gamma^D \tag{1.12i}$$

It is important to note that the Navier-Stokes equations require no a priori boundary conditions on the pressure. Velocity boundary conditions applied to the momentum equations are sufficient to allow the determination of both velocity and pressure.

As mentioned earlier, solutions of this initial boundary value problem do not only depend on initial data and boundary values or the viscosity, but also on the Reynolds number. In viscous fluids governed by the Navier-Stokes equations, the solution may show singularities when the domain contains vertices, the boundary values are not continuous, or the type of boundary condition changes between Dirichlet and Neumann.

1.2.2 Existence and uniqueness

The Navier-Stokes equations (1.8) without body forces in two dimensions with Dirichlet boundary conditions possess a unique solution (up to an additive constant for the pressure) for all $t > 0$. In three dimensions, uniqueness of the solution can only be shown for some interval $[0, t_{\text{final}}]$ (cf. [80]).

The effects of nonlinearity in the fluid equations are strikingly different in two dimensions and three dimensions. In fact, existence and uniqueness of regular solutions for all time for the two-dimensional Navier-Stokes equations are classical results by Leray 1933, [83], whereas the analog in three dimensions is a Clay prize problem (cf. [31]). Kato, [75], has proven the existence of a global unique regular solution in three dimensions under the restrictive assumption of small initial data.

One crucial difference between two and three dimensions is the constraint that equations

(1.8) impose in two dimensions on the evolution of vorticity, $\omega = \nabla \times u$,

$$\frac{D\omega}{Dt} = \frac{\partial \omega}{\partial t} + (u \cdot \nabla)\omega = (\omega \cdot \nabla)u + \nu \Delta \omega. \tag{1.13}$$

In two dimensions, the vorticity is a scalar field multiplied by a unit vector perpendicular to the two dimensional plane of motion. Hence, the term $(\omega \cdot \nabla)u$ vanishes in two dimensions, and although (1.13) remains nonlinear, it is simpler than the three dimensional equation. Thus, in two dimensions, the vorticity is a scalar quantity that is conserved along the trajectories of the fluid particles (cf. [11, 21]).

According to Fefferman (cf. [31]), "Fluids are important and hard to understand. There are many fascinating problems and conjectures about the behavior of the Euler and Navier-Stokes equations. Since we don't even know whether these solutions exist, our understanding is at a very primitive level. Standard methods from PDE appear inadequate to settle the problem. Instead, we probably need some deep, new ideas."

The importance of computational fluid dynamics is mainly due to the fact that in general the solution cannot be found analytically. Only numerical approximations are possible (assuming that there is a solution at all); in particular for all real-life problems (cf. [50]). Therefore, the development of accurate and efficient numerical algorithms is crucial!

1.3 Modeling fluid flow

1.3.1 Other model equations

In principle, the Navier-Stokes equations describe a wide range of flows in continuum mechanics: the motion of incompressible or compressible, Newtonian or non-Newtonian flows are governed by the Navier-Stokes equations. However, it is so far hardly possible to compute instationary flows with acceptable accuracy in an acceptable amount of time when simulating real-life phenomena. The usual approach is to use approximations either at the modeling level or at the algorithmic level or at both levels.

There is a large number of simpler approximations to specific flow problems (cf. [88]). These equations of motion are usually obtained by modification of the Navier-Stokes equations. For example, the Stokes equations describe flow at very low Reynolds numbers, such that inertial forces can be neglected when compared to viscous forces. The standard equations of inviscid flow are the Euler equations. Very often, especially in computational fluid dynamics, the Euler equations are used to model flow far from the body and the boundary layer equations (cf. [104]) for flow close to the body.

Turbulent flows are typically modeled by the Reynolds equations or the κ-ε-model or large-eddy simulations (see also [50, 88, 96]).

Other simplified equations are the Boussinesq approximation, which neglects compressibility except to calculate buoyancy forces, the Hele-Shaw equation (see also [124, 125]), the shallow water equation (cf. [67]), and the potential equation (cf. [89]), to name only a few. Examples and visualizations of such special fluid flows which can be modeled by the above equations can be found in [27].

1.3.2 Computational fluid dynamics

Simplifications at the modeling level need to be handled with great care. Any error introduced by such a simplification will be inherent to the whole simulation, regardless of the numerical methods applied.

Even nowadays, numerical simulation of flow for the design of aircrafts is typically not computed by entirely solving the full Navier-Stokes equations but by solving different equations (from the ones mentioned above) in different regions of the flow (cf. [71]). Usually, the Euler equations or the potential equation are used to simulate the free flow sufficiently far away from the aircraft. The flow near the wings and body is simulated by solving the boundary layer equations. However, boundary layer separation can only be simulated sufficiently accurate by solving the Navier-Stokes equations (cf. [54, 89]).

Solving the Navier-Stokes equations sufficiently accurate is therefore of great importance in many fields of science and engineering. The need for efficient numerical algorithms is as important as the need for increasing computer capacity.

From the perspective of hydrodynamics, CFD complements fluid dynamics theory and experiments (cf. figure 1.3). From the perspective of applications, CFD is the only tool which yields reliable and quantitative predictions of fluid flow.

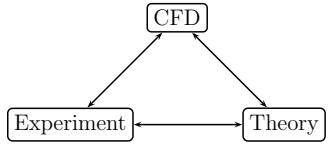
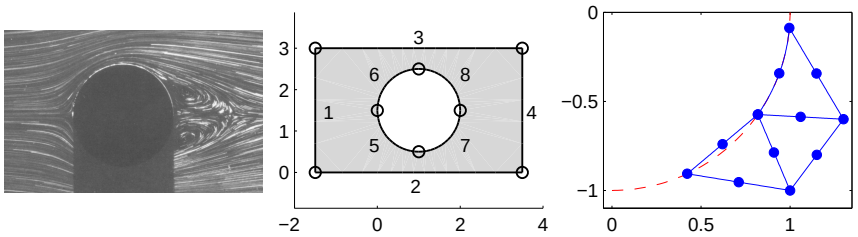


Figure 1.3: CFD complements fluid dynamics theory and experiments

1.4 Notational introduction

From a mathematical point of view, the area occupied by a fluid is referred to as the spatial domain, or just domain, which may be stationary or transient. In general, this is only an approximation to the area occupied by the fluid in reality.

When performing a simulation, the physical domain, i.e. the area occupied by the real fluid, is represented (or possibly sometimes approximated) by some kind of geometry model via boundary modeling. The geometry model is then approximated by a (mathematical) spatial domain, e.g., via finite element triangularization, possibly using isoparametric elements. Figure 1.4 shows this approximation process, where a three-dimensional flow is modeled in two dimensions using isoparametric and affine finite elements. The expressions spatial domain, geometry and geometry model will be used equivalently, if there is no danger of confusion.



(a) 3D physical domain (b) 2D geometry model (c) Section of mathematical domain near boundary element 7

Figure 1.4: Approximation of the spatial domain

The spatial dimension is denoted by d with $d = 2$ or $d = 3$. In some cases it is also denoted by `sdim`. Space coordinates are denoted by x_α where $\alpha = 1, 2, \ldots, d$. Several continuous quantities[4] and discrete quantities will be needed to describe a fluid flow numerically, most notably, velocity and pressure and their discrete analogons. For discrete quantities, we distinguish between constrained and eliminated solution vectors. Notation of typical fluid dynamics quantities is as shown in table 1.2, where the α-index serves as a spatial index.

	discrete		continuum
	constrained	eliminated	
velocity	u_h	u_e	u
velocity in x_α-direction	$u_{\alpha h}$	$u_{\alpha e}$	u_α
pressure	p_h	p_e	p
solution vector	y_h	y_e	y

Table 1.2: Notation of major discrete and continuous variables and quantities.

[4] In the sense of the hypothesis of continuity for fluids (see also [15, 47, 81, 103]).

As a rule of thumb: Greek indices are spatial indices, i.e. $\alpha = 1, \ldots, d$, for example. Latin indices are discretization indices or some other kind of indices, i.e. $i = 1, \ldots, n$ for some $n \in \mathbb{N}$, for example. Therefore, u_{hi} is the ith component of the discrete velocity vector u_h which is an approximation to the velocity u due to spatial discretization via the Galerkin finite element method. And $u_{\alpha hj}$ is the jth component of the discrete velocity vector $u_{\alpha h}$ which is an approximation to the velocity u_α in x_α-direction. In some cases, if there is no confusion, indices may be omitted.

The number of (constrained) degrees of freedom, i.e. the total number of (constrained) velocity and pressure unknowns is denoted by N_{DOF}. The number of degrees of freedom minus those determined by essential boundary conditions is denoted by N_{DOFe}. A complete list can be found in appendix C.3.

Chapter 2

Spatial Discretization

There are several spatial discretization approaches used for the numerical solution of the incompressible Navier-Stokes equations. Most prominent are the finite difference method, the pseudospectral method, the finite volume method, and the finite element method.

The finite difference method is the oldest method for the numerical solution of partial differential equations, believed to have been introduced by Euler in the 18th century (cf. [32]). It is also the first spatial discretization method used in computational fluid dynamics (see for example [12, 13, 14, 58] or references in [50]). On structured grids, the finite difference method is simple and effective. Accurate, i.e. higher order, approximations can be easily achieved, especially on regular grids (cf. [36]). In principle, the finite difference method can be used for the spatial discretization of complex domains (cf. [51, 50, 68]), however this is cumbersome, especially the correct treatment of Neumann boundary conditions.

The spectral method is best suited for very high spatial accuracy. It is however not really flexible and difficult to apply to complicated domains (cf. [95]), and therefore rarely used for numerical simulations of realistic fluid flow phenomena.

According to [48], the finite volume method has probably been (and still is) the most popular spatial discretization method in computational fluid dynamics. It can accommodate any type of grid, so it is suitable for complex geometries. The method is conservative by construction, i.e. linear conservation laws are inherently satisfied (cf. [122]). According to [32], the finite volume approach is perhaps the simplest to understand and to implement. All terms that need to be approximated have physical meaning, which is why it is so popular with engineers. However, properties of the original differential operators do, in general, not transfer to the finite volume discretized operators.

Another popular discretization scheme in computational fluid dynamics is the (Galerkin) finite element method. It is very well suited for unstructured grids, i.e. complex geometries can be handled easily. Its main advantages are the inherent ability to easily and accurately apply the appropriate physical boundary conditions even on complex domains

– especially the Neumann type, and especially at outflow regions – is a real asset. Global physical linear conservation laws are either satisfied automatically or can be made to do so with a slight change in formulation. If the original differential operator is self-adjoint, so too are the discretizations of the operator via the finite element method. As mentioned above, this is, in general, not true for the finite volume method (see also [48]).

A principle drawback, which is shared by any method that uses unstructured grids, is that the matrices of the linearized equations are not as well structured as those for regular grids, thus making them more difficult to solve. But, as we will see shortly, there is no way to find a remedy if a realistic simulation is the objective.

Numerical solutions of fluid flow problems are only approximate solutions which always include at least three kinds of systematic errors (see also [32]):

- *Modeling errors*, which are defined as the difference between the actual flow and the exact solution of the mathematical model,
- *Discretization errors*, defined as the difference between the exact solution of the continuous equations and the exact solution of the algebraic systems of the equations obtained by discretizing these equations, and
- *Convergence errors*, defined as the difference between the iterative and exact solution of the algebraic equations systems.

It is important to be aware of the existence of these errors, and even more to try to distinguish one from another. Otherwise it will be hardly possible to judge the actual quality of a numerical simulation. Since any modeling error will be inherent to the numerical solution, it is important to minimize the modeling error. Major considerations are directed to the governing equations (cf. chapter 1) and the correct representation of the spatial domain.

Arbitrarily shaped geometries are clearly essential. In principle, it is possible to use finite differences on complicated geometries. It is shown in [50, 115] how to treat curved boundaries and interfaces without disturbing the overall regularity of grids. Nevertheless, by far the most flexible and accurate approach is obtained by utilizing unstructured grids, which advises the use of the finite volume or finite element method. For the reasons mentioned above, we use the Galerkin finite element method with Lagrange multipliers to discretize the incompressible Navier-Stokes equations in space, which is described in the remainder of this chapter.

2.1 Preprocessing

The finite element method is a technique for obtaining approximate solutions of weak forms, i.e. it is a method of discretizing the weak form with the result that the underlying function

spaces become finite dimensional – and thus amenable to representation via computer. The approximate solution is represented as an expansion in a convenient (and finite) set of linearly-independent local basis functions (piecewise-polynomials) with coefficients to be determined. The finite element method comprises approximations at several stages, mainly approximation of the domain, for example by a polygonal or Bézier shaped domain, numerical evaluation of integrals, and the finite element spaces themselves. This section describes the preprocessing required in order to set up the finite element method, i.e. definition of geometry, mesh generation, and the choice of finite element spaces.

2.1.1 Definition of geometry

For simplicity, we suppose that the physical domain $\Omega \in \mathbb{R}^d$ is connected. For a non-connected region, every connected subregion can be treated separately. To define the spatial domain, we use the concept of boundary modeling, which is the process of defining a region in terms of its boundary. Another popular concept, especially in computer aided design, is solid modeling (cf. [19]). However, it turns out that boundary modeling is most general although sometimes cumbersome.

The partitioning of the boundary of a two-dimensional domain is now described (the extension to $d = 3$ is similar). We partition $\partial\Omega$ into N_Γ boundary segments Γ_i, so that

$$\partial\Omega = \bigcup_{i=1}^{N_\Gamma} \Gamma_i$$

and

$$\Gamma_i \subset \Gamma_1^D \cap \Gamma_2^D \quad \text{or} \quad \Gamma_i \subset \Gamma_1^N \cap \Gamma_2^D \quad \text{or} \quad \Gamma_i \subset \Gamma_1^D \cap \Gamma_2^N \quad \text{or} \quad \Gamma_i \subset \Gamma_1^N \cap \Gamma_2^N. \tag{2.1}$$

Γ_α^D denotes that part of the boundary $\partial\Omega$ where the solution u_α is constrained by essential boundary conditions. The part of the boundary where u_α suffices a natural boundary condition is denoted by Γ_α^N (see also section 1.2.1). In particular, this means that the type of boundary conditions for the solution components $u_1, \ldots, u_d$ do not change on a boundary segment. Of course, the boundary conditions themselves, whether of Dirichlet or Neumann type, may (and in general will) depend on space and time.

Each boundary segment Γ_i is then represented by a parameterized curve

$$g_i : [s_{1i}, s_{2i}] \to \mathbb{R}^2, \quad s_{1i}, s_{2i} \in \mathbb{R}.$$

For most applications occurring in practice, a sufficient and practical way of describing the boundary segments is by means of piecewise Bézier curves which has been done for all of the following numerical simulations. Any other curve describing the boundary is also possible, of course. For $d = 3$, (2.1) is extended analogously and each Γ_i is represented by a parameterized surface g_i.

2.1.2 Mesh generation

Triangularization

Let the spatial domain $\Omega \subset \mathbb{R}^d$ be a connected polygonal or polyhedral region as described in the previous section. In practice it is enough if the physical domain can be approximated sufficiently well by a polygonal or polyhedral region Ω. We start by breaking up the domain $\bar{\Omega}$ into a set of N_Δ triangles or tetrahedrons (d-simplices), $\mathcal{T}_h = \{\Delta_i\}_{i=1}^{N_\Delta}$. The partition is assumed to be non-degenerate so that the intersection of two triangles is a point or an edge of the triangles and analogously in three dimensions. In particular, the partition is supposed to be conformal:

Definition 2.1 *A triangular (tetrahedral) decomposition $\mathcal{T}_h = \{\Delta_i\}_{i=1}^{N_\Delta}$ of $\bar{\Omega} \in \mathbb{R}^d$ with N_Δ closed triangles (tetrahedrons) Δ_i is called conformal triangularization[1] (tetrahedronization[1]) if*

1. $\bar{\Omega} = \bigcup_{\Delta_i \in \mathcal{T}_h} \Delta_i$
2. $\mathrm{int}(\Delta_i) \cap \mathrm{int}(\Delta_j) = \emptyset$ *if* $i \neq j$
3. $\Delta_i \cap \Delta_j \neq \emptyset \Longrightarrow \Delta_i \cap \Delta_j$ *is either*
   - *a single point or a joint edge ($d = 2$)*
   - *a single point or a joint edge or a joint side ($d = 3$)*

   of both Δ_i and Δ_j.

In addition, the partition of two triangles is assumed to satisfy a minimum angle condition, i.e. each angle in the triangularization is greater than or equal to some fixed angle θ_0. This implies that the diameters of neighboring triangles are roughly the same and only a fixed number (independent of the diameter) of triangles can meet at any point.

Mesh generation

The simplest finite element tesselations are structured meshes, i.e. meshes with regular distribution of nodes according to some pattern. The pattern may change near the boundary (see also [77] and references therein). Structured meshes permit the efficient solution of algebraic systems because matrices of linearized equations are also structured. For real-life problems with complex geometries, which require automatic meshing, this approach is not useful or not even doable.

[1]Both, triangularization and tetrahedronization, are sometimes referred to as tesselation in literature.

For complicated geometries, unstructured grids need to be employed in order to minimize the spatial discretization error. In this work, Femlab®'s automatic grid generator is used to generate unstructured grids. The grid generator is based on the Delaunay algorithm (cf. [16]). In two dimensions, the Delaunay triangularization of the convex hull $\bar{G}$ of a given set of isolated points excels all other possible triangularizations of $\bar{G}$ (with the same boundary nodes) due to the fact that its minimal interior angle is maximal. In three dimensions, however, this is not true anymore. Another feature of the Delaunay triangularization in two dimensions is that the sum of the two angles opposite a common side of two triangles is bounded by π. This is a requirement for maximum principles for finite element methods. See [41, 77] for details.

The mesh data is stored in a node point matrix

$$\mathtt{P} = (p_1, \ldots, p_{N_N}) \in \mathbb{R}^{d \times N_N}$$

which contains the node points p_i, $i = 1, \ldots, N_N$, of the mesh; a boundary element matrix

$$\mathtt{E} \in \mathbb{R}^{7 \times N_B} \tag{2.2}$$

which contains, among other quantities, indices of the starting and ending node points of every boundary element; and an element matrix

$$\mathtt{T} \in \mathbb{R}^{(d+2) \times N_\Delta} \tag{2.3}$$

which contains indices of the node points defining the vertices of the mesh elements and their subdomain number. The element matrix (2.3) defines a mapping which assigns locally numbered vertices $p_1^{\Delta_i}, \ldots, p_{d+1}^{\Delta_i} \in \mathbb{R}^d$ to the vertices of every d-simplex Δ_i defined by the grid points of the mesh.

Tesselation quality measures

Assessment of mesh quality is an important requirement in the selection of a finite element mesh. It is well known that badly shaped finite elements can lead to inaccurate and unstable approximations. If the mesh's quality is poor, the simulation is spoiled. Typically, a planar triangularization is usually required to satisfy a minimal angle condition (see [6, 107, 108]), although this has been shown in [2] to be too restrictive and can be replaced by a condition which limits the maximal allowable angle. More recently, several measures of element quality have been proposed (cf. [3, 28, 34, 37, 91]) based on the dimensionless rations of various geometric parameters. A comparison of quality measures can be found in [92]. The following quality measures $0 \leq q_\Delta \leq 1$ are used in this work:

$$q_\Delta = \min_{\Delta_i \in \mathcal{T}_h} 4\sqrt{3} \frac{A_\Delta(\Delta_i)}{\sum_{j=1}^{3} h_j^2} \qquad \text{and} \qquad q_\Delta = \min_{\Delta_i \in \mathcal{T}_h} \frac{216}{\sqrt{3}} \frac{V_\Delta(\Delta_i)}{(\sum_{j=1}^{6} h_j^2)^{\frac{3}{2}}}$$

for $d = 2$ and $d = 3$, respectively, where $A_\Delta(\Delta_i)$ and $V_\Delta(\Delta_i)$ denote the area and volume, respectively, and h_j the side lengths of an element Δ_i. Optimal quality is obtained for equilateral elements, i.e. $q_\Delta = 1$, and degenerated elements yield $q_\Delta = 0$. If $q_\Delta > 0.6$ the mesh may be considered to be of acceptable quality. If not, the mesh needs to be smoothed and optimized (see also [19, 77]).

2.1.3 Finite element spaces

Throughout this work, the solution components are approximated by C^0-elements. The resulting matrices are sparse and the accuracy for real-life problems is in general sufficient. C^n-elements yield higher spatial accuracy but at the same time generate less sparse matrices which makes them less favorable for efficient numerical simulations. On the other side of the spectrum, discontinuous elements yield the sparse matrices with the lowest bandwidth but suffer from significant loss of accuracy. Therefore, C^0-elements may be considered as the golden mean for most simulations.

Function spaces

Let $L^2(\Omega)$ denote the space of functions that are square integrable in the Lebesgue sense with respect to the domain $\Omega \subset \mathbb{R}^d$. Let $s \in \mathbb{N}$ and

$$H^s(\Omega) = \{\phi \in L^2(\Omega) \,|\, D^m\phi \in L^2(\Omega), m = 1, \dots, s\}$$

where $D^m\phi$ denotes any and all weak partial derivatives of order m, and let

$$H_0^s(\Omega) = \{\phi \in H^s(\Omega) \mid \phi = 0 \text{ on } \partial\Omega\},$$
$$H_{\alpha,0}^s(\Omega) = \{\phi \in H^s(\Omega) \mid \phi = 0 \text{ on } \Gamma_\alpha^D\},$$
$$H_{\alpha,e}^s(\Omega) = \{\phi \in H^s(\Omega) \mid \phi = u_{\alpha D} \text{ or } \phi = u_{\alpha\text{in}} \text{ on } \Gamma_\alpha^D\}.$$

Using the above definitions, the Sobolev spaces for velocity and pressure are

$$V := H_1^1(\Omega) \times \cdots \times H_d^1(\Omega) \tag{2.4a}$$
$$V_0 := H_{1,0}^1(\Omega) \times \cdots \times H_{d,0}^1(\Omega) \tag{2.4b}$$
$$W := L^2(\Omega). \tag{2.4c}$$

Moreover, let

$$V_e := H_{1,e}^1(\Omega) \times \cdots \times H_{d,e}^1(\Omega). \tag{2.4d}$$

The approximations of these function spaces are all based on the finite dimensional space of piecewise polynomial functions

$$X_k := \{\phi \in L^2(\Omega) \mid \phi|_{\Delta_i} \in \mathcal{P}_k(\Delta_i), \forall\, \Delta_i \in \mathcal{T}_h\} \cap C^0(\Omega) \tag{2.5}$$

where

$$\mathcal{P}_k(\Delta_i) = \{P : \Delta_i \to \mathbb{R} \mid P(x_1, \dots, x_d) = \sum_{|\alpha| \le k} c_\alpha x_1^{\alpha_1} \dots x_d^{\alpha_d}\}.$$

Finite dimensional function spaces

For computations on d-simplices, Cartesian coordinates are somewhat awkward. Barycentric coordinates (a.k.a. local coordinates), $\lambda_1, \dots, \lambda_{d+1} \in \mathbb{R}$ are more natural. For any $x \in \Delta_i$, they are defined in such a way that the following holds:

$$x = \sum_{j=1}^{d+1} \lambda_j p_j^{\Delta_i} \qquad \text{and} \qquad 1 = \sum_{j=1}^{d+1} \lambda_j$$

Interpreting the first d Barycentric coordinates $\lambda_1, \dots, \lambda_d$ of a d-simplex Δ_i as local Cartesian coordinates maps Δ_i onto the reference d-simplex Δ_{ref} (in parameter space). Vice versa, the reference d-simplex is mapped (from parameter space or space in local Cartesian coordinates) to the d-simplex Δ_i in the spatial domain Ω by 0-linear transforms $F_i : \Delta_{\text{ref}} \to \Delta_i$,

$$x = F_i(\lambda_1, \dots, \lambda_d) = J_{\Delta_i} \begin{pmatrix} \lambda_1 \\ \vdots \\ \lambda_d \end{pmatrix} + p_1^{\Delta_i}$$

with $J_{\Delta_i} = (p_2^{\Delta_i} - p_1^{\Delta_i}, \dots, p_{d+1}^{\Delta_i} - p_1^{\Delta_i}) \in \mathbb{R}^{d,d}$. So F_i maps knots and sides onto each other and therefore triangles are mapped to triangles (cf. [77]).

Construction of shape functions

In order to define shape functions for velocity and pressure, the grid is augmented by Lagrange points on each d-simplex of the grid. They are defined as follows:

Definition 2.2 *A d-simplex Δ_i is said to be a Lagrange element of type k, if there are Lagrange points*

$$p_\alpha = \sum_{j=1}^{d+1} \frac{\alpha_j}{|\alpha|} p_j \in \Delta_i$$

for all multi-indices $\alpha = (\alpha_1, \dots, \alpha_{d+1})$ with $|\alpha| = k$ and $\alpha_j \in \mathbb{N}$.

Example 1 *Triangular Lagrange reference elements of type k for $k = 1, 2, 3$ are shown in figure 2.1. Their $s = 1 + 2 + \dots (k + 1)$ Lagrange points are distributed in $k + 1$ rows.*

We are now in the position to specify nodal basis functions defined by $\mathcal{T}_h$ and so-called Lagrange $P_{k(u)}P_{k(p)}$-elements. Lagrange elements of type $k(u)$ and $k(p)$, respectively, are

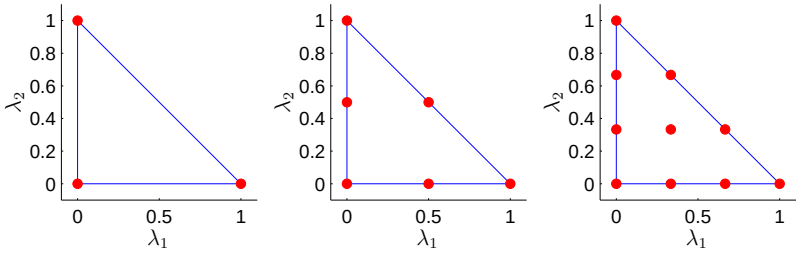


Figure 2.1: Lagrange points on triangular reference elements.

used for the velocity components and the pressure, respectively. The pressure's Lagrange points are denoted by $\widehat{p}_i^p$, $i = 1, \ldots, N_p$ and the velocity Lagrange points by $\widehat{p}_i^u$, $i = 1, \ldots, N_T$. For simplicity of notation, we will skip the superscripts on the Lagrange points if there is no danger of confusion, since they can be distinguished from the domain of the subscript. Nodal shape functions ϕ_i^p for the pressure are then defined by

$$\phi_i^p(\widehat{p}_j^p) = \delta_{ij} \qquad \text{and} \qquad \phi_i^p|_{\Delta_l} \in \mathcal{P}_{k(p)}(\Delta_l)$$

for $i, j = 1, \ldots, N_p$ and $l = 1, \ldots, N_\Delta$. Similarly, nodal shape functions ϕ_i^α for the velocity components u_α, are defined by

$$\phi_i^\alpha(\widehat{p}_j^u) = \delta_{ij} \qquad \text{and} \qquad \phi_i^\alpha|_{\Delta_l} \in \mathcal{P}_{k(u)}(\Delta_l)$$

for $i, j = 1, \ldots, N_T$ and $l = 1, \ldots, N_\Delta$. So $\{\phi_i^p\}_{i=1}^{N_p}$ is a nodal basis for the space X_k as defined in (2.5), and the sets $\{\phi_i^\alpha\}_{i=1}^{N_T}$ are d bases for the space X_{k+1}. We are now in the position to approximate the Sobolev spaces defined at the beginning of this section by spaces of finite dimensions.

The discrete approximations of (2.4) are, in terms of the shape functions, given by

$$V^h = \{v = (v_1, \ldots, v_d)^T \,|\, v_\alpha(x) = \sum_{i=i}^{N_\alpha^T} s_i \phi_i^\alpha(x),\ s_i \in \mathbb{R}\}$$

$$V_0^h = \{v \in V^h \,|\ v_\alpha|_{\Gamma_\alpha^D = 0}\} = \{v = (v_1, \ldots, v_d)^T \,|\, v_\alpha(x) = \sum_{i=i}^{N_\alpha} s_i \phi_i^\alpha(x),\ s_i \in \mathbb{R}\}$$

$$V_e^h = \{v \in V^h \,|\ v_\alpha|_{\Gamma_\alpha^D} = u_{\alpha D} \text{ or } v_\alpha|_{\Gamma_\alpha^D} = u_{\alpha \text{in}}\}$$

$$W^h = \{v \,|\, v(x) = \sum_{i=i}^{N_p} s_i \phi_i^p(x),\ s_i \in \mathbb{R}\}$$

2.2 Weak form with Lagrange multipliers

This section discusses how different weak forms of the incompressible Navier-Stokes equations are actually achieved and how they are approximated with the finite element method in order to obtain the discrete equations. Additionaly, the formulation of the boundary conditions is described. The procedure utilizes a Lagrange-multiplier ansatz, which will allow to impose Dirichlet and Neumann boundary conditions at the same time. The discrete equations are then obtained by first discretizing the partial differential equation in space while neglecting all constraints, i.e. Dirichlet boundary conditions. The constraints will then be discretized separately. In the resulting discrete system, the discrete constraints will be eliminated in a separate elimination process.

Advantages of this approach are that the expensive assembly of the discrete partial differential equation is independent of the constraints (1.12c)–(1.12d). In the case of transient boundary conditions, for example, the time-dependent constraints can be assembled independent of the original partial differential equation.

First, the Lagrange-multiplier ansatz will be described for a general class of advection-diffusion equations. Then, it will be shown how to apply this procedure to the Navier-Stokes equations. We try to stick to a concise notation in order to keep things simple.

2.2.1 A common weak formulation and its approximation

By weak formulation, we mean the sequence of steps needed to go from a given system of partial differential equations and boundary conditions to a specific weak form of them. There are several different weak forms of the Navier-Stokes equations. A weak form of the (not so common) total stress form (1.12a)–(1.12b) will be derived now since it is used most often in the remainder of this work. The actual implementation, however, is done differently and is described in the following section. How to obtain other weak forms and their advantages or disadvantages will also be discussed in the following sections.

The objective is to find $u \in V_e$ and $p \in W$, so that

$$\int_\Omega \frac{\partial u}{\partial t} \cdot v \, dV + \int_\Omega (u \cdot \nabla) u \cdot v \, dV = \int_\Omega (\nabla \cdot (\nu(\nabla u + (\nabla u)^T)) - \nabla p) \cdot v \, dV + \int_\Omega b \cdot v \, dV \qquad \forall v \in V_0$$

$$\int_\Omega (\nabla \cdot u) q \, dV = 0 \qquad \forall q \in W.$$

Applying Green's formula to the weak form of the advection term yields

$$\begin{aligned}\int_\Omega \frac{\partial u}{\partial t}\cdot v\,dV + \int_\Omega (u\cdot\nabla)u\cdot v\,dV &= -\int_\Omega (\nu(\nabla u + (\nabla u)^T)) - Ip) : \nabla v\,dV \\ &\quad + \int_{\Gamma_N} (\nu(\nabla u + (\nabla u)^T) - Ip)nv\,dA + \int_\Omega b\cdot v\,dV \\ &= -\int_\Omega (\nu(\nabla u + (\nabla u)^T)) : \nabla v\,dV - \int_\Omega p\nabla\cdot v\,dV \\ &\quad + \int_{\Gamma_N} (\nu(\nabla u + (\nabla u)^T) - Ip)nv\,dA + \int_\Omega b\cdot v\,dV\end{aligned}$$

$\forall v \in V_0$. The trial functions u, p and test functions v, q are now approximated by $u_h := (u_{1h},\dots,u_{dh})^T \in V_e^h$, $p_h, q_h \in W^h$, and $v_h := (v_{1h},\dots,v_{dh})^T \in V_0^h$

$$\begin{aligned} u_{\alpha h} &= \widehat{u}_\alpha + \sum_{j=1}^{N_\alpha} u_{\alpha j}\phi_j^\alpha \quad \text{with} \quad \widehat{u}_\alpha = \sum_{j=1}^{N_\alpha^D} \widehat{u}_{\alpha j}\phi_j^\alpha \\ p_h &= \sum_{j=1}^{N_p} p_j\phi_j^p \\ v_{\alpha h} &= \sum_{j=1}^{N_\alpha} v_{\alpha j}\phi_j^\alpha \\ q_h &= \sum_{j=1}^{N_p} q_j\phi_j^p \end{aligned}$$

where ϕ_j^α, ϕ_j^p are nodal basis functions defined by $\mathcal{T}_h$ and Lagrange $P_{k(u)}P_{k(p)}$-elements. We have tacitly introduced N_α and N_α^D, the number of not constrained and constrained components of $u_{\alpha h}$:

$$\begin{aligned} N_T &= N_\alpha + N_\alpha^D \\ &= \#\text{ velocity-nodes} \in \Omega \cup \Gamma_\alpha^N + \#\text{ velocity-nodes} \in \Gamma_\alpha^D \end{aligned}$$

where N_T denotes the total number of velocity Lagrange points. The individual integrals are then evaluated numerically. This kind of implementation is the most common one and described in detail in [48], for example. We will, however, make use of a different kind of finite element implementation, namely a finite element method that utilizes a Lagrange-multiplier ansatz, which allows to impose Dirichlet and Neumann boundary conditions at the same time. The advantage of this procedure is that major parts of the assembly, i.e. the assembly of the equation itself, are independent of the boundary conditions which will be imposed in a separate second step.

