hp-FEM for Two-component Flows with Applications in Optofluidics

Zur Erlangung des akademischem Grades eines

Doktors der Naturwissenschaften

von der Fakultät für Mathematik
des Karlsruher Instituts für Technologie (KIT)
genehmigte

Dissertation

von

M. Sc. Eng. Staffan Ronnås

aus Hägersten, Schweden

Tag der mündlichen Prüfung: 27. Juni 2012

Referent: Prof. Dr. Vincent Heuveline
Korreferent: Prof. Dr. Willy Dörfler
Korreferent: Prof. Dr. Thomas Wihler

Abstract

This thesis is concerned with the application of hp-adaptive finite element methods to a mathematical model of immiscible two-component flows. With the aim of simulating the flow processes in microfluidic optical devices based on liquid-liquid interfaces, we couple the time-dependent incompressible Navier-Stokes equations with a level set method to describe the flow of the fluids and the evolution of the interface between them. A least-squares finite element formulation is used to stabilize the advection equation governing the level set function. Furthermore, we review the theory related to hp-adaptive finite element methods, and discuss our own implementation in detail. Finally, we present an hp-adaptive strategy for the solution of the two-component flow model in the stationary case, and illustrate the method with numerical results for a dynamically reconfigurable waveguide.

Zusammenfassung

Diese Arbeit behandelt die Anwendung hp-adaptiver Finite Elemente Methoden auf ein mathematisches Modell für die Strömung zweier nicht mischbarer Flüssigkeiten. Das Ziel ist die numerische Simulation von mikroskopischen Strömungsvorgängen in optischen Komponenten, deren Konfiguration von der Grenzschicht zwischen Flüssigkeiten beeinflusst wird. Dazu werden die zeitabhängigen inkompressiblen Navier-Stokes Gleichungen zur Beschreibung des Flusses mit einer Level-Set-Methode zur Beschreibung der Evolution der Grenzschicht verknüpft. Zur Stabilisierung der Advektions-Gleichung für die Level-Set Funktion wird eine Least-Squares Finite Elemente Formulierung verwendet. Weiter wird die Theorie zu hp-adaptiven Finite Elemente Methoden beschrieben und deren hier durchgeführte Implementierung ausführlich diskutiert. Zum Abschluss wird eine hp-adaptive Vorgehensweise zur Lösung des Flussmodells für den stationären Fall gezeigt. Numerische Resultate für einen dynamisch konfigurierbaren Wellenleiter werden zur Veranschaulichung dieser Vorgehensweise dargestellt.

ACKNOWLEDGMENTS

It is a great pleasure to thank those who have supported me during my PhD studies and in preparing this work.

First and foremost, I express my gratitude to Prof. Dr. Vincent Heuveline, who offered me the possibility to join his institute and immerse myself in the research that resulted in this thesis. Along the way, he has provided guidance, encouragement, and insightful discussions; and he was always willing to help me find my way out of dead ends.

Furthermore, I am grateful to Prof. Dr. Willy Dörfler for his time and effort to provide feedback and constructive criticism which has helped me improve this work. I also thank Prof. Dr. Thomas Wihler for inviting me to Bern and for his valuable input during our discussion there.

Many thanks go to the co-workers of the Institute of Applied and Numerical Mathematics IV, for the wonderful working atmosphere and team spirit that they all take part in creating. Without the daily rituals at the coffee machine or the excursions to the Mensa, I don't know where I would have found the motivation to finish this work. I especially want to thank Martin Baumann, Thomas Gengenbach, and Michael Schick who took part in correcting the thesis. A particular thanks also to Jun.-Prof. Dr. Jan-Philipp Weiß, without whom I would never have joined the group in the first place.

I also extend my gratitude to the students and professors of the Research Training Group 1294, who helped broaden my mathematical perspectives through the many interesting seminars, courses, and personal discussions.

On a more personal note, I am deeply grateful to the members of my family and my extended family, who are always there when I need them. Although I am far away, it feels as though they lived next door, and coming home is always as if I was never gone.

I am truly thankful to the Grella family, for the many invitations over weekends and holidays, all the practical assistance, and the introduction into the German way of life that they have given me. In many ways, I see them as my second family.

Finally, my deepest thanks to Katharina, for her patience and loving support throughout these years. Every day, she makes me remember that there is something greater in life than maths and computers.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

This is an (incomplete) list of mathematical symbols and their meanings in this work. Boldface is used to indicate vector quantities.

Operators

$\cdot$	Scalar-product.	Δ	Laplace operator.
$\times$	Cross-product.	$\mathcal{I}$	Identity operator.
∇	Gradient operator.	$\ll\cdot\gg$	Average operator.
$\nabla\cdot$	Divergence operator.	$[\![\cdot]\!]$	Jump operator.
$\nabla\times$	Curl operator.	$\frac{\mathrm{D}}{\mathrm{D}t}$	Material derivative.

Geometrical Quantities

$\mathbb{R}, \mathbb{C}$	Real and complex numbers.	$\boldsymbol{n}$	Normal vector
Ω	Domain in $\mathbb{R}^d$.	κ	Curvature.
$\boldsymbol{x}$	Point in physical space.	δ	Dirac delta distribution.
Γ	Interface between fluids.	H	Heaviside function.

Physical Quantities and Model Variables

$\boldsymbol{u}$	Velocity field.	μ	Dynamic viscosity.
p	Pressure.	ϕ	Level set function.
σ	Stress tensor.	τ	Interfacial tension coefficient.
ρ	Density.		

$\tilde{\boldsymbol{E}}, \tilde{\boldsymbol{H}}$	Time-dependent electric and magnetic fields.
$\boldsymbol{E}, \boldsymbol{H}$	Amplitude of time-harmonic electric and magnetic fields.

Objects in Finite Element Analysis

$\mathcal{L}$	Abstract linear operator.
$L^2(\Omega)$	Space of square-integrable functions on Ω.
$H^1(\Omega)$	Space $\psi \in L^2(\Omega) : \nabla\psi \in L^2(\Omega)$.
V, V_h	Cont. and disc. space.
ψ, ψ_h	Cont. and disc. functions.
$\mathcal{M}$	Mesh.
K	Cell of a mesh.
h	Cell size
L	Legendre polynomial.
ℓ	Lobatto polynomial.
Δt	Time-step size.
η, η_K	Error indicator (global and local).

1 INTRODUCTION

Since their commercial introduction 60 years ago, computers have profoundly influenced modern society. Whereas some uses of computers have only emerged more recently with the introduction of the personal computer and the expansion of the Internet, they were already at that time being employed for solving problems in diverse fields of science and engineering. Both the availability and the performance of these general-purpose machines have increased tremendously, especially in the last 20 years, and society as well as the individual are becoming more and more reliant on them.

What might seem at first glance to be a revolutionary development, should however rather be seen as part of an evolutionary process, where the description of natural phenomena and technical processes is increasingly being formalized as mathematical models, and questions about these phenomena and processes are to a growing extent being answered by solving problems based on such models. The complexity of these problems often motivates the use of the various approximation techniques that are collectively known as *numerical methods*, in order to be able to obtain a solution. The basis for most of these methods was developed several hundred years ago, and they have been employed ever since. The advent of the computer, and the possibility to perform enormous amount of computations reliably and quickly, has merely extended the degree of detail in the models, and the accuracy of the results, that is achievable.

Nevertheless, computing is increasingly being seen as a third type of scientific activity, which complements the well-established branches theory and experimentation. That the term *Computational Science* is overtaking *Scientific Computing* in popularity shows that this view is becoming more and more widespread. This idea must, however, be considered in the context of the use of mathematical modeling and numerical approximation as described above. This is indeed complementary to theory and experimentation in the sense that the objective is not to explore reality directly, but rather to create useful abstractions, and to apply them to specific problems.

The methodology of Computational Science consists of several levels of abstraction, as shown schematically in Fig. 1.1. Starting from a problem arising in a domain of science or engineering, an abstract mathematical model is created, which attempts to capture the significant aspects of the problem. Although several simplifications are typically made at this point, the resulting mathematical problem is often too complex to solve analytically. Hence, numerical methods are

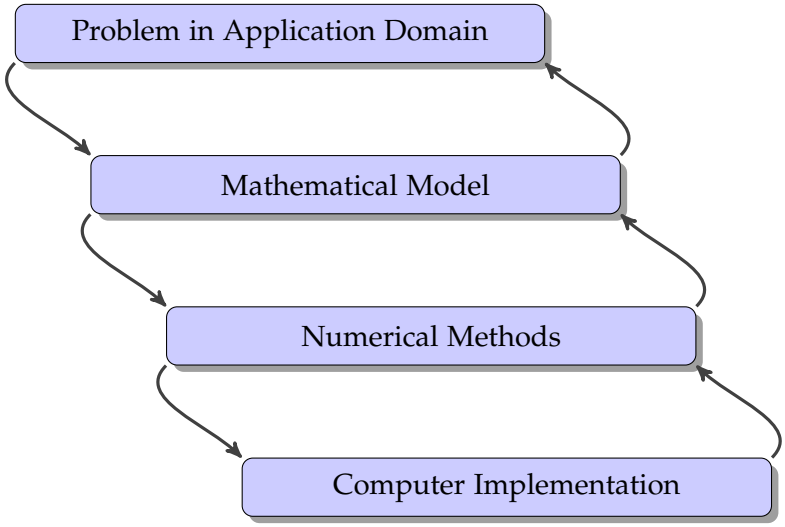


Figure 1.1: Levels of abstraction in computational science.

employed to provide a simplified problem whose solution will approximate that of the original problem. Solving even this simplified problem is often only practical with a computer, which will require a computer implementation of the problem in some form. This last step can consist in using existing software, programming new software, or even designing specialized hardware.

The work presented in this thesis spans all four abstraction levels in this picture. Motivated by problems arising from the emerging domain of optofluidics, it develops a mathematical model which is designed to solve problems involving two-component flows arising in this area. Furthermore, it treats the question of applying hp-adaptive finite element methods for solving problems with this model, and describes a corresponding software implementation.

In this sense, the presented work treats all four abstraction levels of the Computational Science methodology for a specific problem. In the rest of this chapter, the two major themes of the thesis will be introduced in some more detail, before the rest of the document is outlined briefly.

1.1 OPTOFLUIDICS AND TWO-COMPONENT FLOWS

Optofluidics is a scientific domain that integrates technologies from two separate areas: microfluidics and optics, with the aim of creating efficient and adaptable devices for use in chemical and biological analysis, communication systems and imaging. The thematic focus of the research in optofluidics is the interaction of fluids and light at the microscopic level. Overviews of this rapidly developing field are provided in [47, 64].

One of the main topics in optofluidics is the development of optical elements that can be integrated on a microfluidic platform. Besides

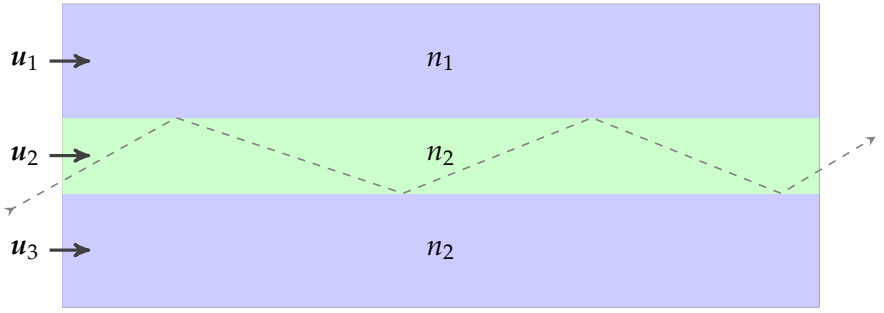


Figure 1.2: Optofluidic waveguide with a liquid-liquid interface.

being an important component in so-called Lab-on-a-Chip devices [34], the use of a fluid medium brings with it advantages which could make such elements preferable to their solid-state counterparts in certain situations. Using a fluid medium makes it possible to dynamically reconfigure the element to adapt it to new operating conditions, for instance changing the focal length of a lens or the characteristic wavelength of a laser. Another advantage is the facility of achieving high-quality surfaces at low production cost.

In this work, we consider the example of a simple microfluidic waveguide, first described in [140], which is shown schematically in Fig. 1.2. Two fluids, with different refractive indices n_1 and n_2, flow in a microchannel. Light passes through the inner *core* fluid, which has a higher refractive index than the outer *cladding* fluid, and it will thus be totally reflected if the angle of incidence is greater than the critical angle, which is determined by the ratio of n_1 to n_2. The geometry of the waveguide is determined by the interaction of the two fluids, which can be controlled via the inflow velocities $\boldsymbol{u}_1$, $\boldsymbol{u}_2$, and $\boldsymbol{u}_3$.

Several other optical elements based on liquid-liquid interfaces have also been investigated. Examples include lenses [129], gratings [63], and lasers [137].

The central aspect of the flow process in these devices is the interaction between two or more liquids of different chemical composition, so-called *multi-component* flow. In this work, we set up a mathematical model for two-component flow with immiscible fluids. The model consists of the incompressible Navier-Stokes equations coupled with a description of the interface that is based on the Level Set Method.

A specific combination of numerical methods, which include finite element and finite difference methods, is used to discretize the model. As an example of how the model and its discretization can be used, we apply them to simulate the two-component flow in a straight channel, similar to the waveguide presented in [140].

1.2 HP-ADAPTIVE FEM

The second major theme of the thesis is that of hp-adaptive finite element methods. This topic brings together the two main aspects in the design of approximation spaces for the finite element method, namely the choice of the mesh, and the choice of the polynomial degrees in each cell. By combining the possibility to vary the fineness of the mesh through local refinement (*h-FEM*) with the use of an appropriate distribution of polynomial degrees (*p-FEM*), it is possible to accelerate the convergence of the method, and hence obtain accurate solutions with fewer unknowns.

Allowing local variation of both cell size and polynomial degree makes the use hp-FEM more complex than standard FEM, both in terms of implementation and application. The design of *adaptive algorithms*, which attempt to compute optimal meshes and degree distributions with respect to some error measure automatically, is still an important open research question.

With the availability of increasingly powerful computers, one might question whether the use of such advanced discretization methods is really necessary. Can the same results not be obtained more economically with a simpler discretization and a faster computer? For certain problems, this might be the case, but in general the demands on the numerical methods with respect to accuracy, robustness and efficiency, are increasing even faster than the development in computer technology. As the problems to which a method is being applied becomes more complex, the need for sophisticated discretizations that keep the size of the problem moderate while still attaining the required accuracy becomes apparent.

In this context, the issue of software quality should also be mentioned. Numerical software is being used by ever more people, and the expectations with respect to usability and performance are rising correspondingly. In addition to investigating hp-adaptive FEM from a theoretical point of view, we have therefore also devoted much effort to the development of hp-FEM support for the finite element library HiFlow^3 , the details of which are also described in this work.

1.3 OUTLINE OF THE THESIS

The thesis is structured as follows. Chapter 2 derives a model for one- and two-component flow for immiscible, incompressible fluids from fundamental physical principles. For the treatment of the interface between the fluids, the model is extended in Chapter 3 with an interface representation based on a level set function, and a corresponding description of the evolution of this function. Chapter 4 describes the discretization of the coupled flow and interface models using a finite difference method in time, and finite elements in space. For the flow

model, a standard approximation scheme is used, whereas hp-FEM is employed for the interface model. This chapter also describes specific approximations used for the approximation of the interface tension force and the material parameters, which vary in space and time.

Chapter 5 is concerned with the subject of hp-FEM, both from a theoretical and a technical point of view. The described methods are then applied to the investigation of the waveguide example in Chapter 6. The concluding Chapter 7 summarizes the thesis and discusses its wider significance as well as future directions.

2

FLOW MODEL

This chapter presents the mathematical models describing the physical phenomena that will be investigated in the thesis. We first introduce the incompressible Navier-Stokes equations, which describe the flow of a volume-preserving Newtonian fluid. In the second part, a model for two-component flow of immiscible fluids is derived.

2.1 GOVERNING EQUATIONS FOR INCOMPRESSIBLE FLUID FLOW

The theory of fluid mechanics is based on the fundamental laws of classical mechanics, taken together with the *continuum hypothesis*, through which one ignores the particle nature of the fluid molecules. This section will introduce the governing equations for incompressible flow with a single fluid component, which forms the basic flow model in this work. The exposition is based on the first three chapters of [125].

Under the continuum hypothesis, one imagines the fluid to consist of a continuous mass, in which even an infinitesimally small volume contains a large number of molecules. The flow variables, such as velocity or pressure, that are defined at each point of the fluid, are considered to be averages of corresponding quantities associated with the individual molecules in a small volume around the point. Some of the flow variables, such as temperature, are only defined at the macroscopic level, as averages of microscopic properties of the molecules, in this case their kinetic energies.

The continuum hypothesis is an idealization, which will lose its validity when the length scale of the situation under consideration is too small. Generally, the characteristic length l of the problem should be much larger than the *mean free path* λ of the molecules in the fluid. The relation between these two length scales is measured by the *Knudsen number* $Kn = \lambda l^{-1}$. The continuum hypothesis is valid when $Kn \ll 1$. A further discussion of the limits of the continuum model can be found e.g. in [78].

Our first goal is to derive the equations that govern the flow of fluids. For this purpose, consider a simply connected domain in d-dimensional space $\Omega_0 \subset \mathbb{R}^d$. Taking this domain to be the *reference configuration*, the deformed state of the fluid at time t, the *physical configuration*, is denoted Ω_t. The mapping $\boldsymbol{\varphi} : \Omega_0 \times [0, \infty) \to \Omega_t$ maps reference points $\boldsymbol{\xi} \in \Omega_0$ to physical points $\boldsymbol{x} = \boldsymbol{\varphi}(\boldsymbol{\xi}, t)$ at a fixed time t. The situation is depicted in Fig. 2.1. The transformation $\boldsymbol{\varphi}$ is required to be smooth and bijective with a smooth inverse.

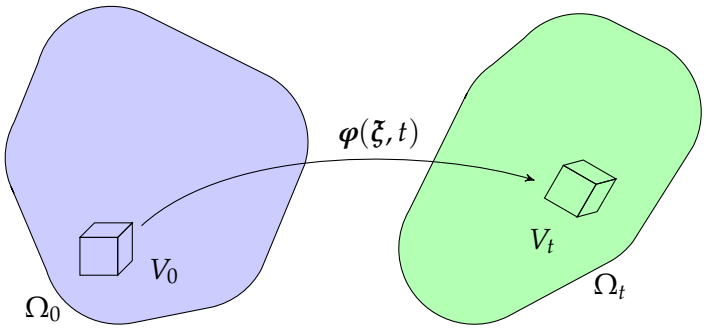


Figure 2.1: The mapping between the reference and physical configurations.

Two different perspectives can be used when considering the evolution of the fluid, corresponding to the two configurations Ω_0 and Ω_t. In the *Lagrangian*, or *material* description, the points $\boldsymbol{\xi} \in \Omega_0$ are used to identify the particles in the fluid. In this context, the vector $\boldsymbol{\xi}$ is called *material point*, and the corresponding coordinates are called *material coordinates*. The alternative *Eulerian* or *spatial* description uses a system of *spatial* coordinates to describe points $\boldsymbol{x} = \boldsymbol{\varphi}(\boldsymbol{\xi}, t)$ in Ω_t. In the Eulerian perspective, one considers the evolution of the flow variables in space, without taking into account the identity of the particles.

In fluid mechanics, the Eulerian description offers the advantage of corresponding directly to the observable quantities of a system. Instead of trying to follow the paths of all particles in the fluid, the Eulerian perspective describes the evolution of flow variables, such as velocity and pressure, as functions of space and time in a fixed reference frame.

However, the Lagrangian description is better suited for the derivation of the equations governing the flow from the basic conservation laws of physics. The strategy that will be followed in this section is to take the conservation laws stated in the reference configuration as the point of departure, and to transform these to Eulerian coordinates.

First, we will introduce some mathematical tools that simplify this transformation. Consider the trajectory of the material point $\boldsymbol{\xi}$, which is described by $\boldsymbol{x}(\boldsymbol{\xi}, t) = \boldsymbol{\varphi}(\boldsymbol{\xi}, t)$. The velocity of the particle relative to a fixed reference frame is given by

$$\boldsymbol{u}\left(\boldsymbol{\xi}, t\right) = \frac{\partial \boldsymbol{\varphi}}{\partial t}\left(\boldsymbol{\xi}, t\right). \tag{2.1}$$

Using the inverse transformation $\boldsymbol{\xi} = \boldsymbol{\varphi}^{-1}(\boldsymbol{x}, t)$, the velocity at a fixed point in space $\boldsymbol{x}$ can then be transformed to spatial coordinates using

$$\boldsymbol{u}\left(\boldsymbol{x}, t\right) = \boldsymbol{u}\left(\boldsymbol{\varphi}^{-1}(\boldsymbol{x}, t), t\right). \tag{2.2}$$

Given a quantity $f(\boldsymbol{x}, t)$ which varies in space and time, we need to be able to determine the evolution of that quantity for the material

point $\boldsymbol{\xi}$ associated with $\boldsymbol{x}$ at time t. This is given by the *substantial* or *material* derivative

$$\frac{\mathrm{D}f}{\mathrm{D}t}(\boldsymbol{x},t) = \frac{\partial f}{\partial t} + \frac{\partial \boldsymbol{x}}{\partial t}\cdot\frac{\partial f}{\partial \boldsymbol{x}} \tag{2.3}$$

$$= \frac{\partial f}{\partial t}(\boldsymbol{x},t) + \boldsymbol{u}(\boldsymbol{x},t)\cdot\nabla f(\boldsymbol{x},t), \tag{2.4}$$

which is nothing else than the total derivative of $f(\boldsymbol{x}(\boldsymbol{\xi},t),t)$ with respect to t, computed using the chain rule. In general, f could be a tensor of any order, and describe for instance the temperature or velocity. The material derivative is used in conservation laws to express the rate of change of a quantity that is bound to a material point using spatial coordinates.

A related result is *Reynolds' Transport Theorem*, which transforms the material derivative of a quantity integrated over a fixed volume of fluid $V_0 \subset \Omega_0$ to an integral of differentiated quantities in spatial coordinates:

Theorem 1. *(Reynolds' Transport Theorem)*
Consider a volume of fluid $V_0 \subset \Omega_0 \subset \mathbb{R}^d$, and the corresponding transformed volume $V_t = \boldsymbol{\varphi}(V_0,t)$ in the physical domain Ω_t. Let $f(\boldsymbol{x},t)$ be a tensor field that is sufficiently smooth for the integrals to be well-defined. Then,

$$\frac{\mathrm{D}}{\mathrm{D}t}\int_{V_t} f(\boldsymbol{x},t)\,\mathrm{d}x = \int_{V_t}\left(\frac{\partial f}{\partial t}(\boldsymbol{x},t) + \nabla\cdot(f(\boldsymbol{x},t)\boldsymbol{u}(\boldsymbol{x},t))\right)\mathrm{d}x. \tag{2.5}$$

A proof can be found e.g. in [130].

This theorem will be used repeatedly in the following derivation of the governing equations for fluid flow from the fundamental laws of physics: conservation of mass, momentum, and angular momentum. In each case, we state the conservation law for a small arbitrary volume $V_t = \boldsymbol{\varphi}(V_0,t)$, where $V_0 \subset \Omega_0$ is a fixed volume of fluid. Theorem 1 is then applied to move the time derivative under the integral sign, and, possibly after some further manipulations, an integral equation is obtained. Since V_t is arbitrary, by the fundamental lemma of the calculus of variations, the corresponding equation for the integrand will also hold point-wise, which yields the desired local equation.

For easier handling of the tensor quantities, we use abstract index notation: the number of indices of an entity is equal to its tensor order. Hence, a vector $\boldsymbol{v}$ is denoted v_i, a tensor τ as τ_{ij}. Furthermore the Einstein summation convention is followed, meaning that an implicit sum is to be performed over repeated indices. We also introduce the symbol $\otimes$ for the dyadic tensor product of two vectors

$$(\boldsymbol{v}\otimes\boldsymbol{w})_{ij} = v_i w_j, \tag{2.6}$$

and

$$(\nabla \cdot \tau)_i = \frac{\partial \tau_{ij}}{\partial x_j}, \tag{2.7}$$

for the divergence of a second-order tensor.