2.2.2 A Lagrange multiplier ansatz

The incompressible Navier-Stokes equations are part of a fairly general general class of partial differential equations, namely general advection-diffusion equations. Let $\Omega \subset \mathbb{R}^d$ be a connected set and

$$y = y(t, x_1, \dots, x_d) = (y_1(t, x_1, \dots, x_d), \dots, y_N(t, x_1, \dots, x_d))^T$$

be the vector of dependent variables, i.e. an unknown function $y : [0, T_{\text{final}}] \times \Omega \to \mathbb{R}^N$ that is to be determined from the system of partial differential equations

$$\sum_{k=1}^{N} d_{a_{lk}} \frac{\partial y_k}{\partial t} + \nabla\cdot\Gamma_l = F_l \qquad \text{in } \Omega \tag{2.6a}$$

$$-n^T\Gamma_l = G_l + \sum_{m=1}^{N_R} \frac{\partial R_m}{\partial y_l}\mu_m \qquad \text{on } \partial\Omega \tag{2.6b}$$

$$0 = R_m \qquad \text{on } \partial\Omega \tag{2.6c}$$

where $d_{a_{lk}} \in \mathbb{R}$ are the mass coefficients, $\Gamma_l(t, x_1, \dots, x_d, y_1, \dots, y_N) \in \mathbb{R}^d$ the flux vector, and $F_l(t, x_1, \dots, x_d, y_1, \dots, y_N) \in \mathbb{R}$ the source term. $G_l \in \mathbb{R}$ is a boundary source term, $\mu_m \in \mathbb{R}$ a Lagrange multiplier, and $R_m \in \mathbb{R}$ restricts the solution component y_l on the boundary. The equation index l ranges from $1, \dots, N$, while the constraint index m ranges from $1, \dots, N_R$.

For simplicity, we introduce the following more concise notation of the above system of partial differential equations,

$$d_a\frac{\partial y}{\partial t} + \nabla\cdot\Gamma = F \qquad \text{in } \Omega \tag{2.7a}$$

$$-n\cdot\Gamma = G + \left(\frac{\partial R}{\partial y}\right)^T \mu \qquad \text{on } \partial\Omega \tag{2.7b}$$

$$0 = R \qquad \text{on } \partial\Omega \tag{2.7c}$$

where $d_a \in \mathbb{R}^{N\times N}$ is the mass matrix, $\Gamma(t, x, y) \in \mathbb{R}^{N\times d}$ the flux matrix, and $F(t, x, y) \in \mathbb{R}^N$ the source term. $G \in \mathbb{R}^N$ is the boundary source term, $\mu \in \mathbb{R}^{N_R}$ a Lagrange multiplier, and $R \in \mathbb{R}^{N_R}$ restricts the solution y on the boundary.

The second equation is referred to as natural boundary condition and the third equation as essential boundary condition. In fact, the natural boundary condition is also called a (generalized) Neumann boundary condition, or in some contexts also mixed or Robin boundary condition. Essential boundary conditions are also called Dirichlet boundary conditions, which are also referred to as constraints.

For all relevant computational fluid dynamics simulations occurring in practice, the constraints of the solution on the boundary are linear in the unknowns. In particular,

the boundary conditions introduced in section 1.2.1 yield linear restrictions of the solution on the boundary. The interaction of natural boundary conditions and essential boundary conditions is explained in appendix A.2.

2.2.3 The Navier-Stokes equations with Lagrange multipliers

As mentioned in section 2.2.1, the weak formulation of the incompressible Navier-Stokes equations can be done in different ways which result in different weak forms. When using the Lagrange multiplier ansatz (2.7) in order to eventually discretize the Navier-Stokes equations, different weak forms are obtained depending on the definition of flux vector and source term in (2.7a). We now present and briefly discuss some possible definitions of the flux vector and source term which eventually yield the most common weak forms.

Pseudo total stress form

If the mass coefficient, flux vector, and source term are defined as

$$d_a = \begin{pmatrix} \varrho I_d & 0_{d\times 1} \\ 0_{1\times d} & 0 \end{pmatrix} \in \mathbb{R}^{d\times d}, \quad \Gamma = \begin{bmatrix} -\eta\nabla u + pI_d \\ 0_{1\times d} \end{bmatrix}, \quad \text{and} \quad F = \begin{bmatrix} b - \varrho(u\cdot\nabla)u \\ \nabla\cdot u \end{bmatrix} \tag{2.8}$$

we call the resulting weak form the pseudo total stress form. In this case, the flux vector does not represent a true physical force because of the simplification (1.7) (see also [48]). Furthermore, this form applies only in the case of constant density and constant viscosity, i.e. it should not be used for non-Newtonian fluid flow simulations. Nevertheless, this is still the most common form used in fluid dynamics (see, e.g., [44, 59, 73, 87, 120, 121]).

Pseudo viscous stress form

Defining mass coefficient, flux vector, and source term as

$$d_a = \begin{pmatrix} \varrho I_d & 0_{d\times 1} \\ 0_{1\times d} & 0 \end{pmatrix} \in \mathbb{R}^{d\times d}, \quad \Gamma = \begin{bmatrix} -\eta\nabla u \\ 0_{1\times d} \end{bmatrix}, \quad \text{and} \quad F = \begin{bmatrix} b - \varrho(u\cdot\nabla)u - \nabla p \\ \nabla\cdot u \end{bmatrix} \tag{2.9}$$

means that integration by parts is not applied to the pressure gradient. This form leads to non-symmetric systems, as will be seen later, and is therefore not very popular in literature (although suggested in [19], for example). Like the previous form, the pseudo viscous stress form applies only in the case of constant density and constant viscosity, i.e. it should not be used for non-Newtonian fluid flow simulations.

Viscous stress form

Besides Newtonian fluids, also non-Newtonian fluids can be modeled with the so-called viscous stress form,

$$d_a = \begin{pmatrix} \varrho I_d & 0_{d\times 1} \\ 0_{1\times d} & 0 \end{pmatrix} \in \mathbb{R}^{d\times d}, \quad \Gamma = -\eta \begin{bmatrix} \nabla u + (\nabla u)^T \\ 0_{1\times d} \end{bmatrix}, \quad \text{and} \quad F = \begin{bmatrix} b - \varrho(u\cdot\nabla)u - \nabla p \\ \nabla\cdot u \end{bmatrix} \tag{2.10}$$

which is used in Femlab's Navier-Stokes application mode, for example (see also [18, 19, 20]). However, the natural boundary conditions of this form are somewhat awkward for outflow boundary conditions. This form leads to non-symmetric systems as well.

Total stress form

True physical forces are modeled by the so-called viscous stress form:

$$d_a = \begin{pmatrix} \varrho I_d & 0_{d\times 1} \\ 0_{1\times d} & 0 \end{pmatrix} \in \mathbb{R}^{d\times d}, \quad \Gamma = \begin{bmatrix} -\eta(\nabla u + (\nabla u)^T) + pI_d \\ 0_{1\times d} \end{bmatrix}, \text{ and } \quad F = \begin{bmatrix} b - \varrho(u\cdot\nabla)u \\ \nabla\cdot u \end{bmatrix} \tag{2.11}$$

Its advantages are that physical outflow boundary conditions as well as boundary conditions on a free boundary are natural boundary conditions. Newtonian and non-Newtonian fluids are described by this form. Furthermore, this form leads to symmetric systems (see section 2.5).

The weak form of the incompressible Navier-Stokes equations based on the total stress form seems to be superior to the other forms mentioned above. This is in accordance with Gresho and Sani [48], for example. Nevertheless, the pseudo total stress form, despite its disadvantages to the total stress form, is probably the most popular form in computational fluid dynamics. If not stated otherwise, the total stress form will be used in the remainder of this work because of the advantages mentioned above. Details about the implementation in two or three dimensions can be found in appendix B.

If the natural boundary condition of the viscous stress form (2.10) is used for outflow boundary conditions, a divergence-free velocity field near the outflow can no longer be guaranteed. This is because $\eta(\nabla u + (\nabla u)^T)n = 0$ is not a traction outflow boundary condition. If Γ_m represents the outflow boundary segment, adding the constraint $R_m := -p$ yields the correct traction outflow boundary conditions. A comparison of this procedure and the total stress form is shown in chapter 5.

Nonlinear advection

For each of the forms described above, the nonlinear advection term may be computed in different ways. For simplicity, we restrict ourselves to two dimensions here. The advection

term may be computed either, the most common way, as

$$(u \cdot \nabla)u = \begin{pmatrix} u_1 \dfrac{\partial u_1}{\partial x_1} + u_2 \dfrac{\partial u_1}{\partial x_2} \\ u_1 \dfrac{\partial u_2}{\partial x_1} + u_2 \dfrac{\partial u_2}{\partial x_2} \end{pmatrix}$$

or, alternatively, using the identities

$$\begin{aligned} u_1 \frac{\partial u_1}{\partial x_1} + u_2 \frac{\partial u_1}{\partial x_2} &= u_1 \frac{\partial u_1}{\partial x_1} + u_2 \frac{\partial u_1}{\partial x_2} + u_1 \left(\frac{\partial u_1}{\partial x_1} + \frac{\partial u_2}{\partial x_2} \right) \\ &= 2u_1 \frac{\partial u_1}{\partial x_1} + u_2 \frac{\partial u_1}{\partial x_2} + u_1 \frac{\partial u_2}{\partial x_2} \\ &= \frac{\partial (u_1^2)}{\partial x_1} + \frac{\partial (u_1 u_2)}{\partial x_2} \end{aligned}$$

and

$$\begin{aligned} u_1 \frac{\partial u_2}{\partial x_1} + u_2 \frac{\partial u_2}{\partial x_2} &= u_1 \frac{\partial u_2}{\partial x_1} + u_2 \frac{\partial u_2}{\partial x_2} + u_2 \left(\frac{\partial u_1}{\partial x_1} + \frac{\partial u_2}{\partial x_2} \right) \\ &= u_1 \frac{\partial u_2}{\partial x_1} + u_2 \frac{\partial u_1}{\partial x_1} + 2u_2 \frac{\partial u_2}{\partial x_2} \\ &= \frac{\partial (u_1 u_2)}{\partial x_1} + \frac{\partial (u_2^2)}{\partial x_2} \end{aligned}$$

provided that $\nabla \cdot u = 0$. The alternative form of the advection term in three dimensions can be derived simultaneously. Other alternative forms of the nonlinear advection term can be found in [48], for example.

Implementation

The field `fem.nseform` determines the weak form and the form of the advection term used for the finite element discretization of the Navier-Stokes equations. See appendix B for details. The type of boundary conditions, i.e. transient or stationary, is controlled by the boolean field `fem.have.tdBCs`.

2.2.4 The (constrained) weak form

The starting point for the Galerkin finite element method applied to the strong form (2.6) is a problem written in weak form. Let $v \in V \times W$ be an arbitrary test function. Multiplying (2.7a) with this function and integrating we get

$$\int_\Omega v^T d_a \frac{\partial y}{\partial t}\, dV + \int_\Omega v^T \nabla \cdot \Gamma\, dV = \int_\Omega v^T F\, dV \tag{2.12}$$

where dV represents the volume element if $d = 3$ or the area element if $d = 2$. The objective is to find $y \in V \times W$, so that (2.12) holds for all $v \in V \times W$. Then, using Green's formula, i.e. integrating by parts, yields

$$\int_\Omega v^T d_a \frac{\partial y}{\partial t}\, dV + \int_{\partial\Omega} v^T \Gamma \cdot n\, dA - \int_\Omega \nabla v \cdot \Gamma\, dA = \int_\Omega v^T F\, dA \qquad \forall v \in V \times W$$

where dA represents the area element if $d = 3$ or the line element if $d = 2$. Employing the Neumann boundary condition (2.7b) yields

$$\int_\Omega v^T d_a \frac{\partial y}{\partial t}\, dV = \int_\Omega (\nabla v \cdot \Gamma + v^T F)\, dV + \int_{\partial\Omega} v^T \left(G + \left(\frac{\partial R}{\partial y} \right)^T \mu \right) dA \tag{2.13a}$$

$$= \sum_{l=1}^{d+1} \int_\Omega (\nabla v_l \cdot \Gamma_l + v_l F_l)\, dV + \sum_{l=1}^{d+1} \int_{\partial\Omega} v_l \left(G_l + \sum_{m=1}^{N_R} \frac{\partial R_m}{\partial y_l} \mu_m \right) \tag{2.13b}$$

$\forall v \in V \times W$. Together with the Dirichlet boundary conditions (2.7c), this is the (constrained) weak form of the original problem, i.e. the strong form (2.7).

2.3 Approximation

The trial function $y \in V \times W$ is now approximated in the finite dimensional space $V^h \times W^h$ by

$$y_h := (u_h, p_h)^T \tag{2.14}$$

with $u_{\alpha h} = (u_{1h}, \ldots, u_{dh}) \in V^h$ and $p_h \in W^h$ which are given by

$$u_{\alpha h} = \sum_{j=1}^{N_T} u_{\alpha j} \phi_j^\alpha \qquad \text{and} \qquad p_h = \sum_{j=1}^{N_p} p_j \phi_j^p.$$

Similarly, the test functions are approximated with the same finite elements, i.e.

$$v_h : \Omega_h \to \mathbb{R}^{d+1}$$

with

$$v_h := (v_{1h}, \ldots, v_{dh}, v_{ph})^T \in V^h \times W^h \tag{2.15}$$

with $v_{\alpha h} : \Omega_h \to \mathbb{R}$ and $v_{ph} : \Omega \to \mathbb{R}$ which are given by

$$v_{\alpha h} = \sum_{j=1}^{N_T} \phi_j^\alpha \qquad \text{and} \qquad v_{ph} = \sum_{j=1}^{N_p} \phi_j^p$$

where $\phi_j^\alpha \in$ and $\phi_j^p \in$ are nodal basis functions defined by $\mathcal{T}_h$ and Lagrange $P_{k(u)}P_{k(p)}$-elements. Note that, unlike in section 2.2.1, Dirichlet boundary conditions are not taken into account at this point.

Since the test functions occur linearly in the integrands of the weak equation (2.13), it is enough that the weak equation holds for every basis function

$$v_{\alpha hj} = \phi_j^\alpha \qquad \text{and} \qquad v_{phj} = \phi_j^p.$$

Substituting (2.14) and (2.15) into the weak form (2.13) eventually yields the discrete weak form with

$$N_{\mathrm{DOF}} = \sum_{i=1}^{d} N_{T\alpha} + N_p$$

equations. The discretization of the boundary term in (2.13b) is described later. Since the finite element data structure is rather complex, only few runs through the data structure should be performed. Therefore, all integrals in discrete weak form which are scalar expressions involving the dependent variables y_l as well as the test functions v_l, and their derivatives, are evaluated elementwise. Finally, every d-simplex is transformed to a reference element on which the actual computations are performed. This process is briefly described in the following where W denotes any of the integrants in the discrete weak form, i.e. the discrete version of (2.13).

$$\int_\Omega W\,dV \approx \int_{\Omega_h} W\,dV = \sum_{i=1}^{N_\Delta} \int_{\Delta_i} W\,dV$$

The integrals on the right-hand side are transformed to the reference element,

$$\begin{aligned}
\int_{\Delta_i} W\,dV &= \int_{\Delta_i} W(x_1,\ldots,x_d)\,dx_1\ldots dx_d && \text{(2.16a)}\\
&= \int_{\Delta_{\mathrm{ref}}} W(F(\lambda_1,\ldots,\lambda_d))\mid \det J_{\Delta_i}\mid d\lambda_1\ldots d\lambda_d && \text{(2.16b)}\\
&= \begin{cases} \displaystyle\int_0^1\int_0^{1-\lambda_1} g(\lambda_1,\lambda_2)\,d\lambda_2 d\lambda_1 & \text{in 2D}\\ \displaystyle\int_0^1\int_0^{1-\lambda_1}\int_0^{1-\lambda_1-\lambda_2} g(\lambda_1,\lambda_2,\lambda_3)\,d\lambda_3 d\lambda_2 d\lambda_1 & \text{in 3D}\end{cases} && \text{(2.16c)}\\
&\approx \sum_{i=1}^{n_p} \omega_i g(\lambda_1^i,\ldots,\lambda_d^i) && \text{(2.16d)}
\end{aligned}$$

with J_{Δ_i} as defined on page 33.

Numerical cubature on d-simplices

The integrals in equations (2.16c) occurring in the components of the load vector L, the stiffness matrix K and the mass matrix D are computed using a quadrature formula (2.16d). The numerical quadrature formula ought to be accurate and cheap. The already available Lagrange points seem to come in handy, but it turns out that the order of quadrature formulas utilizing the given Lagrange points is in general too low for our purposes.

Definition 2.3 *A cubature formula on a d-simplex Δ_d is said to be of order $k+1$ iff*

$$\int_{\Delta_d} p(\lambda)\,dV = \sum_i^{n_p} \omega_i p(\lambda_1^i,\ldots,\lambda_d^i) \qquad \forall p \in \mathcal{P}_k(\Delta_d).$$

In order to evaluate the integrals numerically with sufficient accuracy, more Lagrange points would be needed, i.e. function evaluations at new locations would be necessary. If function evaluations at new locations are necessary, the number of new locations should be minimized while at the same time a high order should be attained.

Any multiple integral over a d-simplex may be transformed into a multiple integral over a d-cube (see [109]). Then, product rules are usually used for quadrature over a d-cube. The main advantage of product quadrature rules is that their derivation is straightforward and simple, for any desired order. The primary disadvantage is inefficiency (even for Gauss quadrature rules) since for high order of the quadrature rule, a relatively large number of points is required (even when one-dimensional Gauss quadrature rules are used), and other quadrature rules are available with many fewer points. The second disadvantage is that the location of the points is unsymmetrical. They are not equally distributed. Except for rules of low degree, a large number of points will be concentrated in a relatively small region near one vertex. Such an arrangement, although theoretically (i.e. with infinite precision) correct, may be undesirable. With finite precision, no vertex should be favored against the others.

For quadrilateral elements, product quadrature rules (also called multidirectional methods, cf. [25]) have less geometrical bias than direct methods; on the other hand, for triangular or tetrahedral elements, direct methods have less geometrical bias than product methods. Therefore, we use non-product cubature rules (cf. [22, 109]) for the computation of matrices and vectors based on the Galerkin finite element method. In fact, we use symmetrical Gaussian quadrature rules which belong to the family of non-product cubature rules.

In two dimensions, integrals over finite elements are computed with two-dimensional symmetrical Gaussian quadrature rules; boundary integrals are computed with ordinary Gaussian quadrature rules. In three dimensions, integrals over finite elements are computed with three-dimensional symmetrical Gaussian cubature rules while boundary integrals are

evaluated with two-dimensional symmetrical Gaussian quadrature. The order of the integration method used for an element and its boundary is set by the field `gporder` (see appendix A). Default order is twice the order of the finite element's shape functions if not specified otherwise. Hence, a minimal number of function evaluations has to be performed in order to reach maximal accuracy.

For completeness, the Gauss points in barycentric (local) coordinates and the corresponding weights for quadratures of order 2, 3, 4, 5, 6, and 7 are given in tables 2.1 and 2.2. Figure 2.2 shows the distribution of the integration points in two dimensions. Finally, we remark that it is far from trivial to find as few points and weights as possible so that the cubature formula is of maximal order and all integration points are inside the triangle. To my knowledge there is, in contrast to the product rules, no complete theory on how to compute integration points and weights for symmetric Gaussian cubature rules in two and three dimensions. Highly nonlinear system of equations have to be solved, typically with Gauss-Newton or Levenberg-Marquardt algorithms. The initial estimate of the points' locations is very critical for finding a solution (see also [26]).

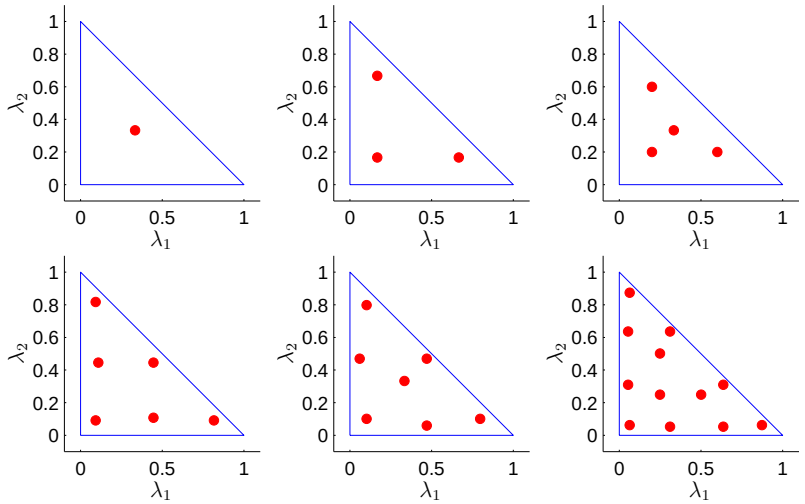


Figure 2.2: Gauss points on triangular reference elements for quadratures of order 2, 3, 4, 5, 6, and 7

order	n_p	p	λ_1	λ_2	ω_p
2	1	1	0.33333333333333	0.33333333333333	0.50000000000000
3	3	1,2,3	0.66666666666667	0.16666666666667	0.16666666666667
4	4	1	0.33333333333333	0.33333333333333	-0.28125000000000
		2,3,4	0.60000000000000	0.20000000000000	0.26041666666667
5	6	1,2,3	0.10810301816807	0.44594849091597	0.11169079483901
		4,5,6	0.81684757298046	0.09157621350977	0.05497587182766
6	7	1	0.33333333333333	0.33333333333333	0.11250000000000
		2,3,4	0.05971587178977	0.47014206410511	0.06619707639425
		5,6,7	0.79742698535309	0.10128650732346	0.06296959027241
7	12	1,2,3	0.50142650965818	0.24928674517091	0.05839313786319
		4,5,6	0.87382197101700	0.06308901449150	0.02542245318510
		7,...,12	0.05314504984482	0.31035245103378	0.04142553780919

Table 2.1: Two-dimensional numerical integration for triangular reference elements

Discretization of constraints

Discretization of (2.13), where the boundary integrals are not taken into account yet, with the tools as described above, yields

$$D(y_h,t)\frac{dy_h}{dt} = L(y_h,t) \tag{2.17}$$

with $D(y_h,t) \in \mathbb{R}^{N_{\mathrm{DOF}}\times N_{\mathrm{DOF}}}$, $y_h(t) \in \mathbb{R}^{N_{\mathrm{DOF}}}$, and $L(y_h,t) \in \mathbb{R}^{N_{\mathrm{DOF}}}$. Now the boundary term in (2.13) and the constraints (2.6c) are discretized.

The Lagrange multipliers are discretized in the following way. Let

$$\Lambda_{mj} = \mu(x_{mj})w_{mj}$$

where x_{mj} are the Lagrange points of the mth boundary mesh element defined by (2.2) and w_{mj} is the length of the appropriate part of this mesh element. The index j denotes the jth Lagrange point of this mesh element.

The boundary integral in (2.13) is approximated as a sum over all boundary mesh elements defined by (2.2). On the mth mesh element, this sum is approximated with a Riemann sum

$$\sum_j \varphi_i^\beta(x_{mj})h^T(x_{mj})\mu(x_{mj})\omega_{mj} = \sum_j \varphi_i^\beta(x_{mj})h^T(x_{mj})\Lambda_{mj}$$

with $\varphi_i^\beta := \phi_i^\alpha$ or $\varphi_i^\beta := \phi_i^p$ and $h := -\frac{\partial R}{\partial y_h}$. This means that the discretization of the weak equation can be written as

$$Dy_h = L - N^T\Lambda.$$

order	n_p	p	λ_1	λ_2	λ_3	ω_p
2	1	1	0.25000000000000	0.25000000000000	0.25000000000000	0.16666666666667
3	4	1,...,4	0.58541019662497	0.13819660112501	0.13819660112501	0.04166666666667
4	5	1	0.25000000000000	0.25000000000000	0.25000000000000	0.13333333333333
		2,...,4	0.50000000000000	0.16666666666667	0.16666666666667	0.07500000000000
5	11	1	0.25000000000000	0.25000000000000	0.25000000000000	0.25000000000000
		2,...,5	0.78571428571429	0.07142857142857	0.07142857142857	0.00762222222222
		6,...,11	0.39940357616680	0.39940357616680	0.10059642383320	0.02488888888889
6	14	1,...,4	0.06734224221010	0.31088591926330	0.31088591926330	0.01878132095300
		5,...,8	0.72179424906733	0.09273525031089	0.09273525031089	0.01224884051939
		9,...,14	0.45449629587435	0.45449629587435	0.04550370412565	0.00709100346285

Table 2.2: Three-dimensional numerical integration for tetrahedral reference elements

Λ is a vector containing all the discretized Lagrange multipliers Λ_{mj} and N is a matrix whose columns contain the elements $\varphi_i^\beta(x_{mj})h(x_{mj})$. And finally, the constraints, i.e. essential boundary conditions (2.6c) are incorporated as follows. The constraints are required to hold pointwise in the Lagrange points of a certain order on each boundary element on Γ^D. Then the discretization of the constraint is

$$0 = R(x_{mj})$$

where x_{mj} are the Lagrange points of the mth boundary mesh element defined by (2.2). The index j denotes the jth Lagrange point of this mesh element. The order of the Lagrange points is set by the field `cporders` (see appendix A). Default value is the maximum order of the shape functions used plus 1. (Each continuous variable is treated separately, i.e. different orders for different variables (u, p) are possible. I.e. order $k(u)+1$ for u and order $k(p)+1$ for p on standard $P_{k(u)}P_{k(p)}$ element, if not stated otherwise). The discretized constraints (2.6c) are then given by

$$M : \mathbb{R}^{N_{\mathrm{DOF}}} \times \mathbb{R} \to \mathbb{R}^{N_{\mathrm{PCB}}}$$

Since the constraints are imposed per boundary element (just like the integrals were evaluated elementwise), a constraint for a common point of two boundary elements is imposed twice. This means that the number of pointwise constraints on the boundary, i.e. boundary segments with essential boundary conditions, is typically in 2D,

$$N_{\mathrm{PCB}} = \sum_{i=1}^{d+1} \#\mathrm{edges}(y_i) \cdot \#\text{Lagrange points}(y_i)$$

where $\#\mathrm{edges}(y_i)$ denotes the number of boundary edge elements with a Dirichlet boundary condition for the variable y_i and $\#\text{Lagrange points}(y_i)$ denotes the number of Lagrange points for the variable y_i on boundary edge elements. For each constraint there is a Lagrange multiplier, and so the Lagrange multipliers are also redundant. This redundancy is removed in the elimination process which is described in the remainder of this chapter.

To complete this section, the spatially discretized problem now reads

$$D(y_h,t)\frac{dy_h}{dt} = L(y_h,t) - N(y_h,t)^T\Lambda \tag{2.18a}$$

$$0 = M(y_h,t) \tag{2.18b}$$

with

$$N(y_h,t) = -\frac{\partial M(y_h,t)}{\partial y_h} \in \mathbb{R}^{N_{\mathrm{PCB}}\times N_{\mathrm{DOF}}} \tag{2.19}$$

and[2] $M \in \mathbb{R}^{N_{\mathrm{PCB}}}$.

[2] In fact, $M : \mathbb{R}^{N_{\mathrm{DOF}}} \times \mathbb{R} \to \mathbb{R}^{N_{\mathrm{PCB}}}$ for a constant discretization, or to be more precise, M is a family of mappings $M_i : \mathbb{R}^{N_{\mathrm{DOF}(i)}} \times [t_i, t_{i+1}) \to \mathbb{R}^{N_{\mathrm{PCB}(i)}}$ in case of a time-dependent discretization.

2.4 Discretized Problem

It was shown in the previous section that the discrete version of the weak form (2.13) is given by equations (2.18). In order to solve this differential algebraic system (with highly redundant algebraic equations (2.18b)), the constraints need to be eliminated. This section described the elimination process, what the linearized problem looks like, and how it is obtained.

2.4.1 Linearized Problem

Consider the stationary problem, i.e. (2.18)-(2.19) with $D(y_h, t) = 0$ and let $L(y_h)$, $M(y_h)$, and $N(y_h)$ be independent of t. Using Taylor series expansion of the load vector $L(y_h)$ and the constraint vector $M(y_h)$ about $y_h^\star$

$$L(y_h) = L(y_h^\star) + \left.\frac{\partial L(y_h)}{\partial y_h}\right|_{y_h=y_h^\star} (y_h - y_h^\star) + \mathcal{O}(||y_h - y_h^\star||^2)$$
$$M(y_h) = M(y_h^\star) + \left.\frac{\partial M(y_h)}{\partial y_h}\right|_{y_h=y_h^\star} (y_h - y_h^\star) + \mathcal{O}(||y_h - y_h^\star||^2)$$

and neglecting second and higher order terms, the discretization of the linearized problem is given by

$$K(y_h^\star)(y_h - y_h^\star) + N(y_h^\star)^T\Lambda = L(y_h^\star)$$
$$N(y_h^\star)(y_h - y_h^\star) = M(y_h^\star)$$

with

$$K(y_h^\star) = -\left.\frac{\partial L(y_h)}{\partial y_h}\right|_{y_h=y_h^\star}$$
$$N(y_h^\star) = -\left.\frac{\partial M(y_h)}{\partial y_h}\right|_{y_h=y_h^\star}.$$

If the original problem is linear, then K and N are independent of $y_h^\star$ and its discretization can be written as

$$Ky_h + N^T\Lambda = L(0) = L(y_h^\star = 0) \tag{2.20a}$$
$$Ny_h = M(0) = M(y_h^\star = 0). \tag{2.20b}$$

2.4.2 Elimination of constraints (nonlinear)

Theorem 2.4 *If $M(y_h(t), t)$ in (2.18) is a linear constraint in $y_h(t)$, then*

$$y_h(t) = \mathcal{N}(t)y_e(t) + y_{h_d}(t) \tag{2.21}$$

where $y_{h_d}(t)$ is a solution of $M(y_h, t) = 0$, and $\mathcal{N}(t) \in \mathbb{R}^{N_{\mathrm{DOF}} \times N_{\mathrm{DOFe}}}$ is a sparse matrix whose columns form an orthonormal basis for the null space of $N(t) \in \mathbb{R}^{N_{\mathrm{PCB}} \times N_{\mathrm{DOF}}}$ (as in (2.19)),

$$\ker(N(t)) = \langle \mathcal{N}(t)_{:,1}, \ldots, \mathcal{N}(t)_{:,N_{\mathrm{DOFe}}} \rangle.$$

Proof. If $M(y_h(t), t)$ is linear in $y_h(t)$, then

$$0 = \frac{\partial M(y_h, t)}{\partial y_h} y_h + M(0, t) \quad \Longleftrightarrow \quad N(t)y_h = M(0, t).$$

Therefore, if $y_d(t)$ is a special solution of the equation on the right-hand side, $y_h(t) = \mathcal{N}(t)y_e(t) + y_d(t)$ is also a solution. Furthermore, $N(t)$ is independent of $y_h(t)$. ◇

Theorem 2.5 *Let $M(y_h(t), t)$ and $\mathcal{N}(t)$ be as in theorem 2.4, then*

$$y_e(t) = \mathcal{N}(t)^T y_h(t). \tag{2.22}$$

Proof. Multiplication of (2.21) by $\mathcal{N}(t)^T$ yields

$$\mathcal{N}(t)^T y_h(t) = y_e(t) + \mathcal{N}(t)^T y_d(t) = y_e(t)$$

since $y_d(t) \in \mathrm{im}(N(t))$. ◇

Theorem 2.6 *Let $M(y_h(t), t)$ be a linear constraint in y_h. The system (2.18) is equivalent to the eliminated system*

$$D_e(y_e(t), t)\frac{dy_e(t)}{dt} = L_e(y_e(t), t) \tag{2.23}$$

with

$$D_e(y_e(t), t) = \mathcal{N}^T D \mathcal{N} \tag{2.24}$$
$$L_e(y_e(t), t) = \mathcal{N}^T(L(\mathcal{N}y_e(t) + y_{h_d}(t), t) - D\frac{dy_{h_d}(t)}{dt} - D\frac{d\mathcal{N}}{dt}y_e(t)). \tag{2.25}$$

with $D = D(\mathcal{N}y_e(t) + y_{h_d}(t), t)$ and $\mathcal{N} = \mathcal{N}(t)$.

$D_e(y_e(t), t)$ is called the eliminated mass matrix and $L_e(y_e(t), t)$ the eliminated load vector.

Proof. The linear[3] constraint $M(y_h, t) = 0$ is eliminated as follows. Let $\mathcal{N} = \mathcal{N}(t)$ and write the DOF solution vector $y_h(t)$ as

$$y_h(t) = \mathcal{N}y_e(t) + y_{h_d}(t),$$

[3]Recall that $N = -\frac{\partial M}{\partial y_h}$, so N is independent of y_h since M is linear in y_h since R is linear in u (see p. 37).

as in theorem 2.4. Since $y_{h_d}(t) \in \mathbb{R}^{N_{\mathrm{DOF}}}$ is a solution vector which satisfies the essential boundary conditions, $y_h(t)$ also satisfies the boundary conditions. Therefore, (2.18) yields

$$\begin{aligned} & D(y_h(t),t)\frac{dy_h(t)}{dt} = L(y_h(t),t) \\ \Longrightarrow \quad & D(\mathcal{N}y_e + y_{h_d},t)\frac{d}{dt}(\mathcal{N}y_e + y_{h_d}) = L(\mathcal{N}y_e + y_{h_d},t) \\ \Longrightarrow \quad & D\mathcal{N}\frac{dy_e}{dt} = L - D\frac{dy_{h_d}}{dt} - D\frac{d\mathcal{N}}{dt}y_e \\ \Longrightarrow \quad & \mathcal{N}^T D\mathcal{N}\frac{dy_e}{dt} = \mathcal{N}^T\left(L(\mathcal{N}y_e + y_{h_d},t) - D\frac{dy_{h_d}}{dt} - D\frac{d\mathcal{N}}{dt}y_e\right) \end{aligned}$$

where we have skipped some of the arguments for simplicity notation. ◇

Theorem 2.7 *If D is independent of y_h, then the Jacobian of the right-hand side of the eliminated system* (2.23) *is*

$$J_e(y_e(t),t) := \frac{\partial L_e(y_e(t),t)}{\partial y_e(t)} = \mathcal{N}(t)^T\left(-K(y_h(t),t)\mathcal{N}(t) - D(y_h,t)\frac{d\mathcal{N}(t)}{dt}\right) \tag{2.26}$$

with

$$K(y_h,t) = -\frac{\partial L(y_h,t)}{\partial y_h}.$$

Proof. Applying theorem 2.5 and the chain rule to (2.25) yields

$$\begin{aligned} \frac{\partial L_e}{\partial y_e} &= \frac{\partial L_e}{\partial y_h}\frac{\partial y_h}{\partial y_e} \\ &= \mathcal{N}^T(-K\mathcal{N} - D\frac{d\mathcal{N}}{dt}) \end{aligned}$$

where the arguments have been omitted for simplicity of notation. ◇

2.4.3 Elimination of constraints (linear)

Theorem 2.8 *The constrained linear system* (2.20) *is equivalent to the eliminated linear system*

$$K_e y_e = L_e \tag{2.27}$$

with

$$\begin{aligned} K_e &= \mathcal{N}^T K\mathcal{N} \\ L_e &= \mathcal{N}^T(L(0) - Ky_{h_d}) \end{aligned}$$

Proof. In order to eliminate the linear constraint M, write the constrained solution y_h according to theorem 2.4 as

$$y_h = \mathcal{N}y_e + y_{h_d}.$$

Then, (2.20) is equivalent to

$$K(\mathcal{N}y_e + y_{h_d}) = L(0).$$

Multiplication with $\mathcal{N}^T$ yields

$$\mathcal{N}^T K\mathcal{N}y_e = \mathcal{N}(L(0) - Ky_{h_d}).$$

◇

2.5 Eliminated Navier-Stokes system

Using the tools described in the previous sections, there are several ways to assemble the discrete Navier-Stokes system. In this section, three modes are introduced in order to assemble the complete discrete Navier-Stokes equations or just parts of it. The coupled mode, decoupled mode, and individual mode are described in the following. The individual mode is the most general one since it gives access to the individual terms of the discrete Navier-Stokes equations. In some situations, however, a less general procedure may suffice and the other modes may be preferred.

2.5.1 Fully coupled mode

The spatially discretized eliminated Navier-Stokes system is given by the differential algebraic equation

$$M_e\frac{dy_e}{dt} = F(y_e) \tag{2.28}$$

with

$$M_e = \begin{pmatrix} M & 0 \\ 0 & 0 \end{pmatrix} \in \mathbb{R}^{N_{\mathrm{DOFe}}\times N_{\mathrm{DOFe}}} \quad \text{and nonsingular} \quad M \in \mathbb{R}^{N_T\times N_T}$$

and $y_e = (u_e, p_h)^T \in \mathbb{R}^{N_{\mathrm{DOFe}}}$, $u_e \in \mathbb{R}^{N_{\mathrm{Te}}}$, $p_h \in \mathbb{R}^{N_{\mathrm{p}}}$, and $M_e = D_e$ (see (2.24)) and $F = L_e$ (see (2.25)). Equation (2.28) is referred to as the fully coupled system. The Jacobian is given by

$$J_e(y_e) := \frac{\partial F(y_e)}{\partial y_e} \in \mathbb{R}^{N_{\mathrm{DOFe}}\times N_{\mathrm{DOFe}}}$$

and computed according to (2.26). Equation (2.28) is the basis for the spatially discretized Navier-Stokes equations. It represents the discrete approximation of (2.6c) and (2.13), i.e. it is the finite element discretization of a specific weak form depending on how flux matrix and source term are defined (cf. section 2.2.3).

This most compact form (2.28) of the discretized Navier-Stokes equations is best suited for fully coupled implicit solution strategies. However, splitting methods require direct access to the discrete advection term and diffusion term, for example. These terms are not amenable individually at this point. In order to suffice the demands of the time integration methods described in chapter 3, the issue of how to compute all individual terms of the discrete Navier-Stokes equations individually is now addressed.

The assembly of individual discrete terms is expensive because the data structure needs to be accessed each time. The velocity and pressure parts can be extracted from the solution vector as follows:

Definition 2.9 *Define* $\mathrm{ind}_u \in \mathbb{R}^{N_{\mathrm{Te}}}$ *and* $\mathrm{ind}_p \in \mathbb{R}^{N_{\mathrm{p}}}$ *so that*

$$u_e = y_{e\,\mathrm{ind}_u} \qquad \textit{and} \qquad p_e = p_h = y_{e\,\mathrm{ind}_p},$$

i.e. $y_e = (u_e, p_h)^T$.

2.5.2 Individual mode

The most common and most important discrete forms of the Navier-Stokes equations will now be described. They are based on the weak forms defined by (2.8) or (2.11), i.e. Green's theorem is also applied to the gradient of the pressure. The discrete equations based upon the forms defined in section 2.2.3 are implemented as described in the following. Suppose that F (as defined in section 2.2.3) is split into its linear and nonlinear parts,

$$F = F_{\mathrm{linear}} + F_{\mathrm{nonlinear}}.$$

Symmetric form

For Newtonian fluids, the pseudo total stress form (2.8) and the total stress form (2.11) yield the discrete Navier-Stokes equations,

$$M\frac{du_h}{dt} = -A(u_h, u_h) + Du_h - Cp_h + f \tag{2.29a}$$
$$C^T u_h = g \tag{2.29b}$$

with

$$A(u_h, u_h) = -L_{e\,\mathrm{ind}_u}$$

where L_e is assembled with $d_a = 0$, $\Gamma = 0$, and $F = F_{\mathrm{nonlinear}}$. All other terms are extracted as follows:

$$\begin{aligned} D &= -K_{e\,\mathrm{ind}_u,\mathrm{ind}_u} \\ C &= K_{e\,\mathrm{ind}_u,\mathrm{ind}_p} \\ C^T &= K_{e\,\mathrm{ind}_p,\mathrm{ind}_u} \\ f &= L_{e\,\mathrm{ind}_u} \\ g &= L_{e\,\mathrm{ind}_p} \\ M &= D_{e\,\mathrm{ind}_u,\mathrm{ind}_u} \end{aligned}$$

with d_a, Γ as in section 2.2.3 and $F = F_{\mathrm{linear}}$. Note that f is the discretized weak form of the force density b in (1.12a) plus contributions from the diffusion term due to Dirichlet boundary conditions. The discrete diffusion term is not identical for the weak forms defined by (2.8) and (2.11).

Remark: The vector g is always generated by inhomogeneous Dirichlet boundary conditions and, because we preclude sources or sinks of mass, it contains mostly zeros. For homogeneous Dirichlet boundary conditions we have $g = 0$, i.e. the discrete equation of continuity then reads $C^T = 0$, which describes contained flow within stationary boundaries (see also [48]).

For non-Newtonian fluids, the symmetric form reads

$$M\frac{du_e}{dt} = -A(u_e, u_e) + D(u_e) - Cp_e + f \tag{2.30a}$$
$$C^T u_e = g \tag{2.30b}$$

where the discrete diffusion term $D(u_e)$ is typically nonlinear. As mentioned in section 1.2, this works only with the total stress form (2.11). In contrast to the Newtonian case, f is now the discretized force density b and does not contain contributions from the diffusion term due to Dirichlet boundary conditions.

Non-Symmetric form

For completeness, we also state the non-symmetric form based on the weak forms defined by (2.9) and (2.10) which reads for Newtonian fluids

$$M\frac{du_e}{dt} = -A(u_e, u_e) + Du_e - Gp_e + f \tag{2.31a}$$
$$C^T u_e = g \tag{2.31b}$$

and for non-Newtonian fluids (only (2.10))

$$M\frac{du_e}{dt} = -A(u_e, u_e) + D(u_e) - Gp_e + f \tag{2.32a}$$
$$C^T u_e = g. \tag{2.32b}$$

The discrete diffusion term is not identical for the weak forms defined by (2.9) and (2.10).

Similar to the symmetric case, f contains contributions from D due to Dirichlet boundary conditions if the fluid is Newtonian. For non-Newtonian fluids, there are no such contributions and f is simply the discrete body force. Finally, for equations (2.29a), (2.30a), (2.31a), and (2.32a), the Jacobian of the advection term is

$$J_A = K_{e\,\mathrm{ind}_u,\mathrm{ind}_u}$$

and the Jacobian of the diffusion term is

$$J_D = -K_{e\,\mathrm{ind}_u,\mathrm{ind}_u}.$$

Of course, $D = J_D$ if the fluid is Newtonian.

Remark 2.10 *The temporal derivatives in* (2.25) *and* (2.26) *of theorem 2.6 and 2.7, respectively, are approximated with central differences,*

$$\frac{dy_{h_d}(t)}{dt} = \frac{y_{h_d}(t+\Delta t_e) - y_{h_d}(t-\Delta t_e)}{2\Delta t_e} + \mathcal{O}(\Delta t_e^2)$$
$$\frac{d\mathcal{N}(t)}{dt} = \frac{\mathcal{N}(t+\Delta t_e) - \mathcal{N}(t-\Delta t_e)}{2\Delta t_e} + \mathcal{O}(\Delta t_e^2).$$

The time step Δt_e *is chosen to be*

$$\Delta t_e = \max\left(\frac{1}{100}(t^n - t^{n-1}), 10^{-10}\right) = \max\left(\frac{1}{100}\Delta t, 10^{-10}\right)$$

where t^n *is the current time where* (2.25) *and* (2.26) *are computed and* t^{n-1} *is the time of the previous time step.*

Remark 2.11 *The matrices and vectors required by the identities* (2.21)*,* (2.22)*, and* (2.25) *are stored in the fields*

$$\texttt{fem.DOF.D} = D(t)$$
$$\texttt{fem.DOF.Nu} = \mathcal{N}(t)$$
$$\texttt{fem.DOF.yd} = y_{h_d}(t)$$
$$\texttt{fem.DOF.t} = t$$

2.5.3 Decoupled mode

The pseudo total stress form (2.8) for Newtonian fluids and the total stress form (2.11) for Newtonian and non-Newtonian fluids yield the discrete Navier-Stokes equations in decoupled mode,

$$M\frac{du_h}{dt} = B(u_h, u_h) - Cp_h$$
$$C^T u_h = g$$

where $B(u_h, u_h)$ denotes a discrete Burgers operator, i.e. advection plus diffusion plus body forces. The assembly procedure is performed similar to the fully coupled mode. The mass, gradient and divergence matrices are extracted,

$$C = K_{e\,\mathrm{ind}_u,\mathrm{ind}_p}$$
$$C^T = K_{e\,\mathrm{ind}_p,\mathrm{ind}_u}$$
$$B = L_{e\,\mathrm{ind}_u} + Cp_h$$

where L_e and K_e are assembled with with d_a, Γ as in section 2.2.3 and $F = F_{\mathrm{linear}} + F_{\mathrm{nonlin}}$. There is no direct access to the discrete advection term, its Jacobian, or the discrete diffusion term. In a similar way, the discrete equations for the forms (2.9) and (2.10) may be obtained.

2.5.4 Mixed finite elements

In order that the spatially discretized Navier-Stokes equations are stable approximations to (2.7c) and (2.13), it is crucial that the space $V_0^h \times W^h$ satisfies the so-called Ladyzhenskaya-Babuska-Brezzi (LBB) condition

$$\inf_{q\in W^h}\sup_{v\in V_0^h}\frac{\int_\Omega q\nabla\cdot v\,dV}{||q||_0||\nabla v||_0} \geq \gamma > 0 \tag{2.34}$$

with a mesh independent constant γ (see [94]). This guarantees that there are no spurious pressure modes (see also section 3.1.2). Mixed finite elements, such as P_2P_1 Lagrange elements satisfy the LBB condition. Error estimates can be found in [82, 94].

2.6 Initial value

Similar to the continuous in space velocity-pressure formulations, the discrete velocity-pressure formulations (such as (2.29), for example) require only initial data for the velocity. The discrete initial velocity field u_0 is supposed to be discretely divergence-free and comply with essential boundary conditions (see also section 1.2.1), i.e. $u_h(0) = u_{0h}$ with

$$C^T u_{0h} = g$$

If $n \cdot u_0 \neq 0$ on Γ^D, then $g \neq 0$ due to contributions from the constraints.

Chapter 3

Numerical Solution Strategies

3.1 Introduction

In this chapter, two classes of discrete projection schemes for the incompressible Navier-Stokes equations are presented. These projection schemes may be regarded as an extension of the discrete versions (cf. [119, 120, 121]) of the classical projection schemes of Chorin (cf. [14]) and Van Kan (cf. [73]) and variants of these. It is known that the projection approach requires only moderately smaller time steps than fully implicitly coupled schemes, but that the work to obtain comparative results with discrete projection methods as solvers is much lower (cf. [120]).

There is a large variety of existing solution schemes for the solution of the incompressible Navier-Stokes equations: fully coupled implicit methods, continuous projection methods and their discrete counterparts are three classes of most common solution schemes (see also [44, 48, 82, 94, 120]). The basic methodology of fully coupled methods is described in the following and it is explained why projection methods can reduce the numerical cost for instationary fluid flow simulations. Then the principal differences between continuous and discrete projection schemes are described and their pros and cons are explained. And finally, two classes of discrete projection schemes are presented and their implementation is described.

Recall the eliminated spatially discretized Navier-Stokes system of equations in its most compact form

$$M_e \frac{dy_e}{dt} = F(y_e), \qquad y_e \in \mathbb{R}^{N_{\mathrm{DOFe}}} \tag{3.1}$$

with constant $M_e = \mathrm{diag}(M, 0_{N_\mathrm{p} \times N_\mathrm{p}})$ which comprises the Newtonian or non-Newtonian and symmetric or non-symmetric forms introduced in section 2.5. Numerical solution of (3.1) with explicit time integration methods[1], such as explicit Runge-Kutta, explicit

[1] By explicit time integration we mean fully explicit time integration not to be confused with half-explicit

multistep methods or exponential time integrators, is not doable straightforward because of the singular mass matrix M_e. This is due to the fact that there is no evolution equation for the pressure in the differential algebraic systems (Newtonian or non-Newtonian and symmetric or non-symmetric form) arising from the incompressible Navier-Stokes equations after spatial discretization. The pressure may be regarded as a Lagrange multiplier which forces the velocity field to remain solenoidal at any instant of time. From another point of view, explicit time integrators cannot be applied directly because of their bounded stability region. The differential equation (3.1) is infinitely stiff as will be seen later (see section 3.4).

Fully coupled methods, a.k.a. fully implicit methods, such as implicit Runge-Kutta methods or BDF methods, circumvent this problem. However, the numerical work required per time step is rather expensive.

Projection schemes have been the methods most commonly used in computational fluid dynamics since Chorin introduced his original scheme in 1967 (see also [12, 13]). They have proven to successfully reduce the numerical cost and are therefore most efficient for instationary fluid flow simulations. Projection methods can be divided into two classes: continuous and discrete projection methods; the former being most popular in computational fluid dynamics and the latter more suited for the finite element method. The pros and cons of continuous and discrete projection methods will be discussed shortly. In both cases, the most important projection methods are based on the implicit Euler method and Crank-Nicolson scheme (see [48, 120] and references therein).

3.1.1 A test problem

As a first test problem, consider the simulation of time-dependent fluid flow for a modified driven cavity problem. The spatial domain Ω is simply the unit square minus some obstacle. The domain is defined using the concept of boundary modeling. The boundary of the domain consists of nine boundary segments Γ_i, i.e. four straight lines and five Bézier splines which define the obstacle (see figure 3.1). If not specified otherwise, the fluid's density is $\varrho = 1$ and its viscosity is $\eta = 0.1$. The fluid is subject to no-slip boundary conditions on the top boundary of the square (boundary segment 3) and on the boundary of the obstacle (boundary segments 5–9). On the other three boundaries (boundary segments 1, 2, and 4), slip boundary conditions are applied. The flow is driven by the top boundary of the square, moving from left to right with $u|_{\text{bs }3} = 2$. Velocity and pressure are approximated with P_2P_1 Lagrange elements and the mesh is generated as follows: an initial Delaunay triangularization is performed where the distribution of the boundary nodes depends on the curvature of the boundary. This initial mesh consists of $N_\Delta = 272$ elements, i.e. $N_{\text{DOFe}} = 1191$ eliminated unknowns based on P_2P_1 Lagrange elements, and its quality is $q_\Delta = 0.75$ (cf. figure 3.2(a)). In a second step, the mesh is uniformly refined which yields

time integration.

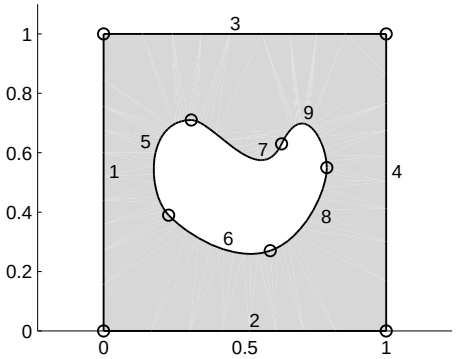


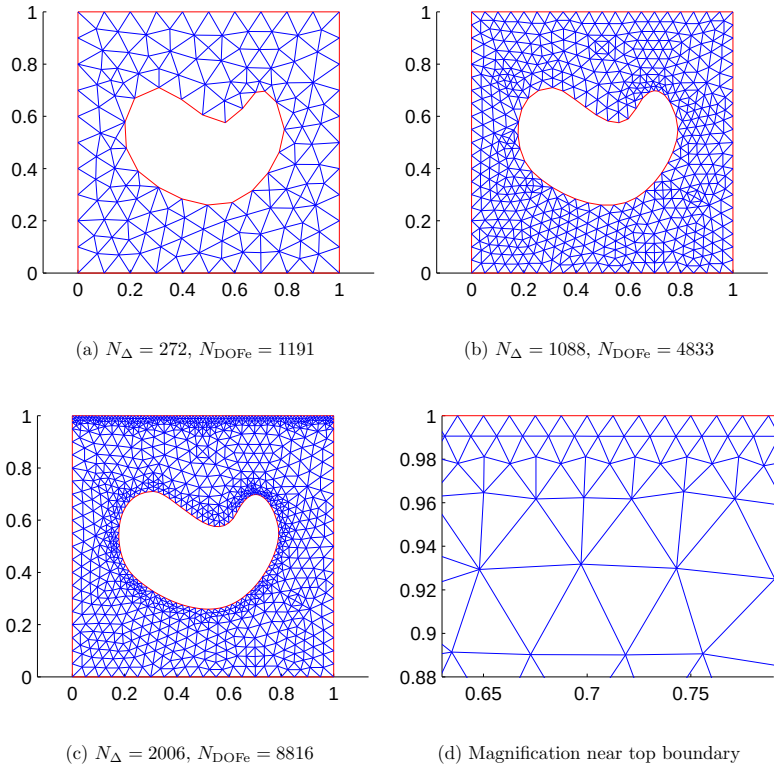
Figure 3.1: Driven cavity with obstacle

$N_\Delta = 1088$ elements, i.e. $N_{\text{DOFe}} = 4833$ eliminated unknowns (cf. figure 3.2(b)). The mesh quality drops to $q_\Delta = 0.61$ due to new nodes on the curved boundary segments. A uniform refinement in a domain with only straight boundary segments would not affect the mesh quality. In the third step, the triangularization near the no-slip boundaries of the obstacle is refined once and the triangularization near the moving top boundary is refined twice. This triangularization then consists of $N_\Delta = 2006$ elements, i.e. $N_{\text{DOFe}} = 8816$ eliminated unknowns, and its quality decreases to $q_\Delta = 0.53$. Clearly, the mesh quality has again decreased since the mesh is no longer a Delaunay triangularization. Nevertheless, this triangularization is better suited to resolve boundary layers than a Delaunay triangularization based on the same boundary nodes would be. Eventually, the mesh is jiggled in order to improve the mesh quality which yields $q_\Delta = 0.81$ while the number of elements, $N_\Delta = 2006$, remains unchanged (cf. figure 3.2(c)). Figure 3.2(d) shows how the elements become smaller towards the top moving boundary in order to resolve the boundary layers.

3.1.2 Pressure modes

We will now briefly explore the fact that C (as well as G) in (2.32) may or may not have full rank. This depends on the physical boundary conditions of the problem, their translation into boundary conditions for the initial boundary value problem, the weak formulation, and the choice of elements. All this will later significantly influence the performance of the numerical time integration schemes as well as the ingredients necessary for those schemes. Recall that either $C = G$ or C and G differ by an integration by parts of ∇p.

There are $N_{\text{Te}} = \sum_{\alpha=1}^{d} N_\alpha$ velocity equations and N_{p} constraint equations among the



(a) $N_\Delta = 272$, $N_{\text{DOFe}} = 1191$

(b) $N_\Delta = 1088$, $N_{\text{DOFe}} = 4833$

(c) $N_\Delta = 2006$, $N_{\text{DOFe}} = 8816$

(d) Magnification near top boundary

Figure 3.2: Mesh generation

$N_{\text{DOFe}} = N_{\text{Te}} + N_{\text{p}}$ equations in (3.1). The numbers N_{Te} and N_{p} depend on the choice of elements, the boundary conditions, and the triangularization. Of the N_{Te} momentum equations, the N_{p} constraints due to conservation of mass leave only $N_{\text{Te}} - N_{\text{p}}$ *effective* momentum equations if and only if C (and G) have full rank. If C (and G) are rank deficient, say $r := \text{rank}(C) < N_{\text{p}}$, then there are $N_{\text{Te}} - r$ effective momentum equations.

There is, at most, one physically meaningful vector in the null space of G called the hydrostatic pressure mode denoted by P_H; a constant vector. When is P_H present? Only when $n \cdot u$ is specified on all of Γ! Then, the pressure is only determinable up to an additive constant (cf. [48]), namely a multiple of the hydrostatic pressure mode. In other words, if traction boundary conditions, such as free boundary or outflow boundary conditions, are employed at any portion of Γ – even just at a single point – then G no longer contains P_H in its null space. A traction boundary condition is an additional condition that ties the pressure to the velocity field, i.e. it can be interpreted as a constraint for the pressure which removes the hydrostatic pressure mode.

However, if natural boundary conditions of the unsymmetric forms (2.31) and (2.32) are used to model outflow or free boundary conditions, then the hydrostatic pressure mode is present. Since natural boundary conditions of the unsymmetric forms don't model true physical forces, we do not use them to model outflow or free boundary conditions. Instead, we always use the traction conditions introduced in section 1.2.1 which models true physical forces.

All null vectors of G except P_H are spurious numerical artifacts and have no analogs in the continuous case. For example, all equal-order elements generate too many (i.e. redundant) mass balance constraints and thus possess spurious modes. This explains the use of mixed order elements, e.g. the Lagrange P_2P_1 element, a.k.a. Taylor-Hood element introduced in chapter 2. For *appropriate* finite elements, i.e. finite elements which satisfy the LBB condition (2.34), the only possible pressure mode is the hydrostatic pressure mode, which means that there are no spurious pressure modes. Lagrange P_2P_1 elements are appropriate in that sense and therefore guarantee that there are no spurious pressure modes. Having clarified the issue of pressure modes, we now turn to the numerical time integration schemes.

3.1.3 Differentiation index

Differentiating (2.32b) with respect to time and inserting (2.32a) yields the pressure Poisson formulation

$$M\dot{u}_e = -A(u_e, u_e) + D(u_e) - Gp_h + f \tag{3.2a}$$

$$C^T M^{-1} G p_h = C^T M^{-1}(-A(u_e, u_e) + D(u_e) + f) - \dot{g}. \tag{3.2b}$$

Under the assumption of sufficient smoothness, differentiating (3.2b) yields

$$M\dot{u}_e = -A(u_e, u_e) + D(u_e) - Gp_h + f$$
$$C^T M^{-1} G\dot{p}_h = C^T M^{-1}\frac{d}{dt}(-A(u_e, u_e) + D(u_e) + f) - \ddot{g}$$

provided that $C^T M^{-1} G$ is invertible, i.e. if and only if C and G have full rank. This is a system of ordinary differential equations which we obtained after differentiating twice. Therefore, the differential algebraic system (2.32) has differentiation index 2 if and only if there are no pressure modes.

The behavior of BDF and Runge-Kutta methods applied to index-2 systems is well understood (cf. [56]). However, if the concept of differentiation index does not apply, the situation is different (cf. [78, 123]).

A typical situation is, however, that no-slip boundary conditions are specified on all of $\partial\Omega$. In this case, C does not have full rank due to the hydrostatic pressure mode (see also [48]). There are also other cases where this situation occurs. In this case, the concept of differentiation index does not apply to such systems. A generalized index concept – the so-called strangeness index – is suggested in [79] for the case of over- and underdetermined linear differential algebraic systems. It is shown in [123] how to apply the concept of strangeness index to the linearized Navier-Stokes equations. But the concept of strangeness index doesn't reveal any new information about the nonlinear equations.

From a computational point of view, we take care of possible pressure modes by additional projections. This guarantees that the pressure mode remains constant and doesn't evolve unphysically or even explode due to roundoff errors.

3.2 Fully coupled methods

In order to motivate the idea behind projection methods, let us briefly have a look at fully coupled implicit methods. The scheme most frequently used in practice is the θ-scheme (cf. [103, 105, 120]). Applied to (3.1), it reads

$$M_e\frac{y_e^{n+1} - y_e^n}{\Delta t} = \theta F(y_e^{n+1}) + (1-\theta)F(y_e^n) \tag{3.3}$$

with $0 \le \theta \le 1$. For $\theta = 1$, this yields the backward Euler method and for $\theta = 0$ we get the half-explicit Euler method (cf. [56]) since M_e is singular. $\theta = \frac{1}{2}$ yields the Crank-Nicolson scheme (see also [73] for a continuous in space variant) which reads, e.g. in the case of the

most general[2] non-Newtonian non-symmetric form (2.32),

$$\begin{aligned} M\frac{u_e^{n+1} - u_e^n}{\Delta t} =& \frac{1}{2}(-A(u_e^{n+1}, u_e^{n+1}) + D(u_e^{n+1}) + f^{n+1}) \\ &+ \frac{1}{2}(-A(u_e^n, u_e^n) + D(u_e^n) + f^n) - \frac{1}{2}G(p_h^{n+1} + p_h^n) \end{aligned} \tag{3.4a}$$
$$C^T u_e^{n+1} = g^{n+1} \tag{3.4b}$$

provided that u^n is discretely divergence-free. Solving the coupled nonlinear equations (3.4) requires the solution of linear systems $Ay_e = b$ with

$$A = \begin{pmatrix} M + c_1(J_A - J_D) & c_2 G \\ C^T & 0_{N_\mathrm{p}\times N_\mathrm{p}} \end{pmatrix} \in \mathbb{R}^{N_{\mathrm{DOFe}}\times N_{\mathrm{DOFe}}} \tag{3.5}$$

and $c_1 = c_2 = \frac{\Delta t}{2}$, $J_A = J_A(u, u)$, and $J_D = J_D(u)$ in every step of the Newton iteration. There is no really cheap way to solve this discrete saddle point problem. In fact, the efficient solution of linear systems with system matrix (3.5) is still an ongoing field of research (cf. [35, 119, 120, 121]). Note that the iteration matrix A can be singular if G is rank deficient which is actually the case for many important fluid dynamics problems although this phenomenon is not very often taken into account (see for example [20, 59, 65, 87, 119, 120]). In particular, A is singular in case of the test problem described in section 3.1.1. Anyway, this problem can be fixed, e.g. by augmenting the linear system and thus removing the rank deficiency, i.e.

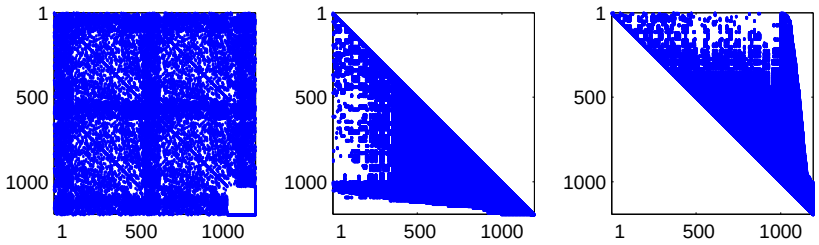
$$\widetilde{A} = \begin{pmatrix} A & (0, P_H^T)^T \\ (0, P_H^T) & 0 \end{pmatrix} \in \mathbb{R}^{(N_{\mathrm{DOFe}}+1)\times(N_{\mathrm{DOFe}}+1)} \tag{3.6}$$

is non-singular with $(0, P_H^T)^T \in \ker(A)$. We will return to this issue in section 3.7.2.

If the mesh is not too fine, direct methods may be applied in a first naive approach. If A is non-singular, the simple LU-decomposition in conjunction with symmetric reverse Cuthill-McKee reordering (cf. [43]) yields acceptable results. However, if A is singular and the rank deficiency is removed according to (3.6), a direct solution method even with symmetric reverse Cuthill-McKee reordering results in an enormous fill-in and is not feasible, not even on coarse meshes. Figure 3.3 illustrates this for the test problem of section 3.1.1 using only the coarsest mesh as shown in figure 3.2(a). The fill-in is prohibitive and the problem cannot be solved on the fine mesh with this strategy.

Clearly, iterative methods need to be employed for large scale simulations in order to get around the fill-in bottleneck of direct methods (see [17] for a comparison of linear solvers). Nevertheless, the solution of linear systems remains one of the most expensive

[2] The equations (2.29), (2.30), (2.31) may be regarded as special cases of (2.32).



(a) $\widetilde{A}$ (0.2% fill-in) (b) L (17.1% fill-in) (c) U (17.1% fill-in)

Figure 3.3: Sparsity pattern of $\widetilde{A}$ and its LU-decomposition with reordering

parts of a fluid flow simulation! Typically, a pressure-Schur-complement approach is used which requires the solution of linear systems $A_p p = b$ with

$$A_p = C^T(M + c_1(J_A - J_D))^{-1}c_2 G \in \mathbb{R}^{N_p \times N_p}, \qquad c_1, c_2 \in \mathbb{R}. \tag{3.7}$$

Using this approach, non-symmetric linear systems with a new solution dependent dense matrix A_p (of course, this dense matrix should never be explicitly computed) must be solved at every time step and in every step of the Newton iteration or at least every couple of iterations, if the simplified Newton method is applied. As before, A_p can also be singular. Both strategies are, in any case, very expensive and in particular solving (3.6) with a direct method is prohibitive for fine meshes. The iterative solution of the discrete saddle point problems (3.5), (3.6) and the corresponding Schur complement equation (3.7) is described in section 3.7.2.

Solving (3.1) with (fully) implicit Runge-Kutta methods is even more expensive per time step compared to implicit θ-methods. In general, nonlinear systems with $s \cdot N_{\mathrm{DOFe}}$ unknowns need to be solved for an s-stage implicit Runge-Kutta method. Diagonally implicit Runge-Kutta methods require the solution of s linear systems with N_{DOFe} unknowns per Newton iteration. BDF methods are cheaper per time step than the latter two methods, since only one linear system with N_{DOFe} unknowns needs to be solved per Newton iteration step, i.e. the numerical work per time step is comparable to that of implicit θ-methods. But BDF methods require a sequence of previous steps which may not always be available (see section 3.5 and chapter 4).

Nevertheless, all of these methods have similar or worse sparsity patterns than the Crank-Nicolson scheme. Of course, iterative methods should be employed in all of these time integration schemes for large scale simulations. But one downside remains: the simultaneous solution for the new velocity and the new pressure requires large or dense linear

systems to be solved in every time step. And as mentioned before, the matrix is solution dependent, i.e. it changes within every step of the Newton iteration, or at least every time step, if the simplified Newton method is applied and it is non-symmetric.

The assembly of matrices and vectors via the finite element method as described in chapter 2 and the solution of discrete saddle point problems (3.5), (3.6) or discrete pressure Poisson problems (3.7) are most time consuming in a fluid flow simulation. Therefore, a practical idea is to try to reduce the number of those linear systems which have to be solved per time step. It would be particularly nice if those fewer linear systems always had the same (if possible symmetric) matrix at least as long as the step size remains constant. In contrast to fully coupled solution strategies, projection methods have these desired properties. However, this usually comes at the cost of smaller step sizes and lower order of the methods. Projection methods are generally based on some kind of modification of standard time integration methods for differential algebraic equations. Their order is typically lower than that of the underlying time integration scheme. Nevertheless, they have proven to be efficient and most popular in computational fluid dynamics.

Half-explicit methods require small step sizes compared to implicit or semi-implicit methods due to their bounded stability region and the stiffness inherent in the velocity part of the equations. The maximal possible step sizes of the half-explicit Euler method applied to the previous test example are given in table 3.1. The maximal step sizes are very

N_{DOF}	1374	5196	9587
N_Δ	272	1088	2006
Δt	0.002	0.0002	0.0001

Table 3.1: Maximal step size for half-explicit Euler method

small and they decrease as the grid gets finer. Although there are half-explicit methods of higher order (cf. [56]), this phenomenon in principle still exists because of their bounded stability domain and the CFL-condition.

Since fully coupled, i.e. fully implicit, methods are expensive and half-explicit methods require extremely small step sizes, implicit or semi-implicit projection methods are inbetween the latter two in some sense. Due to these reasons, projection schemes have been the preferred methods in computational fluid dynamics since Chorin introduced his original scheme in 1967 (see also [12, 13]). Typically, the pressure at the new time level is approximated and later on the velocities and the pressure are corrected, requiring that the discrete divergence will be zero. For large scale simulations, iterative methods are necessary ingredients, too.