We start with the principle of mass conservation. The total mass contained in the volume V_t is given by

$$M(V_t, t) = \int_{V_t} \rho(\boldsymbol{x}, t)\, \mathrm{d}x, \tag{2.8}$$

where ρ is the mass density of the fluid. Since V_t always contains the same materia, its mass is conserved, which is expressed using the material derivative as

$$\frac{\mathrm{D}M}{\mathrm{D}t} = 0. \tag{2.9}$$

Applying Theorem 1 gives

$$\frac{\mathrm{D}M}{\mathrm{D}t} = \int_{V_t} \left(\frac{\partial \rho}{\partial t} + \nabla \cdot (\rho \boldsymbol{u}) \right) \mathrm{d}x = 0. \tag{2.10}$$

Since V_t is chosen arbitrarily, the integrand has to vanish point-wise, which results in a local conservation law described through a partial differential equation (PDE), known as the *continuity equation*:

$$\frac{\partial \rho}{\partial t} + \nabla \cdot (\rho \boldsymbol{u}) = 0. \tag{2.11}$$

The second physical principle is the conservation of momentum, also known as Newton's second law. Again, we consider the spatial subset V_t corresponding to a fixed material volume V_0. Let $\boldsymbol{p}$ be the momentum of V_t, given by

$$\boldsymbol{p}(t) = \int_{V_t} \rho\left(\boldsymbol{x}, t\right) \boldsymbol{u}\left(\boldsymbol{x}, t\right)\, \mathrm{d}x. \tag{2.12}$$

We consider next the forces acting on V_t. These can be divided into forces acting on the volume and forces acting on the surface. The former include forces acting at a distance, such as gravity. When considering non-inertial reference frames, i.e. in meteorological applications, this term can also include "fictitious" forces, such as the Coriolis force. The sum of these forces are represented with the force density $\boldsymbol{f}$. The surface forces are the result of direct contact between the volume and its surroundings, and can be represented using the *stress tensor* σ. The total force acting on V_t is given by

$$\boldsymbol{F} = \int_{V_t} \boldsymbol{f}\left(\boldsymbol{x}, t\right) \rho\left(\boldsymbol{x}, t\right)\, \mathrm{d}x + \int_{\partial V_t} \sigma\left(\boldsymbol{x}, t\right) \cdot \boldsymbol{n}_S\left(\boldsymbol{x}, t\right)\, \mathrm{d}s \tag{2.13}$$

$$= \int_{V_t} \left(\boldsymbol{f}\left(\boldsymbol{x}, t\right) \rho\left(\boldsymbol{x}, t\right) + \nabla \cdot \sigma\left(\boldsymbol{x}, t\right) \right) \mathrm{d}x, \tag{2.14}$$

where ∂V_t is the closed boundary of V_t and $\boldsymbol{n}_S$ is the outward-pointing normal to this boundary. The second equality results from applying the divergence theorem to the surface integral in the first equality.

The conservation of momentum for V_t can be expressed as

$$\frac{\mathrm{D}\boldsymbol{p}}{\mathrm{D}t} = \boldsymbol{F}, \tag{2.15}$$

which, by application of Theorem 1, yields

$$\int_{V_t} \left(\frac{\partial(\rho \boldsymbol{u})}{\partial t} + \nabla \cdot (\rho \boldsymbol{u} \otimes \boldsymbol{u}) \right) \mathrm{d}x = \int_{V_t} (\rho \boldsymbol{f} + \nabla \cdot \sigma) \, \mathrm{d}x. \tag{2.16}$$

In differential form, the following equation is obtained:

$$\frac{\partial(\rho \boldsymbol{u})}{\partial t} + \nabla \cdot (\rho \boldsymbol{u} \otimes \boldsymbol{u}) = \rho \boldsymbol{f} + \nabla \cdot \sigma. \tag{2.17}$$

By the product rule, $\nabla \cdot (\rho \boldsymbol{u} \otimes \boldsymbol{u}) = \boldsymbol{u} \nabla \cdot (\rho \boldsymbol{u}) + \rho \, (\boldsymbol{u} \cdot \nabla) \, \boldsymbol{u}$. Substituting the continuity equation (2.11) into this expression, we can reduce (2.17) to the *non-conservative* form

$$\rho \frac{\partial \boldsymbol{u}}{\partial t} + \rho \, (\boldsymbol{u} \cdot \nabla) \, \boldsymbol{u} = \rho \boldsymbol{f} + \nabla \cdot \sigma. \tag{2.18}$$

As a direct consequence of a third physical law, *conservation of angular momentum*, it can be shown that the stress tensor σ is symmetric. Again we will consider the physical volume V_t that corresponds to an arbitrary material volume V_0. For simplicity, only the three-dimensional case is considered here, but the situation in two dimensions is analogous. The angular momentum $\boldsymbol{L}(t)$ of $V_t \subset \mathbb{R}^3$ is given by

$$\boldsymbol{L}(t) = \int_{V_t} \boldsymbol{x} \times (\rho(\boldsymbol{x}, t) \, \boldsymbol{u}(\boldsymbol{x}, t)) \, \mathrm{d}x. \tag{2.19}$$

The conservation of angular momentum can be expressed by the balance equation

$$\frac{\mathrm{D}\boldsymbol{L}}{\mathrm{D}t} = \int_{V_t} \boldsymbol{x} \times (\rho \boldsymbol{f}) \, \mathrm{d}x + \int_{\partial V_t} \boldsymbol{x} \times (\boldsymbol{n} \cdot \sigma) \, \mathrm{d}s, \tag{2.20}$$

where the integral on the right-hand side represents the total moment of the volume and surface forces acting on V_t. The surface integral can be converted to a volume integral using the divergence theorem for tensors. We make use of the Levi-Civita tensor ϵ_{ijk} to express the cross product as $\boldsymbol{v} \times \boldsymbol{w} = \epsilon_{ijk} v_j w_k \boldsymbol{e}_i$ where $\boldsymbol{e}_i$ denotes the canonical basis

vectors in $\mathbb{R}^d$. Moreover, we denote by $\epsilon : \tau$ the tensor contraction $\sum_{j,k=1}^{3} \epsilon_{ijk}\tau_{jk}$. For $i = 1, 2, 3$, we have

$$\begin{aligned} \boldsymbol{e}_i \cdot \int_{\partial V_t} \boldsymbol{x} \times (\boldsymbol{n} \cdot \sigma) \, \mathrm{d}s &= \int_{\partial V_t} \epsilon_{ijk} \boldsymbol{x}_j \boldsymbol{n}_l \sigma_{lk} \, \mathrm{d}x \\ &= \int_{V_t} \partial_l \left(\epsilon_{ijk} \boldsymbol{x}_j \sigma_{lk} \right) \, \mathrm{d}x \\ &= \int_{V_t} \epsilon_{ijk} \left(\boldsymbol{x}_j \partial_l \sigma_{lk} + \sigma_{jk} \right) \, \mathrm{d}x \\ &= \int_{V_t} \left(\boldsymbol{x} \times \nabla \cdot \sigma + \epsilon : \sigma \right) \, \mathrm{d}x, \end{aligned} \tag{2.21}$$

where the identity $\partial_l \boldsymbol{x}_j = \delta_{lj}$ (δ_{lj} denotes the Kronecker delta) is used to reduce over the indices of σ in the second term. Considering now the left-hand side of Equation (2.20), Theorem 1 leads to

$$\frac{\mathrm{D}\boldsymbol{L}}{\mathrm{D}t} = \int_{V_t} \frac{\partial}{\partial t} \left(\boldsymbol{x} \times (\rho \boldsymbol{u}) \right) + \nabla \cdot \left((\boldsymbol{x} \times \rho \boldsymbol{u}) \otimes \boldsymbol{u} \right) \, \mathrm{d}x \tag{2.22}$$

$$= \int_{V_t} \boldsymbol{x} \times \left(\rho \frac{\partial \boldsymbol{u}}{\partial t} + \boldsymbol{u} \frac{\partial \rho}{\partial t} + \left(\nabla \cdot (\rho \boldsymbol{u}) \, \boldsymbol{u} \right) + \rho \boldsymbol{u} \cdot \nabla \boldsymbol{u} \right) \, \mathrm{d}x. \tag{2.23}$$

Using Equations (2.11) and (2.18), $\frac{\partial \rho}{\partial t}$ and $\rho \frac{\partial \boldsymbol{u}}{\partial t}$ can be substituted to cancel out all terms containing the velocity:

$$\frac{\mathrm{d}\boldsymbol{L}}{\mathrm{d}t} = \int_{V_t} \boldsymbol{x} \times (\rho \boldsymbol{f} + \nabla \cdot \sigma) \, \mathrm{d}x. \tag{2.24}$$

Comparing this left-hand side with Equation (2.20) and using the volume form obtained in Equation (2.21) reduces the balance of angular momentum to a simple condition on the stress tensor σ:

$$\int_{V_t} (\epsilon : \sigma) \, \mathrm{d}x = 0. \tag{2.25}$$

Since V_t is arbitrary, the integrand must vanish at each point in the domain Ω_t, and hence $\epsilon : \sigma(\boldsymbol{x}, t) = 0$ for all $(\boldsymbol{x}, t) \in \Omega_t \times [0, \infty)$. Expanding this condition gives the requirement $\sigma_{jk} = \sigma_{kj}$ for $j \neq k$, which implies that the tensor is symmetric.

The connection between fluid mechanics and thermodynamics is made by the last fundamental conservation law, conservation of energy. Since this work will only consider isothermal flows, where the temperature of the fluid is assumed to be constant, the energy balance equation will be decoupled from the mechanical balance equations. Since it will not be needed in the rest of the work, its derivation is omitted.

At this point, the variables ρ, $\boldsymbol{u}$, and σ have been introduced, and the following equations have been derived:

$$\frac{\partial \rho}{\partial t} + \nabla \cdot (\rho \boldsymbol{u}) = 0 \qquad \text{in } \Omega, \tag{2.26}$$

$$\rho \frac{\partial \boldsymbol{u}}{\partial t} + \rho \left(\boldsymbol{u} \cdot \nabla \right) \boldsymbol{u} = \rho \boldsymbol{f} + \nabla \cdot \sigma \qquad \text{in } \Omega, \tag{2.27}$$

$$\sigma = \sigma^T \qquad \text{in } \Omega. \tag{2.28}$$

where $\Omega \subset \mathbb{R}^d$, $d = 2, 3$. The system is thus under-determined, and more equations are necessary in order to be able to specify a unique solution. These equations are provided by a *constitutive relation* which describes the material properties of the fluid under study. More specifically, the constitutive relation is an experimentally determined equation that specifies how the stress tensor σ varies as a function of the other variables.

Since there are many different kinds of fluids, one cannot expect the same relation to hold for all of them. Several different classifications of fluids according to their constitutive relation have been developed. The most commonly used model is that of the *Newtonian fluid*, in which the stress tensor is given as a function of the velocity $\boldsymbol{u}$ and pressure p by a linear relation known as the *Cauchy-Poisson law*:

$$\sigma = -(p + \lambda \nabla \cdot \boldsymbol{u})\mathcal{I} + 2\mu D(\boldsymbol{u}), \tag{2.29}$$

with the *rate of deformation tensor*

$$D(\boldsymbol{u}) = \frac{1}{2}\left(\nabla \boldsymbol{u} + \nabla \boldsymbol{u}^T\right). \tag{2.30}$$

The material parameter μ is called *dynamic viscosity*, and λ is the *bulk viscosity*. In general, it is a tensor quantity that varies in space, but for the case of an homogeneous, isotropic medium it can be represented with a scalar constant. The fluid is *viscous* if $\mu > 0$ and *inviscid* if $\mu = 0$. There are also models for non-Newtonian fluids, where μ may depend on e.g. the shear rate (*shear thinning* and *shear thickening* fluids), the history of the material (*thixotropic* and *rheopectic* fluids), and external electric or magnetic fields (*electrorheological* and *magnetorheological* fluids). Additionally, many fluids show other types of non-linear behavior. A good reference for the theory of non-Newtonian fluids is [26].

In this work, only viscous Newtonian fluids will be considered. Furthermore, only liquid phases will be treated, and the flow will be assumed to be *incompressible*, meaning that the volume of each fluid element V_0 remains constant in time. Applying Theorem 1 to the physical volume element $V_t = \boldsymbol{\varphi}(V_0, t)$, we obtain

$$\frac{\mathrm{D}V}{\mathrm{D}t} = \frac{\mathrm{D}}{\mathrm{D}t}\int_{V_t} 1 \,\mathrm{d}x = \int_{V_t} \nabla \cdot u \,\mathrm{d}x. \tag{2.31}$$

Taken locally, the *incompressibility condition* $\nabla \cdot u = 0$ is thus derived. For a *homogeneous* fluid, in which the density satisfies $\nabla \rho = 0$, the continuity equation reduces to the trivial equation $\frac{\partial \rho}{\partial t} = 0$ for incompressible flows. The density is then constant both in time and in space. Furthermore the constitutive equation (2.29) reduces to

$$\sigma = -p\mathcal{I} + 2\mu D(\boldsymbol{u}). \tag{2.32}$$

Combining Equations (2.18), (2.31), and (2.32), gives the *incompressible Navier-Stokes equations*:

$$\rho\left(\frac{\partial \boldsymbol{u}}{\partial t} + \boldsymbol{u}\cdot\nabla\boldsymbol{u}\right) = \rho\boldsymbol{f} + 2\nabla\cdot(\mu D(\boldsymbol{u})) - \nabla p \qquad \text{in } \Omega, \tag{2.33}$$

$$\nabla\cdot\boldsymbol{u} = 0 \qquad \text{in } \Omega. \tag{2.34}$$

In the context of incompressible flow, the mass density ρ and viscosity μ are normally considered to be known material parameters. The unknown variables that remain are the velocity $\boldsymbol{u}$ and pressure p.

In order to formulate a well-posed problem for the Navier-Stokes equations on a bounded domain Ω, they must be completed with appropriate boundary conditions. The boundary conditions to be chosen depends on the physical situation. If Ω is bounded by a solid wall, for instance, it is often natural to assume that the fluid will not move relative to the wall. If the wall itself does not move, this results in the so-called *no-slip condition* $\boldsymbol{u} = 0$. More generally, a *slip condition* is given by $\boldsymbol{u} = \boldsymbol{u}_d$, where $\boldsymbol{u}_d$ is a given velocity field defined on the boundary. For problems involving channel flows, it is often natural to prescribe pressure differences between the inflow and outflow boundaries, which is closely related to the specification mean fluxes over these boundaries. This topic is discussed in detail in [68]. In the context of finite element discretizations, it is common to use the *natural* boundary conditions associated with the weak formulation at outflow boundaries. For some choices of the weak formulation, this *do-nothing* condition does not have a direct physical interpretation, whereas for other choices, this corresponds to a specification of the stress on the boundary.

2.2 TWO-COMPONENT FLOW MODEL

Having established a model describing instationary incompressible flow for a single fluid component, we now turn to the question of how to model flows with several components. We shall limit ourselves to the case of two components, since this corresponds to the configurations encountered in the rest of the thesis, but a similar approach could also be used for flows involving three or more components.

The model described here is based on that used in [54] for two-component flows without mass transport or surfactants. It is assumed that the fluids are incompressible, viscous, and immiscible; and that each fluid component is a homogeneous composition, with constant dynamic viscosity and density. Furthermore, it is assumed that no phase transitions occur.

We shall treat the interface as a perfectly sharp dividing surface, at which the density and viscosity of the fluids are taken to be discontinuous. The interaction over the surface is modeled by a interfacial tension force acting on the interface, which arises due to the differ-

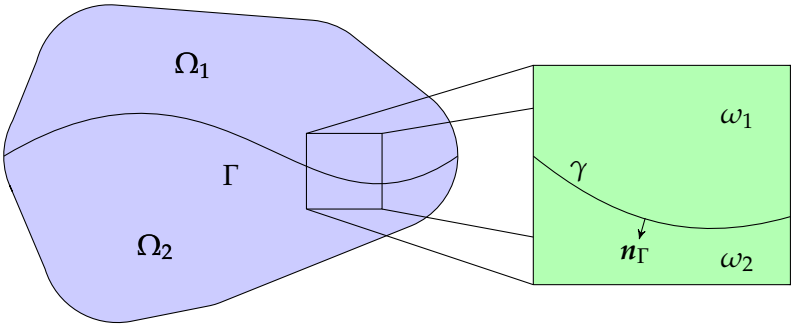


Figure 2.2: Fluid with two components.

ence in inter-molecular forces between the components. This model was originally developed by Gibbs [82, 120] in the second half of the 19th century.

It is well established that even for immiscible fluids, the idea of a sharp interface with discontinuous material parameters is an idealization. In reality, there will be a thin interface layer between the fluid components, with a smooth (but steep) transition of the viscosity and density. This was discussed already by Lord Rayleigh and J. van der Waals at the end of the 19th century [104, 139].

It would be possible to work with such a *diffuse interface* model directly, using an *order parameter* to describe the degree of mixing between the fluid components in the interface layer. The evolution of the order parameter can be derived from thermodynamic considerations and is described by a modified version of the Cahn-Hilliard equation. Coupling this equation with the Navier-Stokes model for fluid flow yields a *phase field* model, which can be used to simulate the interaction of the fluids in the overlapping region. A comprehensive derivation of such a model is presented in [60]. An overview of the development of diffuse-interface models and their applications can be found in [5].

The use of a phase field model is appropriate in situations where the interfacial width is comparable in size to the typical length scale of the flow. The former is typically on the scale of hundreds of nanometers, which is much smaller than the scales encountered in the present work. From a numerical point of view, the phase field model has the advantage of only dealing with smooth quantities, thus avoiding the problems associated with the discontinuous material parameters. On the other hand the Navier-Stokes/Cahn-Hilliard equations contain a fourth-order operator, whose numerical treatment is difficult due to the large condition numbers of the discretized operators. Furthermore, the fact that the width of the transition region is very small leads to large gradients which must be resolved numerically. The relation between the diffuse interface and sharp interface approaches are explored in detail in [120]. In summary, a sharp interface model can be interpreted as a diffuse interface model in which

the unknown variation of the variables in the interface layer is represented by corresponding *excess* quantities that are associated with the interface itself. For instance, the fact that the mass density ρ will not be constant in the interface layer is modeled by introducing a *surface mass density* ρ_s defined on the dividing surface. The bulk quantity ρ is then assumed to be piecewise constant.

The introduction of excess surface quantities impacts the governing equations. To continue with the example of mass, the total mass at time t in a volume V_t that contains a piece γ_t of the interface Γ is then given by

$$M = \int_{V_t} \rho(\boldsymbol{x}, t)\, \mathrm{d}x + \int_{\gamma_t} \rho_s(\boldsymbol{x}, t)\, \mathrm{d}s. \tag{2.35}$$

where the surface integral has been added to the volume integral in Equation (2.8). The conservation of mass would be stated as

$$\frac{\mathrm{d}M}{\mathrm{d}t} = \frac{\mathrm{d}}{\mathrm{d}t} M = \frac{\mathrm{d}}{\mathrm{d}t} \left(\int_{V_t} \rho(\boldsymbol{x}, t)\, \mathrm{d}x + \int_{\gamma_t} \rho_s(\boldsymbol{x}, t)\, \mathrm{d}s \right). \tag{2.36}$$

This expression is expanded in [120], in which a *surface transport theorem* that generalizes Theorem 1 is derived. Since the present work is only concerned with homogeneous fluid components without surfactants, we will follow the approach in [54] and make the assumption that the dividing surface can be chosen in such a way that $\rho_s(\boldsymbol{x}, t) = 0$ in all of Ω and at all times t. This is known as the *clean interface* assumption. Under this assumption, the governing equations derived in Section 2.1 will hold in each component separately, which greatly simplifies the model. For details on more general models, we refer to [120] and [54].

In order to derive a two-component sharp interface model, we need to first introduce some notations. To keep track of the space occupied at time t by the two components, the domain Ω at time t is partitioned into two open subsets $\Omega_1(t)$ and $\Omega_2(t)$, such that $\Omega_1(t) \cap \Omega_2(t) = \emptyset$ (see Fig. 2.2). By $\Gamma = \overline{\Omega_1(t)} \cap \overline{\Omega_2(t)}$ we denote the interface between the components, and by $\boldsymbol{n}_\Gamma$ a unit normal to Γ pointing from Ω_1 into Ω_2. The two fluid components have separate but constant densities ρ_i and dynamic viscosities μ_i, but are both Newtonian, with stress tensors $\sigma_i = -p\mathcal{I} + \mu_i \left(\nabla \boldsymbol{u} + \nabla \boldsymbol{u}^T \right), i = 1, 2$. Conservation of mass and momentum holds separately for each of the two components, which yields the incompressible Navier-Stokes equations in Ω_i, $i = 1, 2$:

$$\rho_i \left(\frac{\partial \boldsymbol{u}}{\partial t} + (\boldsymbol{u} \cdot \nabla)\, \boldsymbol{u} \right) = \nabla \cdot \sigma_i + \rho_i \boldsymbol{f}_i \qquad \text{in } \Omega_i, \tag{2.37}$$

$$\nabla \cdot \boldsymbol{u} = 0 \qquad \text{in } \Omega_1 \cup \Omega_2. \tag{2.38}$$

The interface Γ will move as the shapes of the two fluid components evolve in time. As we shall see in the following, only the normal

component of the velocity v_Γ of the interface will be significant in the model. Since it has been assumed that the components will not mix, meaning that there will be no mass transfer across the surface, the set of fluid particles that lie on the interface remains constant in time. Since these particles are bound to the surface, the normal component of their velocity $\boldsymbol{u}$ must be equal to the normal component of the velocity of the interface itself:

$$\boldsymbol{v}_\Gamma \cdot \boldsymbol{n}_\Gamma = \boldsymbol{u} \cdot \boldsymbol{n}_\Gamma. \tag{2.39}$$

This relation defines the coupling between on the one hand the flow equations, which follow the fluid velocity $\boldsymbol{u}$, and on the other hand the evolution of the interface, which follows the interface velocity $\boldsymbol{v}_\Gamma$.

Under the clean interface assumption, and assuming that the fluids are viscid, the velocity will be continuous at the interface:

$$[\![\boldsymbol{u}(\boldsymbol{x}, t)]\!]_\Gamma = 0, \text{ for } \boldsymbol{x} \in \Omega, \tag{2.40}$$

where the notation

$$[\![f]\!]_\Gamma = \lim_{\epsilon \to 0^+} \left(f(x - \epsilon \boldsymbol{n}_\Gamma) - f(x + \epsilon \boldsymbol{n}_\Gamma)\right), \tag{2.41}$$

is used to denote the jump of a function over Γ. It should be noted that the continuity of the velocity is not in any way obvious: the pressure p will in general be discontinuous over the interface.

When considering the flow in the entire domain Ω, an additional contact force, known as the *interfacial tension force*, will appear on the interface Γ between the fluid components. This force arises due to the difference of molecular interaction forces on the two sides of the interface, which causes an excess force that is not accounted for by the stress tensor. As explained in [54], the interfacial tension force acting on a small subset $\gamma \subset \Gamma$ of the interface between two fluid volumes $\omega_1 \subset \Omega_1$ and $\omega_2 \subset \Omega_2$ (see Fig. 2.2 for a two-dimensional representation) is defined by

$$\boldsymbol{F}_S = \int_{\partial\gamma} \tau \boldsymbol{\nu} \, \mathrm{d}l, \tag{2.42}$$

where $\boldsymbol{\nu}$ is a unit normal to $\partial\gamma$ which is tangential to γ, and τ is the *interfacial tension coefficient*, which is a material property of the combination of component fluids. Note that the integral is taken over the boundary of the two-dimensional surface γ, and hence the units of τ are those of force per distance. Using a divergence theorem for surface integrals, the following holds:

$$\int_{\partial\gamma} \tau \boldsymbol{\nu} \mathrm{d}l = \int_\gamma \left(\nabla_\Gamma \tau - \tau \kappa \boldsymbol{n}_\Gamma\right) \mathrm{d}s, \tag{2.43}$$

where ∇_Γ denotes the surface gradient, and $\kappa = \nabla \cdot \boldsymbol{n}_\Gamma$ the mean curvature. In this work, only situations in which τ is constant will be

considered, which implies that $\nabla_\Gamma \tau = 0$. If surfactants were present, this would in general not be true.