3.3 Classical projection methods

In computational fluid dynamics, projection methods have been introduced as a useful and efficient way to significantly reduce the computational cost of time-dependent incompressible viscous flow simulations in the velocity-pressure formulation (see also [73] and references therein). The method has been given by various names, such as splitting method, pressure correction method, or fractional step method. This means that the system (1.12a)–(1.12b) is split into a series of simpler equations, such as advection-diffusion equations and Poisson equations, for example, and explicit or implicit updates.

The pressure correction method has originally been developed for finite difference methods. It is a special method for incompressible flows. In its original form, the pressure-correction method consists of two steps. In the first step, the momentum equation is solved where some approximation for the pressure is used. The continuity equation is not taken into account and the resulting velocity field may be considered as an intermediate velocity field. In the next step, this intermediate velocity field is projected onto the space of divergence-free vector fields. There are several projection strategies. Typically, a Poisson-type equation for the pressure is solved which is implicitly introduced by the projection step. The pressure correction method is strongly coupled with the type of time discretization.

The projection approach, which goes back to Chorin [12, 13, 14] and Temam [112, 113] is certainly one of the most controversial issues in computation fluid dynamics of incompressible viscous fluid flow, the controversy stemming from the boundary conditions satisfied by the pressure (see also [48, 84, 97, 98, 99] and references therein). The rigorous underlying analysis and understanding is still incomplete (cf. [100, 101]), particularly concerning spurious pressure boundary layers. Additionally, smaller time steps have to be chosen due to the more explicit character of the schemes.

There are two approaches for applying projection methods: continuous in space projection methods (referred to as projection methods in this work) like the methods of Chorin [12, 13, 14], Goda [45], Van Kan [73] which belong to the class of Rothe methods and discrete projection methods such as Turek's (cf. [121]) discrete counterparts of Chorin and Van Kan which belong to the methods of lines class.

So far, little attention has been paid to improving the efficiency of the discrete methods by incorporating other schemes than implicit Euler or Crank-Nicolson or variants thereof. Discrete projection methods based on implicit Euler and Crank-Nicolson scheme have been extensively studied by Turek (cf. [118, 119, 120, 121]).

In the remainder of this chapter, it is shown how the advantages of discrete projection methods can be carried over to BDF and implicit Runge-Kutta methods. The result are more efficient integrators than discrete versions of Chorin and Van Kan. In addition, the techniques suggested in [119] may also be applied.

3.3.1 The pressure correction method - continuous approach

Applying the θ-method to the continuous in space equations (1.8) yields

$$\begin{aligned}\frac{u^{n+1}-u^n}{\Delta t} =& \theta\left(\nu\Delta u^{n+1} - (u^{n+1}\cdot\nabla)u^{n+1} - \nabla p^{n+1} + b^{n+1}\right)\\ &+ (1-\theta)\left(\nu\Delta u^n - (u^n\cdot\nabla)u^n - \nabla p^n + b^n\right) \qquad (3.8a)\\ \nabla\cdot u^{n+1} =& 0 \qquad (3.8b)\end{aligned}$$

provided that u^n is divergence-free. For simplicity we assume Dirichlet boundary conditions

$$u = w \qquad \text{on } \Gamma.$$

Again, $\theta = 0$ yields the (continuous in space) half-explicit Euler method, i.e. p^n is considered as a Lagrange multiplier which keeps the velocity at the new time level, u^{n+1}, divergence-free. In other words, although the velocity seems to be advanced explicitly in time, the pressure at the old time level, p^n, needs to be determined implicitly in order to keep u^{n+1} divergence-free.

Now, because of better stability, consider $\theta = \frac{1}{2}$ and $\theta = 1$. In the first step of the pressure-correction method, the pressure at the new time level, p^{n+1}, in the momentum equation is replaced by some known pressure $\widetilde{p}^{n+1}$. This yields an intermediate velocity field $\widetilde{u}^{n+1}$ satisfying

$$\begin{aligned}\frac{\widetilde{u}^{n+1}-u^n}{\Delta t} &= \theta\left(\nu\Delta\widetilde{u}^{n+1} - (\widetilde{u}^{n+1}\cdot\nabla)\widetilde{u}^{n+1} - \nabla\widetilde{p}^{n+1} + b^{n+1}\right)\\ &\quad + (1-\theta)\left(\nu\Delta u^n - (u^n\cdot\nabla)u^n - \nabla p^n + b^n\right) \qquad (3.9a)\\ \widetilde{u}^{n+1} &= w \qquad \text{on } \Gamma. \qquad (3.9b)\end{aligned}$$

In the second step, $\widetilde{u}^{n+1}$ is projected onto the space of divergence-free vector fields,

$$\begin{aligned}\frac{u^{n+1}-\widetilde{u}^{n+1}}{\Delta t} + \nabla q &= 0 \qquad (3.9c)\\ \nabla\cdot u^{n+1} &= 0 \qquad (3.9d)\\ u^{n+1}\cdot n &= w\cdot n \qquad \text{on } \Gamma, \qquad (3.9e)\end{aligned}$$

where q may be regarded as auxiliary pressure. This projection step is performed by solving the (auxiliary) pressure Poisson equation

$$\Delta q = \frac{1}{\Delta t}\nabla\cdot\widetilde{u}^{n+1}$$

and then performing two explicit updates

$$\begin{aligned}u^{n+1} &= \widetilde{u}^{n+1} - \Delta t\nabla q\\ p^{n+1} &= \widetilde{p}^{n+1} + \frac{q}{\theta}.\end{aligned}$$

Classical approaches are those of Chorin with $\theta = 1$, $\widetilde{p} = 0$ where the auxiliary pressure q is simply the pressure at the new time level t^{n+1} and of Van Kan with $\theta = \frac{1}{2}$, $\widetilde{p} = p^n$ where q is a pressure increment; both using finite differences for the spatial discretization. Chorin's scheme has since been used repeatedly within finite difference [76], finite volume [122], finite element (see [48] and references therein) and spectral element (see [95] and references therein) solvers.

Although there are no boundary conditions for the pressure in the Navier-Stokes equations, boundary conditions for the auxiliary pressure are required in order to solve the pressure Poisson equation. Multiplying (3.9c) by the unit normal vector yields

$$\frac{\partial q}{\partial n} = -\frac{u^{n+1}\cdot n - \widetilde{u}^{n+1}\cdot n}{\Delta t} \qquad \text{on } \Gamma \tag{3.10}$$

which reduces to $\frac{\partial q}{\partial n} = 0$. This boundary condition is most widely used and at the same time has caused controversial discussion ever since in literature (see [100, 101] and references therein).

The correct boundary conditions can be obtained from (3.8) by taking the inner product between the momentum equation and the unit normal vector of the boundary,

$$\frac{\partial p^{n+1}}{\partial n} = (-(u^{n+1}\cdot\nabla)u^{n+1}) + \nu\Delta u^{n+1} + b)\cdot n. \tag{3.11}$$

The right-hand side of this equation is in general not zero and is independent of the discretization parameter Δt, so that errors introduced by (3.10) may be of $\mathcal{O}(1)$. However, imposing the boundary condition (3.11) involves terms at the new time level and therefore leads to a coupled system which is not what we want, since splitting methods are employed in order to get rid of the velocity-pressure coupling.

The simple boundary condition (3.10) works well in practice but prevents splitting methods to be of higher order. Only if the boundary conditions are sufficiently good approximations of the correct boundary conditions (3.11) can continuous splitting methods obtain more than order two. It is shown in [74] how the correct boundary conditions (3.11) can be approximated sufficiently well by using approximations of Adams-Bashforth and Adams-Moulton type (cf. [74]). This approach seems to be tedious and to my knowledge has not been applied to complicated domains yet. Since the issue of boundary conditions for the projection step is one of two major differences between the continuous and the discrete approach, let us have a closer look at this issue.

3.3.2 The pressure correction method - discrete approach

The continuous in space approach described in the previous section is most common in computational fluid dynamics, especially when finite differences or pseudospectral methods

are used for the spatial discretization. The so-called discrete projection method belongs to the methods of lines class. We shall first carry out a semi-discretization, i.e. proceed with spatial discretization while retaining the temporal derivative. Time marching is then treated separately in a second step.

Applying the θ-method to the discrete Navier-Stokes equations yields (3.3). Again, because of better stability, consider $\theta = \frac{1}{2}$ and $\theta = 1$. In the first step of the pressure-correction method, the pressure at the new time level p_h^{n+1} in the momentum equation is replaced by some known pressure $\widetilde{p}_h^{n+1}$. This yields an intermediate velocity field $\widetilde{u}_e^{n+1}$ satisfying

$$M\frac{\widetilde{u}_e^{n+1} - u_e^n}{\Delta t} = \theta\left(D(\widetilde{u}_e^{n+1}) - A(\widetilde{u}_e^{n+1},\widetilde{u}_e^{n+1}) - G\widetilde{p}_h^{n+1} + f^{n+1}\right) \tag{3.12a}$$

$$+ (1-\theta)\left(D(u_e^n) - A(u_e^n,u_e^n) - Gp_h^n + f^n\right). \tag{3.12b}$$

In the second step, $\widetilde{u}_e^{n+1}$ is projected onto the space of divergence-free vector fields,

$$M\frac{u_e^{n+1} - \widetilde{u}_e^{n+1}}{\Delta t} + Gq_h = 0 \tag{3.12c}$$

$$C^T u_e^{n+1} = 0. \tag{3.12d}$$

This projection step is performed by solving the pressure Poisson equation

$$C^T M^{-1} G q_h = \frac{1}{\Delta t} C^T \widetilde{u}^{n+1} \tag{3.13}$$

and then performing two updates

$$u_e^{n+1} = \widetilde{u}_e^{n+1} - \Delta t M^{-1} G q_h$$
$$p_h^{n+1} = \widetilde{p}_h + \frac{q_h}{\theta}.$$

These discrete counterparts of Chorin's method ($\theta = 1$, $\widetilde{p} = 0$) and Van Kan ($\theta = \frac{1}{2}$, $\widetilde{p} = p^n$) have extensively been studied by Turek (cf. [118, 119, 120, 121].

The major differences between continuous and discrete projection methods are twofold. On the one hand, discrete projection methods are easier than continuous projection methods in the sense that boundary conditions for the pressure in the projection step are automatically inherent. On the other hand, discrete projection methods are more difficult in the sense that the pressure Poisson equation (3.13) is significantly more expensive to solve than in the continuous case. At the same time, (3.13) is simpler than (3.7), i.e. the matrix is constant in time and symmetric whenever the forms (2.29) or (2.30) are used.

From a more general point of view, the discrete projection methods introduced above obey the following methodology:

- ☞ Fix an approximate pressure $\widetilde{p}_h$
- ☞ Solve $M\dot{\widetilde{u}}_e = B(\widetilde{u}_e) - G\widetilde{p}_h$ for $\widetilde{u}_e$ where B denotes a discrete Burgers operator
- ☞ Project $\widetilde{u}_e$ onto the discretely divergence-free space
- ☞ Update the pressure, i.e. $p_h^{n+1} = \phi(\widetilde{p}_h, \Delta t, \lambda_h)$ with λ_h as in (3.16) and ϕ depends on the solver in the second step

Using this methodology allows to construct discrete projection methods and classify them according to how the approximate pressure has been chosen, how the discrete Burgers operator has been constructed and solved, and how the projection step has been performed. In this notation (i.e. type of splitting / solver for the velocity step), the discrete implicit projection methods based on the θ-method are

$$\begin{aligned}
\text{Chorin/IEM:} \quad & M\dot{\widetilde{u}}_e = B_{\text{Chorin}}(\widetilde{u}_e), && \widetilde{p}_h = 0 && \phi = \frac{1}{\Delta t}\lambda \\
\text{Van Kan/IEM:} \quad & M\dot{\widetilde{u}}_e = B_{\text{IEM}}(\widetilde{u}_e) - G\widetilde{p}_h, && \widetilde{p}_h = p_h^n && \phi = p_h^n + \frac{1}{\Delta t}\lambda_h = \widetilde{p}_h + \frac{1}{\Delta t}\lambda_h \\
\text{Van Kan/CNS:} \quad & M\dot{\widetilde{u}}_e = B_{\text{CNS}}(\widetilde{u}_e) - G\widetilde{p}_h, && \widetilde{p}_h = p_h^n && \phi = p_h^n + \frac{2}{\Delta t}\lambda_h = \widetilde{p}_h + \frac{2}{\Delta t}\lambda_h
\end{aligned}$$

with λ_h as in (3.16). The method in the middle is, however, rarely used in practice compared to the other two methods. The two possible approaches, namely the fully coupled (FC) approach and the discrete projection (DP) approach, may be summarized as follows:

FC: First treat the nonlinearity by an outer iteration and obtain linear indefinite subproblems which can be solved by a coupled or splitting approach (cf. [120]).

DP: Split the coupled problem and obtain problems in the velocity unknowns (Burgers equation) as well as in the pressure unknowns (pressure Poisson problems). Then treat the nonlinear problems in the velocity unknowns.

3.3.3 Concomitant pressure field

In contrast to the continuous in space equations (1.12) which do not require initial data for the pressure, all discrete projection methods (except those of Chorin type) do require an initial pressure. Given a discrete velocity field u_e, the concomitant pressure field can be obtained by differentiating (2.32b) with respect to time and plugging into (2.32a). This yields

$$C^T M^{-1} G p_h(t) = C^T M^{-1}(-A(u_e, u_e) + D(u_e) + f(t)) - \dot{g}(t).$$

The efficient solution of the pressure Poisson equation is described in section 3.7.2. Note that the issue with boundary conditions for the pressure Poisson equation does, in contrast to chapter 1, not arise here.

3.3.4 Projection step: L^2-projection

This section deals with the treatment of the incompressibility condition (1.12b) in the Navier-Stokes equations by the so-called L^2-projection. The idea is to force the condition $\nabla \cdot u = 0$ that the solution has to verify by projecting (a generally non-divergence-free) predicted value of the solution onto a space of divergence-free vector-valued functions, the projection being in the sense of $L^2(\Omega)$ (see also [44]).

The Helmholtz decomposition theorem states that any vector field satisfying appropriate boundary conditions can be orthogonally decomposed into a solenoidal part, i.e. with zero divergence, and an irrotational part, i.e. with zero curl:

Theorem 3.1 (Helmholtz-Hodge decomposition) *A sufficiently smooth vector field $\widetilde{u}$ on Ω can be uniquely decomposed in the form*

$$\widetilde{u} = u + \nabla\lambda \tag{3.14}$$

where u has zero divergence, i.e. $\nabla \cdot u = 0$, and is parallel to $\partial\Omega$, i.e. $u \cdot n = 0$ on $\partial\Omega$.

A proof can be found in [15, 52]. The best divergence-free approximation in the L^2-sense of a given velocity field $\widetilde{u} \in V_0$ is achieved by the velocity field $u \in V_0$ which minimizes

$$\begin{aligned}
J(u) &= \frac{1}{2}\int_\Omega (u - \widetilde{u}) \cdot (u - \widetilde{u})\, dV \\
&= \frac{1}{2}\int_\Omega u \cdot u\, dV - \int_\Omega \widetilde{u} \cdot u\, dV + \underbrace{\int_\Omega \widetilde{u} \cdot \widetilde{u}\, dV}_{=\text{ const.}}
\end{aligned}$$

subject to the constraint

$$\int_\Omega (\nabla \cdot u) w\, dV = 0$$

$\forall w \in W$. The assigned Lagrange functional is

$$\mathcal{L}(u, \lambda) = \frac{1}{2}\int_\Omega u \cdot u\, dV - \int_\Omega \widetilde{u} \cdot u\, dV + \int_\Omega (\nabla \cdot u)\lambda\, dV$$

which yields the saddle point problem

$$\int_\Omega u \cdot v \, dV + \int_\Omega (\nabla \cdot v)\lambda \, dV = \int_\Omega \widetilde{u} \cdot v \, dV \qquad \forall v \in V_0 \tag{3.15a}$$

$$\int_\Omega (\nabla \cdot u) w \, dV = 0 \qquad \forall w \in W \tag{3.15b}$$

(see also [5]). This is the weak form of the Helmholtz-Hodge decomposition (3.14). Discretizing (3.15) with the tools from section 2.2 yields the discrete L^2-projection at time t,

$$Mu_e + G\lambda_h = M\widetilde{u}_e \tag{3.16a}$$

$$C^T u_e = g(t). \tag{3.16b}$$

Solving (3.16a) for u_e and plugging it into (3.16b) yields

$$\lambda_h = (C^T M^{-1} G)^{-1} C^T \widetilde{u}_e - (C^T M^{-1} G)^{-1} g(t) \tag{3.17}$$

provided that C and G have full rank. Then, (3.16a) and (3.17) yield

$$\begin{aligned} u_e &= \widetilde{u}_e - M^{-1} G \lambda_h \\ &= \widetilde{u}_e - M^{-1} G (C^T M^{-1} G)^{-1} (C^T \widetilde{u}_e - g(t)) \\ &= (I - M^{-1} G (C^T M^{-1} G)^{-1} C^T) \widetilde{u}_e + M^{-1} G (C^T M^{-1} G)^{-1} g(t) \\ &= P_0 \widetilde{u}_e + M^{-1} G (C^T M^{-1} G)^{-1} g(t) \end{aligned}$$

where $P_0 := I - M^{-1} G (C^T M^{-1} G)^{-1} C^T$ is called the L^2-projection matrix. Note that P_0 is indeed a solenoidal projection, i.e.

$$C^T P_0 \widetilde{u}_e = C^T u_e - g(t) = 0$$

and P_0 projects gradients to zero, i.e.

$$P_0 M^{-1} G = 0.$$

For the remainder of this section, suppose that $C = G$. Then P_0 is an orthogonal projection via the discrete L^2-inner product, $(x, y) = x^T M y$. With $Q_0 := I - P_0$,

$$(P_0 \widetilde{u}_e, Q_0 \widetilde{u}_e) = (P_0 \widetilde{u}_e)^T C (C^T M^{-1} C)^{-1} C^T \widetilde{u}_e = (C^T P_0 \widetilde{u}_e)^T (C^T M^{-1} C)^{-1} C^T \widetilde{u}_e = 0$$

because $C^T P_0 \widetilde{u}_e = 0$. The discrete relation

$$\widetilde{u}_e = u_e + M^{-1} C \lambda_h = P_0 \widetilde{u}_e + Q_0 \widetilde{u}_e$$

mimics (3.14). Testing the orthogonality of u_e and $M^{-1} C \lambda_h$, however, yields

$$\begin{aligned} ||\widetilde{u}_e||_0^2 &= (u_e + M^{-1} C \lambda_h)^T M (u_e + M^{-1} C \lambda_h) \\ &= u_e^T M u_e + (M^{-1} C \lambda_h)^T M (M^{-1} C \lambda_h) + 2 u_e^T C \lambda_h \end{aligned}$$

but $u_e^T C \lambda_h = \lambda_h^T C^T u_e = \lambda_h^T g(t)$ and thus we obtain

$$||\widetilde{u}_e||_0^2 = ||u_e||_0^2 + ||M^{-1} C \lambda_h||_0^2 + 2 \lambda_h^T g(t).$$

Hence, L^2-orthogonality between u_e and $M^{-1} C \lambda_h$ is obtained only for $g(t) = 0$ which, in this case, would require $n \cdot u = 0$ on Γ_D (or $\Gamma_D = \emptyset$).

For completeness, we state that there is also a so-called H^1-projection which is equivalent to a Stokes problem while the L^2-projection is equivalent to solving a Darcy flow problem (see also [48]). It is in principle possible to replace all L^2-projections in this work by H^1-projections.

3.4 Discrete multistep projection methods

3.4.1 Multistep methods

BDF methods are multistep methods for the numerical solution of systems of ordinary differential equations $\frac{dy}{dt} = f(t, y)$. In order to apply a BDF method to the differential algebraic equation (3.1), the singular mass matrix $M_e = \mathrm{diag}(M, 0_{N_\mathrm{p} \times N_\mathrm{p}})$ is replaced by $M_e^\varepsilon = \mathrm{diag}(M, \varepsilon I_{N_\mathrm{p} \times N_\mathrm{p}})$ with $\varepsilon > 0$ and the BDF method is formally applied to

$$\frac{dy_e}{dt} = (M_e^\varepsilon)^{-1} F(y_e). \tag{3.18}$$

Multiplication by M_e^ε then yields

$$L_{\Delta t,k}^n (M_e^\varepsilon y_e^{n+1}) = F(y_e).$$

The operator $L_{\Delta t,k}^n$ represents the BDF scheme approximation of $\frac{d}{dt}$,

$$L_{\Delta t,k}^n (M y^{n+1}) = \frac{1}{\Delta t} \sum_{m=0}^{k} \beta_m M y^{n+1-m} \tag{3.19}$$

with appropriate coefficients β_i, $i = 1, \ldots, k$. Then, let $\varepsilon \to 0$ which yields the BDF method applied to the differential algebraic system (3.1). This procedure is known as the ε-embedding method (cf. [56]). Note that for $\varepsilon \to 0$ the equation (3.18) becomes infinitely stiff.

In case of the discrete Navier-Stokes equations (2.32), this results in

$$L^n_{\Delta t,k}(Mu_e^{n+1}) = -A(u_e^{n+1}, u_e^{n+1}) + D(u_e^{n+1}) - Gp_h^{n+1} + f^{n+1} \tag{3.20a}$$
$$C^T u_e^{n+1} = g^{n+1} \tag{3.20b}$$

provided that u_e^n is discretely divergence-free. For $k = 1$ and $\beta_0 = 1$, $\beta_1 = -1$ we get the (fully coupled) implicit Euler method. For $k = 2$ we set $\beta_0 = \frac{3}{2}$, $\beta_1 = -2$, $\beta_2 = \frac{1}{2}$ and for $k = 3$ we use $\beta_0 = \frac{11}{6}$, $\beta_1 = -3$, $\beta_2 = \frac{3}{2}$, $\beta_3 = \frac{1}{3}$ (see [24]). (3.20) represents a fully coupled nonlinear system of equations. Although it is more efficient than the discrete θ-method (if k is sufficiently large), the cost per time step is approximately the same. Therefore, velocity and pressure are decoupled by introducing a projection step.

3.4.2 Velocity-pressure decoupling

In the following, it is shown how velocity and pressure can be decoupled by introducing a projection step in order to reduce the numerical cost per time step. This is a generalization and extension of the continuous in space approach suggested by [61, 62, 63, 64] for the Stokes equations and by [59, 65] for the continuous in space linearized Navier-Stokes equations. These approaches are based on pseudospectral discretizations and are only demonstrated to work on very simple geometries and with homogeneous Dirichlet boundary conditions. Furthermore, boundary conditions for the projection step are chosen to be too simple for this approach, namely homogeneous Neumann boundary conditions (cf. section 3.3.1). In contrast to these approaches, the discrete approach presented here takes care of pressure modes, works on arbitrary geometries in combination with a large variety of physically relevant boundary conditions and the nonlinearity is treated fully implicitly which allows the simulation of non-Newtonian fluid flows. If desired, the nonlinear advection term may also be treated explicitly in the case of Newtonian fluids (see section 3.7.1).

In order to reduce the computational cost, consider the following splitting scheme:

$$\frac{1}{\Delta t}\left(\beta_0 M\widetilde{u}_e^{n+1} + M\sum_{m=1}^{k}\beta_m u_e^{n+1-m}\right) = -A(\widetilde{u}_e^{n+1}, \widetilde{u}_e^{n+1}) + D(\widetilde{u}_e^{n+1}) - G\widetilde{p}_l^{n+1} + f^{n+1} \tag{3.21}$$

with pressure correction

$$\beta_0 M\frac{u_e^{n+1} - \widetilde{u}_e^{n+1}}{\Delta t} + G(p_h^{n+1} - \widetilde{p}_l^{n+1}) = 0$$
$$C^T u_e^{n+1} = g^{n+1}$$

and

$$\widetilde{p}_l^{n+1} = \sum_{m=0}^{l-1}\gamma_m p_h^{n-m} \tag{3.22}$$

with appropriate coefficients γ_m, $m = 0, \ldots, l-1$. For $l = 1$ we have $\gamma_0 = 1$ and for $l = 2$ $\gamma_0 = 2$, $\gamma_1 = -1$, for $l = 3$: $\gamma_0 = 3$, $\gamma_1 = -3$, $\gamma_2 = 1$. For $k = 1$ and $l = 0$ this is the discrete projection method of Chorin type, i.e. Chorin/IEM in section 3.3.2. The projection step is realized, in principle, by solving (3.17), i.e.

$$C^T M^{-1} G\lambda_h = C^T\widetilde{u}_e - g^{n+1}$$

and then performing two updates

$$u_e^{n+1} = \widetilde{u}_e - M^{-1}G\lambda_h$$
$$p_h^{n+1} = \widetilde{p}_h + \frac{\beta_0}{\Delta t}\lambda_h.$$

Using this approach, the nonlinear iteration in order to solve (3.21) is only in the velocity unknowns due to the pressure decoupling, i.e. only "small" linear systems need to be solved. Therefore, the nonlinear iteration is significantly cheaper compared to a fully coupled approach. The linear systems which need to be solved within the nonlinear iteration always have full rank, no matter whether there are pressure modes or not. In other words, pressure modes don't affect the velocity step. In addition, only one "large" linear system or a pressure Poisson equation must be solved per time step in the projection step. If there are pressure modes, they have to be taken care of only once per time step. Theoretically, the approximations u_e^{n+1} are discretely divergence-free. In practice, however, this depends on how the projection step is actually performed (see section 3.7.2).

3.5 One-step methods vs. multistep methods

Discrete multistep methods require information from k previous time steps *at every grid point* from the current time level in order to advance velocity and pressure at the grid points to the new time level. For time-dependent domains, providing this information can be an intricate, tedious, and in many situations inaccurate or even impossible task. Figure 3.4 shows a simple example of a time-dependent domain with a moving mesh. From an Eulerian point of view, at $t = 1.2$ there is no history of the velocity field available at grid points near the left part of the boundary of the moving circle since there has been no fluid at these positions at the previous time steps. The larger the time steps, the more inaccurate and difficult it is to obtain information via extrapolation within the spatial domain of the velocity field at grid points where there has been no fluid at all at previous times

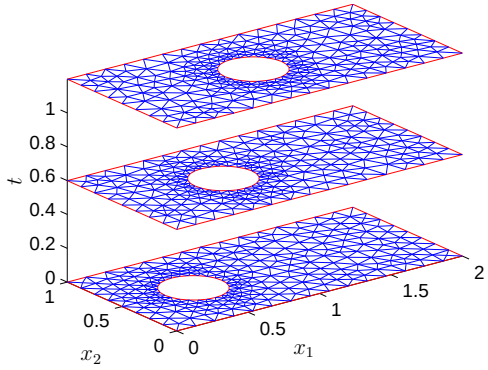


Figure 3.4: Example of a time-dependent domain

steps. The history of fluid particles, i.e. the history of the (moving) grid points themselves (Lagrangian point of view), is of course known, but the history of the velocity field at the (for the moment fixed) grid points is not always known. Discrete multistep methods are therefore awkward or not well suited for transient domains. One-step methods, such as Runge-Kutta methods, require only information from the current time level in order to advance velocity and pressure to the new time level. It will therefore be shown in the next section how the projection approach can be applied to Runge-Kutta methods which then may be used for fluid flow simulations in transient domains.

3.6 Discrete one-step projection methods

3.6.1 Diagonally implicit Runge-Kutta methods

In principle, Runge-Kutta methods, like any other solver for systems of ordinary differential equations, can be used to advance the velocity in the second step of a projection scheme (see p. 70); say of Chorin or Van Kan type. However, this is not very efficient since the velocities at the inner stages don't satisfy the constraint, i.e. they are not divergence-free in general. When solving a differential algebraic equation with Runge-Kutta methods, the solutions at the inner stages should satisfy the constraint. Implicit Runge-Kutta methods require the solution of nonlinear systems with $s \cdot N_{\mathrm{DOFe}}$ unknowns in every time step (see [56]). Computationally more promising than implicit Runge-Kutta methods are diagonally implicit Runge-Kutta methods. They require the solution of s nonlinear systems with N_{DOFe} unknowns per time step. In order to apply a diagonally implicit Runge-Kutta

method to the differential algebraic equation (3.1), the ε-embedding method is again used. Replace the singular mass matrix M_e by

$$M_e^\varepsilon = \begin{pmatrix} M & 0 \\ 0 & \varepsilon I_{N_\mathrm{p} \times N_\mathrm{p}} \end{pmatrix}$$

with $\varepsilon > 0$ and formally apply the Runge-Kutta method to $\frac{dy_e}{dt} = (M_e^\varepsilon)^{-1} F(y_e)$. Multiplication by M_e^ε yields

$$M_e^\varepsilon y_i = M_e^\varepsilon y_e^n + \Delta t \sum_{j=1}^{i} a_{ij} F(y_j) \qquad \text{for } i = 1, \ldots, s$$

$$M_e^\varepsilon y_e^{n+1} = M_e^\varepsilon y_e^n + \Delta t \sum_{i=1}^{s} b_i F(y_i)$$

and finally, let $\varepsilon \to 0$ which gives

$$M u_i = M u_e^n + \Delta t \sum_{j=1}^{i} a_{ij} (-A(u_j, u_j) + D(u_j) + f_j^n) - G p_j) \tag{3.23a}$$

$$C^T u_i = g_i^n \tag{3.23b}$$

for $i = 1, \ldots, s$ with $f_i^n = f(t_n + c_j \Delta t)$ and $g_i^n = f(t_n + c_i \Delta t)$. The new velocity and new pressure are obtained from

$$M u_e^{n+1} = M u_e^n + \Delta t \sum_{i=1}^{s} b_i (-A(u_i, u_i) + D(u_i) + f_i^n) - G p_i) \tag{3.23c}$$

$$p_h^{n+1} = p_h^n + \Delta t \sum_{i=1}^{s} b_i l_i \tag{3.23d}$$

where the l_i are defined by

$$p_i = p_h^n + \Delta t \sum_{j=1}^{s} a_{ij} l_j.$$

In contrast to (3.20), the new velocity u_e^{n+1} is not divergence-free in general. Therefore, an additional projection may be performed.

3.6.2 Velocity-pressure decoupling

It will now be shown how velocity and pressure can be decoupled by introducing a projection step in order to reduce the numerical cost per time step. As in the case of BDF

methods, the idea of discrete projection methods of Chorin or Van Kan type is extended to diagonally implicit Runge-Kutta methods. The approach presented here takes care of pressure modes, works on non-trivial geometries with a large variety of physically relevant boundary conditions and the nonlinearity is treated fully implicitly which allows the simulation of non-Newtonian fluid flows. In the case of Newtonian fluids, the nonlinear advection term may also be treated explicitly (see section 3.7.1).

The fully coupled nonlinear system of equations (3.23) is decoupled by approximating p_i in the ith stage of the method by a known pressure $\widetilde{p}_i$. This yields an intermediate velocity $\widetilde{u}_i$ which is, in general, not divergence-free. By introducing a projection step, the velocity is corrected so that it is discretely divergence-free up to some tolerance and then the pressure is corrected. The principle procedure of the method is shown in algorithm 3.1.

Using this approach, the nonlinear iteration in order to solve (3.24) is only in the velocity unknowns due to the pressure decoupling. Therefore, the nonlinear iteration is significantly cheaper compared to a fully coupled approach. The linear systems which need to be solved within the nonlinear iteration always have full rank independent of the presence of pressure modes. Similar to BDF methods, pressure modes don't affect the velocity step. In addition, only one "large" linear system or a pressure Poisson equation must be solved per inner stage of the method in the projection step. If there are pressure modes, they have to be taken care of only once per time step. This is described in section 3.7.2.

The simplest choice for the approximate pressure $\widetilde{p}_i$ for $i = 1, \dots, s$ is $\widetilde{p}_i := 0$ which corresponds to Chorin's method if the velocity is advanced with the implicit Euler method. Choosing $\widetilde{p}_i := p^n$ for $i = 1, \dots, s$ or $\widetilde{p}_1 := p^n$ and $\widetilde{p}_i := p_{i-1}$ for $s = 2, \dots, s$ is also possible. In general, the approximate pressure at the ith stage should be chosen so that

$$\begin{aligned} p_0 &= p_h^n \\ \widetilde{p}_i &= \sum_{m=0}^{i-1} \gamma_{im} p_m \qquad \text{for } i = 1, \dots, s \end{aligned} \tag{3.26}$$

with appropriate coefficients γ_{im}. In principle, it would be possible to use information from previous time steps in order to obtain a better "guess" for $\widetilde{p}_i$, just like in the case of multistep methods. However, this would make the one-step method become some kind of multistep method, which is not what we want. It turns out that if the order of the extrapolations or interpolations (3.26) is too high, the method may become unstable and therefore the weights γ_m have to be chosen carefully.

3.6.3 Projected velocity-pressure decoupling

Note that although the velocities u_i, $i = 1, \dots, s$, at the intermediate stages are discretely divergence-free (up to some given tolerance), this is generally not true for the approxima-

Algorithm 3.1

STEP 1: For $i = 1, \dots, s$ do:

Step 1: Solve

$$\begin{aligned} M\widetilde{u}_i = Mu_e^n &+ \Delta t \sum_{j=1}^{i-1} a_{ij} \left(-A(t_n + c_j\Delta t, u_j, u_j) + Du_j + f(t_n + c_j\Delta t) - Gp_j\right) \\ &+ \Delta t a_{ii} \left(-A(t_n + c_i\Delta t, \widetilde{u}_i, \widetilde{u}_i) + D\widetilde{u}_i + f(t_n + c_i\Delta t) - G\widetilde{p}_i\right) \end{aligned} \tag{3.24}$$

for $\widetilde{u}_i$ where $\widetilde{p}_i$ is a hopefully good approximation to p_i.