Let us now derive the flow model for the two-component fluid system. Consider, at a given point in time, a volume element $\omega = \omega_1 \cup \omega_2$, again with $\omega_1 \subset \Omega_1$ and $\omega_2 \subset \Omega_2$. As in the previous section, the conservation of momentum for ω is given by

$$\int_\omega \rho \left(\frac{\partial \boldsymbol{u}}{\partial t} + (\boldsymbol{u} \cdot \nabla)\, \boldsymbol{u} \right) \mathrm{d}x = \int_\omega \rho \boldsymbol{f} \,\mathrm{d}x + \int_{\partial\omega} \sigma \cdot \boldsymbol{n} \,\mathrm{d}s - \int_\gamma \tau\kappa \boldsymbol{n}_\Gamma \,\mathrm{d}s. \tag{2.44}$$

Since ρ_i and σ_i are discontinuous over γ, we split the integration according to the sub-domains, to obtain

$$\begin{aligned} \int_\omega \rho \left(\frac{\partial \boldsymbol{u}}{\partial t} + (\boldsymbol{u} \cdot \nabla)\, \boldsymbol{u} - \boldsymbol{f} \right) \mathrm{d}x &= \sum_{i=1}^{2} \int_{\omega_i} \rho_i \left(\frac{\partial \boldsymbol{u}}{\partial t} + (\boldsymbol{u} \cdot \nabla)\, \boldsymbol{u} - \boldsymbol{f}_i \right) \mathrm{d}x, \\ \int_{\partial\omega} \sigma \cdot \boldsymbol{n} \,\mathrm{d}s &= \sum_{i=1}^{2} \int_{\partial\omega_i} \sigma_i \cdot \boldsymbol{n}_i \,\mathrm{d}s - \int_\gamma [\![\sigma]\!]_\Gamma \boldsymbol{n}_\Gamma \,\mathrm{d}s \\ &= \sum_{i=1}^{2} \int_{\omega_i} (\nabla \cdot \sigma_i) \,\mathrm{d}x - \int_\gamma [\![\sigma]\!]_\Gamma \boldsymbol{n}_\Gamma \,\mathrm{d}s. \end{aligned} \tag{2.45}$$

Substitution into Equation (2.44) yields

$$\sum_{i=1}^{2} \int_{\omega_i} \rho_i \left(\frac{\partial \boldsymbol{u}}{\partial t} + (\boldsymbol{u} \cdot \nabla)\, \boldsymbol{u} - \boldsymbol{f}_i \right) - \nabla \cdot \sigma_i \,\mathrm{d}x = - \int_\gamma \tau\kappa \boldsymbol{n}_\Gamma + [\![\sigma]\!]_\Gamma \boldsymbol{n}_\Gamma \,\mathrm{d}s. \tag{2.46}$$

According to the momentum balance equation for each component taken separately (see Equation (2.37)), each term on the left equals zero separately. Since the volumes $\omega_i \subset \Omega_i$ and hence $\gamma \subset \Gamma$ were chosen arbitrarily, we obtain the following interface condition for the stress tensor point-wise on Γ:

$$[\![\sigma]\!]_\Gamma \boldsymbol{n}_\Gamma = -\tau\kappa \boldsymbol{n}_\Gamma. \tag{2.47}$$

To complete the model, the system of PDEs needs to be extended with appropriate initial and boundary conditions. As initial condition, we simply specify the velocity at $t = 0$ with a given function $\boldsymbol{u}^0$: $\boldsymbol{u}(\boldsymbol{x}, 0) = \boldsymbol{u}^0(\boldsymbol{x})$. At the boundaries, two types of conditions will be considered. On the subset $\partial\Omega_D$, a Dirichlet condition represented by the function $\boldsymbol{u}_D(\boldsymbol{x}, t)$ for the velocity will be imposed. On the rest of the domain, $\partial\Omega_N = \partial\Omega \setminus \partial\Omega_D$, we will impose a homogeneous condition for the normal stress: $\sigma \boldsymbol{n} = 0$. This corresponds to fixing the external pressure to be zero, and will be used to model the outflow of channels. This condition corresponds to the natural boundary condition of the weak formulation presented in Section 4.4.

In summary, the following model with a PDE system in each subdomain together with a set of boundary, initial, and interface conditions has been derived:

$$\rho_i \left(\frac{\partial \boldsymbol{u}}{\partial t} + (\boldsymbol{u} \cdot \nabla) \boldsymbol{u} \right) = \rho_i \boldsymbol{f}_i + \nabla \cdot \sigma_i \qquad \text{in } \Omega_i, \tag{2.48}$$

$$\nabla \cdot \boldsymbol{u} = 0 \qquad \text{in } \Omega_i, \tag{2.49}$$

$$\boldsymbol{u} = \boldsymbol{u}_D \qquad \text{on } \partial\Omega_D, \tag{2.50}$$

$$\sigma \boldsymbol{n} = 0 \qquad \text{on } \partial\Omega_N, \tag{2.51}$$

$$\boldsymbol{u}(\boldsymbol{x}, 0) = \boldsymbol{u}^0 \qquad \text{in } \Omega, \tag{2.52}$$

$$[\![\boldsymbol{u}]\!]_\Gamma = 0 \qquad \text{on } \Gamma, \tag{2.53}$$

$$[\![\sigma]\!]_\Gamma \boldsymbol{n}_\Gamma = -\tau\kappa \boldsymbol{n}_\Gamma \qquad \text{on } \Gamma, \tag{2.54}$$

$$\boldsymbol{v}_\Gamma \cdot \boldsymbol{n}_\Gamma = \boldsymbol{u} \cdot \boldsymbol{n}_\Gamma \qquad \text{on } \Gamma. \tag{2.55}$$

3

INTERFACE MODEL

This chapter presents a model based on the Level Set method for simulating the evolution of the interface in the flow model presented in Chapter 2. The aim is to reformulate certain aspects of the problem stated in Equations (2.48) – (2.55) to the standard form of a time-dependent boundary value problem, which can then be discretized and solved. First, in Section 3.1, the scope of the interface model is discussed, with a description of the various parts of the flow model that it refines. Next, Section 3.2 motivates the choice of the Level Set method with a comparison of the advantages and disadvantages of different approaches existing in the literature. Finally, in Section 3.3, we describe the Level Set method, and how it can be applied to transform the flow model to the desired form.

3.1 REQUIREMENTS FOR THE INTERFACE MODEL

The fluid flow model described in Equations (2.48) – (2.55) is physically and mathematically plausible, but has the drawback of being posed on two separate domains, with a set of jump conditions on the interface. One could imagine treating the problem using a domain decomposition approach, but the fact that the interface is moving, and the difficulties related to incorporating the interfacial tension condition in Equation (2.54), makes this option less attractive. Instead, we want to consider the problem on the whole domain $\Omega = \Omega_1 \cup \Omega_2$, and would thus like to avoid treating the interface conditions (2.53) – (2.55) explicitly. The first of these conditions, the continuity of the velocity (2.53), is automatically satisfied through the use of such a single-domain model, if it is assumed that $\boldsymbol{u}$ is continuous inside Ω. The other interface conditions will require more specialized treatment, which will be described in the following.

Regardless of whether a single-domain or multi-domain approach is used, it is necessary to follow the evolution of the interface $\Gamma(t)$. This will require the proposed model to include a mechanism for updating the interface as time progresses, using the condition (2.55).

It is straightforward to pose the Navier-Stokes equations (2.48) – (2.49) on the whole domain Ω, if one introduces global functions $\rho(\boldsymbol{x}, t)$ and $\mu(\boldsymbol{x}, t)$ for the density and viscosity parameters, respec-

tively. These functions are piecewise constant, with different values for each of the two fluid components:

$$\rho|_{\Omega_i(t)} = \rho_i, \tag{3.1}$$

$$\mu|_{\Omega_i(t)} = \mu_i. \tag{3.2}$$

To evaluate $\rho(\boldsymbol{x}, t)$ and $\mu(\boldsymbol{x}, t)$, it will be necessary to determine to which fluid a given point $\boldsymbol{x}$ belongs at time t. How this is accomplished depends on the representation chosen for the interface, which will be discussed in the next section.

Finally, the jump condition (2.54) for the normal stress corresponds to a interfacial tension force localized on the interface. In the weak formulation of the single-domain model, that will be presented in Chapter 4, it is easier to work with this force directly, instead of a jump. A third requirement on the interface representation is therefore that it must provide a way to describe functions that are localized to the interface.

In summary, the use of a single-domain model will require an interface representation that facilitates the following things:

- following the evolution of the interface in time;
- finding the fluid component to which any point in the domain belongs; and
- representing functions that are localized to the interface.

The next section will discuss what approaches have been explored by other authors, and motivate why the Level Set method is appropriate for the problem at hand.

3.2 COMPARISON OF INTERFACE REPRESENTATIONS

Much work has been devoted to the problem of constructing numerical representations of interfaces. The different approaches are usually classified into *front-tracking* and *front-capturing* methods, depending on whether the interface is represented explicitly with for instance a mesh, or implicitly using e.g. a set of particles or a parameterized function. The front-tracking methods are based on Lagrangian coordinates that evolve in time, whereas the front-capturing methods are based on a fixed Eulerian coordinate system.

3.2.1 *Front-tracking Methods*

Among the front-tracking methods, there are several alternatives for the choice of the interface representation. A good overview of the different possibilities is provided in [132]. Here we will describe the basic scheme, in which the front is represented with a surface mesh

containing a set of vertices at positions $\boldsymbol{x}_i$. The mesh is evolved in time by updating the vertex positions according to the kinematic law

$$\frac{\mathrm{d}\boldsymbol{x}_i}{\mathrm{d}t} = \boldsymbol{v}_\Gamma\left(\boldsymbol{x}_i, t\right), \tag{3.3}$$

using for instance the forward Euler scheme:

$$\boldsymbol{x}_i\left(t_{n+1}\right) = \boldsymbol{x}_i\left(t_n\right) + \boldsymbol{v}_\Gamma\left(\boldsymbol{x}_i\left(t_n\right), t_n\right)\left(t_{n+1} - t_n\right). \tag{3.4}$$

The front-tracking method was developed to a large extent by Glimm and his coworkers (see the references in [53]). Among other contributions, they apply this technique to the simulation of shock waves in 3D in [53] and the Rayleigh-Taylor instability in [52]. An application of the front-tracking method for simulating bubbles in incompressible, inviscid flow using the front-tracking method is described in [133].

There are some major drawbacks of the front-tracking method, that arise as a direct consequence of using an explicit representation. As the mesh that represents the interface evolves in time, the spatial distribution of the vertices will often become uneven, so that *mesh restructuring* becomes necessary to maintain an accurate representation of the front. The restructuring can take the form of refinement and coarsening, or modification of the vertex positions, or both. These operations tend to require highly dynamic data structures and are generally very complex, especially in three dimensions. A further difficulty arises when the topology of the interface changes, such as in simulations where bubbles merge or split. Recently, the wish to handle topology changes has lead to the development of hybrid methods which combine front-tracking with aspects of front-capturing, as described e.g in [119].

The main advantages of front-tracking are the possibility to track the interface very accurately, and the fact that mass is naturally conserved as a result of using a Lagrangian representation.

3.2.2 *Front-capturing Methods*

There are several types of front-capturing methods, among which the most notable are the *Marker-and-Cell* (MAC), the *Volume-of-Fluid* (VOF), and the *Level Set* methods.

The Marker-and-Cell method was first developed in the 1960s, but is still widely used today. The original method, described in [61], uses marker particles that are transported along with the fluid. The interface can then be reconstructed by testing in which cells of a fixed Cartesian grid the particles are located. The original method is limited to two-dimensional, rectangular domains, and incompressible flows. Since then, it has been extended to eliminate these restrictions,

producing variations that can handle curved boundaries, compressible flows and three-dimensional problems. For a recent review of the literature in this area, see [85].

The Volume-of-Fluid method was made popular through [69]. That article, which was written with the problem of free surfaces in mind, uses a volume-fraction function F that assigns to each cell the fraction of it that is filled with fluid. The same method can be applied to two-component flows by picking one of the components to be described by F, and the other by $1 - F$.

The evolution of F is governed by the partial differential equation

$$\frac{\partial F(\boldsymbol{x}, t)}{\partial t} + \boldsymbol{v}_\Gamma(\boldsymbol{x}, t) \cdot \nabla F(\boldsymbol{x}, t) = 0. \tag{3.5}$$

This PDE can be solved using a variety of techniques, including finite difference and finite volume methods. A strong emphasis is placed on being able to impose the conservation of the quantity of each fluid at the cell level by assuring that what leaves one cell will enter another.

The VOF method addresses the memory and run-time costs associated with tracking the positions of a large number of marker particles in the MAC method by replacing this data with F, which can be interpreted as a density of marker particles. Apart from this, the MAC and VOF methods are closely related, and mostly share the same strengths and weaknesses.

In contrast to the front-tracking method, the MAC and VOF methods only give an indirect representation of the interface Γ. To be able to evaluate the material parameters and the geometrical properties of the interface, an approximation of Γ must be reconstructed from the data at hand. For VOF, different methods have been devised for this purpose, the most common being *Simple Line Interface Calculation* (see [92]) and *Piecewise Linear Interface Construction* (see [106] and references therein).

The main advantage of the indirect representation is the ability of dealing transparently with topological changes. Fundamentally, this is due to the fact that the "fluid markers", represented explicitly or implicitly in the MAC and VOF methods respectively, do not have to be ordered, whereas the vertices of the interface mesh in the front-tracking method have to respect the ordering imposed by the mesh topology.

The VOF method can also be viewed as tracking of the characteristic function for Ω_1, χ_{Ω_1}, as described in [54]. The volume-fraction function F at a point $\boldsymbol{x}$, be expressed in terms of χ_{Ω_1} via an integral over an arbitrary, small neighborhood $U(\boldsymbol{x})$ around $\boldsymbol{x}$:

$$F(\boldsymbol{x}) = |U(\boldsymbol{x})|^{-1} \int_{U(\boldsymbol{x})} \chi_{\Omega_1} \mathrm{d}x. \tag{3.6}$$

Through substitution into Equation (3.5) we have

$$|U|^{-1}\left(\frac{\partial}{\partial t}\int_U \chi_{\Omega_1}\mathrm{d}x + \boldsymbol{v}_\Gamma \cdot \nabla \int_U \chi_{\Omega_1}\mathrm{d}x\right) = 0, \tag{3.7}$$

and formally exchanging the order of the derivation and integration operations yields

$$\int_U \left(\frac{\partial \chi_{\Omega_1}}{\partial t} + \boldsymbol{v}_\Gamma \cdot \nabla \chi_{\Omega_1}\right)\mathrm{d}x = 0. \tag{3.8}$$

This integral equation is formally equivalent to the following differential advection equation for χ_{Ω_1}:

$$\frac{\partial \chi_{\Omega_1}}{\partial t} + \boldsymbol{v}_\Gamma \cdot \nabla \chi_{\Omega_1} = 0, \tag{3.9}$$

which, however, is not well-defined everywhere since χ_{Ω_1} is discontinuous.

The Level Set method, which was pioneered in [101], uses a different indicator function $\phi(\boldsymbol{x}, t)$ to represent the interface implicitly, but evolves it with the same advection equation:

$$\frac{\partial \phi(\boldsymbol{x}, t)}{\partial t} + \boldsymbol{v}_\Gamma(\boldsymbol{x}, t) \cdot \nabla \phi(\boldsymbol{x}, t) = 0. \tag{3.10}$$

In contrast to χ_{Ω_1} in the VOF method, ϕ is chosen to be at least continuous, and possibly smoother. Furthermore, it is constructed in such a way that the interface $\Gamma(t)$ can be characterized as a level set of ϕ. The level set function can be chosen in several ways, but the most popular method is to let $\phi(\cdot, t)$ be an approximate signed distance function with respect to $\Gamma(t)$:

$$\begin{aligned} |\phi(\boldsymbol{x}, t)| &= \min_{\boldsymbol{y}\in\Gamma(t)} |\boldsymbol{x} - \boldsymbol{y}| && \text{in } \Omega, \\ \phi(\boldsymbol{x}, t) &< 0 && \text{in } \Omega_1, \\ \phi(\boldsymbol{x}, t) &> 0 && \text{in } \Omega_2. \end{aligned} \tag{3.11}$$

With this representation, $\Gamma(t)$ is the zeroth level set of ϕ:

$$\Gamma(t) = \{\boldsymbol{x} \in \Omega : \phi(\boldsymbol{x}, t) = 0\}. \tag{3.12}$$

The fact that $\Gamma(t)$ can be characterized directly in terms of the indicator function is another difference with the VOF method, where $\Gamma(t)$ is instead the boundary of the support of χ_{Ω_1}.

This direct characterization of $\Gamma(t)$ in terms of the level set function is a distinct advantage over the MAC and VOF methods. Like these other Eulerian methods, the Level Set method can also handle topological changes without special measures. Another advantage is the

possibility of directly extracting geometrical information from ϕ without needing to explicitly reconstruct the interface. The unit normal at each point of $\Gamma(t)$ is given by

$$\boldsymbol{n}_\Gamma = \frac{\nabla\phi}{|\nabla\phi|}, \tag{3.13}$$

and the mean curvature of $\Gamma(t)$ by

$$\kappa = \nabla\cdot\boldsymbol{n}_\Gamma = \nabla\cdot\left(\frac{\nabla\phi}{|\nabla\phi|}\right). \tag{3.14}$$

This ability of directly computing the gradient and curvature of the interface, which are needed for the evaluation of the interfacial tension force, is not found in any of the other methods described in this chapter.

The main drawback of the Level Set method is that, depending on the choice of numerical method for solving Equation (3.10), the mass of the different components will in general not be conserved. This issue is addressed for instance in [95, 96].

An early and very influential application of the Level Set method to incompressible fluid flow is described in [127]. Closely related to two-component flows is the simulation of free surfaces which was explored in [20].

The Level Set method has a much wider range of applications than two-component or free boundary flow simulations. Examples include image processing, surface evolution, mesh generation and computer vision. Good overviews can be found in the books [98, 100, 118] and the review article [99]. In contrast to the case with two-component flows, the evolution of the level set in these applications is of governed by the some aspect geometry of the interface itself, rather than an externally imposed velocity. The most common model involves *mean curvature flow*, which is related to the solution of Stefan problems in the modeling of phase transitions in matter. This has applications e. g. in the simulation of dendritic growth [50]. The Level Set representation also lends itself to shape optimization, as is shown in [105], which treats the optimization of band-gaps in photonic crystals.

3.3 INTERFACE MODEL USING THE LEVEL SET METHOD

On the basis of the analysis presented in Section 3.2, the choice of interface representation for this work has fallen on the Level Set method. We now describe the method used in more detail, and explain how it fulfills the requirements presented in Section 3.1.

As already mentioned, there is a great deal of freedom in how the level set function ϕ is chosen. Osher and Fedkiw argue in [98] the advantages of choosing $\phi(\boldsymbol{x}, t)$ to be a signed distance function, which include its smoothness away from the interface, and its favorable properties when computing derived quantities.

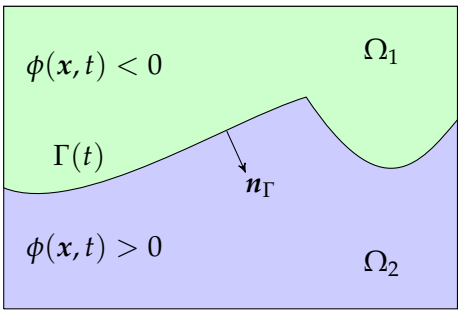


Figure 3.1: The notation used in the two-component model with the level set representation of the interface.

In practice, the signed distance property will not be preserved as the level set function is evolved in time. This is due to the fact that the function is evolved with different speeds at different points in space. Several methods have been proposed to periodically modify the level set function to restore the signed distance property. One popular approach, called *reinitialization*, was first suggested in [127], and includes solving the following PDE to steady state:

$$\frac{\partial \tilde{\phi}}{\partial \tau} + S(\phi)\left(1 - \|\nabla \tilde{\phi}\|\right) = 0, \tag{3.15}$$

where $S(\phi)$ is a smoothened sign function evaluated for the original level set function, and τ is a variable corresponding to a pseudo-time that is only used to reach the stationary state, where $\frac{\partial \tilde{\phi}}{\partial \tau} = 0$, and hence $\|\nabla \tilde{\phi}\| = 1$. The modified level set function $\tilde{\phi}$ can then replace ϕ.

Another approach is the *Fast Marching Method* [117, 118] which uses a discrete approach similar to the famous Dijkstra's method to compute a distance function on a grid, given initial values at the interface.

The former method has the drawback that it will in general not preserve the interface exactly during the re-initialization. Furthermore, being nonlinear, (3.15) is in some sense more difficult to solve than the linear advection equation (3.16) below, which is used to propagate the level set itself. The Fast Marching Method is more attractive from this point of view, but has not yet been extended to be used in a finite element setting with locally varying refinement levels and polynomial degrees, which is the setting that we consider in this work.

We will therefore leave the precise definition of the level set function open, and only impose some arbitrarily chosen sign conventions. These are illustrated in Fig. 3.1. The normal $\boldsymbol{n}_\Gamma$ from Ω_1 to Ω_2, and the corresponding curvature of Γ can be computed from Equations (3.13) and (3.14).

The evolution of ϕ will be governed by Equation (3.10). This fulfills the first requirement of Section 3.1. Since $\nabla\phi$ is normal to the interface, v_Γ can be substituted by $\boldsymbol{u}$ using Equation (2.55):

$$\frac{\partial\phi}{\partial t} + \boldsymbol{u}\cdot\nabla\phi = 0. \tag{3.16}$$

To completely determine the evolution of the level set function, the PDE (3.16) must be extended with initial and boundary conditions. The initial condition takes the form of a level set function $\phi^0(\boldsymbol{x})$, which is such that its zero level set coincides with the interface $\Gamma(0)$ at $t = 0$. As for the boundary conditions, it suffices to fix the value of ϕ on the inflow part $\partial\Omega_{\text{in}} = \{\boldsymbol{x} \in \partial\Omega : \boldsymbol{w}(\boldsymbol{x},t)\cdot\boldsymbol{n}(\boldsymbol{x}) < 0\}$ of the boundary. We represent the corresponding Dirichlet data with the function $\phi^{\text{in}}(\boldsymbol{x},t)$, $\boldsymbol{x} \in \partial\Omega_{\text{in}}$.

Next, we need to be able to determine whether a given point $\boldsymbol{x}$ belongs to Ω_1 or Ω_2, so that the material properties can be evaluated correctly. This information is directly available from the sign of ϕ. By introducing the Heaviside step function

$$H(\phi(\boldsymbol{x},t)) = \begin{cases} 1, & \phi(\boldsymbol{x},t) > 0, \\ 0, & \phi(\boldsymbol{x},t) \le 0, \end{cases} \tag{3.17}$$

the global density and viscosity functions can conveniently be defined as functions of ϕ through

$$\rho(\boldsymbol{x},t) = \rho_1 + H(\phi(\boldsymbol{x},t))(\rho_2 - \rho_1), \tag{3.18}$$

and

$$\mu(\boldsymbol{x},t) = \mu_1 + H(\phi(\boldsymbol{x},t))(\mu_2 - \mu_1), \tag{3.19}$$

respectively.

The third requirement is the need to represent functions that are localized on the interface. Similar to what was done with the Heaviside function above, the level set representation of the interface Γ makes it easy to define a Dirac measure for Γ:

$$\delta_\Gamma(t) = \delta(\phi(\boldsymbol{x},t)). \tag{3.20}$$

Multiplying any function with $\delta_\Gamma(t)$ localizes the function to the interface, where $\phi = 0$.

Having addressed the requirements from Section 3.1, we are now ready to reformulate the two-component flow model. This results

in the following set of differential and algebraic equations for the variables $\boldsymbol{u}$, p and ϕ:

$$\rho\left(\frac{\partial \boldsymbol{u}}{\partial t}+(\boldsymbol{u}\cdot\nabla)\,\boldsymbol{u}\right)=\rho\boldsymbol{f}\,(\boldsymbol{x},t)+\nabla\cdot\sigma \qquad \text{in } \Omega, \tag{3.21}$$

$$\nabla\cdot\boldsymbol{u}=0 \qquad \text{in } \Omega, \tag{3.22}$$

$$\boldsymbol{u}=\boldsymbol{u}_D \qquad \text{on } \partial\Omega_D, \tag{3.23}$$

$$\sigma\boldsymbol{n}=0 \qquad \text{on } \partial\Omega_N, \tag{3.24}$$

$$\boldsymbol{u}(\boldsymbol{x},0)=\boldsymbol{u}^0 \qquad \text{in } \Omega, \tag{3.25}$$

$$[\![\sigma]\!]_\Gamma\boldsymbol{n}_\Gamma=-\tau\kappa\boldsymbol{n}_\Gamma \qquad \text{on } \Gamma, \tag{3.26}$$

$$\frac{\partial\phi}{\partial t}+\boldsymbol{u}\cdot\nabla\phi=0 \qquad \text{in } \Omega, \tag{3.27}$$

$$\phi=\phi^{\text{in}} \qquad \text{on } \partial\Omega_{\text{in}}, \tag{3.28}$$

$$\phi(\boldsymbol{x},0)=\phi^0(\boldsymbol{x}) \qquad \text{in } \Omega, \tag{3.29}$$

where

$$\sigma\,(\boldsymbol{x},t)=-p\,(\boldsymbol{x},t)\,\mathcal{I}+2\mu\,(\boldsymbol{x},t)\,D(\boldsymbol{u}) \qquad \text{in } \Omega, \tag{3.30}$$

$$\rho\,(\boldsymbol{x},t)=\rho_1+H(\phi(\boldsymbol{x},t))\,(\rho_2-\rho_1) \qquad \text{in } \Omega, \tag{3.31}$$

$$\mu\,(\boldsymbol{x},t)=\mu_1+H(\phi(\boldsymbol{x},t))\,(\mu_2-\mu_1) \qquad \text{in } \Omega, \tag{3.32}$$

$$\boldsymbol{n}_\Gamma=\frac{\nabla\phi}{|\nabla\phi|} \qquad \text{on } \Gamma, \tag{3.33}$$

$$\kappa=\nabla\cdot\boldsymbol{n}_\Gamma=\nabla\cdot\left(\frac{\nabla\phi}{|\nabla\phi|}\right). \qquad \text{on } \Gamma. \tag{3.34}$$

Here, $\mathcal{I}$ represents the identity matrix. This revised set of equations is the basis for the variational problem whose formulation and discretization will be the topic of the next chapter.

4

VARIATIONAL FORMULATION AND DISCRETIZATION

This chapter describes the discretization of the two-component flow model. In Section 4.1, we first introduce a splitting of the model into a flow problem based on the Navier-Stokes equations, and an interface problem based on the advection equation for the level set function. The finite element method, upon which the spatial discretization of these two problems is based, is then introduced in an abstract setting in Section 4.2. Sections 4.3 and 4.4 describe the discretization of the interface and flow models using the θ-method in time and finite elements in space. Some details concerning the approximation of the interfacial tension force and the piecewise continuous material parameters are discussed in Sections 4.5 and 4.6. Finally, Section 4.7 provides a summary of the entire solution procedure.

4.1 TWO-STEP MODEL EVOLUTION

The mathematical model derived in Chapters 2 and 3 describes a relatively complex process, that couples several different underlying phenomena. Altogether, if d denotes the dimension of the problem, there are $d+2$ PDE in $d+2$ variables. The PDE describe the evolution only of the velocity and level set variables, whereas the evolution of p is determined implicitly to satisfy the incompressibility condition. The PDE are of different type: the level set equation is a linear hyperbolic equation, whereas the Navier-Stokes system is nonlinear and of mixed type. The evolution processes are coupled in both directions: ϕ influences the flow variables in a rather complicated, nonlinear manner, and $\boldsymbol{u}$ in turn drives the movement of ϕ, albeit in a more straightforward way.

One could imagine discretizing the entire model at once, which would result in a large nonlinear time-dependent problem that would have to be linearized, discretized in time and space, and then solved numerically. In order to limit the complexity of both the exposition in the thesis and the numerical implementation, we will instead split the model into two sub-problems corresponding to the Navier-Stokes equations and the Level Set equations, respectively. The coupling between the problems will then be treated at the discrete level, by solving the two problems in succession inside each time-step.

Letting $(t_n) \in [0, T), n = 0, 1, \ldots$ denote a monotonically increasing sequence of time-steps that will be used in the time discretization, Fig. 4.1 gives an overview of the two-step computation. At the be-

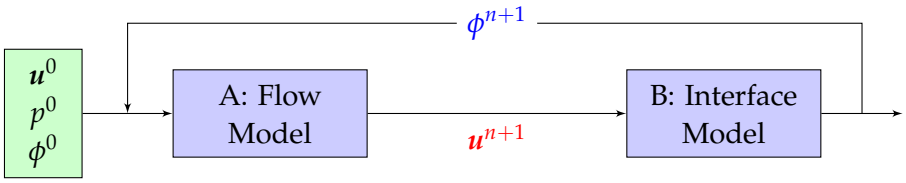


Figure 4.1: Schematic view of the coupling between the flow model and interface model.

ginning of each time-step, the velocity and pressure are updated by solving sub-problem A. The updated values are then used to compute the level set function for the next time-step in sub-problem B. The details of solving the sub-problems will be discussed in the following sections of this chapter.

4.2 SPATIAL DISCRETIZATION WITH FINITE ELEMENT METHODS

The finite element method is used to derive the discretization of the PDE in the spatial dimensions. Since its invention, which can be traced back to the 1940s, the finite element method has become popular among practitioners of mathematics and several engineering disciplines alike. At first, the development was driven mainly by members of the engineering community, who sought to solve problems in structural mechanics. The mathematical theory of finite elements began in the second half of the 1960s, with the establishment of convergence proofs and a priori error estimates. Much progress was made in the 1970s to extend the method from linear elliptic and parabolic problems to nonlinear models and problems with a hyperbolic character. This theory made use of the then recent developments in functional analysis and approximation theory. For a more complete overview of the early development of the finite element method, see [93].

The finite element method has several advantages over other discretization techniques, such as the finite difference or finite volume methods. By considering a weak form of the PDE, it is possible to solve problems with irregular data in a robust way. Such irregular data can include discontinuous material properties, forces applied to isolated points, and domains with re-entrant corners. Furthermore, through the use of unstructured meshes and curved elements, it is possible to obtain accurate approximations also on the complex domains that occur in industrial and scientific applications. More details can be found e. g. in [28, 35, 46].

In this section, we introduce a mathematical concepts that form the basis of finite element methods. These concepts will be applied in Sections 4.3 and 4.4 to discretize the two parts of the two-component flow model.

4.2.1 *Abstract FEM Framework*

To set the stage for the coming sections which describe the discretization in detail, this section provides an outline of how a discrete problem is derived from a continuous boundary value problem for a PDE. We generally follow the approach taken in [46], with suitable simplifications to achieve a level of generality that is appropriate for this work.

Consider the following abstract boundary value problem for a linear PDE posed on a domain Ω:

$$\mathcal{L}u(\boldsymbol{x}) = b(\boldsymbol{x}), \qquad \forall \boldsymbol{x} \in \Omega, \tag{4.1}$$

$$\mathcal{B}u(\boldsymbol{x}) = g(\boldsymbol{x}), \qquad \forall \boldsymbol{x} \in \partial\Omega. \tag{4.2}$$

Here $\mathcal{L}$ and $\mathcal{B}$ are linear partial differential operators acting respectively on the solution function u and its trace on the boundary of Ω. The right hand side functions b and g are typically known data parameters of the problem.

We consider first Equation (4.1) in isolation, and leave the enforcement of the boundary conditions expressed in Equation (4.2) to later. In order for Equation (4.1) to be well-defined, $u(\boldsymbol{x})$ has to lie in the domain of $\mathcal{L}$ and b in its range, for every point $\boldsymbol{x} \in \Omega$. For an operator of order m, this implies that b and $u \in C^m(\Omega)$ is required for the problem to be well-posed. Often such restrictions are too strict for the mathematical model that one would like to use.

The first ingredient of a finite element method is therefore the reformulation of the PDE into a weaker problem, based on the *weighted residual method*. We introduce a space of *test functions* V, and state the *weighted residual equation*:

$$\int_\Omega \left(\mathcal{L}u(\boldsymbol{x}) - b(\boldsymbol{x})\right) \psi(\boldsymbol{x}) \,\mathrm{d}x = 0, \qquad \forall \psi \in V, \tag{4.3}$$

where the integration is to be interpreted in the sense of Lebesgue. At this stage, the restrictions on the solution and data can already be weakened by interpreting the derivatives in $\mathcal{L}$ as *weak derivatives*, and choosing an appropriate test space V. The concept of weak derivative, in this context is defined as follows (this follows [28]):

Definition 1. *Weak Derivative*

Let $v \in L^1_{loc}(\Omega)$, the space of locally Lebesgue integrable functions on Ω. Furthermore, let $\alpha = (\alpha_1, \ldots, \alpha_d)$ be a multi-index, and $D^\alpha = (\partial^{\alpha_1}, \ldots, \partial^{\alpha_d})$ the corresponding derivative operator. Provided that such a function exists, $w \in L^1_{loc}(\Omega)$ is the weak derivative with index α of v if

$$\int_\Omega v D^\alpha \varphi \,\mathrm{d}x = (-1)^{|\alpha|} \int_\Omega w\varphi \,\mathrm{d}x, \quad \forall \varphi \in C_0^\infty(\Omega), \tag{4.4}$$

where $C_0^\infty(\Omega)$ is the set of all infinitely differentiable functions with compact support in Ω.

Depending on the order of these derivatives, the choice of V usually falls upon a member of the family of *Sobolev* spaces:

Definition 2. *Sobolev Spaces*
The Sobolev space $W^{k,p}(\Omega)$ with index (k,p) is the set of L^p-functions whose weak derivatives up to order k are also L^p-functions:

$$W^{k,p}(\Omega) = \{w \in L^p(\Omega) : D^\alpha w \in L^p(\Omega), \forall \alpha, |\alpha| \leq k\}. \tag{4.5}$$

In this work, we will exclusively deal with spaces where $p = 2$, and use the widespread notation $H^k(\Omega) = W^{k,2}(\Omega)$. Note that these spaces are Hilbert spaces, which is not true for $W^{k,p}(\Omega)$ when $p \neq 2$.

For operators whose order m is even, it is often advantageous to transform the integral with e. g. Green's identity, to have a different operator applied to u, and thereby lowering the regularity requirements further. Introducing the notation a and b for the resulting bilinear and linear forms, respectively; and a *solution space* U, the transformed problem then reads

$$\begin{aligned} &\text{Seek } u \in U \text{ such that} \\ &\qquad a(u,\psi) = b(\psi), \qquad \forall \psi \in V. \end{aligned} \tag{4.6}$$

In order to be able to use this formulation in a meaningful way, one must establish that the problem is *well-posed* in the sense that it has a unique solution, which depends continuously on the data. The following theorem, which is discussed and proven in [46, Chapter 2] (in the more general setting of Banach spaces), gives sufficient and necessary conditions for well-posedness of (4.6):

Theorem 2. *Well-posed weak problem*
Let U and V be Hilbert spaces, $a : U \times V \to \mathbb{R}$ a bounded bilinear form, and $b : V \to \mathbb{R}$ a bounded linear form. Then, the problem described in (4.6) is well-posed if and only if:

$$\exists \beta > 0 : \inf_{u \in U} \sup_{\psi \in V} \frac{a(u,\psi)}{\|u\|_U \|\psi\|_V} \geq \beta, \tag{4.7}$$

and

$$\forall \psi \in V, \quad \text{if } a(u,\psi) = 0, \forall u \in U, \text{then } \psi = 0. \tag{4.8}$$

In general, one should also try to determine to what extent the solution of the weakened problem and that of the original problem correspond. Often, a sufficiently smooth solution to the weak problem will also be a solution of the boundary value problem for the PDE. It should be noted that in some cases, the latter was derived from an integral formulation, which might in itself yield a weak problem, without having to go via a PDE model.

For the treatment of the boundary conditions expressed in Equation (4.2), there are two choices. One possibility is to impose these conditions in a strong form by adding them as restrictions on the solution space U, and modifying V accordingly. The boundary conditions can also be enforced weakly by casting the boundary residual $\mathcal{B}u - g$ into a weak form, and incorporating this into a and b, respectively.

The details of the weak form and the choice of the solution and test spaces U and V lead to different types of methods. Some possibilities include *collocation methods*, *subdomain methods*, *least-squares methods* and *Galerkin methods*. These all have in common that they can be expressed using the weighted residual equation (4.3), and hence they are classified as weighted residual methods. Only the last two types are considered in this work.

The solution and test spaces are of infinite dimension. In order to find a numerical approximation of the solution, it is necessary to reduce the problem to a finite dimension. This discretization is performed by replacing U and V in Equation (4.6) by the finite-dimensional spaces U_h and V_h, respectively. In the general case, a and b will also be approximated, by a_h and b_h. This *non-consistency* can be due, e. g. to the use of quadrature instead of exact integration in evaluation of the integrals, but will not be discussed here further. More details can be found e. g. in [46, Chapter 2.3] or [28, Chapter 10].

The result is the discrete weak problem:

Seek $u_h \in U_h$ such that

$$a(u_h, \psi_h) = b(\psi_h), \qquad \forall \psi_h \in V_h. \tag{4.9}$$

Most commonly, one chooses $U_h \subset U$ and $V_h \subset V$, in which case the resulting method is called *conforming*, but other choices, which lead to *non-conforming* methods, are also possible.

The question of whether the discrete problem is well-posed is in general independent of whether the continuous problem is well-posed. Only in the special case of a consistent, conforming method, where $U = V$, and where the bilinear form is additionally *coercive*, i. e. $\exists \gamma > 0 : \forall u \in U, a(u,u) \geq \gamma \|u\|_U^2$, is it known that if the continuous problem is well-posed, then the same holds for the discrete problem. In other cases, the conditions in Theorem 2 have to be demonstrated to hold for $(u_h, \psi_h) \in U_h \times V_h$. In this context, the first condition Theorem 2 is often called the *inf-sup* or *Ladyzhenskaya-Babuška-Brezzi* (LBB) condition.

As will be described in the following, the discrete spaces are defined via the construction of global basis functions, so that $U_h = \operatorname{span}\{\varphi_i\}$ and $V_h = \operatorname{span}\{\psi_j\}$. In most cases, the discrete spaces are chosen to have the same dimension N, which makes it possible to derive a linear system of equations whose solution vector characterizes the solution u_h to Equation (4.9).

The linear system is obtained by expanding u_h as a linear combination of φ_i:

$$u_h = \sum_{j=1}^{N} u_j \varphi_j, \tag{4.10}$$

and substituting into Equation (4.9):

$$\begin{gathered} \text{Seek } \boldsymbol{u} \in \mathbb{R}^N \text{ such that } \forall \psi_h \in V_h, \\ a\left(\sum_{j=1}^{N} u_j \varphi_j, \psi_h\right) = b\left(\psi_h\right). \end{gathered} \tag{4.11}$$

Due to linearity, it suffices that the equation holds for the basis functions in V_h, which yields

$$a\left(\sum_{j=1}^{N} u_j \varphi_j, \psi_i\right) = b\left(\psi_i\right), \qquad \forall i = 1, \ldots, N. \tag{4.12}$$

This is equivalent to the linear system of equations

$$\begin{gathered} A\boldsymbol{u} = b, \\ \text{where } A_{ij} = a\left(\varphi_j, \psi_i\right) \\ \text{and } b_i = b\left(\psi_i\right). \end{gathered} \tag{4.13}$$

4.2.2 *Construction of the Finite Element Space*

A *finite element method* is most commonly understood as a method in which the discrete solution and test spaces are constructed from a basis consisting of piecewise polynomial shape functions. This basis is constructed by first constructing a mesh $\mathcal{M}$ that partitions Ω into a set of cells of simple shapes. To each cell K, one associates a *finite element*, defined as follows (see [46], Chapter 1.2):

Definition 3. *Finite Element*
A finite element is a triplet $(K, \mathcal{P}, \Sigma)$, where

- *K is a compact, connected subset of $\mathbb{R}^d$ with non-empty interior,*
- *$\mathcal{P}$ is a vector space of functions defined on K,*
- *Σ is a set of linear forms $\sigma_i : \mathcal{P} \to \mathbb{R}, i = 1, \ldots, m$, which form a basis for $\mathcal{L}(P; \mathbb{R})$. $\{\sigma_i\}$ are called the local degrees of freedom.*

Definition 4. *Local Shape Functions*
The local shape functions of a finite element $(K, \mathcal{P}, \Sigma)$, is the set of basis functions $\varphi_i, 1, \ldots, dim(P)$ for $\mathcal{P}$ such that $\sigma_i(\varphi_j) = \delta_{ij}, 1 \leq i, j \leq dim(P)$.

The concrete definition of a finite element can be either be performed by choosing the basis of local degrees of freedom first, and then letting the local shape functions be the dual of this basis; or the other way around. The most commonly used elements belong to the class of *Lagrange* elements where the local degrees of freedom are defined as function evaluation at a set of nodal points $\boldsymbol{x}_i \in K$. An example of the second method is the definition of the elements based on Lobatto polynomials in Section 5.4.2.

The definition of the mesh and the elements is used to define a set of global basis functions ψ_k for the discrete spaces U_h and V_h. An important feature of finite element methods is that the support of each global basis function is localized to one or a few cells only. This property, which arises naturally from the construction described above, ensures that the matrix of the resulting linear system is sparse, and thus makes it possible to efficiently solve problems with millions of unknowns.

In order to obtain a conforming method, the functions in the discrete spaces often have to fulfill some continuity conditions along the interfaces between cells. In a *continuous Galerkin* method, these continuity conditions are imposed by adding constraints on the values of the degrees of freedom.

It is clear that the choice of U_h directly influences how accurate the solution is, and how fast it can be computed. A larger solution space possibly increases the accuracy of the result, but also the number of unknown variables, which strongly influence the effort needed to compute the solution. Additionally, the choice of basis is important, since it influences the condition number of the matrix, which again plays an important role for the computational cost. The automatic construction of a sequence of spaces which finds a good balance in this trade-off is the topic of *adaptive* methods, which will be treated in Chapter 5.

4.3 LEAST-SQUARES VARIATIONAL FORMULATION OF THE INTERFACE MODEL

This section deals with the treatment of sub-problem B introduced in Section 4.1. We begin by stating an instationary boundary value problem with the PDE for the level set function (3.27) and the associated side conditions. Next we perform semi-discretization for the time variable, and proceed to derive a variational formulation of the problem based on the least-squares weighted residual method. This will later form the basis for computing a finite element solution for the problem.

Assume that $\boldsymbol{w}(\boldsymbol{x}, t)$ is a velocity field that is continuous on a time-space cylinder $\Omega \times (0, T)$. A Dirichlet boundary condition for the level set function is provided on the inflow boundary $\partial\Omega_{in} = \{\boldsymbol{x} \in$

$\partial\Omega : \boldsymbol{w}(\boldsymbol{x},t)\cdot\boldsymbol{n}(\boldsymbol{x}) < 0\}$ by the function ϕ^{in}. For a general velocity field $\boldsymbol{w}$, $\partial\Omega_{in}$ could vary in time, but we restrict the discussion in the following to situations where this is not the case. An initial condition for the level set function is given by $\phi^0(\boldsymbol{x})$. The interface model corresponds to Equations (3.27)–(3.29):

$$\frac{\partial\phi}{\partial t} + \boldsymbol{w}\cdot\nabla\phi, = 0 \qquad (\boldsymbol{x},t)\in\Omega\times(0,T), \tag{4.14}$$

$$\phi\left(\boldsymbol{x},t\right) = \phi^{in}\left(\boldsymbol{x},t\right), \qquad (\boldsymbol{x},t)\in\partial\Omega_{in}\times(0,T), \tag{4.15}$$

$$\phi\left(\boldsymbol{x},0\right) = \phi^{0}\left(\boldsymbol{x}\right), \qquad \boldsymbol{x}\in\Omega. \tag{4.16}$$

To be able to solve this problem numerically, a discretized version of it, which yields an approximate solution, will be presented below. The discretization proceeds in two steps, according to Rothe's method. First a semi-discretized form for the time dimension is set up using the standard θ-method. This corresponds to a sequence of boundary value problems, each of which serves to advance the solution one step in time. Each of these boundary value problems is based on a first-order advection-reaction equation, for which it is well known that a standard finite element discretization will not be stable. We therefore apply the least-squares weighted residual formalism to derive a stabilized variational problem that can be solved with the finite element method to yield an approximation of the solution.

The θ-method for discretizing ordinary differential equations (ODE) is really a family of methods, parameterized by the real variable $\theta\in[0,1]$. It includes the well-known explicit Euler ($\theta = 0$) and implicit Euler ($\theta = 1$) methods, as well as the second-order trapezoidal method ($\theta = 0.5$). A description of the method can be found in standard textbooks on the numerical solution of ODE, e. g. [75].

Following a common convention, the semi-discretized variables receive a superscript to indicate the corresponding time-step. For instance, $\phi^n = \phi(t_n)$. We also introduce the length of the time-step $\Delta t_n = t_{n+1} - t_n$. The semi-discretized PDE can then be stated in residual form as

$$R(\phi^{n+1}) = \frac{\phi^{n+1}-\phi^n}{\Delta t_n} + (1-\theta)\,\boldsymbol{w}^n\cdot\nabla\phi^n + \theta\boldsymbol{w}^{n+1}\cdot\nabla\phi^{n+1} = 0, \tag{4.17}$$

or equivalently as

$$\phi^{n+1} + \theta\Delta t_n\boldsymbol{w}^{n+1}\cdot\nabla\phi^{n+1} = \phi^n - (1-\theta)\,\Delta t_n\boldsymbol{w}^n\cdot\nabla\phi^n. \tag{4.18}$$

From the second form, this problem can thus be interpreted as an *advection-reaction* equation for the level set ϕ^{n+1}.

For each value of θ, Equation (4.17) provides a scheme for advancing the level set function ϕ^n to the next time-step, in the form of a PDE that depends only on the spatial variable. As described in

Section 4.2.1, in order to discretize such a problem with the finite element method, the first step is to choose a space V of test functions, and formulate the weighted residual equation

$$\int_\Omega \psi R(\phi^{n+1})\,\mathrm{d}x = 0, \qquad \forall \psi \in V. \tag{4.19}$$

Since there are only first derivatives of ϕ^{n+1} in the residual, it is not necessary to perform integration by parts in this case, and the natural choice of test space is $V = L^2(\Omega)$. The solution is required to have weak derivatives in the direction of $\boldsymbol{w}$ only, so the solution space $S = \{\phi \in L^2(\Omega) : \boldsymbol{w} \cdot \nabla\phi \in L^2(\Omega)\}$. It can be shown (see e. g. [46], Chapter 5), that (4.18) is a *Friedrichs' system*, and that the associated differential operator

$$\begin{aligned} &A : S_0 \to L^2(\Omega) \\ &\phi \mapsto \phi + \theta\Delta t_n \boldsymbol{w} \cdot \nabla\phi, \end{aligned} \tag{4.20}$$

where

$$S_0 = \{\phi \in S : \phi|_{\partial\Omega_{in}} = 0\}, \tag{4.21}$$

is an isomorphism under the additional condition that almost everywhere in Ω,

$$1 - \frac{1}{2}\theta\Delta t_n \nabla\cdot\boldsymbol{w} > 0. \tag{4.22}$$

In the present case, the continuous velocity field $\boldsymbol{w}$ will be divergence-free as it is the solution of an incompressible Navier-Stokes equation, and the discrete velocity field will at least be approximately divergence-free. Hence, we assume that this condition will be fulfilled.

In this case the problem 4.19 is well-posed, but it is not coercive. With a standard Galerkin method, where the discrete test and solution spaces are the same, it can be shown that the coefficient β in the LBB condition for the discrete problem will tend to zero as the mesh is refined. As the approximation space is enlarged, the discrete problem will hence become less and less well-posed, which typically manifests itself by the appearance of large oscillating disturbances in the solution. The standard Galerkin formulation is thus unstable for this type of first-order problems.

In order to still be able to employ finite elements for the discretization, we will therefore use a different formulation, which leads to stable discrete problems. This formulation is derived via the Least-Squares Weighted Residual Method (LSWRM), which yields modified weights for the residual. By interpreting these weights as different test functions, one can see that this least-squares stabilization is closely related to other common stabilization techniques, such as

the Stabilized Upwind Petrov-Galerkin (SUPG) and Galerkin/Least-Squares (GaLS) methods.

In the literature, the latter methods seem to be widely employed, whereas the use of least-squares formulations is not so common. For the second-order, singularly perturbed problems that arise e.g. in convection-dominated flows, direct application of the LSWRM to the second-order problem will lead discrete problems with very large condition numbers, which seems to have given this method something of a bad reputation. In general, one should always reformulate the problem to a system of first-order equations before using the LSWRM. Good introductions to LSWRM, with applications to a wide range of problems, are given in [91] and [46]. The use of this method specifically for solving a Level Set equation in the context of two-phase fluid flow, which forms the basis for the method employed here, has been presented in [32].