Step 2: Project intermediate velocity $\widetilde{u}_i$ onto the discretely divergence-free space, i.e. solve

$$\begin{aligned} Mu_i + G\lambda &= M\widetilde{u}_i \\ C^T u_i &= g(t_n + c_i\Delta t) \end{aligned}$$

as described in section 3.7.2.

Step 3: Update the pressure

$$p_i := \widetilde{p}_i + \frac{1}{\Delta t a_{ii}}\lambda$$

STEP 2: Compute new velocity, i.e. solve

$$Mu_e^{n+1} = Mu_e^n + \Delta t \sum_{i=1}^{s} b_i \left(-A(t_n + c_i\Delta t, u_i, u_i) + Du_i + f(t_n + c_i\Delta t) - Gp_i\right)$$

for u^{n+1}.

STEP 3: Compute new pressure:

Step 1: Solve

$$p_i = p_h^n + \Delta t \sum_{j=1}^{i} a_{ij} l_j$$

for l_i, $i = 1, \dots, s$ provided that $\mathcal{A} = (a_{ij})_{i,j}$ is invertible.

Step 2: Compute

$$p_h^{n+1} = p_h^n + \Delta t \sum_{i=1}^{s} b_i l_i \tag{3.25}$$

STEP 4: Bump n and goto STEP 1

tions u_h^{n+1} of $u(t_{n+1})$. For applications where this is not acceptable, one may regard the new velocity u_e^{n+1} in the nth time step again as an intermediate velocity $\widetilde{u}_e^{n+1}$ and then force it to be discretely divergence-free by performing an additional projection step. Of course, (3.25) should then no longer be used to compute the new pressure field. Instead, the concomitant pressure field should be computed (see section 3.3.3). The new steps are shown in algorithm 3.2, which replace the corresponding steps of algorithm 3.1.

Algorithm 3.2

STEP 2' Compute new velocity:

Step 1: Solve

$$M\widetilde{u}_e^{n+1} = Mu_e^n + \Delta t \sum_{i=1}^{s} b_i \left(-A(t_n + c_i\Delta t, u_i, u_i) + Du_i + f(t_n + c_i\Delta t) - Gp_i\right)$$

for $\widetilde{u}_e^{n+1}$.

Step 2: Project $\widetilde{u}_e^{n+1}$ onto the discretely divergence-free space, i.e. solve

$$\begin{aligned} Mu_e^{n+1} + G\lambda_h &= M\widetilde{u}_e^{n+1} \\ C^T u_e^{n+1} &= g^{n+1}. \end{aligned}$$

STEP 3': Compute the concomitant pressure field, i.e. solve

$$C^T M^{-1} G p_h^{n+1} = C^T M^{-1}\left(-A(t_{n+1}, u_e^{n+1}, u_e^{n+1}) + Du_e^{n+1} + f(t_{n+1})\right) - \dot{g}^{n+1}$$

for p_h^{n+1}.

The quantity $\dot{g}^{n+1}$ is approximated with central finite differences in a similar way as shown in remark 2.10. The implementation of the projection step and the post projection are described in section 3.7.2.

3.7 The velocity and projection steps

3.7.1 Treatment of the advection term

If not specified otherwise, the advection term is treated fully implicitly for the methods in sections 3.3.2, 3.4.2, 3.6.2, 3.6.3. A common approach in computational fluid dynamics is to treat the nonlinear advection term explicitly while retaining the implicit treatment of the diffusion term. Such methods are called semi-implicit methods, which basically solve

a modified Stokes system. However, this only makes sense as long as the diffusion term is linear, i.e. when the fluid is Newtonian. In order to allow for non-Newtonian fluids, the advection term as well as the possibly nonlinear diffusion term are both treated implicitly. Explicit treatment of the advection term may be forced by setting the field `fem.advection` as described in section A.3.

3.7.2 The discrete projection step

The discrete projection step can either be performed by solving the fully coupled system (3.16), i.e.

$$Ay_e = b \tag{3.27}$$

with

$$A = \begin{pmatrix} M & G \\ C^T & 0_{N_\mathrm{p}\times N_\mathrm{p}} \end{pmatrix}, \qquad y_e = \begin{pmatrix} u_e \\ \lambda_h \end{pmatrix}, \qquad b = \begin{pmatrix} \widetilde{f} \\ g(t) \end{pmatrix}, \qquad \widetilde{f} = M\widetilde{u}_e$$

or by decoupling velocity u_e and auxiliary pressure λ_h,

$$A_p\lambda_h = C^T M^{-1}\widetilde{f} - g \tag{3.28a}$$

$$Mu_e = \widetilde{f} - G\lambda_h \tag{3.28b}$$

with

$$A_p = C^T M^{-1} G.$$

The advantage of this (auxiliary) pressure Schur complement approach is that the linear system (3.28a) is of much smaller dimension than the original fully coupled system (3.27). However, the matrix A_p is dense and cannot be explicity computed. In addition, the system is poorly conditioned on unstructured grids in complicated domains (cf. [48]).

The matrices A and A_p in (3.27) and (3.28a), respectively, can be symmetric or non-symmetric and they can be singular or have full rank, depending on the weak formulation (see sections 2.2.3 and 2.5) and the boundary conditions. These four situations require different solution techniques. The forms (2.30) and (2.29) are most important since they yield symmetric systems.

Iterative solution of the projection step, part I

In order to solve the (auxiliary) pressure Poisson equation (3.28a) efficiently, the pressure Schur complement techniques suggested in [119, 120, 121] are applied and extended to simulations with pressure modes. Let K be a preconditioning operator which is carefully chosen, i.e. "close to M" and cheap, so that $B = (C^T K^{-1} G)^{-1}$ resembles a good preconditioner. Then, the general formulation of the discrete projection step (3.28) reads as follows:

Given λ^0, then solve for $l = 1, \ldots, L$:

$$\widetilde{\lambda}^l = \lambda^{l-1} + \alpha(C^T K^{-1} G)^{-1} \left(C^T M^{-1} (\widetilde{f} - G\lambda^{l-1}) - g \right) \tag{3.29a}$$

and compute

$$\lambda^l := \widetilde{\lambda}^l - P_H (P_H^T \widetilde{\lambda}^l) \tag{3.29b}$$

if there are pressure modes or set

$$\lambda^l := \widetilde{\lambda}^l \tag{3.29c}$$

otherwise. Finally, determine u_e via

$$M u_e = \widetilde{f} - G\lambda^L + \frac{1}{\alpha}(\alpha I - M K^{-1}) G(\lambda^L - \lambda^{L-1}). \tag{3.29d}$$

The iteration counter $L \geq 1$ is either explicitly chosen by the user, e.g. as a fixed number, or by a stopping criterion, i.e. until the residual drops below some chosen tolerance. α is a relaxation parameter; typically $\alpha = 2$.

This scheme is a (relaxed by α) Richardson iteration for the Schur complement equation (3.28), with the special preconditioner $C^T K^{-1} G$ and an orthogonalization step. Some properties of the discrete projection step in the case of no pressure modes are summarized in the following proposition.

Proposition 3.2 *Suppose there are no pressure modes.*

1. *For $K = M$ and $\alpha = 1$, the additive term in* (3.29d) *vanishes and the solution is obtained after $L = 1$ iteration.*

2. *The solution u_e in* (3.29d) *is discretely divergence free, i.e.*

$$C^T u_e = g.$$

Proof. For $K = M$, (3.29d) yields (3.28b). To prove the second statement, equation (3.29a) for $l = L$ yields

$$\begin{aligned} \frac{1}{\alpha}(C^T K^{-1} G)(\lambda^L - \lambda^{L-1}) &= C^T M^{-1} \left(\widetilde{f} - G\lambda^{L-1} \right) - g \\ &= -C^T M^{-1} G \lambda^{L-1} + C^T M^{-1} \widetilde{f} - g. \end{aligned}$$

Multiplication of (3.29d) from left by $C^T M^{-1}$ yields

$$\begin{aligned} C^T u_e &= C^T M^{-1} \widetilde{f} - C^T M^{-1} G \lambda^L + \frac{1}{\alpha} C^T \left(\alpha M^{-1} - K^{-1} \right) G(\lambda^L - \lambda^{L-1}) \\ &= C^T M^{-1} \widetilde{f} - C^T M^{-1} G \lambda^L + C^T M^{-1} G(\lambda^L - \lambda^{L-1}) - \frac{1}{\alpha} C^T K^{-1} G(\lambda^L - \lambda^{L-1}) \\ &= C^T M^{-1} \widetilde{f} - C^T M^{-1} G \lambda^{L-1} - \frac{1}{\alpha} C^T K^{-1} G(\lambda^L - \lambda^{L-1}) \\ &= C^T M^{-1} \widetilde{f} - C^T M^{-1} G \lambda^{L-1} + C^T M^{-1} G \lambda^{L-1} - C^T M^{-1} \widetilde{f} + g \\ &= g. \end{aligned}$$

◇

The operator K^{-1} should be a good approximation to M^{-1} and cheap to apply. A good choice is

$$K = M_l$$

where the lumped mass matrix M_l is defined as

$$M_{l_{ii}} = \sum_{j=1}^{N_{\mathrm{Te}}} M_{ij}, \qquad i = 1, \ldots, N_{\mathrm{Te}}.$$

The simplest choice $K = I$ doesn't work for most examples presented in this work.

The convergence of the iteration scheme (3.29) applied to the test problem of section 3.1.1 has been studied. The numerical tests show that the performance of the iteration scheme is superior for problems with hydrostatic pressure mode compared to problems without hydrostatic pressure mode if the approximations within the iteration are orthogonalized with respect to the pressure mode. The results of the projection step applied to the test problem are shown in section 3.7.6.

When the symmetric forms (2.29) or (2.30) are used, then the conjugate gradient method may be applied instead. If not specified otherwise, it is assumed that $G = C$ in the following so that A and A_p become symmetric.

Iterative solution of the projection step, part II

The pressure Poisson equation (3.28a) can also be solved using the conjugate gradient method. The preconditioned conjugate gradient method (PCG) can be viewed as a conjugate gradient method applied to the preconditioned system

$$BAx = Bf. \tag{3.31}$$

If $A, B \in \mathbb{R}^{N_{\mathrm{p}} \times N_{\mathrm{p}}}$ are symmetric positive definite and B is a preconditioner for A, i.e. $B \approx A^{-1}$, then

$$||x - x_k||_A \leq 2 \left(\frac{\sqrt{\kappa(BA)} - 1}{\sqrt{\kappa(BA)} + 1} \right)^k ||x - x_0||_A, \tag{3.32}$$

which implies that PCG converges faster with smaller condition number $\kappa(BA)$ (see [46]). Even more important are cluster effects around 1, i.e. only very few eigenvalues of BA are close to zero or very large compared to the eigenvalues of the original matrix A. From (3.31) and the convergence estimate (3.32), it follows that the efficiency of the PCG method depends on two main factors: the action of B and the size of $\kappa(BA)$ or the eigenvalues of BA not clustered around 1. Hence, a good preconditioner should have the properties that the action of B is relatively cheap to compute and that $\kappa(BA)$ is relatively small w.r.t. $\kappa(A)$. Nevertheless, the algorithm described in part I seems to yield better results.

In order to handle the situation where A_p is singular, the PCG method is equipped with an additional projection in order to keep the solution orthogonal to a possible hydrostatic pressure mode. This PCG algorithm for (3.28a), i.e. $Ax = b$ with $A := A_p$, $x := \lambda_h$, and $b := C^T M^{-1} \widetilde{f} - g$, is shown in algorithm 3.3. Similar to the previous approach, $K = M_l$ and $B = (C^T K^{-1} C)^{-1}$ is used as a preconditioner. Algebraic multigrid is used to apply the preconditioner (see section 3.7.5). In addition, algebraic multigrid is also used to perform the matrix-vector multiplications within the conjugate gradient method. Since the matrix $A = CM^{-1}C$ is dense, the matrix-vector products $y := Ax$ in algorithm 3.3 are computed by first solving $Mz = Cx$ (see section 3.7.5) and then computing $y = C^T z$. In other words, for every matrix-vector multiplication within the PCG method, a linear system has to be solved. In a second step, u_e is computed via (3.28a).

Algorithm 3.3 AMG-preconditioned CG for (3.28a), solve $Ax = b$

STEP 1: Select starting vector $x_0 \in \mathbb{R}^{N_\mathrm{p}}$ and tolerance $\mathtt{tol} > 0$.

STEP 2: Compute $r_0 = Ax_0 - b$, $s_0 = Br_0$ and initialize $m = 0$, $d_0 = s_0$, $\delta_0 = r_0^T s_0$.

STEP 3: Compute the approximate solution as follows:

while $||r_k|| \geq \mathtt{tol}$ and $m < m_{\max}$ **do**
- Compute $\delta'_m = d_m^T A d_m$
- Set $\alpha_m = \delta_m / \delta'_m$
- Set $\widetilde{x}_{m+1} = x_m - \alpha_m d_m$
- Compute $x_{m+1} = \widetilde{x} - p_H (p_H^T \widetilde{x})$
- Compute $r_{m+1} = r_m - \alpha_m A d_m$
- Compute $s_{m+1} = B r_{m+1}$
- Set $\delta_{m+1} = r_{m+1}^T s_{m+1}$
- Set $\beta_m = \delta_{m+1} / \delta_m$
- Set $d_{m+1} = s_{m+1} + \beta_m d_m$
- Bump m

end while

Direct solution of the projection step

If there are no pressure modes, A can be factored as

$$A = \begin{pmatrix} R^T & 0 \\ S^T & T^T \end{pmatrix} \begin{pmatrix} R & S \\ 0 & -T \end{pmatrix}$$

by applying two Cholesky decompositions: first, compute the Cholesky decomposition of M,

$$R^T R := M,$$

then solve

$$R^T S = C$$

for S and then compute the Cholesky decomposition

$$T^T T := S^T S.$$

If there are pressure modes as in the test problem of section 3.1.1, the rank deficiency may be removed as shown in (3.6). Applying a LU-decomposition with reverse Cuthill-McKee reordering works in principle, but the fill-in is prohibitive and only problems on very coarse meshes can be solved as already mentioned in section 3.2. Therefore, iterative methods as described above are strongly recommended whenever there are pressure modes present, even on coarse grids.

3.7.3 A remark on robustness

It would be ideal to choose the weights in (3.22) and (3.26) so that $\widetilde{p}_l^{n+1}$ and $\widetilde{p}_i$ are high order extrapolations (and for some Runge-Kutta methods also possibly interpolations) to $p^{n+1} \approx p(t_{n+1})$ and $p_i \approx p(t_n + c_i \Delta t)$, respectively.

The most critical point for higher order extrapolation techniques is that the numerical computations show a strong coupling between robustness of the resulting time stepping scheme with the underlying shape of the spatial mesh and the accuracy of the projection step. While these higher order extrapolation techniques work well on meshes without complex geometrical details (an accurate projection is then also easier to obtain due to lower condition numbers), the situation may be different in complicated geometries with highly unstructured meshes with largely varying sizes. Then the decoupled solution strategies with high order extrapolation are more prone to instabilities compared to those not using extrapolations of the highest possible order, since large amplification factors for certain components of the solution vector may arise which can lead to instabilities during evolution of time. For perturbed data, the error of the interpolation polynomial is estimated by the following theorem.

Theorem 3.3 *Let $x \in \Omega$ and let $\bar{p}(\cdot, x) : \mathbb{R} \to \mathbb{R}$ and $\widehat{p}(\cdot, x) : \mathbb{R} \to \mathbb{R}$ be the Lagrange interpolating polynomials defined by $(t_n + c_i\Delta t, \bar{p}_i(x))$ and $(t_n + c_i\Delta t, \widehat{p}_i(x))$, respectively, for $i = 0, \ldots, m$. Then*

$$\max_{t\in[t_n,t_{n+1}]} |\bar{p}(t,x) - \widehat{p}(t,x)| \leq \Lambda_m \max_{i=0,\ldots,m} |\bar{p}_i(x) - \widehat{p}_i(x)|$$

with the Lebesgue constant Λ_m.

A proof can be found in [69]. Since the Lebesgue constant grows exponentially with the number of nodes used for the interpolation, the error introduced by inaccurate projection steps can be amplified significantly.

If instabilities occur, the order of the extrapolation must be reduced. The numerical computations in various geometries have shown that doing so prevents instabilities. However, the decoupled BDF and Runge-Kutta methods of sections 3.4 and 3.6 have worked well in the presented test examples, although the decoupled BDF approach seems to be more stable than the decoupled Runge-Kutta approach. Numerical experiments show that algorithm 3.1 works best if pressure modes are present. If there are no pressure modes, the projected velocity-pressure decoupling approach should be used.

3.7.4 Treatment of the nonlinear subproblems

For non-Newtonian fluids or for Newtonian fluids where the advection term is treated implicitly (cf. section 3.7.1), the methods in sections 3.3.2 3.4.2, 3.6.2, and 3.6.3 require the solution of a nonlinear system of equations in the eliminated velocity unknowns in every time step or in every stage of the method. These nonlinear equations are solved by the simplified Newton method. The tolerance for Newton's method is chosen to be $NliTol = NliTol(\Delta t) = c\Delta t$ with $c \in \mathbb{R}$. The solution of the linear systems within Newton's method is described in the next section.

3.7.5 Treatment of the linear subproblems by AMG

The projection methods presented in sections 3.4.2, 3.6.2, and 3.6.3 as well as the iterative solution of the discrete projection step in section 3.7.2 require the solution of linear systems

$$Mu = b \tag{3.33}$$

and

$$(M + c_1\Delta t J_A - c_2\Delta t J_D))u = b, \qquad c_1, c_2 \in \mathbb{R}. \tag{3.34}$$

These linear subproblems are solved with an algebraic multigrid (AMG) solver (cf. [7, 30, 66, 110, 111]) based on the Ruge-Stüben algorithm (cf [102]). In contrast to geometric

multigrid methods, AMG requires no a priori given hierarchy of coarse grids. In fact, the construction of a problem-dependent hierarchy – including the coarsening process itself, the transfer operators as well as the coarse-grid operators – is part of the AMG algorithm, based solely on algebraic information contained in the given system of equations (cf. [116]).

Geometrically oriented approaches, however, can hardly cope with the complex geometries occurring in simulations of real-life problems. There is generally no natural grid hierarchy which could easily be exploited. But even if there was such a hierarchy, the coarsest level would still be required to be fine enough to resolve the geometry to some extend. For industrially relevant configurations, such coarse grids would still be much too fine for efficient multilevel solutions (cf. [116]). For example, there is typically no hierarchy of coarse grids a priori available for grids coming form an unknown source. Even in cases where a hierarchy of coarse grids is available, remeshing due to moving meshes in time-dependent domains may destroy a coarse grid hierarchy.

In principle, it is possible to generate coarse grids from any given fine grid by applying an automatic coarsening algorithm, e.g., Femlab®'s meshscale algorithm. A geometric multigrid solver may then be applied. However, AMG is the more elegant and more efficient approach for solving the linear subproblems. For example, the performance of AMG within the context of finite differences on complicated domains has been studied in [51]. It is shown in the following that AMG also works very well on the unstructured grids in all numerical simulations presented in this work.

Although AMG has originally been developed for symmetric, positive definite problems, it can also be applied to non-symmetric systems. However, the convergence behavior is not clear in this case. The performance of AMG applied to non-symmetric convection-dominated systems is studied in the following. However, it should be said that at this point, AMG for non-linear problems is still a field of major ongoing research and there is no well-settled approach known yet.

If not specified otherwise, Gauß-Seidel smoothing is used within AMG V-cycles with ν_1 pre-smoothing relaxations and ν_2 post-smoothing relaxations (abbreviated by AMG(ν_1,ν_2) in the following), typically $\nu_1 = \nu_2 = 2$. The grid coarsening is performed until the number of unknowns on the coarsest grid drops below 50. A direct method is then used to solve on the coarsest level.

Let d^m, $m \in \mathbb{N}\backslash\{0\}$, denote the defect of AMG(ν_1,ν_2) after m V-cycles. Then the defect reduction factors are defined as

$$q^{(m)} := \frac{||d^m||}{||d^{m-1}||}.$$

If not specified otherwise, the Euclidean norm is used. The convergence factor ρ of the

AMG(ν_1,ν_2) method can be estimated by the average defect reduction factor $\widehat{q}^{(m)}$, i.e.

$$\rho \approx \widehat{q}^{(m)} := \sqrt[m]{q^{(m)}q^{(m-1)}\cdots q^{(1)}} = \sqrt[m]{\frac{||d^m||}{||d^0||}}$$

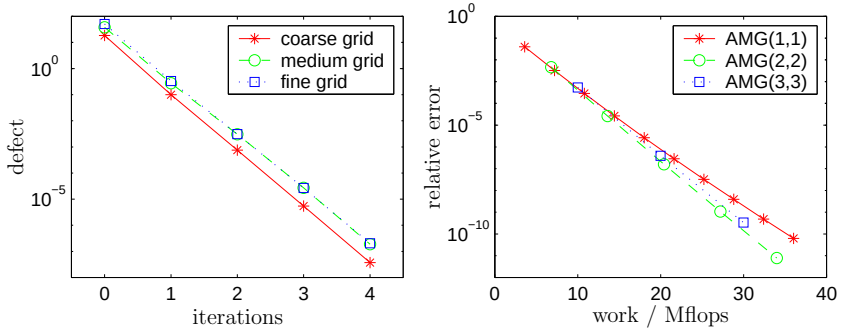
(see also [116]).

The convergence histories and rates of AMG(2,2) applied to (3.33) on different grids of the first test problem (see section 3.1.1) are shown in table 3.2 and figure 3.5(a). It can be seen that the convergence is essentially grid independent. Note that the different grids are not regular refinements of each other and therefore true grid independent convergence can not be expected.

grid	$q^{(1)}$	$q^{(2)}$	$q^{(3)}$	$q^{(4)}$	$\widehat{q}^{(4)} \approx \rho$
coarse	0.0051	0.0074	0.0075	0.0066	0.0066
medium	0.0072	0.0111	0.0090	0.0070	0.0084
fine	0.0062	0.0092	0.0085	0.0074	0.0077

Table 3.2: Convergence rates on different grids

The efficiency of AMG(ν_1,ν_2) applied to (3.33) on the fine grid of section 3.1.1 is shown in figure 3.5(b). It turns out that $\nu_1 = \nu_2 = 2$ is the optimal choice for this test problem. The performance of AMG(1,2), AMG(2,1), AMG(3,2), AMG(2,3), etc. has been found to be inferior to that of AMG(2,2).



(a) Grid independent convergence of AMG

(b) Efficiency of AMG

Figure 3.5: AMG convergence

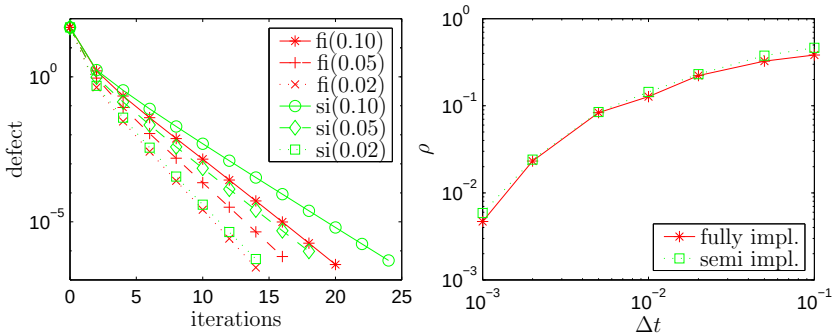
AMG in the nonlinear iteration

The convergence histories of AMG(2,2) applied to the linear systems (3.34) occurring within the (simplified) Newton method have been studied on the finest grid of the test problem. The convergence history of AMG depends on the temporal step size of the underlying time integration scheme and whether the advection term is treated fully implicitly or explicitly, as in the case of semi-implicit methods. The results are shown in figure 3.6(a) for different step sizes. The convergence rates are shown in table 3.3. Figure 3.6(b) shows

	$\Delta t = 0.1$	0.05	0.02	0.01	0.005	0.002	0.001
fully implicit	0.3832	0.3260	0.2219	0.1272	0.0834	0.0232	0.0047
semi implicit	0.4638	0.3764	0.2305	0.1446	0.0847	0.0241	0.0059

Table 3.3: Convergence rates $\rho(\Delta t)$ for different step sizes

the dependence of the convergence factor on the temporal step size for both, fully implicit and semi-implicit methods. These numerical tests show that the convergence factors are



(a) Convergence depending on the step size (fi(Δt)=fully implicit, si(Δt)=semi implicit)

(b) Convergence factor depending on the step size

Figure 3.6: AMG convergence in the velocity step

excellent for both fully implicit and semi implicit treatment of the velocity step. In fact, as the time step size decreases, the efficiency of AMG increases and therefore contributes to a lower increase of numerical work of the time stepping schemes for small step sizes. The performance of AMG in three dimensions is studied in section 5.4.

3.7.6 Implementation

Several discrete projection methods have been implemented. The discrete multistep projection methods have been implemented in the codes SpexBDF(k,l) with k and l as defined in section 3.4.2. An example of a discrete diagonally implicit Runge-Kutta projection method as described in section 3.6.2 has been implemented in the code SpexSDIRK which is based on the Runge-Kutta method shown in table 3.4 (cf. [55]). The coefficients γ_{im} in (3.26) are

γ	γ	
$1-\gamma$	$1-2\gamma$	γ
	$\frac{1}{2}$	$\frac{1}{2}$

Table 3.4: SDIRK method with $\gamma = \frac{3+\sqrt{3}}{6}$

defined as $\gamma_{10} = 1$, $\gamma_{20} = 1-(1-\gamma)/\gamma$, $\gamma_{21} = (1-\gamma)/\gamma$, or $\gamma_{10} = 1$, $\gamma_{20} = 0$, and $\gamma_{21} = 1$ which yields a more robust scheme (cf. section 3.7.3).

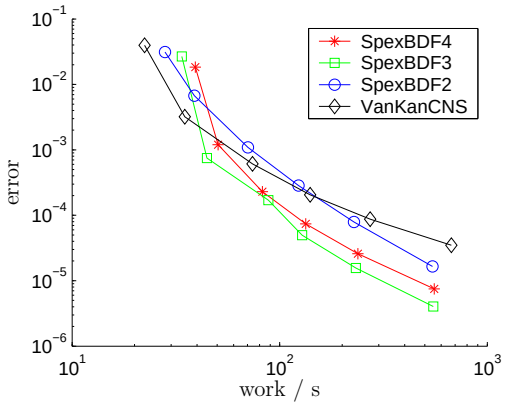


Figure 3.7: Work vs. accuracy

The performance of the discrete multistep projection methods compared to the discrete version of Van Kan's projection method, which is one of the most popular and most efficient methods used in practice (cf. [118, 120, 121]), applied to the test problem of section 3.1.1, is shown in figure 3.7. The error has been measured in the infinity norm. It can be seen that SpexBDF(3,2) is the most efficient code. This is due to the fact that SpexBDF(4,l) is unstable for $l = 3$ and therefore the order of the extrapolation had to be

decreased, i.e. SpexBDF(4,2) was used instead. Nevertheless, this numerical example shows the advantages, the good performance of SpexBDF(3,2), and the limits of the velocity-pressure decoupling approach, i.e. SpexBDF(4,2) is less efficient than SpexBDF(3,2).

The convergence behavior of the projection step is shown in figure 3.8. The results of this simulation are illustrated in terms of the magnitude of the velocity and a streamline plot in figure 3.9.

Multistep projection methods are typically more efficient than Runge-Kutta projection methods due to lower numerical cost per time step. However, as mentioned in section 3.5, there are situations where multistep methods are awkward, nonpractical, and difficult to apply and Runge-Kutta methods should therefore be used. Such an example is shown in sections 5.2 and 5.3.

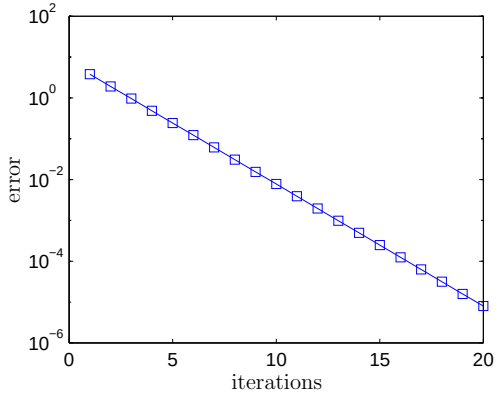
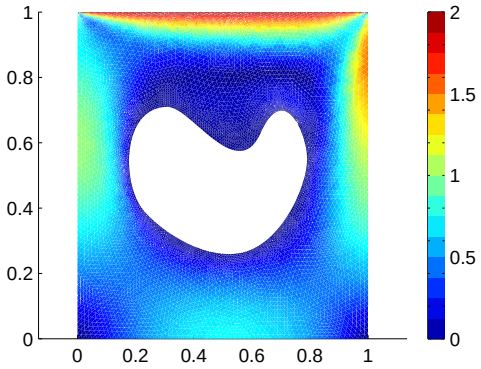
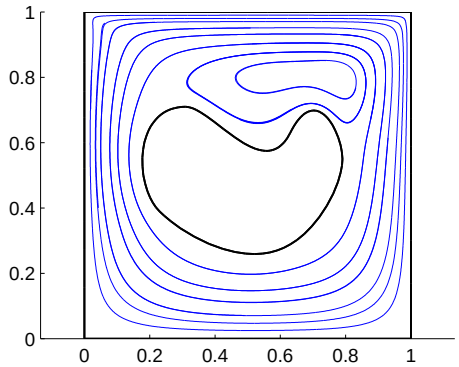


Figure 3.8: Convergence of projection step



(a) Magnitude of velocity



(b) Streamlines

Figure 3.9: Flow of the test problem

Chapter 4

Extensions

The simulation techniques described in the previous chapters belong to the field of direct numerical simulations. This means that the fluid motion is resolved numerically and the forces which move the fluid particles are computed rather than modeled (cf. [70, 72]).

This chapter discusses the simulation of more complicated fluid flows, namely completely instationary flows within transient domains. Examples of fluid flows in transient domains are those with free boundaries such as water in a river, multi-fluid flows such as water-oil mixtures, multi-phase flows such as gas-liquid mixtures, or problems involving fluid-structure interaction. The methods presented in this chapter are based on body and boundary fitted moving unstructured grids. Such grids give optimal spatial resolution, but may yield poor condition numbers of the linear systems in the projection step, for example. The moving boundaries are computed by tracking fluid particles on the boundaries. This kind of simulation is called Lagrangian. The fluid equations are then solved in every grid point as described in the previous chapters, i.e. in their Eulerian form.

The coupling between the fluid and a structure can be twofold, i.e. one way or two way: The coupling is one way if the motion of the structure influences the fluid flow but the action of the fluid flow upon the structure is negligible. A rotating fan or a ship's propeller are examples of fluid structure interaction problems with one-way coupling. The situation is more complicated when the motion of the structure influences the fluid flow and at the same time the structure's motion is influenced by the fluid flow. This is called a two-way coupling. Examples are wind power plants and a boulder sinking in water.

In the following, extensions of the solvers to free boundary problems as well as fluid-structure interaction, particularly solid-liquid flows, with two-way coupling are presented. In solid-liquid flows, the structures (also denoted as moving obstacles) are moved by Newton's laws under the action of hydrodynamic forces computed from the direct numerical solution of the fluid equations.

To perform a direct solution in the above sense, therefore, one must simultaneously integrate the Navier-Stokes equations and the equations of rigid-body motions. These

equations are coupled through no-slip condition on the moving obstacles' boundaries, and through hydrodynamic forces and torques that appear in the equations of rigid-body motion (cf. [72]). These hydrodynamic forces and torques must, of course, be those arising from the computed motion of the fluid, and so are not known in advance, but only as the integration proceeds. It is crucial that no approximation of these forces and torques be made – other than due to the numerical discretization itself – so that the overall simulation will yield a solution of the exact coupled initial value problem up to numerical truncation error.

Accurate modeling of interfacial flows such as fluid structure interaction, and free boundary flows demands a realistic representation of interface and boundary topology. Therefore, boundary and body fitted unstructured grids are used to resolve the transient topology. The numerical techniques presented in the previous chapters have been extended to solve and simulate fluid-structure interaction problems as well as free boundary problems or possibly a combination of both. These extensions open the gate to simulating highly interesting non-standard fluid flow problems. The fluid-structure interaction process itself, the coupling with the Navier-Stokes equations, and the implementation in particular are rather complicated, elaborate, and expensive compared to simulations in stationary domains. The same can be said about simulations with free boundaries. In the following two sections, only the basic ideas of the algorithms are presented and we abstain from describing details of the implementation, data structure, etc.

4.1 Fluid-structure interaction

As an object moves through a fluid, the viscosity of the fluid acts on the moving object with a force that resists the motion of the object. At the same time, the fluid flow depends on the moving obstacle. The fluid velocity on each structure boundary must be constrained to match the rigid-body motion of the structure.

Two recent examples of a fluid-structure interaction simulation can be found in [9, 10]. Their approach is based on a finite differences discretization with staggered grid and Chorin/EEM is used for the time integration. The approach presented here is based on body fitted unstructured moving grids for optimal spatial resolution and implicit or semi-implicit discrete projection methods.