The derivation of the variational formulation via the LSWRM will now be described. Assuming that there exists a unique solution $\phi^{n+1} \in S$ to (4.17), that solution will also fulfill

$$\left\| R(\phi^{n+1}) \right\|_0^2 = \int_\Omega R(\phi^{n+1})^2 \, \mathrm{d}x = 0. \tag{4.23}$$

Since the norm is non-negative, the solution to the original problem can be determined by seeking the minimizer of the norm. A necessary condition for this minimizer is that its first variation vanishes for all test functions $\delta\phi \in S$. We thus obtain the following condition on ϕ^{n+1}, which is to hold for all $\delta\phi \in S$:

$$\begin{aligned}
&\frac{\mathrm{d}}{\mathrm{d}s}\left(\left\| R(\phi^{n+1} + s\delta\phi) \right\|_0^2 \right)\Big|_{s=0} = 0, \\
\Leftrightarrow \frac{\mathrm{d}}{\mathrm{d}s} \int_\Omega \left(R(\phi^{n+1}) + s\left(\frac{\delta\phi}{\Delta t_n} + \theta \boldsymbol{w}^{n+1} \cdot \nabla\delta\phi \right) \right)^2 \mathrm{d}x \Big|_{s=0} &= \\
\int_\Omega 2R(\phi^{n+1}) \left(\frac{\delta\phi}{\Delta t_n} + \theta \boldsymbol{w}^{n+1} \cdot \nabla\delta\phi \right) \mathrm{d}x + & \\
\int_\Omega 2s \left(\frac{\delta\phi}{\Delta t_n} + \theta \boldsymbol{w}^{n+1} \cdot \nabla\delta\phi \right) \mathrm{d}x \Big|_{s=0} &= 0, \\
\Leftrightarrow \int_\Omega R\left(\phi^{n+1}\right) \left(\frac{\delta\phi}{\Delta t_n} + \theta \boldsymbol{w}^{n+1} \cdot \nabla\delta\phi \right) \mathrm{d}x &= 0.
\end{aligned}$$

The variational problem to be solved in each time-step, with the Dirichlet boundary condition imposed strongly, can then be stated as follows:

$$\text{Seek } \phi^{n+1} \in S_{\text{in}}, \text{ such that} \quad \forall \delta\phi \in S_0, \tag{4.24}$$

$$\begin{aligned}
&\int_\Omega \left(\phi^{n+1} + \theta \Delta t_n \boldsymbol{w}^{n+1} \cdot \nabla \phi^{n+1} \right) \left(\delta\phi + \theta \Delta t_n \boldsymbol{w}^{n+1} \cdot \nabla \delta\phi \right) \mathrm{d}x = \\
&\int_\Omega \left(\phi^n - (1-\theta)\, \Delta t_n \boldsymbol{w}^n \cdot \nabla \phi^n \right) \left(\delta\phi + \theta \Delta t_n \boldsymbol{w}^{n+1} \cdot \nabla \delta\phi \right) \mathrm{d}x.
\end{aligned} \tag{4.25}$$

Here $S_{in} = \{\phi \in S : \phi|_{\partial\Omega_{in}} = \phi^{in}\}$.

As mentioned previously, this method can be seen as a weighted residual equation with weight functions $\psi = \delta\phi + \theta\Delta t_n \boldsymbol{w}^{n+1}\cdot\nabla\delta\phi$. In this sense, it can be seen as a *Petrov-Galerkin* method where the space of test functions is different from the solution spaces. In this respect, this method is similar to SUPG stabilization, the only difference being that an additional, usually mesh-dependent, weight α is given to the gradient term in the SUPG test functions: $\psi_{\text{SUPG}} = \delta\phi + \alpha\theta\Delta t_n \boldsymbol{w}^{n+1}\cdot\nabla\delta\phi$.

Both methods can be thought of as adding artificial diffusion in the direction of the flow, which corresponds to an *upwind* discretization. The extra parameter in the SUPG method makes it possible (or necessary, depending on your point of view), to fine-tune the amount of stabilization, taking into account the local resolution of the mesh. Since usually, $\delta < 1$, the SUPG can be less diffusive, and hence degrade the accuracy less, than the least squares method. However, the latter has the advantage of being symmetric and coercive, which further implies that the discrete problem is well-posed as long as a conforming approximation space is chosen.

Working with the space S is quite impractical, since the definition of this space depends on on the velocity field $\boldsymbol{w}$, which varies both in time and space. In this work, we therefore restrict ourselves to the solution of this problem in the more standard space $H^1_{\text{in}}(\Omega) = \{\phi \in H^1(\Omega) : \phi|_{\partial\Omega_{in}} = \phi^{in}\} \subset S_{in}$, using test functions from $H^1_0(\Omega) = \{\phi \in H^1(\Omega) : \phi|_{\partial\Omega_{in}} = 0\} \subset S_0$. The corresponding variational problem reads:

$$\text{Seek } \phi^{n+1} \in H^1_{\text{in}}(\Omega), \text{ such that} \quad \forall\delta\phi \in H^1_0(\Omega), \tag{4.26}$$

$$\int_\Omega \left(\phi^{n+1} + \theta\Delta t_n \boldsymbol{w}^{n+1}\cdot\nabla\phi^{n+1}\right)\left(\delta\phi + \theta\Delta t_n \boldsymbol{w}^{n+1}\cdot\nabla\delta\phi\right)\,\mathrm{d}x = \int_\Omega \left(\phi^n - (1-\theta)\,\Delta t_n \boldsymbol{w}^n\cdot\nabla\phi^n\right)\left(\delta\phi + \theta\Delta t_n \boldsymbol{w}^{n+1}\cdot\nabla\delta\phi\right)\,\mathrm{d}x. \tag{4.27}$$

For the finite element discretization of this problem, we will use approximation spaces defined through an hp-adaptive enlargement process, in which the mesh resolution and polynomial degrees are allowed to vary locally. Such spaces and their construction will be discussed in Chapter 5. In order to assure that the corresponding discrete problems are well-posed, the discrete solution and test spaces are assumed to be H^1-conforming: $S^{\text{hp}}_{\text{in}} \subset H^1_{\text{in}}(\Omega)$ and $S^{\text{hp}}_0 \subset H^1_0(\Omega)$, respectively. This means that the functions of the space are required to be globally continuous. Fulfilling this requirement is one of the complications that follow when using hp-spaces, and it will discussed in Section 5.4.6.

Given discrete spaces of dimension N of test and solution functions, and a common basis $\xi_i, i = 1,\ldots,N$, for these spaces, we can expand the solution according to $\phi^{n+1} = \sum_{j=1}^N \phi_j^{n+1}\xi_j$. The variational

problem (4.26) is then equivalent to the linear system of equations $\sum_{j=1}^{N} A_{ij}\phi_j^{n+1} = b_i$, where

$$A_{ij} = \int_\Omega \left(\xi_j + \theta\Delta t_n \boldsymbol{w}^{n+1}\cdot\nabla\xi_j\right)\left(\xi_i + \theta\Delta t_n \boldsymbol{w}^{n+1}\cdot\nabla\xi_i\right)\,\mathrm{d}x, \tag{4.28}$$

and

$$b_i = \int_\Omega \left(\phi^n - (1-\theta)\,\Delta t_n \boldsymbol{w}^n\cdot\nabla\phi^n\right)\left(\xi_i + \theta\Delta t_n \boldsymbol{w}^{n+1}\cdot\nabla\xi_i\right)\,\mathrm{d}x. \tag{4.29}$$

The matrix is symmetric and positive definite, which makes it possible to use the efficient and robust Conjugate Gradient method for the solution. A numerical of the convergence of this model and the implementation that we used is presented in Section 6.3.

4.4 MIXED WEAK FORMULATION OF THE FLOW MODEL

The next step is to discretize the flow model (sub-problem A) which was stated at the end of Section 3. We recall the initial boundary value problem for the Navier-Stokes equations described by (3.21) – (3.26):

$$\rho\left(\frac{\partial \boldsymbol{u}}{\partial t} + \boldsymbol{u}\cdot\nabla\boldsymbol{u}\right) = \rho\boldsymbol{f} + \nabla\cdot\sigma(\boldsymbol{u},p) \qquad \text{in } \Omega, \tag{4.30}$$

$$\nabla\cdot\boldsymbol{u} = 0 \qquad \text{in } \Omega, \tag{4.31}$$

$$\boldsymbol{u} = \boldsymbol{u}_D \qquad \text{on } \partial\Omega_D, \tag{4.32}$$

$$\sigma\boldsymbol{n} = 0 \qquad \text{on } \partial\Omega_N, \tag{4.33}$$

$$[\![\sigma]\!]_\Gamma \boldsymbol{n}_\Gamma = -\tau\kappa\boldsymbol{n}_\Gamma \qquad \text{on } \Gamma, \tag{4.34}$$

$$\boldsymbol{u}(\boldsymbol{x},0) = \boldsymbol{u}^0 \qquad \text{in } \Omega. \tag{4.35}$$

Our first step is to discretize the momentum balance equation with respect to the time variable. For this purpose, we rewrite (4.30) as

$$\rho\frac{\partial \boldsymbol{u}}{\partial t} = F(\boldsymbol{u},t), \tag{4.36}$$

where

$$F(\boldsymbol{u},t) = \rho\boldsymbol{f} + \nabla\cdot\sigma - \rho\boldsymbol{u}\cdot\nabla\boldsymbol{u}, \tag{4.37}$$

and apply the θ-method with a constant time-step Δt.

$$\left(\frac{\rho^{n+1}\boldsymbol{u}^{n+1} - \rho^n\boldsymbol{u}^n}{\Delta t}\right) = \theta F(\boldsymbol{u}^{n+1}, t^{n+1}) + (1-\theta)\,F(\boldsymbol{u}^n, t^n). \tag{4.38}$$

Here we have followed the notation introduced in Section 4.3 with for the definition of the time-step Δt, and the use of the superscript index n.

The method is unconditionally stable for $\theta \geq 0.5$, whereas the use of $\theta = 0$ (explicit Euler method) generally requires very small time-steps to ensure stability. Since the trapezoidal method ($\theta = 0.5$) is the only method of this class which is of second order, this choice is especially attractive. A drawback of this scheme, however, is that it requires the specification of an initial condition for the pressure, which is not available in our model. We avoid this by using the implicit Euler ($\theta = 1$) method when solving the first time-step, and the trapezoidal method for the remaining time-steps.

The semi-discretization in time yields a sequence of boundary value problems which can be solved to obtain approximations $(\boldsymbol{u}^{n+1}, p^{n+1})$ of the velocity and pressure variables. $\boldsymbol{u}^0$ is given by the initial condition (4.35), and $(\boldsymbol{u}^{n+1}, p^{n+1})$ is the solution of

$$\rho^{n+1}\boldsymbol{u}^{n+1} - \theta\Delta t F^{n+1} = \rho^n \boldsymbol{u}^n + (1-\theta)\,\Delta t F^n \qquad \text{in } \Omega, \tag{4.39}$$

$$\nabla\cdot\boldsymbol{u}^{n+1} = 0 \qquad \text{in } \Omega, \tag{4.40}$$

combined with the boundary and interface conditions (4.32) – (4.34) evaluated at time t^{n+1}.

The next step is to derive weak formulations of these problems, in order to apply the finite element method. For the approximation of $\boldsymbol{u}$, we introduce the solution space

$$U = \{\boldsymbol{u} \in H^1(\Omega)^d : \boldsymbol{u}|_{\partial\Omega} = \boldsymbol{u}_D\}, \tag{4.41}$$

and test space

$$V = \{\boldsymbol{v} \in H^1(\Omega)^d : \boldsymbol{v}|_{\partial\Omega} = 0\}. \tag{4.42}$$

For the pressure, we define

$$Q = \begin{cases} L^2(\Omega), & \text{if } \partial\Omega_N \neq \emptyset, \\ L_0^2(\Omega) = \left\{p \in L^2(\Omega) : \int_\Omega p\,\mathrm{d}x = 0\right\}, & \text{if } \partial\Omega_N = \emptyset, \end{cases} \tag{4.43}$$

which will act as both solution space and test space. The constraint on the second space is needed since in the absence of the boundary condition on $\partial\Omega_N$, the pressure would only be determined up to a constant. This is due to the fact that only the gradient of p appears in the PDE.

To derive the weak formulation, (4.39) and (4.40) are multiplied with test variables $\boldsymbol{v} \in V$ and $q \in Q$ and integrated over Ω:

$$\int_\Omega \boldsymbol{v}\cdot\left(\rho^{n+1}\boldsymbol{u}^{n+1} - \rho^n\boldsymbol{u}^n\right)\mathrm{d}x - \theta\Delta t\int_\Omega \boldsymbol{v}\cdot F^{n+1}\,\mathrm{d}x - (1-\theta)\,\Delta t\int_\Omega \boldsymbol{v}\cdot F^n\,\mathrm{d}x = 0, \tag{4.44}$$

$$\int_\Omega q\nabla\cdot\boldsymbol{u}^{n+1}\,\mathrm{d}x = 0. \tag{4.45}$$

The term with the stress tensor σ which is part of F^n and F^{n+1} includes second derivatives of the solution variables. We can transform these integrals using Green's identity, in order to transfer one derivative to the test function. For simplicity, we omit the time index, and consider only the integrand $\boldsymbol{v} \cdot \nabla \cdot \sigma$. Substituting the constitutive relation (3.30), and splitting the integral over the two sub-domains $\Omega_k, k = 1, 2$ gives

$$-\int_\Omega \boldsymbol{v} \cdot \nabla \cdot \sigma \, \mathrm{d}x = \sum_{k=1}^{2} \left(\int_{\Omega_k} \boldsymbol{v} \cdot \nabla \cdot (p\mathcal{I}) \, \mathrm{d}x - 2 \int_{\Omega_k} \mu_k \boldsymbol{v} \cdot \nabla \cdot D(\boldsymbol{u}) \, \mathrm{d}x \right). \quad (4.46)$$

The two terms in the sum are evaluated using the divergence theorem:

$$\begin{aligned} \int_{\Omega_k} \boldsymbol{v} \cdot \nabla \cdot (p\mathcal{I}) \, \mathrm{d}x &= \int_{\Omega_k} \boldsymbol{v} \cdot \nabla p \, \mathrm{d}x \\ &= -\int_{\Omega_k} p \nabla \cdot \boldsymbol{v} \, \mathrm{d}x + \int_{\partial\Omega_k} p \boldsymbol{v} \cdot \boldsymbol{n}_k \, \mathrm{d}s, \quad (4.47) \\ -\int_{\Omega_k} \mu_k \boldsymbol{v} \cdot \nabla \cdot D(\boldsymbol{u}) \, \mathrm{d}x &= -\int_{\partial\Omega_k} \mu_k D(\boldsymbol{u}) \boldsymbol{n}_k \cdot \boldsymbol{v} \, \mathrm{d}s \\ &\quad + \int_{\Omega_k} \mu_k \mathrm{tr}\left(D(\boldsymbol{u}) \cdot D(\boldsymbol{v}) \right) \, \mathrm{d}x, \quad (4.48) \end{aligned}$$

where

$$\mathrm{tr}\left(D(\boldsymbol{u}) \cdot D(\boldsymbol{v}) \right) = \frac{1}{4} \sum_{i=1}^{d} \sum_{j=1}^{d} (\partial_i u_j + \partial_j u_i)(\partial_i v_j + \partial_j v_i)$$

denotes the trace operator.

Using the fact that $v = 0$ on $\partial\Omega_D$, and that $\partial\Omega \cap \overline{\Omega}_k = \partial\Omega_k \setminus \Gamma$, we can reformulate the boundary integrals:

$$\begin{aligned} \sum_{k=1}^{2} \int_{\partial\Omega_k} p \boldsymbol{v} \cdot \boldsymbol{n}_k \, \mathrm{d}s &= \int_{\partial\Omega_N} p \boldsymbol{v} \cdot \boldsymbol{n} \, \mathrm{d}s + \sum_{k=1}^{2} \int_{\Gamma} p \boldsymbol{v} \cdot \boldsymbol{n}_k \, \mathrm{d}s \quad (4.49) \\ -2 \sum_{k=1}^{2} \int_{\partial\Omega_k} \mu_k D(\boldsymbol{u}) \boldsymbol{n}_k \cdot \boldsymbol{v} \, \mathrm{d}s &= -2 \int_{\partial\Omega_N} \mu D(\boldsymbol{u}) \boldsymbol{n} \cdot \boldsymbol{v} \, \mathrm{d}s \\ &\quad - 2 \sum_{k=1}^{2} \int_{\Gamma} \mu_k D(\boldsymbol{u}) \boldsymbol{n}_k \cdot \boldsymbol{v} \, \mathrm{d}s. \quad (4.50) \end{aligned}$$

Summing these two contributions, and substituting $\boldsymbol{n}_1 = -\boldsymbol{n}_2 = \boldsymbol{n}_\Gamma$ gives the following total boundary term:

$$\begin{aligned} \int_{\partial\Omega_N} \left(p\mathcal{I} - 2\mu D(\boldsymbol{u}) \right) \boldsymbol{n} \cdot \boldsymbol{v} \, \mathrm{d}s + \int_{\Gamma} \left[\!\left[\left(p\mathcal{I} - 2\mu D(\boldsymbol{u}) \right) \boldsymbol{n}_\Gamma \right]\!\right]_\Gamma \cdot \boldsymbol{v} \, \mathrm{d}s \\ = -\int_{\partial\Omega_N} \sigma \boldsymbol{n} \cdot \boldsymbol{v} \, \mathrm{d}s - \int_{\Gamma} \left[\!\left[\sigma \boldsymbol{n}_\Gamma \right]\!\right]_\Gamma \cdot \boldsymbol{v} \, \mathrm{d}s, \quad (4.51) \end{aligned}$$

where $[\![\cdot]\!]_\Gamma$ represents the jump over the interface Γ. The first of these terms will be omitted from the weak formulation, which corresponds to imposing the *natural* boundary condition (4.33) at each time-step t^n.

In the second term, the original interface tension force can be substituted for the jump term in the PDE model, using (4.34). At time t^n, the interface condition is then imposed by including in the weak form the following contribution:

$$f_\Gamma^n(\boldsymbol{v}) = -\int_{\Gamma(t^n)} \tau\kappa^n \boldsymbol{n}_\Gamma^n \cdot \boldsymbol{v}\, \mathrm{d}s \tag{4.52}$$

It is convenient to also introduce a set of linear and bilinear forms that represent the other terms in the weak form. We follow the notation in [54] and make the following definitions:

$$m^n(\boldsymbol{u},\boldsymbol{v}) = \int_\Omega \rho^n \boldsymbol{u}\cdot\boldsymbol{v}\,\mathrm{d}x, \qquad (\boldsymbol{u},\boldsymbol{v}) \in U\times V, \tag{4.53}$$

$$a^n(\boldsymbol{u},\boldsymbol{v}) = \int_\Omega 2\mu^n \mathrm{tr}\left(D(\boldsymbol{u})\cdot D(\boldsymbol{v})\right)\,\mathrm{d}x, \quad (\boldsymbol{u},\boldsymbol{v}) \in U\times V, \tag{4.54}$$

$$b(\boldsymbol{u},q) = -\int_\Omega q\nabla\cdot\boldsymbol{u}\,\mathrm{d}x, \qquad (\boldsymbol{u},q) \in U\times Q, \tag{4.55}$$

$$b^*(\boldsymbol{v},p) = -\int_\Omega p\nabla\cdot\boldsymbol{v}\,\mathrm{d}x, \qquad (\boldsymbol{v},p) \in V\times Q, \tag{4.56}$$

$$c^n(\boldsymbol{u},\boldsymbol{w},\boldsymbol{v}) = \int_\Omega \rho^n\left(\boldsymbol{u}\cdot\nabla\boldsymbol{w}\right)\boldsymbol{v}\,\mathrm{d}x, \qquad (\boldsymbol{u},\boldsymbol{w},\boldsymbol{v}) \in U\times U\times V, \tag{4.57}$$

$$f_V^n(\boldsymbol{v}) = \int_\Omega \rho^n \boldsymbol{f}\cdot\boldsymbol{v}\,\mathrm{d}x, \qquad \boldsymbol{v}\in V. \tag{4.58}$$

To shorten the weak formulation, we gather the integral contributions from F^n in the following operator:

$$G^n(\boldsymbol{v}) = b^*(\boldsymbol{v},p^n) + a^n(\boldsymbol{u}^n,\boldsymbol{v}) + c^n(\boldsymbol{u}^n,\boldsymbol{u}^n,\boldsymbol{v}) + f_V^n(\boldsymbol{v}) + f_\Gamma^n(\boldsymbol{v}). \tag{4.59}$$

Using these definitions, (4.44) – (4.45) can be rewritten as follows:

$$m^{n+1}(\boldsymbol{u}^{n+1},\boldsymbol{v}) + \theta\Delta t G^{n+1}(\boldsymbol{v}) - m^n(\boldsymbol{u}^n,\boldsymbol{v}) + (1-\theta)\Delta t G^n(\boldsymbol{v}) = 0, \qquad \forall \boldsymbol{v}\in V, \tag{4.60}$$

$$b(\boldsymbol{u}^{n+1},q) = 0, \qquad \forall q\in Q. \tag{4.61}$$

This is a nonlinear problem due to the term $c(\boldsymbol{u}^{n+1},\boldsymbol{u}^{n+1},\boldsymbol{v})$. Among the many methods for solving such problems, the Newton method stands out for its ability to achieve quadratic convergence. In order to describe this method, we denote the two unknown variables at time t^{n+1} together as $\boldsymbol{w} = \left(\boldsymbol{u}^{n+1}, p^{n+1}\right) \in U\times Q$, and the corresponding test variables as $\boldsymbol{\eta} = (\boldsymbol{v},q) \in W = V\times Q$. The nonlinear weak problem at time t^{n+1} can be expressed in residual form as

$$R(\boldsymbol{w}) = m^{n+1}(\boldsymbol{u}^{n+1},\boldsymbol{v}) + \theta\Delta t G^{n+1}(\boldsymbol{v}) + b(\boldsymbol{u}^{n+1},q) + r(\boldsymbol{u}^n,p^n) = 0, \quad \forall \boldsymbol{\eta}\in W, \tag{4.62}$$

where r contains the terms corresponding to the previous time-step t^n.

The Newton method is an iterative algorithm based on the first-order Taylor expansion of the residual R. Given the approximation $\boldsymbol{w}_k$ at step k, the next approximation $\boldsymbol{w}_{k+1}$ is determined in two steps:

1. Compute a correction by solving the linear system of equations $\nabla R\left(\boldsymbol{w}_k\right) \delta \boldsymbol{w}_k = R\left(\boldsymbol{w}_k\right), \forall \boldsymbol{\eta} \in W$.
2. Update $\boldsymbol{w}_{k+1} = \boldsymbol{w}_k - \delta \boldsymbol{w}_k$.

This procedure is repeated until some convergence criteria, most often related to the norms of R and $\delta \boldsymbol{w}$, are fulfilled.