4.1.1 Initial boundary value problem

In order to simulate the motion of $N_{\text{fsi}} \in \mathbb{N}$ moving obstacles within a fluid, the Navier-Stokes equations are coupled with the equations for rigid body motions. The initial boundary value problem for a fluid-structure interaction problem is therefore given by the complete initial boundary value problem (1.12) augmented by the following equations

for $i = 1, \ldots, N_{\text{fsi}}$ moving structures $\text{fsi}(i)$,

$$m_{\text{fsi}(i)} \frac{du_{\text{fsi}(i)}}{dt} = m_{\text{fsi}(i)} g + F_{\text{fsi}(i)}(u) \tag{4.1a}$$

$$\frac{dx_{\text{fsi}(i)}}{dt} = u_{\text{fsi}(i)} \tag{4.1b}$$

$$I_{\text{fsi}(i)} \frac{d\omega_{\text{fsi}(i)}}{dt} = T_{\text{fsi}(i)}(u) \tag{4.1c}$$

$$\frac{d\theta_{\text{fsi}(i)}}{dt} = \omega_{\text{fsi}(i)} \tag{4.1d}$$

$$F_{\text{fsi}(i)}(u) = \int_{\Gamma_{\text{struct}}} (-pI_{d\times d} + \eta(\nabla u + (\nabla u)^T))n \tag{4.1e}$$

$$T_{\text{fsi}(i)}(u) = \int_{\Gamma_{\text{struct}}} r_{\text{fsi}(i)} \tau^T (-pI_{d\times d} + \eta(\nabla u + (\nabla u)^T))n \tag{4.1f}$$

$$u = u_{\text{fsi}(i)} + \omega_{\text{fsi}(i)} \times r_{\text{fsi}(i)} \qquad \text{on } \Gamma_{\text{fsi}(i)}(t) \tag{4.1g}$$

$$u_{\text{fsi}(i)}(0) = u_{\text{fsi}(i),0} \tag{4.1h}$$

$$x_{\text{fsi}(i)}(0) = x_{\text{fsi}(i),0} \tag{4.1i}$$

$$\theta_{\text{fsi}(i)}(0) = \theta_{\text{fsi}(i),0} \tag{4.1j}$$

$$\omega_{\text{fsi}(i)}(0) = \omega_{\text{fsi}(i),0}. \tag{4.1k}$$

Mass, velocity of the center of mass, position of the center of mass, moment of inertia, angular velocity, and angle of rotation of the ith structure are denoted by $m_{\text{fsi}(i)}$, $u_{\text{fsi}(i)}$, $x_{\text{fsi}(i)}$, $I_{\text{fsi}(i)}$, $\omega_{\text{fsi}(i)}$, and $\theta_{\text{fsi}(i)}$, respectively, and gravity (or any other type of body force) is denoted by g. Equations (4.1e) and (4.1f) represent the force and torque exerted upon the ith structure by the flow. Most important for the simulation is the fluid-structure coupling expressed by the transient no-slip boundary conditions (4.1g). The initial boundary value problem (4.1) describes fluid-structure interaction problems as long as there are no collisions between the moving structures.

4.1.2 (Basic) Algorithm

The idea of body fitted grids is that the nodes on the structure are assumed to move with the obstacle. At each time step, the grid is updated according to the motion of the obstacle. Then a new grid is generated using the new edge elements in order to prevent distortion of the grid and the flow field is projected onto the new grid. The simulation of fluid flow in a time-dependent domain is significantly more expensive since all matrices and vectors are time-dependent. Furthermore, the grid generation within the simulation is very expensive.

The initial boundary value problem (1.12) and (4.1) is solved by the following splitting procedure: first, the fluid equations are solved subject to the correct boundary conditions

on the moving structure which couple fluid motion and rigid body motion. Then, the equations of rigid body motion are integrated. This process is repeated while at the same time the grid is constantly updated.

The main steps of the algorithm are as follows:

- ☞ Compute the new velocity field, i.e. perform one time integration step
- ☞ Compute force and torque exerted upon the structure by the fluid flow
- ☞ Integrate the equations of rigid body motion
- ☞ Compute displacement of moving structure
- ☞ Move nodes on structure
- ☞ Generate new geometry
- ☞ Check mesh quality and possibly compute new triangularization and interpolate the solution to the new grid
- ☞ Repeat this procedure

4.2 Free boundary problems

To treat incompressible, free surface flows, the marker and cell (MAC) method was developed by Harlow and Welch (cf. [58]) as a variation of the particle in cell (PIC) method (cf. [57]). The MAC method is based on a finite difference approach and was the first successful technique for incompressible flows with free boundaries. Particles were used as markers to locate the material in the mesh and, consequently, to define the location of the free-surface. Recent implementations of the MAC method can be found in [10, 50].

In contrast to the popular MAC schemes, the methods presented in this chapter are based on body and boundary fitted moving unstructured grids, also known as interface tracking or boundary tracking methods (cf. [72]). The moving boundaries are computed by tracking fluid particles on the boundaries. If necessary, fluid particles on the boundary are injected, i.e. the boundary is refined by increasing the number of boundary nodes and edge elements on the free boundary.

4.2.1 Boundary condition

The boundary condition on the free boundary is a traction boundary condition, where the traction is given by the surface tension. For $d = 2$, the free boundary condition reads

$$(\eta(\nabla u + (\nabla u)^T) - pI_{2\times 2})n = f_{\text{st}}$$

where the surface tension force is given by

$$f_{\text{st}} = c\frac{\kappa}{R}n$$

where n is the unit outward normal vector in the free boundary, R the radius describing the curvature of the free boundary, and κ a constant depending on the fluid (cf. [48, 50]). $c = \pm 1$ or 0 if concave, convex, or no curvature at all. The discretized surface tension is represented as a time-dependent surface tension vector. Its dimension depends on the number of velocity nodes on the free boundary and therefore is, in general, also time-dependent. The surface tension vector is recomputed every time step.

4.2.2 Computation of surface tension

The computation of the surface tension on the free boundary proceeds in three steps. First, the surface tension in the mesh nodes on the free boundary is approximated. Second, the surface tension at Lagrange nodes which are not grid points on the free boundary is computed. With this information, the surface tension can be computed in every point on every edge element of the free boundary. The process of computing the surface tension for P_2 velocity elements is described in the following.

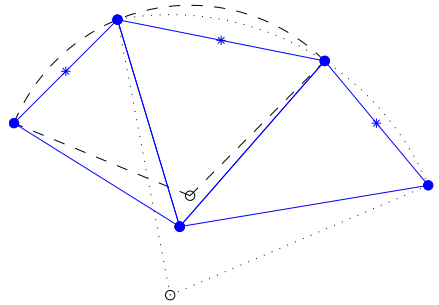


Figure 4.1: Computation of surface tension

Surface tension at vertices: In a first step, the nodes of the triangularization on the free boundary are determined. Figure 4.1 shows three mesh elements (solid lines and solid grid points) with edges on the free boundary (defined by the three upper solid edge elements). Second, the surface tension in these mesh nodes, i.e. vertices of the finite elements (solid dots on the free boundary) is computed via the curvature of the surface. Since the free boundary is polygonal, the curvature in the vertices is computed as the inverse of the radius of the circle trough a given vertex and its two neighboring vertices. Figure 4.1 shows parts of the two curvature circles (dashed and dotted arcs) for two vertices on the free boundary, the corresponding centers ($\circ$), and the corresponding radii (straight dashed and dotted lines). The exact procedure is described in algorithm 4.1.

Surface tension at inner Lagrange nodes: For finite elements of order higher than one, the surface tension also needs to be computed at the inner nodes on boundary edge elements of the free boundary, i.e. inner Lagrange points of boundary edge elements, denoted by $\star$ in figure 4.1. This is done by linear interpolation for every single edge element of the free boundary.

Surface tension on edge element: Finally, the surface tension needs to be defined everywhere on the edge elements and not just at the Lagrange points on the free boundary. This is needed for the numerical evaluation of the weak form via numerical quadrature, for example. Therefore, the surface tension at Lagrange points is used for interpolating the surface tension on every edge element using the velocity shape functions.

Algorithm 4.1 Surface tension at free boundary mesh nodes

Let $\mathtt{P}(:,\mathrm{fbn})$ be the n_{fbn} mesh nodes on the free boundary, so that $\mathtt{P}(:,\mathrm{fbn}(i-1))$ and $\mathtt{P}(:,\mathrm{fbn}(i+1))$ are neighbors of $\mathtt{P}(:,\mathrm{fbn}(i))$.

for $i = 1 : n_{\mathrm{fbn}}$ **do**

Compute outward normal vector n on boundary at $b = \mathtt{P}(:,\mathrm{fbn}(i))$.

Let $a = \mathtt{P}(:,\mathrm{fbn}(i-1))$ and $c = \mathtt{P}(:,\mathrm{fbn}(i+1))$.

Solve $Ax = y$ with

$$A = \begin{pmatrix} b^T - a^T \\ c^T - b^T \end{pmatrix} \quad \text{and} \quad y = \frac{1}{2}\begin{pmatrix} b^T b - a^T a \\ c^T c - b^T b \end{pmatrix}$$

and define $R := ||x||_2$.

Set $f_{\mathrm{surface}}(i) = \frac{\kappa}{R} n$ where κ is a constant depending on the fluid.

end for

4.2.3 (Basic) Algorithm

The idea of moving grids is that the nodes on the free boundary are assumed to move with the flow. At each time step, the grid is updated according to the motion of the free boundary. Then a new grid is generated using the new edge elements in order to prevent distortion of the grid and the flow field is projected onto the new grid. The simulation of fluid flow in a time-dependent domain is significantly more expensive since all matrices and vectors are time-dependent. Furthermore, the grid generation within the simulation is very expensive.

The initial boundary value problem (1.12) and (4.1) is solved as follows: first, the surface tension on the free boundary is computed. Then, the fluid equations are solved subject to the correct boundary conditions on the free boundary. This process is repeated while at the same time the grid is constantly updated.

The basic algorithm for a simulation with a free boundary reads as follows:

- ☞ Determine edge elements, grid points, and Lagrange points on free boundary
- ☞ Compute surface tension at boundary grid points
- ☞ Compute surface tension at Lagrange nodes on free boundary which are not grid points via interpolation
- ☞ Perform one time integration step and compute the new velocity field
- ☞ Compute displacement of free boundary
- ☞ Move nodes on boundary
- ☞ Check mesh quality and possibly compute new triangularization and interpolate the solution to the new grid
- ☞ Repeat this procedure

An example of a numerical simulation of a free boundary fluid flow problem is shown in chapter 5.

Chapter 5

Applications

The numerical methods described in the previous chapters and their inherent subproblems have been tested in several simulations. The following applications comprise a wide range of flow patterns and behaviors in stationary as well as transient domains with and without fluid-structure interaction. The flows and their phenomena are visualized by their velocity vector fields, the magnitude of the velocity, and streamlines. A streamline is a curve that is tangent to the velocity field at an instant of time.

5.1 Flow over a hemisphere

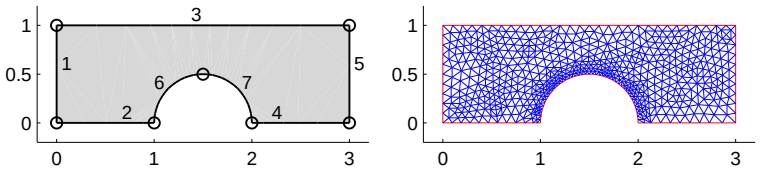
This two-dimensional model problem is known in literature as flow over a hemisphere, although flow over a semicircle would be a more accurate description. In fact, this model problem can been seen as a two-dimensional analog of three-dimensional flow over a hemisphere or a cylinder. In section 5.4, a composition of both situations in three dimensions is presented.

The geometry model and the triangularization are shown in figure 5.1. Boundary segment Γ_1 represents the inlet where the fluid enters the domain with a parabolic velocity profile

$$u_{\text{in}} = u_{\text{in}}(s) = (4s(1-s), 0)^T$$

where $s \in [0,1]$ linearly parameterizes Γ_1. Boundary segment Γ_5 is the outlet where homogeneous traction boundary conditions are applied, i.e. the force acting upon the fluid at the outlet is $f_{\text{out}} = 0 \in \mathbb{R}^2$. No-slip boundary conditions are applied on all other boundaries.

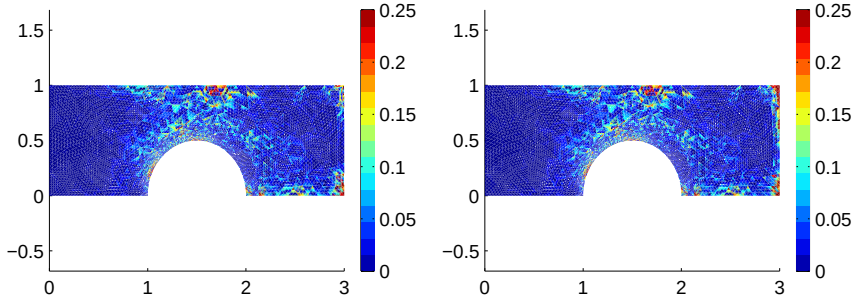
In order to demonstrate the effect of implementations based on different weak forms, the stationary flow has been computed for the weak forms introduced in chapter 2. Figure 5.2 shows the magnitude of the divergence of the velocity field. It can be seen that the total stress form yields better results near the outlet than the viscous stress form. Within the



(a) (b)

Figure 5.1: Geometry model and mesh

domain, there is no significant difference bewteen these two implementations. For the pseudo total stress form and the pseudo viscous stress form, the maximal magnitude of the divergence is about 50% higher compared to the total stress form. In that sense, these implementations don't seem to be competitive with the total stress form, which also excels since it also can be used to model non-Newtonian fluids.



(a) Total stress form (b) Viscous stress form

Figure 5.2: Magnitude of the divergence

Given a geometry model and boundary conditions, the flow also strongly depends on the Reynolds number. Figure 5.3 shows the results for Reynolds numbers $Re = 6$, 60, 600, and 3000 computed with SpexBDF$(3, 2)$. The emergence of vortices behind the obstacle can be observed for Reynolds numbers greater than approximately 500. As the Reynolds number increases, the flow becomes instationary. In order to compute the flow for Reynolds numbers greater than approximately 3000, the spatial discretization must be refined.

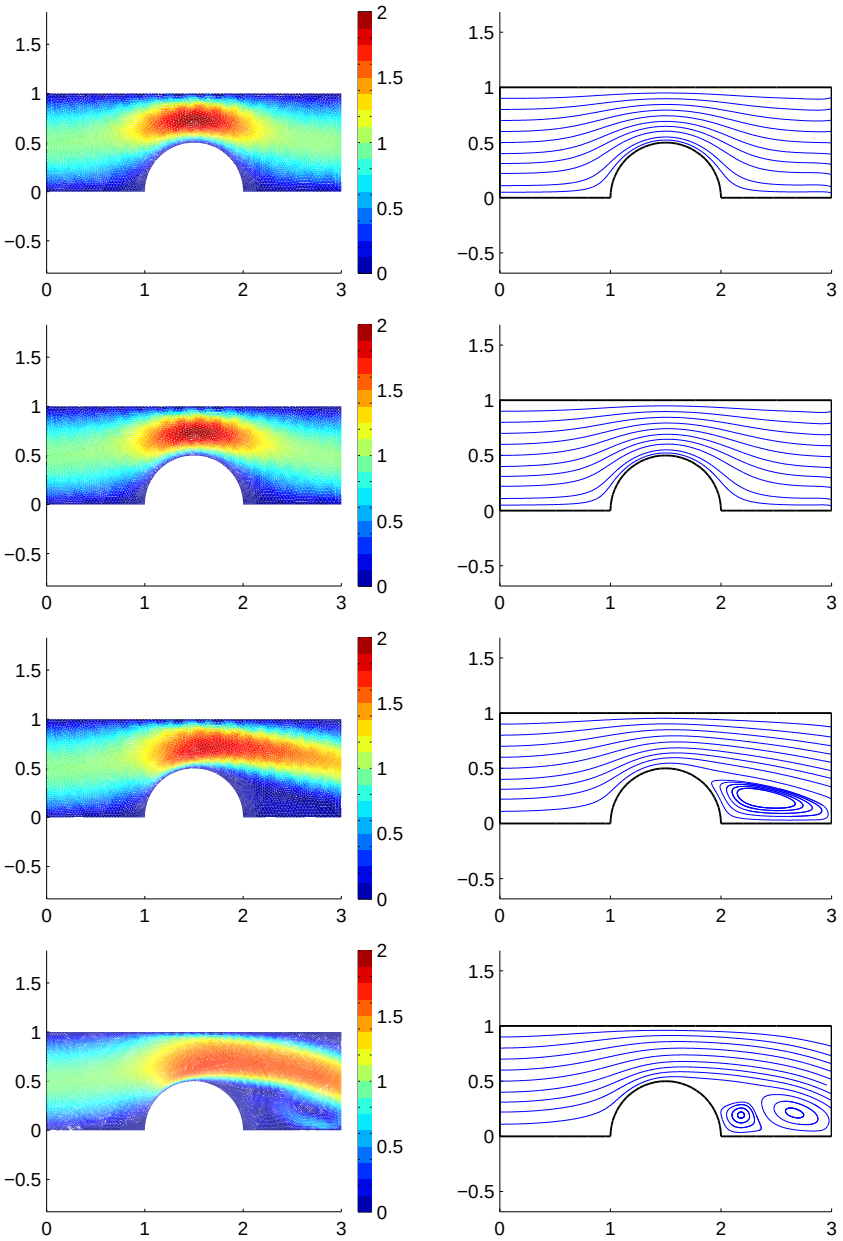


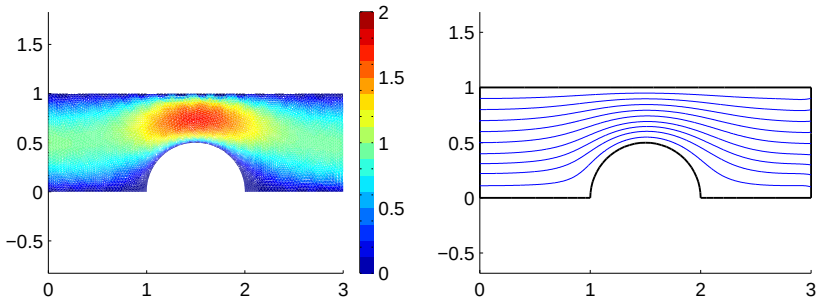
Figure 5.3: Flow at $Re = 6$, 60, 600, and 3000 (from top to bottom)

Non-Newtonian flow

The simulation of non-Newtonian fluid flow is more difficult than the simulation of Newtonian fluid flow in the sense that the convergence behavior is, in general, critical. Only the pseudo total stress form and the total stress form of chapter 2 are appropriate to model the dynamics of non-Newtonian fluids. As an example of a non-Newtonian fluid, consider the following test problem with non-constant viscosity

$$\eta(u) = \eta_0(1 + u_1^2 + u_2^2)$$

with $\eta_0 = 0.1$. The geometry model and the triangularization are the same as for the previous problem (see figure 5.1). At $t = 0$, the fluid is at rest and boundary conditions are a parabolic velocity profile at the inlet, homogeneous traction outflow conditions at the outlet, and no-slip conditions on all other boundaries. For $t = [0, 5]$, the simulation has been performed with SpexBDF$(2, 1)$ in conjunction with the total stress form. The magnitude of the velocity and streamlines are shown in figure 5.4 for $t = 5$. The different behavior of the non-Newtonian fluid flow compared to a Newtonian fluid flow with $\eta = \eta_0$ can be seen by looking at cross sections of the magnitude of the velocity. Figure 5.5(a) shows cross sections at $x = 1.5$ and $x = 2.9$ for both, Newtonian and non-Newtonian fluid flow. It can be seen that the non-Newtonian flow differs significantly from the Newtonian flow and its velocity profile deviates from the more parabolic shaped Newtonian velocity profile.



(a) Magnitude of velocity (b) Streamlines

Figure 5.4: Non-Newtonian flow

Paper pulp is another example of a non-Newtonian fluid which is a suspension of wood fibers in water. In situations where inertial forces are small over interesting timescales, the two phases don't separate and a single velocity field can describe the flow (cf. [18]). Paper pulp is a non-Newtonian fluid because viscous shear forces grow very slowly with the velocity gradients compared to the Newtonian assumption that the shear forces are proportional to the velocity gradients. In two dimensions, the non-Newtonian viscosity for paper pulp is described by the Carreau-Yasuda model (cf. [18])

$$\eta(r) = \frac{\eta_0 + \left(\frac{r}{R}\right)^n \eta_\infty}{1 + \left(\frac{r}{R}\right)^n}$$

with

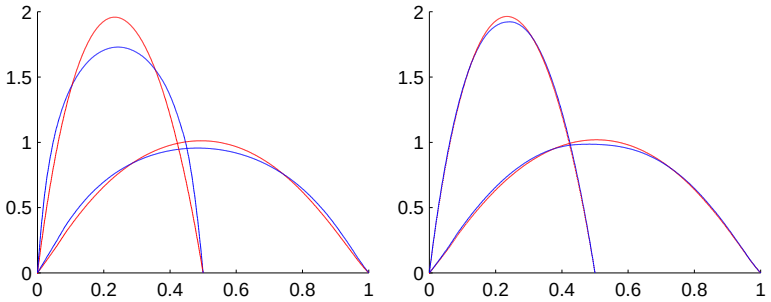
$$r = \sqrt{\frac{1}{2}\left(\left(\frac{\partial u_1}{\partial x_1}\right)^2 + \left(\frac{\partial u_1}{\partial x_2}\right)^2 + \left(\frac{\partial u_2}{\partial x_1}\right)^2 + \left(\frac{\partial u_2}{\partial x_2}\right)^2\right)}$$

and some constants η_0, η_∞, n, and R. If $\eta_0 = \eta_\infty$, then $\eta(r) = \eta_0 = \text{const}$ and the fluid is Newtonian. The geometry model and the triangularization are the same as for the previous problem (see figure 5.1) and let $\eta_0 = 500000$, $\eta_\infty = 2000$, $R = 0.002$, and $n = 1$.

The stationary solution of this problem has been computed as described in the following. The implementation uses the weak form based on the total stress form since it has been shown at the beginning of this chapter that it is superior to the other forms. In order to obtain convergence for this highly nonlinear stationary problem, the instationary Navier-Stokes equations for a Newtonian fluid with $\eta_0 := \eta_\infty$ are solved for $t \in [0, 0.2]$. Then, the solution at $t = 0.2$ is used as initial guess for solving stationary Newtonian flow. It has not been possible to solve the stationary problem without having a good initial guess from the instationary problem. Finally, the stationary Newtonian solution is used as initial guess for computing the stationary non-Newtonian solution. The stationary problems have been solved with Femlab®'s nonlinear solver and the instationary problem has been solved with SpexBDF. Figure 5.5(b) shows a plot of cross section velocity profiles at $x = 1.5$ and $x = 2.9$ for the Newtonian ($\eta_0 = \eta_\infty$) and non-Newtonian ($\eta_0 \neq \eta_\infty$) case. As in the previous test example, the non-Newtonian flow differs from the Newtonian flow and its velocity profile deviates slightly from the more parabolic shaped Newtonian velocity profile.

5.2 A fluid-structure interaction problem

As an example of fluid-structure interaction, consider the situation shown in figure 5.6. A reservoir is filled with fluid and more fluid is pumped into the reservoir through the inlet on the left side while fluid leaves the reservoir through the outlet on the right side. Within the reservoir, there is a solid structure (a circle) which moves under body forces (gravity) and the forces of the fluid acting upon the structure. The domain of the reservoir



(a) (b)

Figure 5.5: Velocity profiles for Newtonian (red) and non-Newtonian (blue) fluid flow

is represented by 18 boundary segments. Boundary segment Γ_1 represents the inlet where the velocity profile of the flow is given as

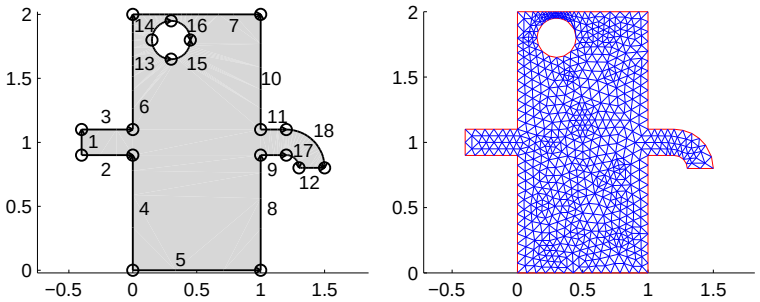
$$u_{\text{in}} = u_{\text{in}}(s) = (s(1-s), 0)^T$$

where $s \in [0,1]$ linearly parameterizes Γ_1. Boundary segment Γ_{12} is the outlet where homogeneous traction boundary conditions are applied, i.e. the force acting upon the fluid at the outlet is $f_{\text{out}} = 0 \in \mathbb{R}^2$ on Γ_{12}. Let the density of the fluid be $\varrho = 1$, the viscosity $\eta = 0.1$, and the body force $b = (0, -2)^T$. The parameters of the structure are as follows: The circle's radius is $r = 0.15$ and it has a homogeneous mass density. Its mass is $m_{\text{fsi}} = 1$ and its moment of ineratia $I_{\text{fsi}} = 1$.

At $t = 0$, the fluid is at rest and the initial position of the center of mass and velocity of the structure are $x_{\text{fsi}}(0) = (0.3, 1.8)^T$ and $u_{\text{fsi}}(0) = (0, 0)^T$, respectively. In addition, the initial angular velocity of the structure is $\omega(0) = 2$ and its initial angular orientation is $\theta(0) = 0$.

As the sphere moves through the fluid, the viscosity of the fluid acts upon the moving sphere with a force that resists the motion of the object. Since the circle rotates at the beginning, it can be observed that the angular velocity of the structure decreases due to frictional forces of the viscous fluid. The evolution of the angular velocity $\omega(t)$ is shown in figure 5.7(a).

As the structure starts moving downward, it also moves slightly to the left since fluid mainly from the upper right follows the circles' path. Then, as the circle comes closer to the inlet, it gets swept away by the stream going mainly from inlet to outlet. Figure 5.7(b) shows the evolution of the horizontal position of the center of mass of the sphere. The gain



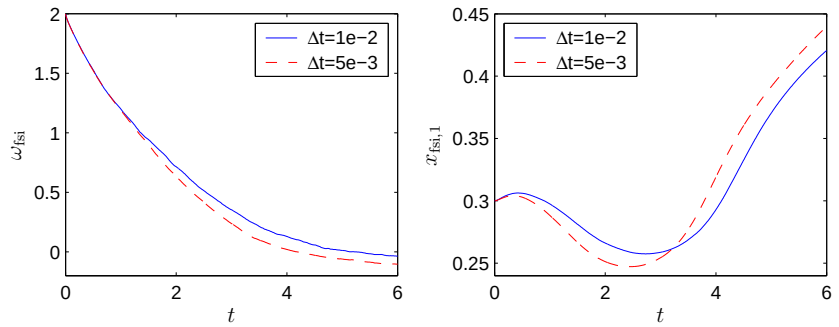
(a) Initial geometry (b) Initial mesh

Figure 5.6: Initial configuration

of horizontal momentum, when the sphere gets caught by the stream, can be clearly seen. The results of this simulation have been computed with SpexSDIRK and the techniques presented in section 4.1. Figure 5.8 shows a sequence of plots of the magnitude of the velocity and figure 5.9 shows the corresponding velocity vector field.

5.3 A free boundary problem

As an example of a free boundary problem and the techniques presented in section 4.2, consider the situation shown in figure 5.10. A reservoir is filled with fluid and the fluid is drained by opening the lower right outlet. There is no lid on top of the reservoir, so that the top boundary of the spatial domain is the fluids' free boundary whose shape is determined within the simulation by the flow itself and the fluids' surface tension on the boundary. Within the reservoir, there are two fixed obstacles which do not move. The domain of the reservoir is represented by 20 boundary segments. Boundary segment 5 represents the free boundary of the fluid and boundary segment 17 is the outlet where homogeneous traction boundary conditions are applied. Suppose the reservoir is filled with a fluid with density $\varrho = 1$ and viscosity $\eta = 0.15$. Let the constant describing the surface tension be $\kappa = 0.01$ and the body force acting upon the fluid is $b = (0, -5)^T$. At $t = 0$, the fluid is at rest and the reservoir is drained by opening the lower right outlet. The results of this simulation are shown in figures 5.11 and 5.12 as sequences of plots of the magnitude of the velocity and the corresponding velocity vector fields, respectively.



(a) Angular velocity (b) Horizontal position

Figure 5.7: Evolution of angular velocity and horizontal position of center of mass

5.4 Flow over a 3D obstacle

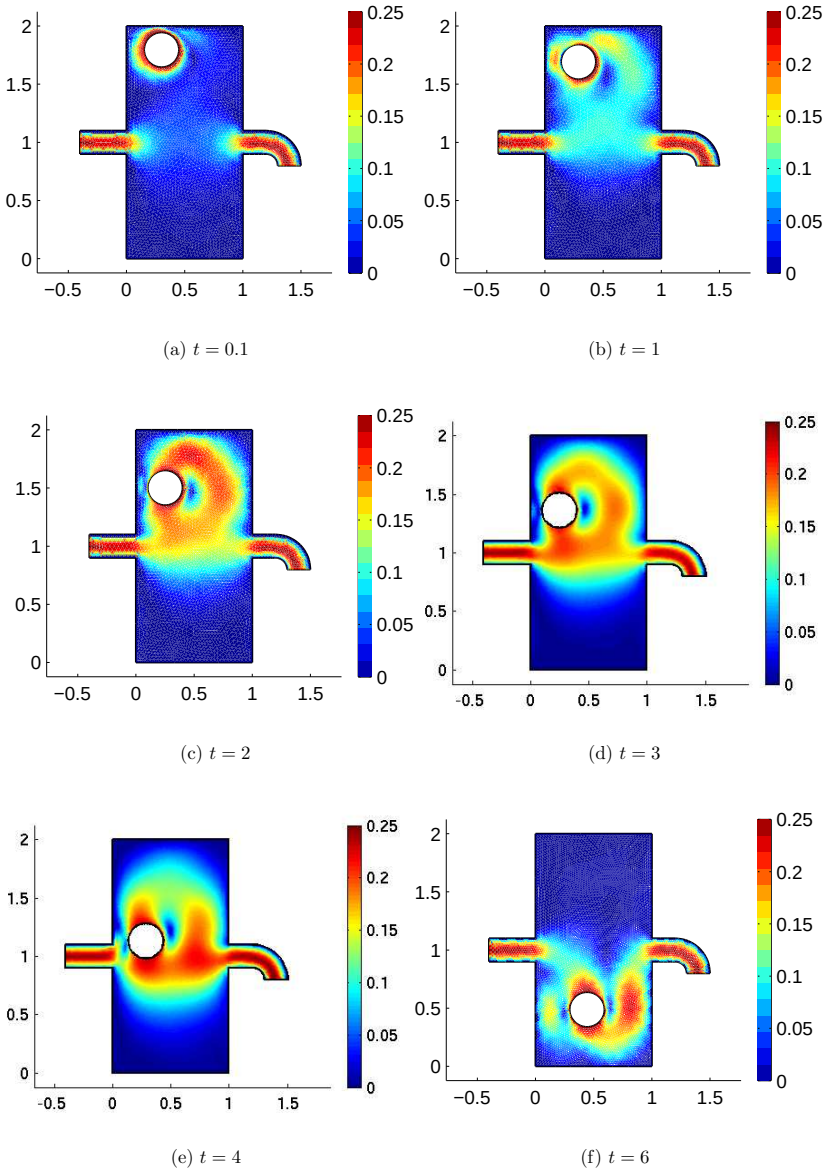
This model problem is an extension of the problem in section 5.1 to three dimensions. The obstacle in the channel consists of a quarter of a sphere and one half of a cylinder. The fluid density is $\varrho = 1$, the viscosity is $\eta = 0.01$, and there are no body forces, i.e. $b = 0$. Three meshes have been generated in order to demonstrate the performance of the velocity and the pressure step of the decoupled methods. The meshes consist of 1491, 3905, and 24034 tetrahedrons with 8204, 20204, and 116220 degrees of freedoms, respectively.