The first step of the iteration requires solving a linear weak boundary value problem, which can be determined by finding the expression for $\nabla R\left(\boldsymbol{w}\right) \delta \boldsymbol{w}$, given $\boldsymbol{\eta} \in W$:

$$\begin{aligned}
\nabla R\left(\boldsymbol{w}\right) \delta \boldsymbol{w} &= \lim_{\epsilon \to 0} \frac{1}{\epsilon} \left\{ R\left(\boldsymbol{w} + \epsilon \delta \boldsymbol{w}\right) - R\left(\boldsymbol{w}\right) \right\} \\
&= \lim_{\epsilon \to 0} \frac{1}{\epsilon} \Big\{ \epsilon \Big(m^{n+1}(\delta \boldsymbol{u}, \boldsymbol{v}) + b(\delta \boldsymbol{u}, q) + b^*(\boldsymbol{v}, \delta p) \\
&\quad + a^{n+1}(\delta \boldsymbol{u}, \boldsymbol{v}) + c^{n+1}(\boldsymbol{u}, \delta \boldsymbol{u}, \boldsymbol{v}) + c^{n+1}(\delta \boldsymbol{u}, \boldsymbol{u}, \boldsymbol{v}) \Big) + \\
&\quad \epsilon^2 c^{n+1}(\delta \boldsymbol{u}, \delta \boldsymbol{u}, \boldsymbol{v}) \Big\} \\
&= m^{n+1}(\delta \boldsymbol{u}, \boldsymbol{v}) + b(\delta \boldsymbol{u}, q) + b^*(\boldsymbol{v}, \delta p) \\
&\quad + a^{n+1}(\delta \boldsymbol{u}, \boldsymbol{v}) + c^{n+1}(\boldsymbol{u}, \delta \boldsymbol{u}, \boldsymbol{v}) + c^{n+1}(\delta \boldsymbol{u}, \boldsymbol{u}, \boldsymbol{v}) \\
&= d^{n+1}(\delta \boldsymbol{u}, \boldsymbol{v}) + b(\delta \boldsymbol{u}, q) + b^*(\boldsymbol{v}, \delta p),
\end{aligned}$$

where

$$\begin{aligned}
d^{n+1}(\delta \boldsymbol{u}, \boldsymbol{v}) = m^{n+1}(\delta \boldsymbol{u}, \boldsymbol{v}) + a^{n+1}(\delta \boldsymbol{u}, \boldsymbol{v}) + \\
c^{n+1}(\boldsymbol{u}, \delta \boldsymbol{u}, \boldsymbol{v}) + c^{n+1}(\delta \boldsymbol{u}, \boldsymbol{u}, \boldsymbol{v}).
\end{aligned}$$

We split the problem according to the test variables, and arrive at the following linearized weak problem to be solved at step k of the Newton iteration for time t^{n+1}:

$$\text{Seek } (\delta \boldsymbol{u}_k, \delta p_k) \in U \times Q, \text{ such that} \tag{4.63}$$

$$d^{n+1}(\delta \boldsymbol{u}_k, \boldsymbol{v}) + b^*(\boldsymbol{v}, \delta p_k) = R_v(\boldsymbol{w}_k, \boldsymbol{v}), \quad \forall \boldsymbol{v} \in V, \tag{4.64}$$

$$b(\delta \boldsymbol{u}_k, q) = 0, \quad \forall q \in Q, \tag{4.65}$$

where the residual in the first equation is given by

$$\begin{aligned}
R_v(\boldsymbol{w}_k, \boldsymbol{v}) = m^{n+1}(\boldsymbol{u}_k, \boldsymbol{v}) - m^n(\boldsymbol{u}^n, \boldsymbol{v}) + (1-\theta)\Delta t G^n(\boldsymbol{v}) + \\
\theta \Delta t \Big(a^{n+1}(\boldsymbol{u}_k, \boldsymbol{v}) + c^{n+1}(\boldsymbol{u}_k, \boldsymbol{u}_k, \boldsymbol{v}) + \\
f_V^{n+1}(\boldsymbol{v}) + f_\Gamma^{n+1}(\boldsymbol{v}) + b^*(\boldsymbol{v}, p_k) \Big).
\end{aligned} \tag{4.66}$$

The final step is to derive a linear system by introducing the finite element spaces $U_h \subset U$, $V_h \subset V$, $Q_h \subset Q$ with global basis functions $\boldsymbol{\psi}_i \in V_h$, $i = 1, \ldots, N_V$ and $\chi_i \in Q_h$, $i = 1, \ldots, N_Q$. The discrete linearized weak problem is obtained by substituting the discrete solutions $\delta \boldsymbol{u}_k^h = \sum_{j=1}^{N_V} x_j \boldsymbol{\psi}_j$, $\delta p_k^h = \sum_{j=1}^{N_Q} y_j \chi_j$ and testing with the discrete basis functions:

$$\sum_{j=1}^{N_V} x_j d^{n+1}(\boldsymbol{\psi}_j, \boldsymbol{\psi}_i) + \sum_{j=1}^{N_Q} y_j b^*(\boldsymbol{\psi}_i, \chi_j) = R_v(\boldsymbol{w}_k^h, \boldsymbol{\psi}_i), \quad i = 1, \ldots, N_V, \tag{4.67}$$

$$\sum_{j=1}^{N_V} x_j b(\boldsymbol{\psi}_j, \chi_i) = 0, \qquad i = 1, \ldots, N_Q. \tag{4.68}$$

Using block matrix notation, the system can be written as

$$\begin{bmatrix} D & B^T \\ B & 0 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} R_V(\boldsymbol{w}_k^h, \boldsymbol{\psi}_i) \\ 0 \end{bmatrix}, \tag{4.69}$$

with

$$D_{ij} = d^{n+1}(\boldsymbol{\psi}_j, \boldsymbol{\psi}_i), \tag{4.70}$$

and

$$B_{ij} = b(\boldsymbol{\psi}_j, \chi_i). \tag{4.71}$$

The system matrix is indefinite, due to the zero block on the diagonal. The reason for this is that the weak formulation does not correspond to an unconstrained energy minimization problem, as is the case for e. g. the Laplace equation or the least-squares variational formulation that was used for the interface model. Here, we are instead dealing with a constrained minimization of the energy, since the incompressibility condition is imposed as part of the weak form, and is not part of the space. The pressure variable acts as the Lagrange multiplier for this constraint, and the solution will be a saddle point of the corresponding optimization problem.

The use of conforming discrete finite element spaces is not enough to guarantee that the discrete, linearized problem is well-posed. Although conformity will guarantee that the bilinear form d^{n+1} is coercive on $U_h \times V$, the same is not necessarily true for b. Hence, one must choose the spaces so that the following LBB condition is fulfilled:

$$\exists \beta_h > 0 : \inf_{q_h \in Q_h} \sup_{\boldsymbol{v}_h \in V_h} \frac{b(\boldsymbol{v}_h, q_h)}{\|\boldsymbol{v}_h\|_V \|q_h\|_Q} \geq \beta_h. \tag{4.72}$$

It is well-known that this condition is not fulfilled for certain pairs of spaces, e. g. with linear approximations for the velocity and constant or linear approximations for the pressure. A popular choice is

the family of *Taylor-Hood* elements, with degree-k elements for the velocity and degree-$(k-1)$ elements for the pressure, where $k \geq 2$. These elements are known to fulfill condition (4.72) (for a proof with irregular meshes, see [66]). In this work, we use quadrilateral elements with bi-quadratic velocity and bilinear pressure approximations. The mesh used is the same as that for the level set function, but for the flow variables, we do not vary the polynomial degree.

4.5 TRANSFORMATION OF THE INTERFACIAL TENSION FORCE

The weak form of the Navier-Stokes equations is commonly used, and its properties overall are well understood. In this particular model, the surface integral that represents the contribution from the interfacial tension force is the most problematic point.

There are two main drawbacks of the formulation presented in Section 4.4. Firstly, the fact that the curvature $\kappa = \nabla \cdot \left(\frac{\nabla \phi}{|\nabla \phi|} \right)$ appears in the integrand poses a strong regularity requirement on ϕ in order for the integral to be well defined. If standard H^1-conforming finite elements are used for the approximation of ϕ, one can expect that the computation of this quantity will lead to large numerical errors. Secondly, the domain of integration is a time-varying surface that is not available explicitly, which makes the use of standard quadrature methods difficult.

The first drawback is addressed by further weakening the expression, and removing the explicit dependence on the curvature. This procedure makes use of the fact that the combination of curvature and surface normal corresponds exactly to the right-hand side of the *Laplace-Beltrami* equation. For a fixed time t,

$$-\Delta_\Gamma \mathcal{I}_\Gamma(\boldsymbol{x}) = \kappa(\boldsymbol{x}) \boldsymbol{n}_\Gamma(\boldsymbol{x}), \qquad \boldsymbol{x} \in \Gamma. \tag{4.73}$$

Here $\mathcal{I}(\boldsymbol{x}) = \boldsymbol{x}$ is the identity operator on Γ, and Δ_Γ denotes the *Laplace-Beltrami* operator. Similarly to the Laplace operator, we have the identity

$$\Delta_\Gamma f = \nabla_\Gamma \cdot \nabla_\Gamma f \tag{4.74}$$

with the tangential derivative

$$\nabla_\Gamma f = (\mathcal{I} - \boldsymbol{n}_\Gamma \boldsymbol{n}_\Gamma^T) \nabla f, \tag{4.75}$$

and surface divergence

$$\nabla_\Gamma \cdot \boldsymbol{f} = \operatorname{tr}(\nabla_\Gamma \boldsymbol{f}). \tag{4.76}$$

As described in [54], there is a corresponding Green's formula which can be used to replace the contribution from the interfacial tension

force with a weaker integral, whose only dependence on ϕ is via the normal vector:

$$f_\Gamma(\boldsymbol{v}) = -\int_\Gamma \tau\kappa\boldsymbol{n}_\Gamma \cdot \boldsymbol{v}\,\mathrm{d}s \tag{4.77}$$

$$= \int_\Gamma \tau\,\Delta_\Gamma \mathcal{I}_\Gamma(\boldsymbol{x}) \cdot \boldsymbol{v}\,\mathrm{d}s \tag{4.78}$$

$$= -\int_\Gamma \tau\,\nabla_\Gamma \mathcal{I}_\Gamma(\boldsymbol{x}) : \nabla_\Gamma \boldsymbol{v}\,\mathrm{d}s \tag{4.79}$$

$$= -\int_\Gamma \tau\,\mathrm{tr}((\mathcal{I} - \boldsymbol{n}_\Gamma \boldsymbol{n}_\Gamma^T)\nabla \boldsymbol{v})\,\mathrm{d}s. \tag{4.80}$$

This result holds for $\boldsymbol{v} \in H^1(U)$, where U is a neighborhood of Γ.

This transformation of the interfacial tension force has been applied in [54], and a variation of it in [20]. A different approach, based on regularizing the curvature by solving a stabilized problem, was proposed in [131].

For computing the surface integral over Γ, we use a second transformation, which involves the Dirac-measure $\delta_\Gamma(\phi)$ introduced in Section 3.3. By multiplying the integrand with δ_Γ we can replace the surface integral by a volume integral:

$$\int_{\Gamma(t)} G(\boldsymbol{x}, t)\,\mathrm{d}s = \int_\Omega \tilde{G}(\boldsymbol{x}, t)\delta_\Gamma(\phi(\boldsymbol{x}, t))\,\mathrm{d}x, \tag{4.81}$$

where $\tilde{G}$ is an appropriate extension of G from Γ to Ω. The natural approach is to extend the normals to Ω via $\boldsymbol{n}_\Gamma(x, t) = \frac{\nabla\phi(x,t)}{|\nabla\phi(x,t)|}$, $\forall \boldsymbol{x} \in \Omega$. In summary, we have the following expression for the contribution from the interfacial tension force:

$$f_\Gamma(\boldsymbol{v}) = -\int_\Omega \tau\mathrm{tr}\left(\left(\mathcal{I} - \boldsymbol{n}_\Gamma \boldsymbol{n}_\Gamma^T\right)\nabla \boldsymbol{v}\right)\delta_\Gamma(\phi(\boldsymbol{x}, t))\,\mathrm{d}x. \tag{4.82}$$

A discrete approximation of δ_Γ will be described in Section 4.6.

4.6 APPROXIMATION OF HEAVISIDE AND DIRAC FUNCTIONS

The weak formulation of the flow problem involves the use of generalized functions in two respects. The first concerns the material parameters ρ and μ, which are piecewise constant functions in each sub-domain Ω_k, $k = 1, 2$. The global functions were introduced in Section 3.3 as

$$\rho(\boldsymbol{x}, t) = \rho_1 + H(\phi(\boldsymbol{x}, t))\,(\rho_2 - \rho_1) \tag{4.83}$$

$$\mu(\boldsymbol{x}, t) = \mu_1 + H(\phi(\boldsymbol{x}, t))\,(\mu_2 - \mu_1)\,, \tag{4.84}$$

where $H(s)$ denotes the Heaviside function. In much of the literature dealing with two-component flows, one replaces H with a smoothened approximation H_ϵ (see e. g. [95, 127]). The reason for this is to improve both the stability of the solution process, and its accuracy. The former aspect is of subordinate importance when a finite

element method is used, since the weak formulation makes it possible to handle discontinuous coefficients robustly. Due to the use of numerical quadrature, accuracy is still a concern, however, and it is therefore advisable to use a smooth approximation also in finite element computations. The choice of H_ϵ involves balancing the resulting approximation error with the quadrature error. This problem is discussed in [131], where the approximation error is analyzed in terms of moments of H_ϵ. In that work, a polynomial approximation is used, to enable the use of exact quadrature. In this work, we follow the approach in [95, 127], and let

$$H_\epsilon(s) = \begin{cases} 1, & s > \epsilon, \\ \frac{1}{2}\left(1 + \frac{s}{\epsilon} + \frac{1}{\pi}\sin\left(\frac{\pi s}{\epsilon}\right)\right), & |s| \leq \epsilon, \\ 0, & s < -\epsilon. \end{cases} \tag{4.85}$$

Although this choice will incur both approximation and quadrature errors, it is very popular in the literature for two-phase flows, and has been applied successfully in e. g. [32, 33, 141]. The parameter ϵ, which governs the size of the interval over which the function is smoothened out, is typically chosen approximately equal to the cell size.

The second respect in which generalized functions are used is formal use of the Dirac distribution $\delta(s)$ in the substitution of the surface integral over Γ with the volume integral over Ω that was applied to the contribution from the interfacial tension force in Section 4.5. This substitution was introduced to solve the problem that the time-varying $\Gamma(t)$ is only known implicitly as the zero level of the level set function ϕ. A common alternative approach (see e. g. [54, 131]) is to construct an explicit approximation of Γ, and performing numerical quadrature using this approximation. Such reconstructions are, however, cumbersome to perform, especially in the setting of the present work, where locally varying polynomial degrees and quadrilateral elements are used. Therefore, just as for the Heaviside function, we use instead a regularized approximation δ_α of the Dirac distribution which is based on a *mollifier function*:

$$\delta_\alpha(s) = \begin{cases} \frac{1}{\beta}\exp\left(-\frac{\alpha^2}{s^2-\alpha^2}\right), & |s| \leq \alpha, \\ 0, & |s| > \alpha, \end{cases} \tag{4.86}$$

where

$$\beta = \int_{\mathbb{R}} \exp\left(-\frac{\alpha^2}{y^2-\alpha^2}\right) \mathrm{d}y. \tag{4.87}$$

The normalization factor β is included to ensure that $\int_{\mathbb{R}} \delta_\alpha(s)\,\mathrm{d}s = 1$. Using δ_α, the surface integrals over Γ can be approximated by

$$\int_{\Gamma(t)} G(\boldsymbol{x}, t)\,\mathrm{d}s = \int_\Omega \tilde{G}(\boldsymbol{x}, t)\delta_\alpha(\phi(\boldsymbol{x}, t))\,\mathrm{d}x, \tag{4.88}$$

where $\tilde{G}$ is an extension of G from Γ to Ω.

Other choices of δ_α are also possible. In particular, one could make it coincide with the derivative of the regularized Heaviside function, in order make the two approximations consistent, and reduce the number of discretization parameters. In the present case, there is no direct connection between the uses of H_ϵ and those of δ_α, so this choice would be for convenience only.

4.7 SUMMARY OF THE SOLUTION PROCEDURE

To conclude this chapter, we summarize the solution procedure for the coupled model. Overall, it contains three nested iterations, which are outlined schematically in Fig. 4.2. The adaptive iteration solves the coupled time-dependent model with successively larger approximation spaces V_h^r, Q_h^r, S_{hp}^r. The extension of the approximation space S_{hp} for level set function is based on the hp-adaptive algorithm described in the next chapter, whereas the spaces V_h and Q_h for the velocity and pressure approximations use Q_2/Q_1 elements on the same mesh as for the level set function.

The evolution of the variables in time is solved using the time-stepping procedures for the two models. In each time-step n, first the semi-discretized flow model (4.60) – (4.61) is solved to yield the updated velocity and pressure, and then the level set function is updated by solving the interface model (4.26).

Whereas the interface problem is simply a linear system, which can be solved e. g. using the CG method, the flow problem is nonlinear. As described in Section 4.4, this problem is linearized using the Newton method. The corresponding inner iteration computes in each step k a correction $\delta \boldsymbol{w}_{k+1}$ is computed by solving the linear boundary value problem (4.63), whereafter the approximation $\boldsymbol{w}_k$ is updated by adding $\delta \boldsymbol{w}_{k+1}$.

The corresponding program is naturally more complex than what Fig. 4.2 indicates, since it also has to deal with initialization of data structures and data output, e. g. in the form of visualizations of the various variables. The main iterative structure in the diagram does however capture the core logic of the program.

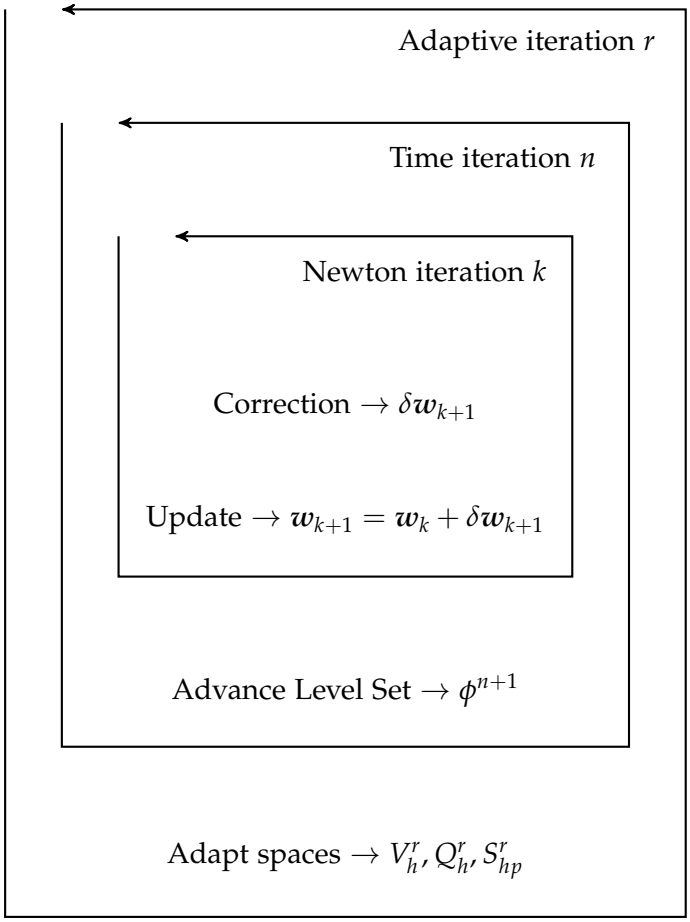


Figure 4.2: Diagram summarizing the nested iterations in the solution procedure.

5

HP-FEM

This chapter describes adaptation of the discrete solution space using hp-FEM techniques. The first part of the chapter (Sections 5.1 to 5.3) gives an overview of the theoretical aspects of successive space extension and error control via hp-adaptive methods. Section 5.4 describes the key aspects of the implementation of support for hp-FEM in the software package HiFlow^3. This is followed by Section 5.5 which presents the results of some numerical experiments conducted to verify the correctness of the implementation. The chapter is concluded by a discussion in Section 5.6 of some work concerning the use of hp-FEM in the context of discontinuous Galerkin formulations.

5.1 SUCCESSIVE SPACE EXTENSION

Section 4.2 described in an abstract setting how the finite element method can be used to discretize a boundary-value problem for a partial differential equation. One of the central decisions to be made in the definition of a FEM is the choice of the discrete approximation spaces U_h and V_h for the solution and test functions, respectively. This choice directly determines the quality of the approximate solution, and strongly influences the computational cost of solving for it. In general, the more accurate the computed solution has to be, the larger U_h (and consequently V_h) has to be chosen.

Most often, it is not possible to determine *a priori* exactly how well the unknown solution u can be approximated in a given space, and it is therefore common to solve the problem successively with a sequence of spaces $U_h^1, U_h^2, \ldots$, and try to estimate the size of the error for each computed solution, which will (hopefully) converge toward zero.

The most common method of constructing the sequence of spaces is to increase the number of elements in each step, either by generating a new mesh (*remeshing*), or by splitting the existing elements into sub-elements (*refinement*). Although remeshing is frequently used in practice, it has some important drawbacks. Computing a mesh for a geometric domain is often computationally expensive, and sometimes requires manual intervention to ensure the quality of the result. With remeshing, this procedure has to be repeated several times instead of being performed just once before the computation. Furthermore, remeshing requires complex transfer operations of discrete functions between the spaces. When using the term *h-FEM*, most authors therefore only include methods based on refinement.

The simplest h-FEM strategy is *global* or *uniform* refinement, where all the cells in the mesh are refined in an identical way. The number of elements, and hence the size of U_h^i, will then grow exponentially, which leads to intractable problems very quickly. For instance, assume that each cell in a two-dimensional computation is split into four cells with each refinement, and that U_h^1 corresponds to a mesh with only one cell. After ten refinement steps, the mesh will have over one million elements, and after fifteen steps the this number will exceed one billion! Most often, only a few steps can be therefore performed before the computational costs become too high. For some problems, a few refinement steps might be enough; but in cases where the error converges slowly, or where a highly accurate solution is required, uniform refinement will in general be too expensive.

More advanced h-FEM strategies are *local* in the sense that they will only refine a subset of the elements, and try to maximize the gain in accuracy for each step. Many problems contain localized features which need to be resolved on the mesh before an accurate solution is obtained. Away from these features, a coarser mesh can be used without increasing the global error of the solution very much.

Another, less commonly used, method of increasing the size of U_h is to raise the polynomial degree p_K of the shape functions. When performed for a fixed mesh, this is referred to as *p-FEM*. Again, it is possible to increase the degree globally for all cells, or locally for just a subset. The use of high-degree elements was pioneered by Szabó and Babuška in the late 1970s [13, 128].

Methods which exploit the possibilities of both h-FEM and p-FEM are categorized as *hp-FEM*. This approach grew out of the work on p-FEM in order to compensate for its weaknesses, which will be described further in Section 5.2. An important contribution that motivated much of the work that followed was a series of papers [55–57] which analyzed in detail the convergence rates of h-, p- and hp-FEM for an elliptic one-dimensional problem with a special type of singular solutions. It was proven that if the combination of mesh refinement and polynomial degrees is chosen in a particular way, then it is possible to obtain exponential convergence toward the correct solution. Exponential convergence is also possible with p-FEM, but only when the solution is analytic, and the main motivation of using hp-FEM is to recover this property for a larger class of problems.

Another approach related to p-FEM is the *spectral element method*, first described by Patera [102]. This is a localized version of the *spectral method*, which in turn was developed by Orszag at the end of the 1960s [97]. Although developed independently, the spectral element method shares with p-FEM the use of unstructured meshes and basis functions whose supports are localized to one or a few cells. The original method uses a special Lagrange basis that interpolates the Chebyshev nodes in each cell, and can be considered a special class

of p-FEM methods. Other methods that use Lagrange bases with special choices of nodes are also often referred to as spectral element methods. An extension of this method to incorporate hp-FEM techniques is described in [79].

Finally, it should be noted that there are other methods of modifying the approximation space to obtain better approximations. In r-adaptive FEM (*r* stands for *relocation*), one modifies the geometry of the computational mesh instead of its topology, in order to obtain a finer resolution where the solution is varying rapidly. An example of its use can be found in [30]. Another recent approach is the use of k-FEM, where the k corresponds to the order of continuity of the global finite element basis. In the context of *isogeometric analysis*, pioneered in [72], one uses B-splines or NURBS as basis functions, instead of the standard C^0 piecewise polynomials. This has several advantages, including better integration with CAD-tools. The use of these basis functions also makes it possible to vary the order of continuity locally, and in this way be able to approximate functions that have both smooth parts and irregular features efficiently.

5.2 CHARACTERISTICS OF H-, P- AND HP-FEM

In order to effectively design an hp-FEM space extension strategy, one must understand the effect of mesh refinement and the polynomial degree on the accuracy of the approximate solution. *A priori* estimates of the difference between the exact solution to a problem and the elements in the discrete solution space U_h are important analytical tools in this context. In contrast to *a posteriori* estimates, the results in a priori analysis can depend on the unknown solution itself, which means that the expressions can in general not be evaluated. They are, however, useful for establishing the convergence rate that can asymptotically be expected, as U_h is extended in some way.

As an example, let us consider a boundary value problem for the Poisson equation on a domain Ω:

$$-\Delta u = f \qquad \text{in } \Omega, \tag{5.1}$$

$$u = 0 \qquad \text{on } \partial\Omega. \tag{5.2}$$

A common variational formulation for this problem is as follows:

Seek $u \in H_0^1(\Omega)$ such that

$$\int_\Omega \nabla u \cdot \nabla \psi \, \mathrm{d}x = \int_\Omega f \psi \, \mathrm{d}x, \qquad \forall \psi \in H_0^1(\Omega), \tag{5.3}$$

where $H_0^1(\Omega) = \{u \in H^1(\Omega) : u|_{\partial\Omega} = 0\}$. In this case, the solution and test spaces are the same, and a conforming discretized problem can be constructed by choosing $U_h = V_h \subset H_0^1(\Omega)$:

Seek $u_h \in V_h$ such that

$$\int_\Omega \nabla u_h \cdot \nabla \psi_h \,\mathrm{d}x = \int_\Omega f \psi_h \,\mathrm{d}x, \qquad \forall \psi_h \in V_h. \tag{5.4}$$

We now consider a sequence of finite element spaces V_h which correspond to a sequence of *quasiuniform* meshes $\{\mathcal{M}_h\}$ with cell size parameter $h = \max_{K\in\mathcal{M}_h} h_K$ and polynomial degree p for all elements. Here h_K is the diameter of the cell K. A quasiuniform family of meshes satisfies $h \leq \tau h_K$ for all K with a constant τ which is independent of the mesh. It was proven in [12] that if the exact solution $u \in H^r(\Omega)$, the following result holds for the error in the H^1-norm:

$$\|u - u_h\|_1 \leq C \frac{h^{\min(p,r-1)}}{p^{r-1}} \|u\|_r . \tag{5.5}$$

This estimate can be used to analyze the convergence rate for h-FEM and p-FEM procedures in the quasiuniform setting. To compare the two methodologies, we express the convergence in terms of the number of degrees of freedom N. For h-FEM with uniform refinement, N is proportional to the number of cells, which grows as h^{-d} if $\Omega \subset \mathbb{R}^d$, which yields $N \in \mathcal{O}(h^{-d})$. For p-FEM, there are p^d degrees of freedom per cell, so $N \in \mathcal{O}(p^d)$. Substituting into 5.5 gives the estimates

$$\|u - u_h\|_1 \leq C N^{-\min(p,r-1)/d} \|u\|_r , \tag{5.6}$$

for h-FEM and

$$\|u - u_h\|_1 \leq C N^{-(\min(p,r-1)+(r-1))/d} \|u\|_r , \tag{5.7}$$

for p-FEM.