The convergence histories and convergence rates of AMG(1,2) applied to (3.33) on these grids are shown in table 5.1 and figure 5.14(a). It can be seen that the convergence is essentially grid independent. Note that the different grids are not regular refinements of eachother (due to curved boundaries and different parameters for the grid generation) and therefore true grid independent convergence can not be expected. The efficiency of

grid	$q^{(1)}$	$q^{(2)}$	$q^{(3)}$	$q^{(4)}$	$q^{(5)}$	$q^{(6)}$	$q^{(7)}$	$q^{(8)}$	$\widehat{q}^{(4)} \approx \rho$
coarse	0.0792	0.0936	0.0895	0.0873	0.0870	0.0872	0.0875	0.0879	0.0873
medium	0.0862	0.1051	0.0967	0.0933	0.0923	0.0922	0.0925	0.0928	0.0938
fine	0.0976	0.1130	0.1016	0.0964	0.0951	0.0953	0.0959	0.0966	0.0988

Table 5.1: Convergence rates on different grids

AMG(ν_1,ν_2) applied to (3.33) on the fine grid is shown in figure 5.14(b). It turns out that $\nu_1 = 1$ and $\nu_2 = 2$ is the optimal choice for this test problem. The performance of

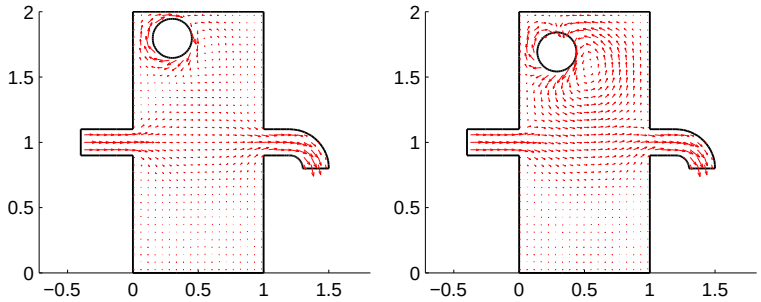


(a) $t = 0.1$ (b) $t = 1$

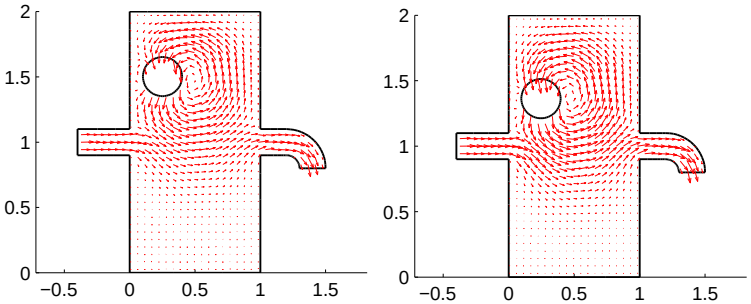
(c) $t = 2$ (d) $t = 3$

(e) $t = 4$ (f) $t = 6$

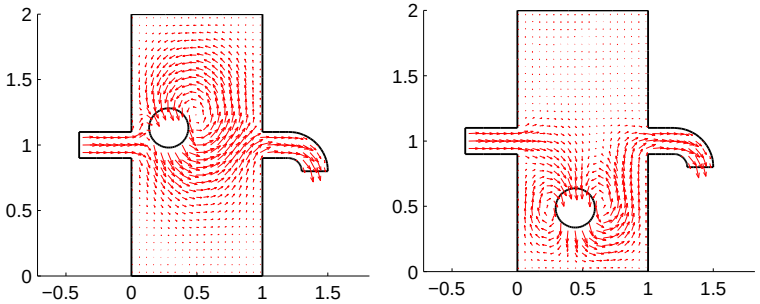
Figure 5.8: Magnitude of velocity



(a) $t = 0.1$ (b) $t = 1$



(c) $t = 2$ (d) $t = 3$



(e) $t = 4$ (f) $t = 6$

Figure 5.9: Velocity field

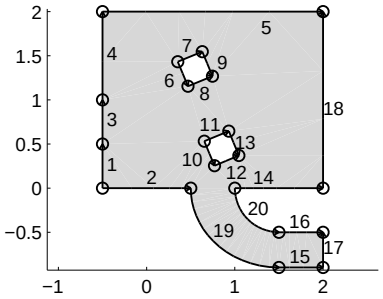
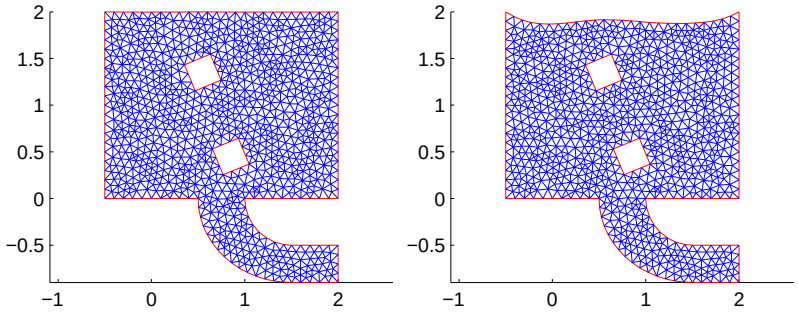


Figure 5.10: Initial geometry

AMG(1,1), AMG(2,1), AMG(2,2), AMG(2,3), AMG(3,3), etc. has been found to be below that of AMG(1,2).

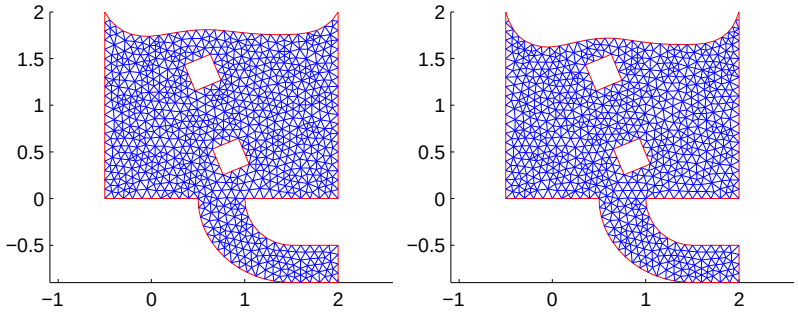
The convergence histories of AMG(1,2) applied to the linear systems (3.34) occurring within the (simplified) Newton method have been studied on the finest grid. The convergence history of AMG depends on the temporal step size of the underlying time integration scheme and whether the advection term is treated fully implicitly or explicitly as in the case of semi-implicit methods. The convergence rates are plotted in figures 5.15(a) and 5.15(b) for both, fully implicit and semi-implicit methods, for different steps sizes. Figure 5.15(c) shows the dependence of the convergence factor on the temporal step size.

It turns out that in this example, the fundamental idea of multigrid, to reduce the high frequency components of the error by smoothing procedures and to take care of the low frequency error components by coarse grid corrections, does not work optimally. Certain error components remain large an are not reduced effectively by the smoothing procedures applied. In this situation, the combination with Krylov subspace methods such as GMRES may excel the convergence properties. This is subject to future work to be done.



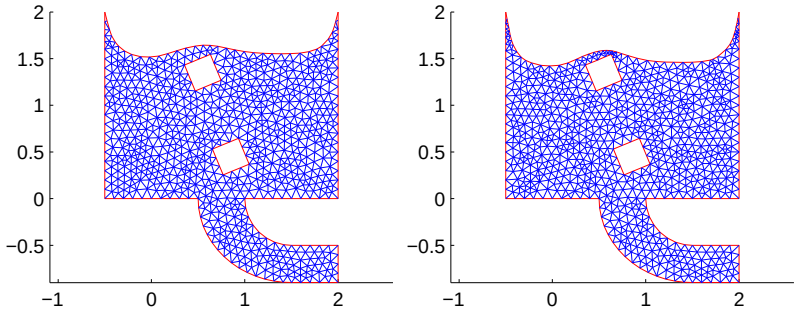
(a) $t = 0.1$

(b) $t = 1$



(c) $t = 2$

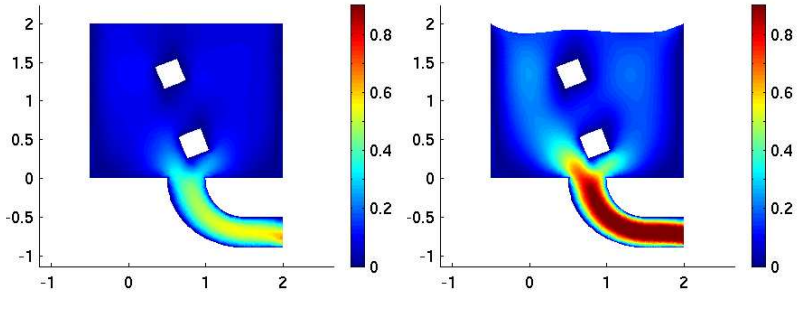
(d) $t = 3$



(e) $t = 4$

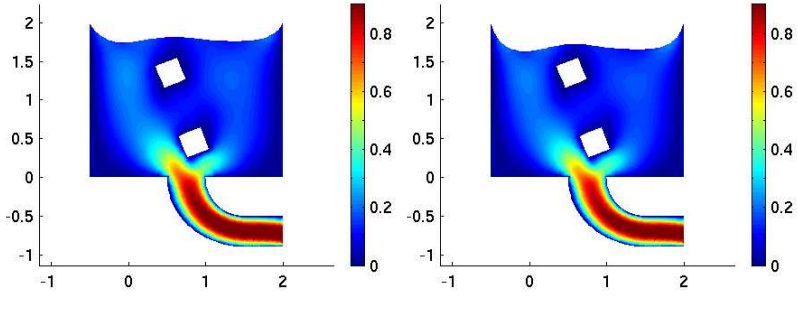
(f) $t = 5$

Figure 5.11: Moving mesh



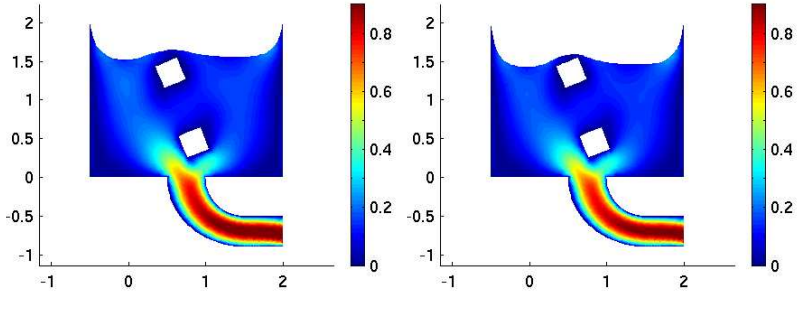
(a) $t = 0.1$

(b) $t = 1$



(c) $t = 2$

(d) $t = 3$



(e) $t = 4$

(f) $t = 5$

Figure 5.12: Magnitude of velocity

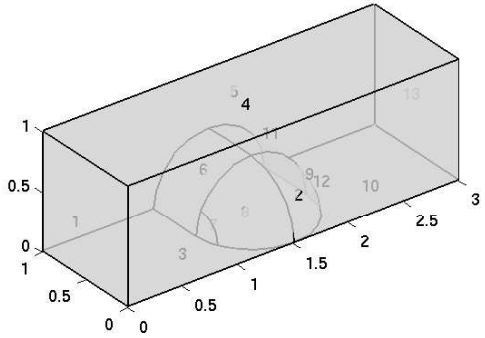
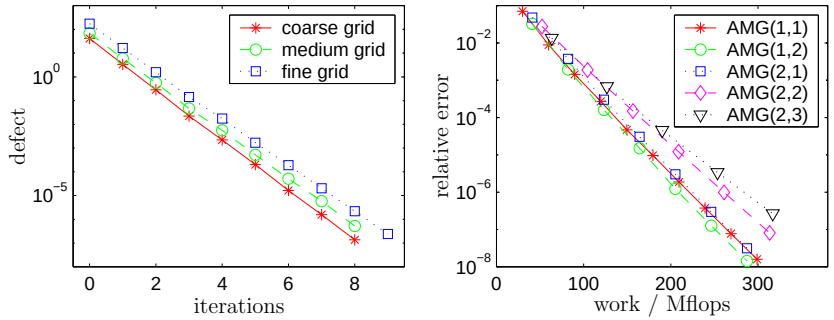


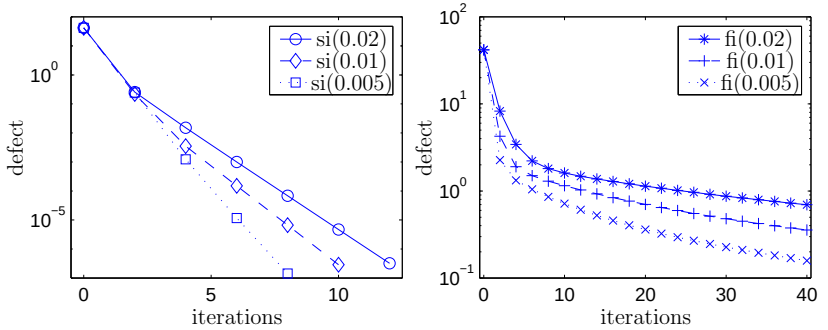
Figure 5.13: The geometry of the 3D model



(a) Grid independent convergence of AMG

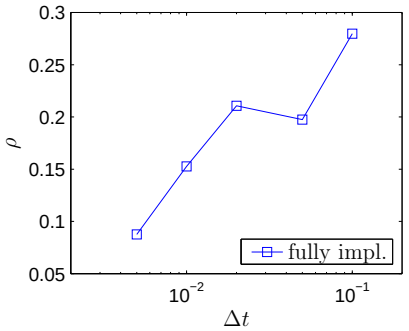
(b) Efficiency of AMG

Figure 5.14: AMG convergence



(a) Convergence depending on step size

(b) Convergence depending on step size



(c) Convergence factor depending on step size

Figure 5.15: AMG convergence in the velocity step

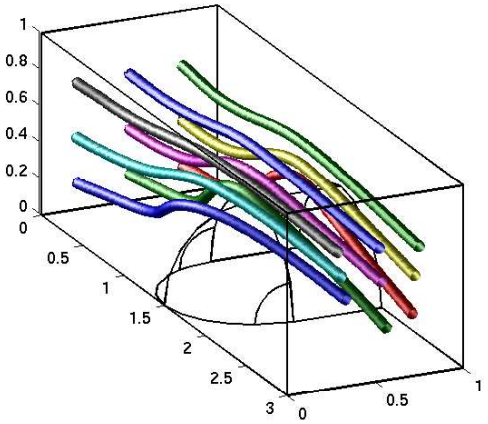
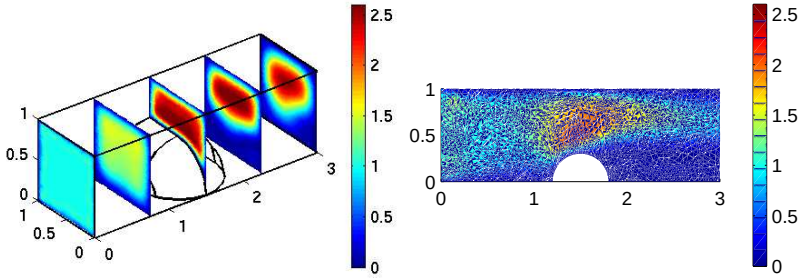
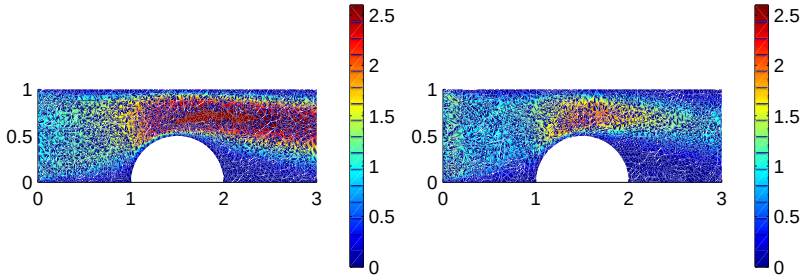


Figure 5.16: Streamlines at $Re = 100$



(a) $x = 0.01, 0.75, 1.5, 2.25, 2.99$ (b) $y = 0.1$



(c) $y = 0.5$ (d) $y = 0.9$

Figure 5.17: Cross sections of magnitude of velocity ($Re = 100$)

Appendix A

Implementation

The first part of this appendix describes the basic methodology for simulating a real-life fluid flow problem numerically. In the second part, the implementation of boundary conditions is illustrated with a simple example. In the third part, the underlying data structure for the numerical simualtions is introduced. For of easy use, flexibility, and compatibility with Femlab®, all data necessary for the simulation is stored in a single Matlab® structure – the `fem` structure. The most important fields of this structure are briefly described in section A.3.

A.1 Modus operandi

Following the methodology of scientific computing, the procedure of numerically simulating a real-life fluid flow problem consists of four major steps:

- Description of the real problem
- Physical and mathematical modeling
- Numerical approximation
- Visualization

The third step, i.e. the actual numerical solution step, consists of the following major steps:

- Initialize model (`initmodel`)
  - Define domain (`initgeom`)
  - Create coarse mesh (`initmesh`)
  - Mesh refinement (optional)

  - Set up weak form (`nseform`)
  - Define boundary conditions

- Generate extended mesh, i.e. create Lagrange points for mixed elements
- Do time integration (e.g., using any of the methods of chapter 3) and adapt mesh if the domain is time-dependent
- Post processing of the solution (`visualization`)

A.2 Natural and essential boundary conditions

The handling and interaction of Neumann boundary conditions (a.k.a. natural boundary conditions) and Dirichlet boundary conditions (a.k.a. essential boundary conditions) for the Lagrange multiplier ansatz presented in section 2.2.2 is illustrated with an example from [19].

Consider the stationary version of (2.6) with $N = d = 2$, i.e. $\Omega \subset \mathbb{R}^2$, $y : \Omega \to \mathbb{R}^2$, and

$$\begin{aligned} \nabla \cdot \Gamma_1(y) &= F_1(y) \qquad \text{in } \Omega \\ \nabla \cdot \Gamma_2(y) &= F_2(y) \qquad \text{in } \Omega \end{aligned}$$

with Neumann boundary conditions

$$\begin{aligned} -n \cdot \Gamma_1(y) &= G_1(y) + \frac{\partial R_1(y)}{\partial y_1}\mu_1 + \frac{\partial R_2(y)}{\partial y_1}\mu_2 \qquad \text{on } \partial\Omega \\ -n \cdot \Gamma_2(y) &= G_2(y) + \frac{\partial R_1(y)}{\partial y_2}\mu_1 + \frac{\partial R_2(y)}{\partial y_2}\mu_2 \qquad \text{on } \partial\Omega \end{aligned}$$

and Dirichlet boundary conditions

$$\begin{aligned} 0 &= R_1(y) \qquad \text{on } \partial\Omega \\ 0 &= R_2(y) \qquad \text{on } \partial\Omega. \end{aligned}$$

For simplicity of notation, arguments will be omitted in the following. Now, consider three cases of boundary conditions.

Case 1: Let $R_1 = R_2 = 0$, i.e. there are no constraints on the solution and we obtain Neumann boundary conditions

$$\begin{aligned} -n \cdot \Gamma_1 &= G_1 \qquad \text{on } \partial\Omega \\ -n \cdot \Gamma_2 &= G_2 \qquad \text{on } \partial\Omega. \end{aligned}$$

Case 2: Let $R_1 = r_1 - y_1$ and $R_2 = r_2 - y_2$. Then, the Dirichlet conditions are simply $y_1 = r_1$ and $y_2 = r_2$. The Neumann boundary conditions become

$$\begin{aligned} -n \cdot \Gamma_1 &= G_1 - \mu_1 \qquad \text{on } \partial\Omega \\ -n \cdot \Gamma_2 &= G_2 - \mu_2 \qquad \text{on } \partial\Omega. \end{aligned}$$

and impose no restrictions on y and therefore can be discarded.

Case 3: Let $R_1 = r_1 - y_1$ and $R_2 = 0$. Then the Dirichlet conditions are $y_1 = r_1$ and $0 = 0$ on $\partial\Omega$ and the Neumann conditions are

$$\begin{aligned} -n \cdot \Gamma_1 &= G_1 - \mu_1 \qquad \text{on } \partial\Omega \\ -n \cdot \Gamma_2 &= G_2 \qquad \text{on } \partial\Omega. \end{aligned}$$

The first Neumann condition can be discarded. Hence, there is a Dirichlet condition for y_1 together with the second Neumann condition.

It is important to note that when mixing Dirichlet and Neumann conditions, the ordering of the equations as well as the ordering of the variables are important (see [19] for details). The situation of boundary conditions becomes more intricate as the number of dependent variables N or the spatial dimension d increases. In addition, different boundary conditions may be applied on different segments of the boundary.

For the fluid dynamics simulations presented in the previous chapters, the spatial dimensions are $d = 2$ and $d = 3$ and the number of dependet variables are $N = 3$ and $N = 4$, respectively. In each of the simulations, several different boundary conditions, such as, e.g. no-slip conditions and traction conditions, were imposed on different segments Γ_i of the boundary $\partial\Omega = \bigcup_{i=1}^{N_\Gamma} \Gamma_i$.

A.3 Data structure

The data structures that define the continuous as well as the discrete problem are stored in a single Matlab® structure – the `fem` structure. The data structures in the fields of the `fem` structure define different aspects of the problem, and various processing stages produce new data structures. For compatibility with Femlab® 2.3, an augmented `fem` structure is used in this work which is based upon Femlab®'s `fem` structure `fem`. However, in order to handle a larger class of fluid dynamics problems and new solvers, it contains several more structure fields than the original `fem` structure. Details about the original `fem` structure can be found in [19].

`const`	Constants
`sdim`	Names of space coordinates
`geom`	Analyzed geometry
`equiv`	Equivalent boundaries
`mesh`	Mesh structure
`xmesh`	Extended mesh structure
`sshape`	Geometry approximation order
`dim`	Names or number of dependent variables
`shape`	Finite element shape functions
`cporder`	Order of Lagrange points
`gporder`	Order of Gauss points
`variables`	Definition of constants
`equ`	Variables, equations, constraints, and initial values on subdomains
`bnd`	Variables, equations, constraints, and initial values on boundaries
`form`	Form of equations (general/coefficient)
`init`	Initial value
`sol`	Solution structure
`u`	Matrix containing solution vectors
`tlist`	Time of solution vectors

Table A.1: Data structure

`model`	Specify a model (cf. chapter 5)
`have`	Properties of the model
`pmode`	Pressure mode (boolean)
`tdgeom`	Time-dependent geometry (boolean)
`tdBCs`	Time-dependent boundary conditions (boolean)
`freebnd`	Free boundary (boolean)
`fsi`	Fluid structure interaction (boolean)
`bnd.free`	Indices to free boundary groups
`bnd.fsi`	Indices to fluid-structure interaction boundary groups
`odefile`	Name of ODEfile
`nseform`	Control weak form (for options see table A.3)
`nse`	Navier-Stokes equation (depending on `fem.nseform`, see also p. 38)

continued on next page

continued from previous page

`da, ga, f`	Mass coefficient, flux vector, and source term
`shape`	Shape functions
`linear`	Linear parts of PDE
`ga`	Flux vector
`f`	Source term
`nonlinear`	Nonlinear partsof PDE
`f`	Source term
`solver`	Time integration method
`strategy`	Optional instead of `solver`
`integrator`	Optional instead of `solver`
`newton`	Options for Newton's method for nonlinear systems
`solver`	Linear solver within Newton's method
`tol`	Tolerance
`itmax`	Maximal number of iterations
`lumping`	boolean
`projection`	Control projection
`method`	Type of projection
`solver`	Solver for projection
`precond`	LU/RCM, AMG
`fc`	Auxiliary storage field
`L,U,P,perm`	
`penalty`	Penalize PPE (boolean)
`time`	Control simulation time
`t0`	Initial time for simulation
`tend`	Final time for simulation
`delt`	Current step size
`tlist`	Solution output time vector. (optional). Equals `tend` if not specified.
`elim`	Storage field for eliminated quantities
`init`	Initial value, with fields `u`, `p`
`sol`	(eliminated) solution vector
`M, D, G, Ct`	Matrices
`f, g`	Vectors
`dof`	Storage field for constrained quantities

continued on next page

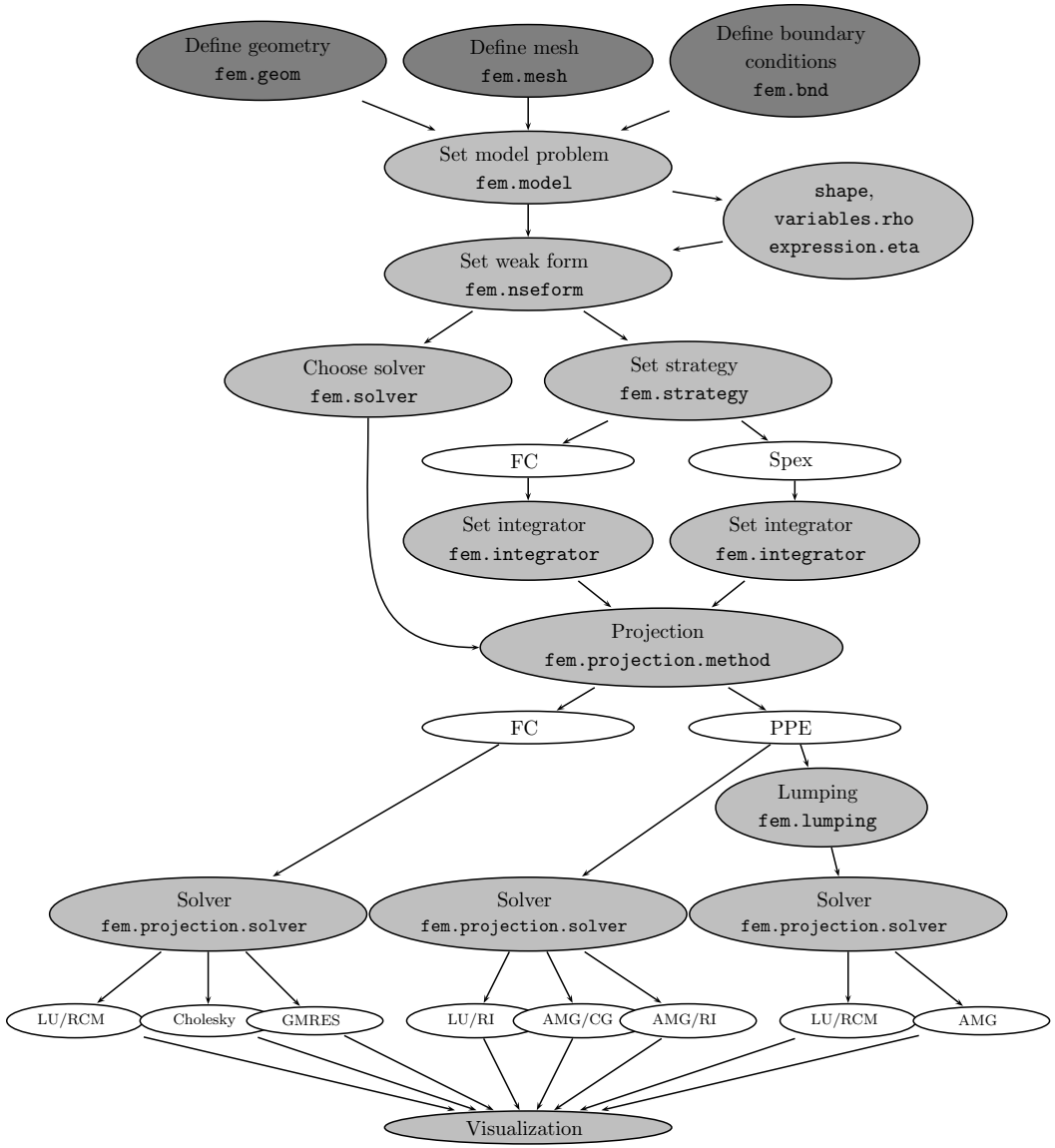
continued from previous page	
`D`	Discretized mass coefficient d_a
`Nu`	$\mathcal{N}$
`yd`	y_{h_d}
`t`	Time when the above quantities were computed
`ind`	Indices
`u, p`	Velocity and pressure DOFs
`ue, pe`	Eliminated velocity and pressure DOFs
`u1, u2, u3`	u_α velocity DOFs
`N`	Dimensions
`y, u, p`	
`ye, ue, pe`	
`amg`	Control algebraic multigrid
`maxlevels`	Max. number of grids
`minnodes`	Min. number of nodes
`smoother`	Smoother for AMG
`preits`	# relaxations
`postits`	# relaxations
`advection`	Treatment of advection term (see section 3.7.1)
`fluid`	Type of fluid

Table A.2: Augmented data structure

`nseform`	Weak form
`pta`	Pseudo total stress form (cf. p. 38)
`pva`	Pseudo viscous stress form (cf. p. 38)
`vsa`	Visous stress form (cf. p. 39)
`tsa`	Total stress form (cf. p. 39)

Table A.3: Options for `fem.nseform` field

A.4 Basic solver options



Appendix B

Weak formulation

In this section, the coefficients used for the strong form (2.7) in order do generate different weak forms of the Navier-Stokes equations (see also section 2.2) are listed. First, the mass coefficient and then the flux vectors and source terms for the pseudo total stress form, the pseudo viscous stress form, the viscous stress form, and the total stress form (see section 2.2.3 on p. 38) in two and three dimensions are listed in detail.

B.1 Mass coefficient

The mass coefficients for the two-dimensional and three-dimensional Navier-Stokes equations in advection-diffusion form (2.7a) are given by

$$d_a = \begin{bmatrix} \varrho & 0 & 0 \\ 0 & \varrho & 0 \\ 0 & 0 & 0 \end{bmatrix} \qquad \text{and} \qquad d_a = \begin{bmatrix} \varrho & 0 & 0 & 0 \\ 0 & \varrho & 0 & 0 \\ 0 & 0 & \varrho & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix},$$

respectively. See [19] for a detailed description of the syntax. The flux vectors and source terms required to generate different weak forms are listed below.

B.2 Pseudo total stress form

`sdim=2`, `nseform=pta`:

$$\Gamma = \begin{bmatrix} -\eta\frac{\partial u_1}{\partial x_1} + p & -\eta\frac{\partial u_1}{\partial x_2} \\ -\eta\frac{\partial u_2}{\partial x_1} & -\eta\frac{\partial u_2}{\partial x_2} + p \\ 0 & 0 \end{bmatrix}, \qquad F = \begin{bmatrix} b_1 - \varrho(u_1\frac{\partial u_1}{\partial x_1} + u_2\frac{\partial u_1}{\partial x_2}) \\ b_2 - \varrho(u_1\frac{\partial u_2}{\partial x_1} + u_2\frac{\partial u_2}{\partial x_2}) \\ \frac{\partial u_1}{\partial x_1} + \frac{\partial u_2}{\partial x_2} \end{bmatrix}$$

`sdim=3`, `nseform=pta`:

$$\Gamma = \begin{bmatrix} -\eta\frac{\partial u_1}{\partial x_1} + p & -\eta\frac{\partial u_1}{\partial x_2} & -\eta\frac{\partial u_1}{\partial x_3} \\ -\eta\frac{\partial u_2}{\partial x_1} & -\eta\frac{\partial u_2}{\partial x_2} + p & -\eta\frac{\partial u_2}{\partial x_3} \\ -\eta\frac{\partial u_3}{\partial x_1} & -\eta\frac{\partial u_3}{\partial x_2} & -\eta\frac{\partial u_3}{\partial x_3} + p \\ 0 & 0 & 0 \end{bmatrix}, \qquad F = \begin{bmatrix} b_1 - \varrho(u_1\frac{\partial u_1}{\partial x_1} + u_2\frac{\partial u_1}{\partial x_2} + u_3\frac{\partial u_1}{\partial x_3}) \\ b_2 - \varrho(u_1\frac{\partial u_2}{\partial x_1} + u_2\frac{\partial u_2}{\partial x_2} + u_3\frac{\partial u_2}{\partial x_3}) \\ b_3 - \varrho(u_1\frac{\partial u_3}{\partial x_1} + u_2\frac{\partial u_3}{\partial x_2} + u_3\frac{\partial u_3}{\partial x_3}) \\ \frac{\partial u_1}{\partial x_1} + \frac{\partial u_2}{\partial x_2} + \frac{\partial u_3}{\partial x_3} \end{bmatrix}$$

B.3 Pseudo viscous stress form

`sdim=2`, `nseform=pva`:

$$\Gamma = -\eta \begin{bmatrix} \frac{\partial u_1}{\partial x_1} & \frac{\partial u_1}{\partial x_2} \\ \frac{\partial u_2}{\partial x_1} & \frac{\partial u_2}{\partial x_2} \\ 0 & 0 \end{bmatrix}, \qquad F = \begin{bmatrix} b_1 - \varrho(u_1\frac{\partial u_1}{\partial x_1} + u_2\frac{\partial u_1}{\partial x_2}) - \frac{\partial p}{\partial x_1} \\ b_2 - \varrho(u_1\frac{\partial u_2}{\partial x_1} + u_2\frac{\partial u_2}{\partial x_2}) - \frac{\partial p}{\partial x_2} \\ \frac{\partial u_1}{\partial x_1} + \frac{\partial u_2}{\partial x_2} \end{bmatrix}$$

`sdim=3`, `nseform=pva`:

$$\Gamma = -\eta \begin{bmatrix} \frac{\partial u_1}{\partial x_1} & \frac{\partial u_1}{\partial x_2} & \frac{\partial u_1}{\partial x_3} \\ \frac{\partial u_2}{\partial x_1} & \frac{\partial u_2}{\partial x_2} & \frac{\partial u_2}{\partial x_3} \\ \frac{\partial u_3}{\partial x_1} & \frac{\partial u_3}{\partial x_2} & \frac{\partial u_3}{\partial x_3} \\ 0 & 0 & 0 \end{bmatrix}, \qquad F = \begin{bmatrix} b_1 - \varrho(u_1\frac{\partial u_1}{\partial x_1} + u_2\frac{\partial u_1}{\partial x_2} + u_3\frac{\partial u_1}{\partial x_3}) - \frac{\partial p}{\partial x_1} \\ b_2 - \varrho(u_1\frac{\partial u_2}{\partial x_1} + u_2\frac{\partial u_2}{\partial x_2} + u_3\frac{\partial u_2}{\partial x_3}) - \frac{\partial p}{\partial x_2} \\ b_3 - \varrho(u_1\frac{\partial u_3}{\partial x_1} + u_2\frac{\partial u_3}{\partial x_2} + u_3\frac{\partial u_3}{\partial x_3}) - \frac{\partial p}{\partial x_3} \\ \frac{\partial u_1}{\partial x_1} + \frac{\partial u_2}{\partial x_2} + \frac{\partial u_3}{\partial x_3} \end{bmatrix}$$