For the case of h-FEM with uniform refinement, the rate of convergence in terms of N will be determined by the fixed polynomial degree p, as long as $p \leq r - 1$. When the solution is not regular enough to lie in $H^{p+1}(\Omega)$, then raising the polynomial degree will not yield a higher convergence rate. With p-FEM, the rate of convergence increases as p is raised successively, until the limit imposed by the regularity of the solution is reached. If the solution is completely regular, the estimate holds for arbitrary large values of r. For the estimate to be useful, however, $\|u\|_r$ cannot grow too fast with r. Under the additional assumption that u is analytic, it can be shown that this does not happen, and that exponential convergence is achieved (a constructive proof for $d = 1$ can be found in [115, Chapter 3]):

$$\|u - u_h\|_1 \leq a \exp(-b N^c). \tag{5.8}$$

In contrast, extension processes based on h-FEM, will always be algebraic convergence rates. This qualitative difference in the convergence between the two paradigms can have a significant impact on the performance of the method, especially when a highly accurate solution is required.

For many problems occurring in practice, the solution is not everywhere smooth, however, and the maximum value of r is in general small. This can occur for several different reasons. The regularity of u depends on the regularity of the functions that constitute the right-hand side of the PDE, its coefficients, and the boundary data. In dimensions larger than one, the shape of the domain also plays an important role. Reentrant corners of non-convex domains, or degenerate boundaries such as slits will limit the regularity.

Furthermore, many problems exhibit solutions that, although being regular to a high degree, have very large gradients. The archetype situation are the *boundary layers* and that occur near the boundary in advection-dominated flows. In this case, reaching the asymptotic range of convergence, in which the convergence rates given by a priori estimates will be observed, will require a fine enough mesh to resolve the sharp variations in the solution.

In most cases, the solution is only irregular in some parts of the domain, and is more or less smooth away from these parts. There are certainly functions which are globally irregular, such as the nowhere differentiable Weierstrass function [76], but these rarely occur as solutions to the partial differential equations that are used to model physical processes. It is therefore reasonable to assume that if one combines mesh refinement in those areas where the solution is irregular with elevation of the polynomial degree where it is smooth, one can hope to achieve the fast convergence of p-FEM, even if the solution is not globally smooth. This idea forms the basis of most hp-FEM strategies.

Much work has been devoted to analyzing whether exponential convergence can be achieved with hp-FEM even when the solution is not globally analytic, and there are several positive results. In [55–57], the situation for one-dimensional problems whose solution has a singularity of type x^α with $0.5 < \alpha < 1$ was analyzed for different h-, p- and hp-methods. It was found that exponential convergence can be achieved in this case by using meshes for which the cell size decreases geometrically toward $x = 0$. The optimal ratio for these meshes, as well as the optimal distribution of polynomial degrees, were also determined.

It is also known that for elliptic problems with sufficiently regular data, it is possible to achieve exponential convergence with hp-FEM. This was first analyzed in [58, 59], which based the analysis on special *countably normed* function spaces. It was later shown (see e. g. [9]

for the two-dimensional case) that solutions to elliptic problems with piecewise analytic data belong to these spaces.

Apart from the possibility of achieving if not exponential, then at least very fast convergence, the use of p-FEM and hp-FEM also has further advantages over pure h-FEM for certain problems. In many applications there is a parameter that has a critical value such that the accuracy of low-order solutions can become severely limited when this value is approached. This phenomenon, called *locking*, is well-known in linear elastic models of structural mechanics, where it can be observed when the value of the Poisson's ratio is close to 0.5, which corresponds to a nearly incompressible material. Performing mesh refinement with $p \leq 4$ will only yield very slow convergence in this case [14]. The use of p-FEM, on the other hand, remains robust with respect to the Poisson's ratio. Similar locking effects are also possible for problems in other domains whose character changes as a parameter approaches a critical value; and the use of p-FEM instead of h-FEM is useful in those cases as well.

Oppositely, there are cases in which the use of h-FEM is necessary to overcome weaknesses of p-FEM. The latter is very sensitive to *pollution* effects, where e. g. a singularity in the boundary data will influence the solution not only in the direct vicinity, but also further away. If p-FEM is used on a coarse mesh in this case, the error due to the singularity will propagate over a large area. Raising the polynomial degree will not be able to diminish this error propagation, whereas refining around the singularity will.

By combining localized h-FEM and p-FEM techniques, one can hope to be able to resolve problems that exhibit locally irregular solutions, locking and pollution errors, if the enlargement process can be controlled to refine close to irregularities and increase the polynomial degree in those areas where the solution is smooth. For synthetically constructed problems, it is sometimes possible to determine an effective enlargement process beforehand. More complex problems in science and engineering will require either manual intervention or an *adaptive* control algorithm to steer the hp-FEM enlargement. The former is generally difficult and expensive, which is why the latter is to be preferred whenever possible. This topic will be discussed further in Section 5.3.

5.3 ADAPTIVE ALGORITHMS

Now that we have established the strengths and weaknesses of extension of the discrete space via h-FEM and p-FEM, respectively, we turn to the question of how the generation of the sequence of approximation spaces can be automatized. *Adaptive* methods use algorithms which take into account the characteristics of the problem and the computed approximations to extend the space.

5.3.1 *Adaptive Strategies for h-FEM*

For pure h-FEM extension, there is a large body of literature on adaptive methods based on *a posteriori* error estimation. The basic idea is to decide where to refine in each step by approximating the local contribution to the error from each cell K with an *error indicator* η_K. The mesh is then refined only where η_K is large. In addition to the large number of articles devoted to the construction and analysis of error indicators (see e. g. [10, 11, 19, 135]), two good overviews of the subject of *a posteriori* error estimation are presented in the books [3] and [136].

It goes without saying that the error indicator should be related to the actual error. This relation can be characterized through two properties, which should be taken into account when evaluating an error indicator η_K in the context of a specific problem. A *reliable* error indicator η_K satisfies

$$\|u - u_h\|_*^2 \leq C \sum_{K \in \mathcal{M}_h} \eta_K^2, \tag{5.9}$$

with some constant $C > 0$. The norm $\|\cdot\|_*$ is problem-dependent. This ensures that the global error is not greatly underestimated.

The reversed inequality is associated to the idea of efficient error indicators. For a *globally efficient* error indicator it holds that

$$\exists\, C : \sum_{K \in \mathcal{M}_h} \eta_K^2 \leq C \|u - u_h\|_*^2 \tag{5.10}$$

and a *locally efficient* error indicator satisfies

$$\exists\, C : \eta_K^2 \leq C \|u - u_h\|_{K,*}^2, \quad \forall K. \tag{5.11}$$

where $\|\cdot\|_{K,*}$ is a local norm related to cell K.

Reliability is usually seen as an absolute requirement, whereas efficiency is not always demanded. When the error indicator is used for driving an adaptive algorithm, global reliability and efficiency together assure that the error indicator will decrease at the same rate as the real error, which prevents exceedingly pessimistic estimation that causes the algorithm to require too many steps to terminate. Local efficiency ensures that a large value of the indicator on a cell implies that the error is also large on that cell, i. e. it prevents false positive indications.

It is generally not possible to prevent false negative indications, due to the fact that the local error is generally influenced by processes on the entire domain. These effects cannot be captured in a local error indicator η, which uses information only from a small neighborhood of K. Hence, the local error can be large although the error indicator is small.

Based on the error indicator, the adaptive algorithm chooses a set $\mathcal{R}$ of elements that should be refined. Perhaps the simplest such

marking strategy chooses all elements whose error indicator is within some fraction θ of the maximum error indicator:

$$\mathcal{R} = \{K : \eta_K \geq \theta \max_K \eta\}. \tag{5.12}$$

The performance of this *Maximum Strategy* depends strongly on the choice of θ. If θ is small, then many cells will be refined, which yields a very large approximate space, and thereby large computational costs. If θ is large, then only a few cells will be refined, but more adaptive steps might be required until the desired tolerance is reached. From experience, values of θ between 0.7 and 0.9 generally yield the best results.

A more sophisticated method is the *Fixed Energy Fraction* or *Dörfler* strategy, which was proposed in [42]. In this case, one chooses $\mathcal{R}$ to be the smallest set such that

$$\sum_{K\in\mathcal{R}} \eta_K^2 \geq \theta^2 \sum_{K\in\mathcal{M}_h} \eta_K^2. \tag{5.13}$$

Again, $\theta \in (0, 1)$ is a user-defined parameter.

It was shown in [42] that this method is convergent, and furthermore it was established in [126] that it is asymptotically of optimal complexity in the sense that the number of operations necessary to obtain an approximate solution with error in the energy norm smaller than some tolerance τ grows as $\mathcal{O}(\tau^{-1/s})$, where s is the asymptotic rate of convergence of the best possible approximation. The setting was restricted to approximating the Poisson equation with linear finite elements in two dimensions in [126], but it has since been generalized in several directions, including non-conforming FEM for mixed formulations [23], and indefinite, non-symmetric elliptic problems [31]. The question of convergence and optimal complexity of adaptive methods remains an active area of research, as there are several settings for which these properties have not yet been established.

Most adaptive methods focus on reduction of the error in a global norm, such as the L^2-, H^1- or energy norm. In practice, however, the aim of a computation is often to obtain an accurate computation of some derived quantity. Consequently, it can be desirable to adapt the mesh to reduce the error in such a *quantity of interest*. This is the idea behind *Goal-oriented adaptivity*, where the quantity of interest is expressed as a linear functional $\mathcal{J}(\cdot)$, and the corresponding error to be minimized is taken to be $|\mathcal{J}(u) - \mathcal{J}(u_h)|$. The dominating computational method to solve such problems is the *Dual Weighted-Residual* method, which is described in [24]. This uses a posteriori error indicators computed by solving a dual problem. Much of the early work on the use of duality methods for a posteriori error indicators was developed by Eriksson and Johnson (see the overview [45] and references therein). A recent application of this technique can be found in [22].

5.3.2 *Adaptive Strategies for hp-FEM*

Most existing approaches to constructing adaptive hp-FEM methods are extensions of existing h-FEM algorithms. The possibility of making a choice between mesh refinement and elevation of the polynomial degree brings a larger freedom to the adaptive process, and therefore requires more information for making decisions. An hp-FEM algorithm has to decide not only where to extend the space, but also how to do it. Existing h-FEM methods based on a posteriori error indicators can be used to answer the first question, but additional consideration is needed to answer the second.

An extensive survey of different hp-adaptive methods is available in [90], and here we only outline the basic ideas. One direction that is followed by many authors is to base the decision between cell refinement and degree elevation on some estimation of the local regularity of the solution. Examples include the methods presented in [83] and [71], which use the coefficients of the approximate solution expanded in a basis of Legendre polynomials to extract information about the local regularity of the solution.

Another method of obtaining information about the regularity is to solve local problems with increased polynomial degrees on each cell, as proposed in [4]. A related approach, which was proposed in [109], is to use separate a posteriori estimates of the error reduction corresponding to mesh refinement and degree elevation. The a posteriori estimates in that work are again computed by solving local problems on each cell for the enriched spaces.

A different school of thought is represented by Demkowicz and his collaborators (see e. g. [37, 38, 40] and references therein). They propose the use of a *reference solution* computed in a discrete space in which the mesh has been globally refined, and the polynomial degree has been increased on each cell, as a point of reference for determining the next adaptation step. For the different available options of cell refinement and degree elevation, the error decrease rate relative to the increase in the number of unknowns of that enlargement can be computed using the difference between the reference solution and its interpolation on the modified element as error measure. Variants of this method have also been described in [121, 124].

The use of a reference solution avoids the need of theoretically derived error estimators, which might not be available for all problems. Since it bases the adaptive control on an estimation of the error function and not just one or a few derived scalar indicators, one can expect to obtain better approximation spaces this way. The drawback is the high cost of computing the reference solution, which can be mitigated to some degree using multigrid techniques.

In later parts of this work, we will use an hp-adaptive strategy developed by Melenk and Wohlmuth in [88]. They make use of an a

```
for each cell K {
  if (η_K^2 > θ η̄_K^2) {
    if (η_K^2 < π_K^2) {
      increase p_K
      π_K^2 = γ_p η_K^2
    } else {
      refine cell K → children K_c
      π_{K_c}^2 = (1/4) γ_h (1/2)^{2 p_K} η_K^2
    }
  } else {
    π_K^2 = γ_n η_K^2
  }
}
```

Figure 5.1: The hp-adaptive algorithm described by Melenk and Wohlmuth.

posteriori indicator η_K for the error, and make the choice between local cell refinement or degree elevation by comparing η_K with a *predicted* error indicator π_K. This prediction corresponds to the error that would be expected based on the previous action taken under the assumption that the solution is locally smooth. If the error indicator is smaller than the predicted error, this is an indication that the solution is really locally smooth, and accordingly the polynomial degree is increased. Otherwise, the cell is refined.

The algorithm, which we will call the *Melenk-Wohlmuth Strategy*, is outlined in Figure 5.1 for the case of quadrilateral elements. It first tests for each cell K if its error indicator is larger than a certain fraction θ of the squared mean indicated error $\bar{\eta}_K^2$. If this first test passes, the error is large enough that the space should be extended locally. The second test between the error indicator and the predicted error indicator determines whether the polynomial degree should be increased, or whether the cell should be refined. In both cases, the predicted error of the cell K or its new children K_c is computed from the current error indicator with formulas that are motivated by a priori analytical arguments. Apart from the threshold coefficient θ, the algorithm also contains the user-defined parameters $\gamma_p, \gamma_h, \gamma_n$, which control how the predicted error indicator is weighted for the case of degree elevation, cell refinement, or no action, respectively.

The search for efficient algorithms for controlling adaptation with hp-FEM is still a field of active research, and none of the methods listed above has been demonstrated to be the ultimate answer. An extensive comparison of the methods in terms of convergence rates for a set of benchmark problems has been presented in [89]. This report also includes a brief comparison of their efficiency in terms of computational cost.

5.4 HP-FEM IMPLEMENTATION IN HIFLOW^3

Given the advantages that the use of an hp-adaptive method can bring, one might expect these techniques to be broadly employed. However, the fact is that support for hp-FEM is only available in a small number of software packages, most of which have been developed in connection with academic research projects. Among the popular commercial applications that are used for the vast majority of FEM computations, the support for hp-adaptive FEM seems to be non-existent, although research in this area has been ongoing for the past thirty years. Whereas advanced meshing techniques for h-adaptivity are commonly available, there is little emphasis on the use of higher-order elements to increase convergence rates and efficiently obtain accurate solutions.

This fact can to some extent be explained by a low demand among users of FEM software for highly accurate numerical solutions, since the relative errors arising from the physical and geometrical modeling will often be on the order of 10^{-2} or 10^{-3} at best. Another reason is probably a lack of familiarity with higher-order methods and their benefits. Since implementing and applying higher-order methods in general and hp-adaptive methods in particular increases the complexity of both writing and using the software, the choice of remaining with standard low-order FEM is understandable.

One of the goals of this work is to show that implementing and using hp-FEM does not have to be excessively complicated. The remainder of this section describes an implementation of the required functionality for performing hp-adaptive FEM computations with quadrilateral meshes with one level of irregularity, that was created by the author as part of the FEM library HiFlow^3 [6, 7, 65, 67, 108]. The implementation builds on the work of Demkowicz and Šolín [37, 38, 123].

5.4.1 *Existing Software*

There have been several efforts to develop solvers and libraries that use hp-FEM. Most of these projects have originated in academic institutions, and hence the user base can be assumed to have been rather small and highly specialized. Bringing the use of adaptive hp-FEM techniques to the larger community of engineers and scientists in fields outside of mathematics remains a challenge.

The only commercial development that the author is aware of is the *PHLEX* kernel, and associated solvers, originally developed at the former Computational Mechanics Company founded by J.T. Oden, and later acquired by Altair Engineering. It is not clear to what extent hp-adaptivity is also available in the commercial HyperWorks products.

Name	Principal Authors	Version	Reference
2Dhp90, 3Dhp90	Demkowicz, Rachowicz, Pardo, et al.	N/A	[39]
Concepts	Frauenfelder, Lage	v2.1.1 2006	[49]
deal.II	Bangerth, Hartmann, Kanschat, Keyser-Herold (hp-FEM)	v7.1 2011	[16, 17]
DUNE-FEM	Dedner, Klöfkorn, Nolte	v2.1 2011	[36]
Hermes	Šolín et al.	v1.0 2011	[121, 123]
HiFlow^3	Heuveline et al.	v1.2 2012	[6, 7]
hpGEM	van der Vegt et al.	v1.0.1 2010	[103]
libMesh	Kirk et al.	v0.7.3 2012	[80]
NGSolve	Schöberl	v4.9.13 2010	
Nektar++	Kirby, Sherwin, et al.	v3.1 2012	[138]

Table 5.1: Existing FEM software with support for hp-FEM.

Among the academic projects, many of these are available as open source projects. Table 5.1 lists the software packages providing support for or using hp-FEM that are known to the author.

The goals of these projects vary greatly, as do the capabilities and quality of the software packages. The work presented in this thesis has been developed using the HiFlow^3 library, which aims to provide general-purpose tools for developing high-performance finite element solvers. The choice to implement support for hp-FEM in this package, instead of using one of the existing codes, was based on the desire to integrate this implementation with the other features of HiFlow^3, including the support for computing on a wide range of parallel computer architectures, ranging from graphic cards (GPU) to large-scale clusters.

The rest of this section describes the extensions of the HiFlow^3 library that provide support for hp-FEM.

5.4.2 *Lobatto Shape Functions*

The most basic component of a finite element method is the definition of the local basis functions or shape functions. For low-order finite elements, the standard approach is to define a set of equidistant nodes $\boldsymbol{\xi}_i$ on the reference cell, and use the associated Lagrange interpolation polynomials as the shape functions. The local degrees of freedom of a function f can then be evaluated simply as $\sigma_i(f) = f(\boldsymbol{\xi}_i)$.

This choice has some drawbacks for defining elements with high polynomial degrees. It is well known that high-degree interpolation using equidistant nodes is an ill-conditioned problem: this is known as Runge's phenomenon. This representation of discrete functions using the Lagrange interpolation polynomials will therefore tend to introduce larger numerical errors, especially toward the boundary of each cell.

Another drawback of the equidistant Lagrange shape functions is that their use will lead to ill-conditioned system matrices as the polynomial degree is increased. It can be shown (see [94]) that for the Laplace operator the condition number grows as $\mathcal{O}(2^{2p})$ with the polynomial degree with these basis functions. For operators coming from other problems the equidistant Lagrange shape functions also lead to ill-conditioned systems [107]. A numerical illustration of this can be found later in this section.

One way to mitigate these problems is to retain the interpolating Lagrange functions, but choose a non-uniform distribution of the nodes. This approach is followed in for instance [29, 87, 107]. It has the advantage that the intuitive interpretation of the degree of freedom functionals being values of the represented function at a given set of nodes is retained. The question of how to choose these nodes optimally for elements in two and three dimensions is an area of active research. A

popular choice is to use the *Fekete* nodes, which are those nodes that maximize the determinant of the Vandermonde matrix for a given geometry and polynomial degree. It has been shown in [27] that on the tensor-product cells the Fekete nodes are exactly the zeros of the Gauss-Lobatto polynomials $(x^2-1)\ell_j'(x)$, which are also used for quadrature. That this basis leads to more a more moderate growth ($\mathcal{O}(p^3)$ or $\mathcal{O}(p^4(1+\log p))$ depending on the exact definition) of the condition numbers is shown in [86].

A different approach is to abandon the requirement of interpolatory basis functions, and turn to so-called *modal* (as opposed to *nodal*) basis functions. Apart from being able to avoid the numerical problems mentioned above, an added advantage of choosing non-interpolatory shape functions is the possibility to construct *hierarchical bases*, for which the set of local shape functions for degree p is a subset of those for degree $p+1$. Hence, there will be polynomials of all degrees from 1 to p, whereas for the Lagrange basis of degree p, all polynomials will be of degree p. This property of hierarchical bases makes it possible to increase and decrease the polynomial degree of an element simply by adding or removing a part of its basis, which we shall see is advantageous for instance for imposing continuity constraints.

In this work, we have chosen to use a hierarchical basis consisting of tensor products of the *integrated Legendre* polynomials defined on the reference element $\hat{K} = [-1,1]$. This construction and similar choices for other reference elements have been advocated by several authors who work with hp-FEM. In particular, we have followed the approach taken in [37, 38, 123] closely.

We begin by defining the one-dimensional integrated Legendre or *Lobatto* polynomials for $t \in [-1,1]$:

Definition 5. *Lobatto Polynomials*

$$\ell_0(t) = \frac{1}{2}(1-t), \tag{5.14}$$

$$\ell_1(t) = \frac{1}{2}(1+t), \tag{5.15}$$

$$\ell_j(t) = \frac{1}{\|L_{j-1}\|_0}\int_{-1}^{t} L_{j-1}(s)\,\mathrm{d}s, \quad j \geq 2, \tag{5.16}$$

where L_j are the Legendre *polynomials, given explicitly by Rodrigues' formula*

$$L_j(t) = \frac{1}{2^j j!}\left(\frac{\mathrm{d}^j}{\mathrm{d}t^j}\left(t^2-1\right)^j\right). \tag{5.17}$$

The Lobatto shape functions for $\hat{K}$ are defined as tensor products of ℓ_j. A sign factor s_{ij} is also needed in order to compensate for the relative orientations of neighboring elements. This sign factor, which

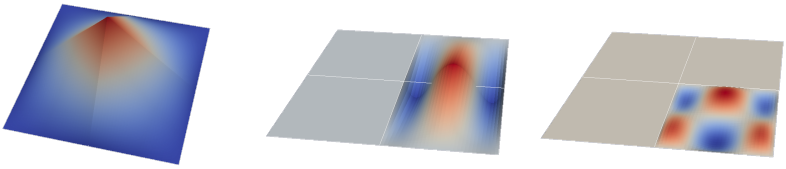


Figure 5.2: Examples of Lobatto vertex (left), edge (middle) and inner (right) shape functions.

is either -1 or 1, is used to make sure that the global basis functions made up of asymmetric shape functions (which occur when $p \geq 3$) match along edges.

Definition 6. *Lobatto Shape Functions for the Quadrilateral Element of Degree p*

$$\varphi_{ij}(\xi,\eta) = s_{ij}\ell_i(\xi)\,\ell_j(\eta), \quad 0 \leq i,j \leq p, (\xi,\eta) \in [-1,1]^2. \tag{5.18}$$

One can observe that for $j \geq 2$, the one-dimensional Lobatto polynomials vanish at the endpoints ± 1. Due to this fact, the two-dimensional shape functions $\varphi_{ij}(x,y)$ can be split into three groups:

- Vertex functions, for which $0 \leq i,j \leq 1$. These are linear in both variables.
- Edge functions, for which $0 \leq i \leq 1, j \geq 2$ or $i \geq 2, 0 \leq j \leq 1$. These vanish on all except one edge, where they are at least quadratic. In the direction perpendicular to this edge, they are linear.
- Interior functions, for which $i,j \geq 2$. These vanish on all edges, and are at least quadratic in both variables in the interior of the cell.

A representative shape function from each group is show in Fig. 5.2.

The local basis functions on each mesh cell K are defined via the pullback by the cell transformation $F_K : \hat{K} \to K$, which is assumed to be a diffeomorphism. We use the notation $\varphi_{K,r}(\boldsymbol{x}) = \varphi_r(F_K^{-1}(\boldsymbol{x}))$. The abstract index r runs over all pairs (i,j). For notational flexibility the index r will in some places in the following be replaced with the index of an entity in the mesh, which should in this case be understood as the set of local basis functions associated with that entity.