B.4 Viscous stress form

`sdim=2`, `nseform=vsa`:

$$\Gamma = -\eta \begin{bmatrix} 2\frac{\partial u_1}{\partial x_1} & \frac{\partial u_1}{\partial x_2} + \frac{\partial u_2}{\partial x_1} \\ \frac{\partial u_1}{\partial x_2} + \frac{\partial u_2}{\partial x_1} & 2\frac{\partial u_2}{\partial x_2} \\ 0 & 0 \end{bmatrix}, \qquad F = \begin{bmatrix} b_1 - \varrho(u_1\frac{\partial u_1}{\partial x_1} + u_2\frac{\partial u_1}{\partial x_2}) - \frac{\partial p}{\partial x_1} \\ b_2 - \varrho(u_1\frac{\partial u_2}{\partial x_1} + u_2\frac{\partial u_2}{\partial x_2}) - \frac{\partial p}{\partial x_2} \\ \frac{\partial u_1}{\partial x_1} + \frac{\partial u_2}{\partial x_2} \end{bmatrix}$$

`sdim=3`, `nseform=vsa`:

$$\Gamma = -\eta \begin{bmatrix} 2\frac{\partial u_1}{\partial x_1} & \frac{\partial u_1}{\partial x_2} + \frac{\partial u_2}{\partial x_1} & \frac{\partial u_1}{\partial x_3} + \frac{\partial u_3}{\partial x_1} \\ \frac{\partial u_1}{\partial x_2} + \frac{\partial u_2}{\partial x_1} & 2\frac{\partial u_2}{\partial x_2} & \frac{\partial u_2}{\partial x_3} + \frac{\partial u_3}{\partial x_2} \\ \frac{\partial u_1}{\partial x_3} + \frac{\partial u_3}{\partial x_1} & \frac{\partial u_2}{\partial x_3} + \frac{\partial u_3}{\partial x_2} & 2\frac{\partial u_3}{\partial x_3} \\ 0 & 0 & 0 \end{bmatrix}, F = \begin{bmatrix} b_1 - \varrho(u_1\frac{\partial u_1}{\partial x_1} + u_2\frac{\partial u_1}{\partial x_2} + u_3\frac{\partial u_1}{\partial x_3}) - \frac{\partial p}{\partial x_1} \\ b_2 - \varrho(u_1\frac{\partial u_2}{\partial x_1} + u_2\frac{\partial u_2}{\partial x_2} + u_3\frac{\partial u_2}{\partial x_3}) - \frac{\partial p}{\partial x_2} \\ b_3 - \varrho(u_1\frac{\partial u_3}{\partial x_1} + u_2\frac{\partial u_3}{\partial x_2} + u_3\frac{\partial u_3}{\partial x_3}) - \frac{\partial p}{\partial x_3} \\ \frac{\partial u_1}{\partial x_1} + \frac{\partial u_2}{\partial x_2} + \frac{\partial u_3}{\partial x_3} \end{bmatrix}$$

B.5 Total stress form

`sdim=2`, `nseform=tsa`:

$$\Gamma = \begin{bmatrix} -\eta 2\frac{\partial u_1}{\partial x_1} + p & -\eta(\frac{\partial u_1}{\partial x_2} + \frac{\partial u_2}{\partial x_1}) \\ -\eta(\frac{\partial u_1}{\partial x_2} + \frac{\partial u_2}{\partial x_1}) & -\eta 2\frac{\partial u_2}{\partial x_2} + p \\ 0 & 0 \end{bmatrix}, \qquad F = \begin{bmatrix} b_1 - \varrho(u_1\frac{\partial u_1}{\partial x_1} + u_2\frac{\partial u_1}{\partial x_2}) \\ b_2 - \varrho(u_1\frac{\partial u_2}{\partial x_1} + u_2\frac{\partial u_2}{\partial x_2}) \\ \frac{\partial u_1}{\partial x_1} + \frac{\partial u_2}{\partial x_2} \end{bmatrix}$$

`sdim=3`, `nseform=tsa`:

$$\Gamma = \begin{bmatrix} -\eta 2\frac{\partial u_1}{\partial x_1} + p & -\eta(\frac{\partial u_1}{\partial x_2} + \frac{\partial u_2}{\partial x_1}) & -\eta(\frac{\partial u_1}{\partial x_3} + \frac{\partial u_3}{\partial x_1}) \\ -\eta(\frac{\partial u_1}{\partial x_2} + \frac{\partial u_2}{\partial x_1}) & -\eta 2\frac{\partial u_2}{\partial x_2} + p & -\eta(\frac{\partial u_2}{\partial x_3} + \frac{\partial u_3}{\partial x_2}) \\ -\eta(\frac{\partial u_1}{\partial x_3} + \frac{\partial u_3}{\partial x_1}) & -\eta(\frac{\partial u_2}{\partial x_3} + \frac{\partial u_3}{\partial x_2}) & -\eta 2\frac{\partial u_3}{\partial x_3} + p \\ 0 & 0 & 0 \end{bmatrix},$$

$$F = \begin{bmatrix} b_1 - \varrho(u_1\frac{\partial u_1}{\partial x_1} + u_2\frac{\partial u_1}{\partial x_2} + u_3\frac{\partial u_1}{\partial x_3}) \\ b_2 - \varrho(u_1\frac{\partial u_2}{\partial x_1} + u_2\frac{\partial u_2}{\partial x_2} + u_3\frac{\partial u_2}{\partial x_3}) \\ b_3 - \varrho(u_1\frac{\partial u_3}{\partial x_1} + u_2\frac{\partial u_3}{\partial x_2} + u_3\frac{\partial u_3}{\partial x_3}) \\ \frac{\partial u_1}{\partial x_1} + \frac{\partial u_2}{\partial x_2} + \frac{\partial u_3}{\partial x_3} \end{bmatrix}$$

Appendix C

Notation

C.1 Variables, constants, etc.

d	Space dimension, $d = 2$ or $d = 3$
α	Spatial index, i.e. $\alpha = 1, \ldots, d$
x_α	Space coordinate
$x = (x_1, \ldots, x_d)^T$	Space variable
Ω	Spatial domain $\Omega \subset \mathbb{R}^d$
$\partial\Omega$	Boundary of Ω
$\Gamma_i \in \partial\Omega$	Boundary segment of $\partial\Omega$ (see p. 29)
Γ_α^D	Dirichlet boundary for velocity in x_α-direction
Γ_α^N	Neumann boundary for velocity in x_α-direction
$\mathrm{int}(\Delta_i)$	Interior of Δ_i
$p = p(t, x)$	Scalar pressure field
$u = u(t, x) = (u_1(t, x), \ldots, u_d(t, x))^T$	Velocity field
$u_\alpha = u_\alpha(t, x)$	Component of velocity field in x_α-direction
t	Time
t_0	Start of simulation
t_{final}	End of simulation
b	Body force
$\varrho \in \mathbb{R}$	Density

continued on next page

continued from previous page	
η	Dynamic viscosity or first coefficient of viscosity
λ	Second coefficient of viscosity
ν	Kinematic viscosity (p. 9)
L_c	Characteristic length
U_c	Characteristic velocity
$Re = \frac{L_c U_c}{\nu}$	Reynolds number, if viscosity is scalar (p. 17)
$\omega = \nabla \times u$	Vorticity
d_a	Mass coefficient (p. 37)
Γ	Flux matrix (p. 37)
F	Source term (p. 37)
G	Boundary source term (p. 37)
R	Restriction (constraints) on boundary, i.e. essential boundary conditions (p. 37)
$\mu = (\mu_1, \ldots, \mu_{N_R})$	Lagrange multiplier (p. 37)
$\sigma_{\mathrm{c,v}} = \eta(\nabla u + (\nabla u)^T) + \lambda(\nabla \cdot u)I$	Viscous stress tensor for a compressible fluid
$\sigma_{\mathrm{c}} = -pI + \sigma_{\mathrm{c,v}}$	Total stress tensor for a compressible fluid
$\sigma_{\mathrm{v}} = \eta(\nabla u + (\nabla u)^T)$	Viscous stress tensor for an incompressible fluid
$\sigma = -pI + \sigma_{\mathrm{v}}$	Total stress tensor for an incompressible fluid

C.2 Operators

$\Delta u_i = \frac{\partial^2 u_i}{\partial x_1^2} + \cdots + \frac{\partial^2 u_i}{\partial x_d^2} \in \mathbb{R}$	Laplacian of $u_i : \mathbb{R}^d \to \mathbb{R}$
$\Delta u = (\Delta u_1, \ldots, \Delta u_d)^T \in \mathbb{R}^d$	Laplacian of $u : \mathbb{R}^d \to \mathbb{R}^d$
$\nabla p = \left(\frac{\partial p}{\partial x_1}, \ldots, \frac{\partial p}{\partial x_d}\right)^T \in \mathbb{R}^d$	Gradient of $p : \mathbb{R}^d \to \mathbb{R}$
$\nabla u = \left(\frac{\partial u}{\partial x_1}, \ldots, \frac{\partial u}{\partial x_d}\right) \in \mathbb{R}^{d,d}$	Gradient of $u : \mathbb{R}^d \to \mathbb{R}^d$
	continued on next page

continued from previous page	
$\nabla \cdot u = \frac{\partial u_1}{\partial x_1} + \cdots + \frac{\partial u_d}{\partial x_d} \in \mathbb{R}$	Divergence of $u : \mathbb{R}^d \to \mathbb{R}^d$
$(u \cdot \nabla)v = u_1 \frac{\partial v}{\partial x_1} + \cdots + u_d \frac{\partial v}{\partial x_d} \in \mathbb{R}^d$	Transport operator of $u, v : \mathbb{R}^d \to \mathbb{R}^d$
$\nabla \times u = \left(\frac{\partial u_3}{\partial x_2} - \frac{\partial u_2}{\partial x_3}, \frac{\partial u_1}{\partial x_3} - \frac{\partial u_3}{\partial x_1}, \frac{\partial u_2}{\partial x_1} - \frac{\partial u_1}{\partial x_2}\right)^T$	Curl operator of $u : \mathbb{R}^3 \to \mathbb{R}^3$
$\nabla \times u := \left(\frac{\partial u_2}{\partial x_1} - \frac{\partial u_1}{\partial x_2}\right) \in \mathbb{R}$	Auxiliary "curl" operator of $u : \mathbb{R}^2 \to \mathbb{R}^2$
$\nabla \times u := \left(\frac{\partial u}{\partial x_2}, -\frac{\partial u}{\partial x_1}\right)^T \in \mathbb{R}^2$	Auxiliary "curl" operator of $u : \mathbb{R}^2 \to \mathbb{R}$

C.3 Discretizations and finite element constructs

$\lambda = (\lambda_1, \ldots, \lambda_{d+1})$	Barycentric coordinate (p. 33)
$\mathcal{P}_k$	Function space of polynomials
$\mathcal{T}_h$	Triangularization (p. 30)
$\Delta_i \in \mathcal{T}_h$	Finite element (triangle or tetrahedron)
Δ_{ref}	Reference element (d-simplex)
N_Δ	# triangles
N_Γ	# boundary segments
N_N	# nodes p_i
N_B	# edges
N_{PCB}	# number of pointwise constraints on boundary (p. 48)
q_Δ	Quality measure (p. 31)
N_α	# u_α-nodes $\in \Omega \cup \Gamma_\alpha^N$
M_α	# u_α-nodes $\in \Gamma_\alpha^D$
$N_{T\alpha} = N_\alpha + M_\alpha$	# u_α nodes
$N_T = \sum_\alpha N_\alpha + M_\alpha$	# of total velocity nodes
$N_{Te} = \sum_\alpha N_\alpha$	# of eliminated velocity nodes
N_{p}	# pressure nodes
$N_{\mathrm{DOF}} = N_T + N_p$	# degrees of freedom
$N_{\mathrm{DOFe}} = N_{Te} + N_{Pe}$	# eliminated degrees of freedom
N_R	# of constraints on boundary (p. 37)
	continued on next page

continued from previous page	
V	Sobolev space for velocity
V_0	Sobolve space for velocity
V_e	Set of trial functions
W	Sobolev space for pressure
V^h	Discrete approximation of V
V_0^h	Discrete approximation of V_0
V_e^h	Discrete approximation of V_e
W^h	Discrete approximation of W
X_h	Space of piecewise polynomial functions
ϕ_i^p	Nodal shape function for pressure
ϕ_i^α	Nodal shape function for u_α

C.4 Matrices and vectors

u_h	Discrete velocity
p_h	Discrete pressure
$y_h = (u_h, p_h)^T$	Discrete solution vector
u_e	Eliminated velocity vector
p_e	Eliminated pressure vector
$y_e = (u_e, p_e)^T$	Eliminated solution vector
Gp	Discrete pressure gradient
Cp	Discrete pressure gradient
$C^T u$	Discrete divergence of velocity
$L(y_h, t)$	Load vector
$K(y_h, t) = -\frac{\partial L(y_h,t)}{\partial y_h}$	Stiffness matrix (p .50)
$A(u_e, u_e)$	Advection term
$D(u_e)$	(Possibly nonlinear) Discrete diffusion term
Du_e	Linear discrete diffusion term
$D(y_h, t)$	Discrete mass coefficient (p. 45)
$M(y_h, t)$	Discrete essential boundary conditions
$N(y_h, t) = -\frac{\partial M(y_h,t)}{\partial y_h}$	Jacobian of discrete essential boundary conditions
$\mathcal{N}(t)$	Null space of $N(t)$ (p. 48)
D_e	(Eliminated) mass matrix (p. 49)
M_e	(Eliminated) mass matrix

continued on next page

continued from previous page	
M_l	Lumped mass matrix
$L_e(y_e, t)$	(Eliminated) load vector (p. 49)
$K_e(y_e, t)$	(Eliminated) stiffness matrix
$J_e(y_e, t)$	(Eliminated) Jacobian matrix (p. 50)
$J_A(u_e)$	Jacobian matrix of advection term
$J_D(u_e)$	Jacobian matrix of diffusion term

List of Figures

List of Tables

Bibliography

[1] P. W. Atkins. *Physical Chemistry*. Oxford University Press, New York, 1998.

[2] I. Babuska and A. Aziz. On the angle condition in the finite element method. *SIAM Journal on Numerical Analysis*, 13(2):214–226, 1976.

[3] T. Baker. Element quality in tetrahedral meshes. In *Proc. 7th Int. Conf. Finite Element Methods in Flow Problems*, pages 1018–1024, Huntsville, 1989.

[4] G. Batchelor. *An Introduction to Fluid Dynamics*. Cambridge University Press, Cambridge, 1967.

[5] D. Braess. *Finite Elements*. Cambridge University Press, Cambridge, 2nd edition, 2001.

[6] J. H. Bramble and M. Zlámal. Triangular elements in the finite element method. *Math. Comp.*, 24:809–820, 1970.

[7] W. L. Briggs, V. E. Henson, and S. McCormick. *A Multigrid Tutorial*. SIAM, Philadelphia, 2000.

[8] D. L. Brown, R. Cortez, and M. L. Minion. Accurate projection methods for the incompressible Navier-Stokes equations. *Journal on Computational Physics*, 168:464–499, 2001.

[9] H.-J. Bungartz, A. Frank, F. Meier, T. Neunhoeffer, and S. Schulte. Fluid structure interaction: 3D numerical simulation and visualization of a micropump. Technical Report SFB-Bericht Nr. 342/06/97, TU München, 1997.

[10] H.-J. Bungartz, A. Frank, F. Meier, T. Neunhoeffer, and S. Schulte. Efficient treatment of complicated geometries and moving interfaces for CFD problems. In H.-J. Bungartz, F. Durst, and C. Zenger, editors, *High Performance Scientific and Engineering Computing*, volume 8 of *Lecture Notes in Computational Science and Engineering*, pages 113–123. Springer, Heidelberg, 1999.

[11] M. Cannone and S. Friedlander. Navier: Blow-up and collapse. *Notices of the AMS*, 50(1):7–13, 2003.

[12] A. J. Chorin. A numerical method for solving incompressible viscous flow problems. *Journal of Computational Physics*, 2:12–26, 1967.

[13] A. J. Chorin. The numerical solution of the Navier-Stokes equations for an incompressible fluid. *Bull. Am. Math. Soc.*, 73:928–931, 1967.

[14] A. J. Chorin. Numerical solution of the Navier-Stokes equations. *Mathematics of Computations*, 22:745–762, 1968.

[15] A. J. Chorin and J. E. Marsden. *A Mathematical Introduction to Fluid Dynamics*. Springer-Verlag, New York, 1993.

[16] T. J. Chung. *Computational Fluid Dynamics*. Cambridge University Press, Cambridge, 2002.

[17] J. Cihlář and P. Angot. Numerical solution of Navier-Stokes systems. *Numerical Linear Algebra with Applications*, 6:17–27, 1999.

[18] Comsol, Inc. *Femlab 2.3 Model Library*, 2001.

[19] Comsol, Inc. *Femlab 2.3 Reference Manual*, 2001.

[20] Comsol, Inc. *Femlab 2.3 User's Guide and Introduction*, 2001.

[21] P. Constantin. A few results and open problems regarding incompressible fluids. *Notices of the AMS*, 42(6):658–663, 1995.

[22] R. Cools and P. Rabinowitz. Monomial cubature rules since 'Stroud': A compilation. *J. Comput. Appl. Math.*, 48:309–326, 1993.

[23] J. G. Currie. *Fundamental Mechanics of Fluids*. MacGraw-Hill, New York, 1993.

[24] P. Deuflhard and F. Bornemann. *Numerische Mathematik II*. de Gruyter, Berlin, 1994.

[25] G. Dhatt and G. Touzot. *The finite element method displayed*. John Wiley & Sons, Chichester, 1982.

[26] D. A. Dunuvant. High degree efficient symmetrical Gaussian quadrature rules for the triangle. *Int. Journal Num. Meth. Engineering*, 21:1129–1148, 1980.

[27] M. V. Dyke. *An Album of Fluid Motion*. Parabolic Press, Stanford, 1982.

[28] J. D. et al. Proposal for 3D unstructured tetrahedral mesh optimization. In *Proc. 7th Int. Mesh. Roundtable*, Dearborn, 1998.

[29] L. Euler. Principles géneraux du movement des fluides. *Mém. Acad. Sci. Berlin*, 11:274–315, 1755.

[30] R. D. Falgout. Adaptive algebraic multigrid. Lawrence Livermore National Laboratory, 2002.

[31] C. L. Fefferman. Existence & smoothness of the Navier-Stokes equation. Available online at `http://www.claymath.org/prizeproblems/navier_stokes.pdf`, 2000. Clay Mathematics Institute.

[32] J. H. Ferziger and M. Perić. *Computational Methods for Fluid Dynamics*. Springer-Verlag, Berlin, 2001.

[33] R. P. Feynman, R. B. Leighton, and M. Sands. *The Feynman Lectures on Physics*, volume 2. Addison-Wesley, Reading, 7th edition, 1977.

[34] D. A. Field. Qualitative measures for initial meshes. *Intl. J. for Num. Meth. Engr.*, 47:887–906, 2000.

[35] P. F. Fischer. Projection techniques for iterative solution of $Ax = b$ with successive right-hand sides. *Comput. Methods Appl. Mech. Engrg.*, 163:193–204, 1998.

[36] B. Fornberg. *A Practical Guide to Pseudospectral Methods*. Cambridge Monographs on Applied and Computational Mathematics. Cambridge University Press, Cambridge, 1998.

[37] P. J. Frey and P. L. George. *Mesh Generation*. Hermes Science Publishing, Oxford, 2000.

[38] G. P. Galdi. *An Introduction to the Mathematical Theory of the Navier-Stokes Equations*, volume 1. Springer-Verlag., New York, 1994.

[39] G. P. Galdi. *An Introduction to the Mathematical Theory of the Navier-Stokes Equations*, volume 2. Springer-Verlag., New York, 1994.

[40] G. P. Galdi, J. G. Heywood, and R. Rannacher. *Fundamental directions in mathematical fluid mechanics*. Birkhäuser, Basel, 2000.

[41] P. George. *Automatic Mesh Generation – Application to Finite Element Methods*. John Wiley & Sons, New York, 1991.

[42] H. Gevgilili and D. Kalyon. Catastrophic failure of the no-slip condition at the wall during torsional flows and development of gross surface irregularities during capillary flow of three polymers. *Society of Plastics Engineers ANTEC Technical Papers*, 48, 2002.

[43] N. Gibbs, W. Poole, and P. Stockmeyer. An algorithm for reducing the bandwidth and profile of a sparse matrix. *SIAM J. Num. Anal.*, 13(2):236–250, 1976.

[44] R. Glowinski. *Numerical Methods for Fluids*, volume 3 of *Handbook of Numerical Analysis*. North-Holland, Amsterdam, 2003.

[45] K. Goda. A multistep technique with implicit difference schemes for calculating two- or three-dimensional cavity flows. *Journal on Computational Physics*, 30:76–95, 1979.

[46] G. H. Golub and C. F. V. Loan. *Matrix Computations*. Johns Hopkins University Press, 1996.

[47] R. A. Granger. *Fluid Mechanics*. Dover Publications, New York, 1995.

[48] P. M. Gresho and R. L. Sani. *Incompressible Flow and the Finite Element Method*, volume 2. John Wiley & Sons, Chichester, 2000.

[49] P. M. Gresho and R. L. Sani. *Incompressible Flow and the Finite Element Method*, volume 1. John Wiley & Sons, Chichester, 2000.

[50] M. Griebel, T. Dornseifer, and T. Neunhoeffer. *Numerical Simulation in Fluid Dynamics: A Practical Introduction*. SIAM, Philadelphia, 1997.

[51] M. Griebel, T. Neunhoeffer, and H. Regler. Algebraic multigrid methods for the solution of the Navier-Stokes equations in complicated geometries. *International Journal for Numerical Methods in Fluids*, 26(3):281–301, 1998.

[52] R. B. Guenther and J. W. Lee. *Partial Differential Equations of Mathematical Physics and Integral Equations*. Prentice Hall, Englewood Cliffs, N.J., 1988.

[53] M. E. Gurtin. *An Introduction to Continuum Mechanics*. Academic Press, San Diego, 1981.

[54] E. Guyon, J.-P. Hulin, L. Petit, and C. D. Mitescu. *Physical Hydrodynamics*. Oxford University Press, Oxford, 2001.

[55] E. Hairer, S. Nørsett, and G. Wanner. *Solving Ordinary Differential Equations I.* Springer-Verlag, Berlin, 1986.

[56] E. Hairer and G. Wanner. *Solving Ordinary Differential Equations II.* Springer-Verlag, Berlin, 2002.

[57] F. H. Harlow. Hydrodynamic problems involving large fluid distortion. *J. Assoc. Comp. Mach.*, 4:137 ff., 1957.

[58] F. H. Harlow and E. Welch. Numerical calculation of time-dependent viscous incompressible flow of fluid with free surface. *Phys. Fluids*, 8:2182–2189, 1965.

[59] H. Haschke and W. Heinrichs. Splitting techniques for the Navier-Stokes equations. In P. Neittaanmk̈i, T. Tiihonen, and P. Tarvainen, editors, *Numerical Mathematics and Advanced Applications*, 2000.

[60] W. Heinrichs. Spectral multigrid techniques for the Navier-Stokes equations. *Computer Methods in Applied Mechanics and Engineering*, 106:297–314, 1993.

[61] W. Heinrichs. Splitting techniques for the pseudospectral approximation of the unsteady Stokes equations. *SIAM Journal on Numerical Analysis*, 30:19–39, 1993.

[62] W. Heinrichs. High order time splitting techniques for the Stokes equations. *J. Scient. Comput.*, 11(4):397–410, 1996.

[63] W. Heinrichs. Operator splitting for the Stokes equations. In G. Wittum, editor, *Proceedings of the Fifth European Multigrid Conference*, pages 101–113, 1998.

[64] W. Heinrichs. Splitting techniques for the unsteady Stokes equations. *SIAM Journal on Numerical Analysis*, 35(4):1646–1662, 1998.

[65] W. Heinrichs and H. Haschke. Splitting techniques with staggered grids for the Navier-Stokes equations. *Journal on Computational Physics*, 168:131–154, 2001.

[66] V. E. Henson. An algebraic multigrid tutorial. Lawrence Livermore National Laboratory, 1999.

[67] U. Hesse. *Implizite Verfahren höherer Ordnung und Mehrgitterverfahren zur Lösung der Shallow-Water-Equations*. PhD thesis, University of Düsseldorf, 1998.

[68] C. Hirt, B. Nichols, and N. Romero. SOLA – A numerical solution algorithm for transient fluid flows. Technical Report Rep. LA-5852, Los Alamos Scientific Lab., 1975.

[69] M. Hochbruck. Numerical analysis. Lecture notes, 2004.

[70] H. H. Hu, N. A. Patankar, and M. Y. Zhu. Direct numerical simulations of fluid-solid systems using the arbitrary Lagrangian-Eulerian technique. *Journal of Computational Physics*, 169:427–462, 2001.

[71] A. Jameson and J. Vassberg. Computational fluid dynamics (CFD) for aerodynamic design: Its current and future impact. In *Proceedings of the 39th AIAA Aerospace Sciences Meeting and Exhibit*, 2001.

[72] D. D. Joseph. Interrogations of direct numerical simulation of solid-liquid flow. Available online at `http://www.efluids.com/efluids/books/joseph.htm`, 2004.

[73] J. V. Kan. A second-order accurate pressure-correction scheme for viscous incompressible flow. *SIAM J. Sci. Stat. Comput.*, 7:870–891, 1986.

[74] G. E. Karniadakis, M. Israeli, and S. A. Orszag. High-order splitting methods for the incompressible Navier-Stokes equations. *Journal of Computional Physics*, 97:414–443, 1991.

[75] T. Kato. Strong L^p solutions of the Navier-Stokes equations in $\mathbb{R}^m$ with applications to weak solutions. *Math. Z.*, 187:471–480, 1984.

[76] J. Kim and P. Moin. Application of a fractional-step method to incompressible Navier-Stokes equations. *Journal of Computational Physics*, 59:308–323, 1985.

[77] P. Knabner and L. Angermann. *Numerik partieller Differentialgleichungen*. Springer-Verlag, Berlin, 2000.

[78] P. Kunkel and V. Mehrmann. Generalized inverses of differential-algebraic operators. *SIAM Journal on Matrix Analysis and Applications*, 17(2):426–442, 1996.

[79] P. Kunkel and V. Mehrmann. A new class of discretization methods for the solution of linear differential-algebraic equations with variable coefficients. *SIAM Journal on Numerical Analysis*, 33(5):1941–1961, 1996.

[80] O. A. Ladyshenskaja. *Funktionalanalytische Untersuchungen der Navier-Stokesschen Gleichungen*. Akademie-Verlag, Berlin, 1965.

[81] L. D. Landau and E. M. Lifshitz. *Hydrodynamik*, volume VI of *Lehrbuch der Theoretischen Physik*. Akademie-Verlag, Berlin, 1966.

[82] H. P. Langtangen, K.-A. Mardal, and R. Winter. Numerical methods for incompressible viscous flow. *Advances in Water Resources*, 25(8-12):1125–1146, 2002.

[83] J. Leray. Études de diverses équations intégrales non linéaires et de quelques problèmes que pose l'hydrodynamique. *J. Math. Pures et Appl.*, 12:1–82, 1933.

[84] M. Marion and R. Temam. Navier-Stokes equations. In *Handbook of Numerical Analysis*, volume 1, pages 197–462. North-Holland, Amsterdam, 1998.

[85] D. Meschede. *Gerthsen Physik*. Springer-Verlag, Berlin, 2003.

[86] C. L. M. H. Navier. Mémoire sur les lois du movement des fluides. *Mém. Acad. Sci. Inst. France*, 6:389–440, 1822.

[87] C. K. Newman. *Exponential Integrators for the Incompressible Navier-Stokes equations*. PhD thesis, Virginina Polytechnic Institute and State University, 2003.

[88] H. Oertel. *Numerische Strömungsmechanik*. Springer-Verlag, Berlin, 1995.

[89] H. Oertel. *Strömungsmechanik*. Vieweg, Braunschweig/Wiesbaden, 1999.

[90] R. Panton. *Incompressible Flow*. John Wiley and Sons, Inc., New York, 2nd edition, 1996.

[91] V. T. Parthasarathy, C. M. Graichen, and A. F. Hathaway. A comparison of tetrahedral quality measures. *Fin. Elem. Anal. Des.*, 15:255–261, 1993.

[92] P. Pébay and T. Baker. Analysis of triangle quality measures. *Math. Comp.*, 72(244):1817–1839, 2003.

[93] N. A. Petersson. Stability of pressure boundary conditions for Stokes and Navier-Stokes equations. *Journal on Computational Physics*, 172:40–70, 2001.

[94] R. Peyret. *Handbook of Computational Fluid Mechanics*. Academic Press, London, 2000.

[95] R. Peyret. *Spectral Methods for Incompressible Viscous Flow*. Springer-Verlag, Heidelberg, 2002.

[96] C. Pozrikidis. *Introduction to Theoretical and Computational Fluid Dynamics*. Oxford University Press, New York, 1997.

[97] A. Prohl. *Projection and quasi-compressibility methods for solving the incompressible Navier-Stokes equations*. Teubner, Stuttgart, 1997.

[98] L. Quartapelle. *Numerical Solution of the Incompressible Navier-Stokes Equations*. Birkhäuser, Basel, 1993.

[99] A. Quarteroni and A. Valli. *Numerical Approximation of Partial Differential Equations*. Springer-Verlag, Berlin, 1994.

[100] R. Rannacher. On Chorin's projection method for the incompressible Navier-Stokes equations. In J. G. H. et al., editor, *The Navier-Stokes equations II - Theory and Numerical Methods*, volume 1530 of *Lecture Notes in Mathematics*, Berlin, 1992. Springer-Verlag.

[101] R. Rannacher. Finite element methods for the incompressible Navier-Stokes equations. Unpublished, University of Heidelberg, 1999.

[102] J. W. Ruge and K. Stüben. Algebraic multigrid. In S. F. McCormick, editor, *Multigrid Methods*, number 3 in Frontiers in Applied Mathematics. SIAM, Philadelphia, 1987.

[103] J.-M. Sautter. Numerical simulation in fluid dynamics. Master's thesis, University of Tübingen, 1997.

[104] H. Schlichting and K. Gersten. *Boundary-Layer Theory*. Springer-Verlag, Berlin, 2000.

[105] P. Schreiber and S. Turek. An efficient finite element solver for the nonstationary incompressible Navier-Stokes equations in two and three dimensions. In *Proc. Workshop 'Numerical Methods for the Navier-Stokes Equations', Heidelberg, Oct. 25-28, 1993*, volume 47 of *Notes on Numerical Fluid Mechanics*. Vieweg, 1993.

[106] B. Simeon. Wissenschaftliches Rechnen in der Festkörpermechanik. Lecture notes, Technical University of Munich, 2002.

[107] G. Strang. Approximation in the finite element method. *Numer. Math.*, 19:81–98, 1972.

[108] G. Strang and G. Fix. *An analysis of the finite element method*. Prentice Hall, Englewood Cliffs, N.J., 1973.

[109] A. H. Stroud. *Approximate Calculation of Multiple Integrals*. Prentice Hall, Englewood Cliffs, N.J., 1971.

[110] K. Stüben. *Algebraic Multigrid (AMG): An Introduction with Applications*. GMD Report No. 70. Gesellschaft für Mathematik und Datenverarbeitung, St. Augustin, Bonn, 1999.

[111] K. Stüben. *A Review of Algebraic Multigrid*. GMD Report No. 69. Gesellschaft für Mathematik und Datenverarbeitung, St. Augustin, Bonn, 1999.

[112] R. Temam. Sur l'approximation de la solution des equations de Navier-Stokes par la méthode des fractionnaires I. *Arch. Rational Mech. Anal.*, 32(2):135–153, 1969.

[113] R. Temam. Sur l'approximation de la solution des equations de Navier-Stokes par la méthode des fractionnaires II. *Arch. Rational Mech. Anal.*, 33(5):377–385, 1969.

[114] J. F. Thompson, Z. U. A. Warsi, and C. W. Mastin. *Numerical Grid Generation*. North-Holland, Amsterdam, 1985.

[115] M. Tome and S. McKee. Gensmac: A computational marker and cell method for free surface flows in general domains. *Journal of Computational Physics*, 110:171–186, 1994.

[116] U. Trottenberg, C. Oosterlee, and A. Schüller. *Multigrid*. Academic Press, London, 2001.

[117] C. Truesdell and K. Rajagopal. *An Introduction to the Mechanics of Fluids*. Birkhäuser, Boston, 2000.

[118] S. Turek. A comparative study of some time-stepping techniques for the incompressible Navier-Stokes equations: From fully implicit nonlinear schemes to semi-implicit projection methods. *International Journal for Numerical Methods in Fluids*, 22:987–1011, 1996.

[119] S. Turek. *Multilevel Pressure Schur Complement Techniques for the Numerical Solution of the Incompressible Navier-Stokes Equations*. Habilitationsschrift, Universität Heidelberg, 1997.

[120] S. Turek. On discrete projection methods for the incompressible Navier-Stokes equations: An algorithmical approach. *Comp. Methods Appl. Mech. Engrg.*, 143:271–288, 1997.

[121] S. Turek. *Efficient solvers for incompressible flow problems: An algorithmic and computational approach*, volume 6 of *Lecture Notes in Computational Science and Engineering*. Springer-Verlag, Berlin, 1999.

[122] H. K. Versteeg and W. Malalasekera. *An Introduction to Computational Fluid Dynamics: The Finite Volume Method*. Addison-Wesley, Reading, 1996.

[123] J. Weickert. Navier-Stokes equations as a differential-algebraic system. Preprint sfb393/96-08, Technische Universität Chemnitz-Zwickau, 1996.

[124] N. Whitaker. Numerical solution of the Hele-Shaw equations. *Journal on Computational Physics*, 90(1):176–199, 1990.

[125] N. Whitaker. Some numerical methods for the Hele-Shaw equations. *Journal on Computational Physics*, 111(1):81–99, 1994.