Fig. 5.3 compares the growth with p of the condition number κ of the stiffness matrix arising from the FE discretization of the 2D Poisson problem (5.3) for the Lobatto basis and the Lagrange basis with equidistant nodes. The condition numbers, which are with respect to the Euclidean matrix norm, were computed in Octave [44] using the command *cond*, which is based on the singular value decomposition. The discretization is based on a uniform quadrilateral mesh of the unit square, with 64x64 cells, giving a constant cell width $h \approx 0.016$.

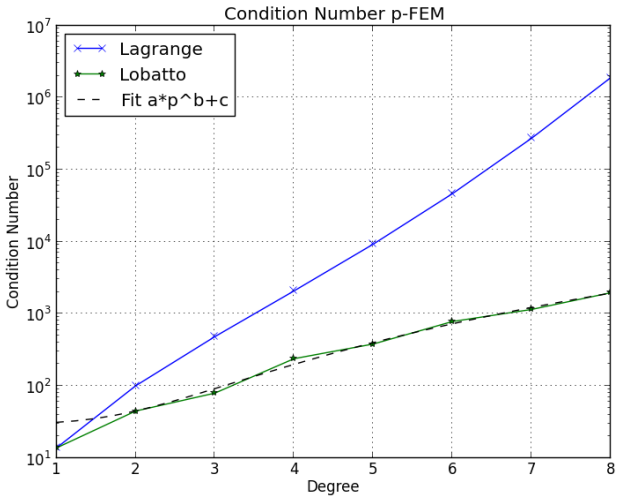


Figure 5.3: Condition number of the stiffness matrix as a function of polynomial degree with Lobatto and Lagrange bases.

The graph shows clearly the explosive growth in κ for the Lagrange basis. For the Lobatto basis, the condition number grows as well, but much slower. The upper and lower bounds on κ have been shown ([73]) to grow as $\mathcal{O}(p^{4(d-1)})$, where d is the dimension.

Fitting a curve (see Fig. 5.3) of the form $y = ap^b + c$ by nonlinear optimization of the least-squares error with respect to the parameters a, b, and c yields the approximate value $b = 3.5$ for the exponent. Taking into account the relatively coarse mesh, and the modification of the matrix to account for Dirichlet boundary conditions, this seems to correspond well with the theoretical prediction.

Even a polynomial growth $\mathcal{O}(p^{4(d-1)})$ of κ is very challenging, particularly in three dimensions. Applying the *static condensation* technique, which forms the Schur complement with respect to the interior degrees of freedom, yields a system that is generally much better conditioned than the original matrix. For instance, it is shown in [86] that static condensation can reduce the bound on the condition number from $\mathcal{O}(p^3)$ to $\mathcal{O}(p)$ for one of the bases considered in that work. It is also possible to use preconditioning techniques based on decomposition and partial orthogonalization of the basis [2, 134], to further improve the conditioning.

The division of the shape functions according to their supports provides an elegant solution to the problem of imposing continuity constraints between neighboring elements which have different polynomial degrees. For each edge e in the mesh, we define $\omega_e = \{K : e \subset K\}$. An *edge degree* p_e can then be associated with e as the minimum of the degrees of the surrounding elements K:

$$p_e = \min_{K \in \omega_e} p_K. \tag{5.19}$$

In order to impose continuity of the global basis, the local basis for each element only includes edge functions of degree up to p_e. Through the use of this intermediary local basis, one avoids the need for imposing constraints for the degrees of freedom shared between neighboring elements with different polynomial degrees, and is thereby able to decrease the total size of the system.

The work of Šolín et al. [123] additionally treats the use of anisotropic polynomial degrees within each element. This provides an additional possibility to tailor the approximation space which is of interest especially in the context of e. g. transport problems with a dominant direction. To reduce the complexity of the implementation, anisotropic polynomial degrees were not considered in this work.

5.4.3 *Global Basis Functions*

In most theoretical expositions of finite elements, the construction of the global basis functions proceeds in two steps. The previous section described the first step of defining the local basis of shape functions on each element. In the second step, the local bases are combined to form global basis functions whose support may consist of more than one cell. Given that the applications addressed in this work have solution and test space $H^1(\Omega)$ (disregarding boundary conditions), the global basis functions must be continuous in order to obtain a conforming method.

For other problems, the conformity requirements would be different. For instance, weak formulations derived from the Maxwell equations contain the spaces of vector-valued functions $H(\mathrm{curl})$ and $H(\mathrm{div})$, which only require continuity along the element interfaces of the tangential and normal components, respectively. It is also possible to work with non-conforming methods, in which one does not impose any further continuity requirements on the global basis functions. One example is the class of *Discontinuous Galerkin* (DG) methods (see e. g. [8, 70]) in which the continuity between elements is imposed in a weak sense. This kind of approach will be discussed in Section 5.6.

At present, our implementation only provides support for solving problems with H^1-conforming global basis functions. Extensions to the more exotic types of conformity are described by several authors (see e. g. [38, 41, 110, 122] for applications to Maxwell problems), and there is nothing which prevents such a development in the future.

The global basis functions are constructed as piecewise combinations of the local basis functions associated with the different entities of the mesh. There are different types of global basis functions, which reflects the separation of the shape functions into vertex, edge and in-

terior functions that was mentioned in Section 5.4.2. The global vertex function associated with a vertex v is defined as

$$\psi_v(\boldsymbol{x}) = \begin{cases} \varphi_{K,v}(\boldsymbol{x}) & \boldsymbol{x} \in K, K \in \omega_v \\ 0 & \boldsymbol{x} \notin K, \forall K \in \omega_v, \end{cases} \tag{5.20}$$

where $\varphi_{K,v}$ is the shape function associated with v in a cell K that is part of the neighborhood of cells ω_v which contain v.

Similarly, the global basis functions associated with an edge e are defined via the shape functions $\varphi^p_{K,e}$ in the cells ω_e that contain e:

$$\psi^p_e(\boldsymbol{x}) = \begin{cases} \varphi^p_{K,e}(\boldsymbol{x}) & \boldsymbol{x} \in K, K \in \omega_e \\ 0 & \boldsymbol{x} \notin K, \forall K \in \omega_e. \end{cases} \tag{5.21}$$

Note that associated with each edge there is one global basis function for each degree $2 \leq p \leq p_e$.

The global interior functions, which have zero trace on the entire boundary of each element are supported on a single cell only. Each global function corresponds to an interior shape function $\varphi_{K,i,j}$ with $2 \leq i, j \leq p_K$, which is extended by zero to the rest of the domain:

$$\psi^{i,j}_{K,b}(\boldsymbol{x}) = \begin{cases} \varphi_{K,i,j}(\boldsymbol{x}) & \boldsymbol{x} \in K \\ 0 & \boldsymbol{x} \notin K. \end{cases} \tag{5.22}$$

5.4.4 *Class Structure*

The continuity of the functions in the solution and test spaces is ascertained by imposing the constraint that the degrees of freedom corresponding to the shape functions on neighboring cells which share part of their support be equal. This constraint is realized in the implementation by assigning degrees of freedom that are shared between all elements supporting a global basis function. A basic requirement is therefore the ability to identify these neighborhood relations, and number the degrees of freedom accordingly.

The HiFlow^3 library contains functionality for dealing with computational meshes, in the form of the *Mesh module* which was developed by the author and Thomas Gengenbach [108]. With the classes in this module, it is possible to describe meshes in two and three dimensions with simplex and tensor-product cells; and perform various operations such as refinement and coarsening. The neighborhood relations between the different types of entities (vertices, edges, faces and cells) can be computed and traversed iteratively.

These capabilities are used in the present work for dealing with the basic topological and geometrical aspects of the finite element space. The additional information that is required in the computational representation of the global finite element space is encapsulated in a separate data structure, whose design has been based on the ideas put forth in [37, 38, 123]. This data structure associates an *Element*

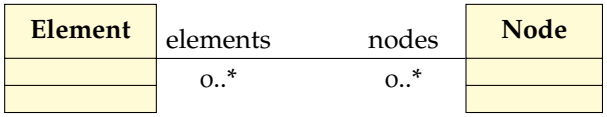


Figure 5.4: UML class diagram showing the relationship between Elements and Nodes.

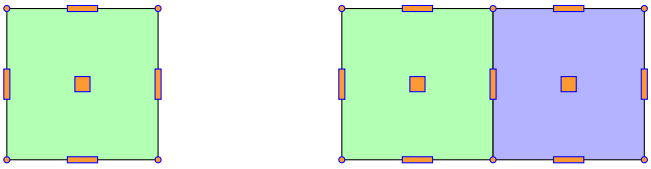


Figure 5.5: The nodes associated to a quadrilateral element (left), and two neighboring elements sharing nodes (right).

with each cell in the mesh, and a set of abstract *Node*s with each Element. The Nodes represent the supports of the global basis functions, and are shared between all Elements representing cells in the support. Each Node therefore contains a list of Elements to which it belongs. Fig. 5.4 shows a UML class diagram depicting this relationship.

There are different types of Nodes, corresponding to the different types of global basis functions. The *type* attribute indicates whether a Node corresponds to a vertex, an edge, a face (in 3D) or the interior of a cell. Fig. 5.5 depicts a quadrilateral element and the associated nodes, as well as how the nodes are shared with a neighboring element. As will be explained in Section 5.4.5, cell refinement creates a hierarchy of Nodes, which are connected via its *parent* and *children* attributes. Fig. 5.6 shows these attributes of the Node class.

The Node and Element objects are organized into the *NodeInfo* container class, which in turn is a part of the *HpSpace* class (see Fig. 5.7). The latter provides logic related to handling multiple variables and the associated degrees of freedom, boundary conditions and constraints.

This class design can be understood as an instance of the *Decorator* design pattern, which is described in [51]. The NodeInfo class decorates the mesh abstraction of the HiFlow^3 library with the Element-

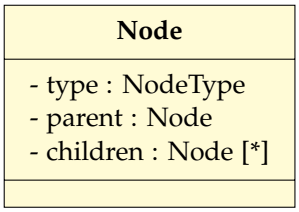


Figure 5.6: Attributes of the class Node.

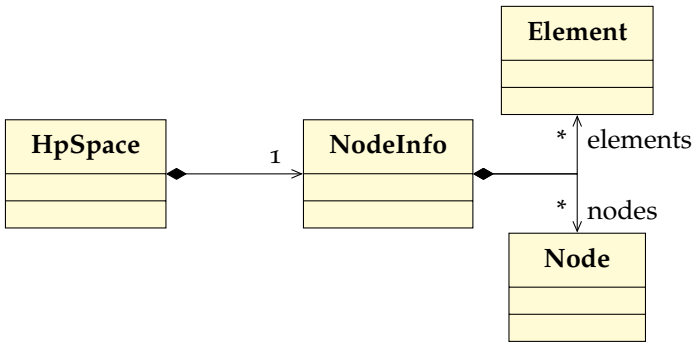


Figure 5.7: Relations between the classes HpSpace, NodeInfo, Node, and Element.

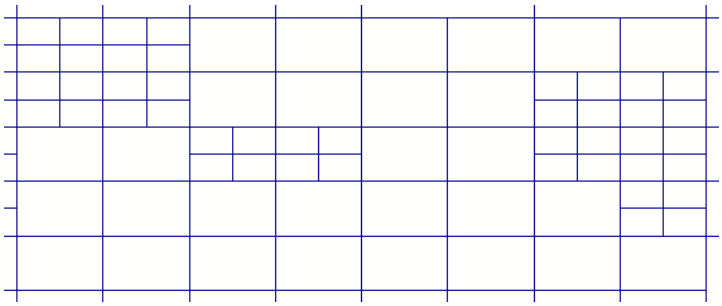


Figure 5.8: Example of irregular mesh with hanging edges.

Node neighborhood relationships, providing each connection between two elements with an object of the class Node. The HpSpace class performs a second decoration, in which the Element and Node objects are given attributes containing the element degrees and degree of freedom indices, respectively. This data is stored in the HpSpace class, rather than in the Element and Node classes, since there needs to be one instance of it per variable.

5.4.5 *Refinement*

The Mesh module of the HiFlow^3 library is able to compute refinements of all its supported cell types, and for some cell types even supports anisotropic refinements and refinements in which the children cells are of a different type than the parent cell. At present, the regularity of the mesh is not constrained, which means that it is possible to work with meshes in which the intersection of two cells does not correspond to an entire sub-entity in both cells. Fig. 5.8 shows an example of an irregular mesh of quadrilaterals. Here some cells have been refined further than their neighbors, which has resulted in some cell intersections which are larger than a vertex, but smaller than an entire edge. We shall refer to such a situation as an *irregular* intersection.

This kind of situation will inevitably appear during local refinement of the mesh. When using triangular or tetrahedral meshes, it is possible to retain regularity of the mesh by performing additional refinements of the cells in a neighborhood of the irregular intersection. The most popular method for this, the *red-green* refinement [18], splits the neighboring cells anisotropically, which degrades the quality of the mesh, but has the desirable property that the children cells have the same type as the parent. For meshes with tensor-product cells, one could use a corresponding method, in which the irregularities are removed by introducing triangles or tetrahedra, and possibly other shapes, such as pyramids, which leads to mixed-type meshes. This makes further refinement and the construction of the finite element space somewhat more complex, but is otherwise an approach worth consideration.

Another common approach for meshes with tensor-product cells, which has been followed in the present work, is to admit meshes that are *one-irregular*, meaning that if a cell K has an edge which is a proper subset of the intersection with a neighboring cell K', than the parent of K has a regular intersection with K'. This translates into the requirement that a cell with an irregular intersection with another cell cannot be further refined, before the irregularity has been removed. Allowing a certain amount of irregularity limits the number of unwanted *regularizing* refinements that must be performed. The major drawback is that the straightforward construction of global basis functions as described in Section 5.4.3 will be able to represent discontinuous functions on irregular meshes. In order to restore H^1-conformity, it is thus necessary to impose *continuity constraints* on the discrete functions. This can be done by formulating additional equations for the degrees of freedom associated with the irregular interface, which will be described in Section 5.4.6.

It is also possible to admit completely irregular meshes, and again impose global continuity using additional constraints on the degrees of freedom. This option has been explored in the work of Šolín et al. [121, 122]. Due to the increased level of complexity in determining the continuity constraints, this approach has not been investigated further in this work.

Since the HiFlow^3 Mesh module presently lacks built-in support for limiting the amount of irregularity of the mesh, this functionality has been implemented using the NodeInfo data structure described in Section 5.4.4. The *refine* function of the NodeInfo class takes as input a set of cells to refine. At present, only isotropic refinement of a quadrilateral cell to four children cells is admitted, but in order to allow future extensions, a flag indicating the requested type of refinement for each cell is also required.

The refinement algorithm is outlined in Figure 5.9. The listings and explanations that follow represent a simplified version of the actual

```
Refine(in: ref) {

  Regularize(inout: ref)

  CreateNodesAndElements(in: ref,
                         out: ref_elem_data)

  RefineMesh(inout: mesh,
             in: ref,
             in: ref_elem_data)

  SwapElements(ref_elem_data)

  ConnectNodes()
}
```

Figure 5.9: Outline of the refinement algorithm.

code, and omit some details which would unnecessarily complicate the exposition.

The first step is to make sure that the refinement will yield a *one-irregular* mesh. This is done in the step *Regularize* (see Figure 5.10), which recursively adds the necessary refinements to make sure that all elements which will be refined have regular interfaces with their neighbors. This procedure uses a stack to keep track of elements which are yet to be refined. At the beginning, all elements for which refinement has been requested are put on the stack. The elements are then visited in the order that they appear on the stack, i. e. *Last In, First Out*. For each element, the associated nodes are traversed searching for one that is constrained, meaning that its parent node is also attached to some element of the current mesh. If such a node is found, the element is put back on the stack to be visited again later, and all elements containing the parent of the constrained node are marked to be refined. Since these refinements can also cause new irregularities to appear, the corresponding elements are also put on the stack to be checked in the coming iterations. The search for constrained nodes is canceled after the first one is found, to avoid adding the same element several times to the stack. If no constrained nodes are found, the element is discarded from the stack, which eventually becomes empty, at which point the procedure is finished. This algorithm is similar to that described in [123].

The next step is to create the new Nodes and Elements in the NodeInfo data structure. Figure 5.11 outlines the procedure *CreateNodesAndElements* which effectuates the modifications. Since the Nodes and Elements are coupled in both directions, so that each Node is connected to several Elements, and vice versa, one has to choose

```
Regularize(inout: ref) {
  Create stack s of elements to be refined.

  while (s not empty)
    Element e := s.pop()

    for (node n of e)
      if (n is constrained)
        s.push(e)
        p := parent node of n

        for (e' element containing p)
          ref.add(e')
          s.push(e')
        skip rest of nodes
}
```

Figure 5.10: Description of the *Regularize* step.

```
CreateNodesAndElements(in: ref,
                       out: ref_elem_data) {
   for (Element e)
     if (e in ref)
       for (Node n connected to e)
         if (n has no children)
           CreateChildrenNodes(n)
       CreateChildrenElements(e, ref_elem_data)

     else
       CopyElement(e, ref_elem_data)
}
```

Figure 5.11: Description of how the new Nodes and Elements are created.

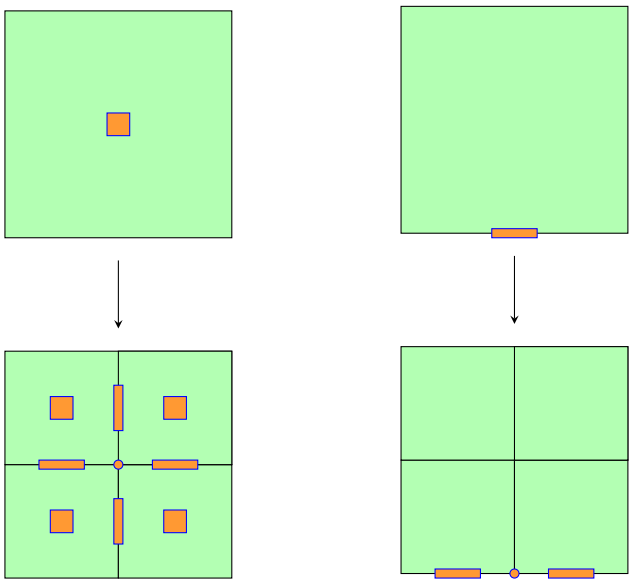


Figure 5.12: Generation of children of an inner node (left) and an edge node (right).

which of the objects to create first. The choice that has been made in this implementation is to first create the children nodes that will be connected to the newly created Elements. This is done in the function *CreateChildrenNodes*, which generates new nodes according to the type of n. This generation is illustrated in Fig. 5.12, which shows that the children of the inner nodes are also associated with the interior of the parent element, whereas the children of the edge nodes are associated with its boundary. For each element which is to be refined, one first creates the children of all its nodes, if they do not already exist, and then create the new elements with the function *CreateChildrenElements*. This function uses a static table which contains the information which children of which nodes in the parent element should be part of each child element.

Whereas refinement extends the existing list of Nodes in the NodeInfo object, it creates a new data structure *ref_elem_data* to keep track of the data for the elements that should be part of the refined space. *CreateChildrenElements* modifies this structure, and the Elements which should not be refined are simply copied over using the function *CopyElement*.

The function *RefineMesh* refines the cells of the Mesh object according to the refinements of the Elements. The fact that this function is called as a part of the *Refine* procedure again illustrates the use of the NodeInfo class as a Decorator for the Mesh class. Finally the function *ConnectNodes* reestablishes the link from Nodes to Elements, by iterating over all Elements, and letting each one add itself to each of the Nodes that it contains.

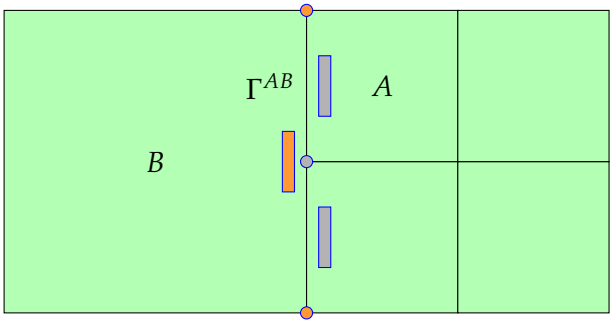


Figure 5.13: Canonical hanging node situation for 2D quadrilateral meshes.

5.4.6 *Constraints*

As described in the previous sections, the implementation permits the use of meshes which contain irregular intersections, or *hanging nodes*. Hence, the corresponding discrete space $\tilde{V}_h = \text{span}\{\tilde{\psi}_i\}$, $i = 1 \dots N$, will contain discontinuous functions, and thus not be a subset of $H^1(\Omega)$. A common approach to obtain a conforming discrete space is to construct a subspace V_h of $H^1(\Omega)$ by projecting the basis functions $\tilde{\psi}$ onto a set of continuous functions ψ_j, $j = 1 \dots M$, and letting $V_h = \text{span}\{\psi_j\}$.

The projection operator can be determined by considering the constraints that have to be imposed on the values of the degrees of freedom to ensure that all discrete functions are continuous. Consider the situation depicted in Fig. 5.13. In order for the discrete function to be globally continuous, it is necessary and sufficient that continuity be imposed along the irregular intersections of this type in the mesh. Consider a discrete function w_h, and its restrictions w_h^A and w_h^B to the cells A and B, respectively. For w_h to be continuous over the interface Γ^{AB}, it must hold that

$$w_h^A(\boldsymbol{x}) = w_h^B(\boldsymbol{x}), \qquad \forall \boldsymbol{x} \in \Gamma^{AB}. \tag{5.23}$$

Let $\mathcal{S}_A^{AB}$ and $\mathcal{S}_B^{AB}$ be the indices of the shape functions on A and B with non-zero trace on Γ^{AB}, respectively, and let $\iota_K(r)$ be the mapping from indices of local shape functions to indices of global basis functions on cell K. Then

$$w_h^A(\boldsymbol{x}) = \sum_{r \in \mathcal{S}_A^{AB}} \tilde{w}_{\iota_A(r)} \varphi_{A,r}(\boldsymbol{x}), \tag{5.24}$$

$$w_h^B(\boldsymbol{x}) = \sum_{s \in \mathcal{S}_B^{AB}} \tilde{w}_{\iota_B(s)} \varphi_{B,s}(\boldsymbol{x}), \tag{5.25}$$

where $\tilde{w}_i$ are the unknown values of the degrees of freedom.

The continuity condition (5.23) is then equivalent to

$$\sum_{r \in \mathcal{S}_A^{AB}} \tilde{w}_{\iota_A(r)} \varphi_{A,r}(\boldsymbol{x}) = \sum_{s \in \mathcal{S}_B^{AB}} \tilde{w}_{\iota_B(s)} \varphi_{B,s}(\boldsymbol{x}), \qquad \forall \boldsymbol{x} \in \Gamma^{AB}. \tag{5.26}$$

It is then possible to obtain expressions for the values of the degrees of freedom in $\mathcal{S}_A^{AB}$ in terms of those in $\mathcal{S}_B^{AB}$ by evaluating Equation (5.26) at a set of distinct points $\{\boldsymbol{x}_k \in \Gamma^{AB}\}$, $k = 1, \ldots, |\mathcal{S}_A^{AB}|$. We introduce the matrices $\kappa_{kr}^A = \varphi_{A,r}(\boldsymbol{x}_k)$ and $\kappa_{ks}^B = \varphi_{B,s}(\boldsymbol{x}_k)$, and obtain the equality

$$\kappa^A \tilde{w}^A = \kappa^B \tilde{w}^B. \tag{5.27}$$

Since the shape functions are linearly independent, κ^A is invertible, and we can solve for $\tilde{w}^A$:

$$\tilde{w}^A = \left(\kappa^A\right)^{-1} \kappa^B \tilde{w}^B = P^{AB} \tilde{w}^B. \tag{5.28}$$

The local constraint matrix $P^{AB} = \left(\kappa^A\right)^{-1} \kappa^B$ with coefficients p_{rs} is a local representation of the constraints that have to be imposed on the degrees of freedom in order to enforce continuity. In this case, the constraints have been formulated for the degrees of freedom on cell A in terms of those on cell B. Other choices are also possible, but this method is straightforward and seems to be the most common in the literature.

By considering all the irregular intersections in the mesh, it is possible to construct in this manner a set of constraints which when applied will yield a continuous discrete function. In this process, the number of unconstrained degrees of freedom will be decreased from N to M, and $N - M$ constraints will be imposed. The full process can be summarized as a global constraint matrix $P \in \mathbb{R}^{NxM}$, which maps the degrees of freedom from the reduced global basis to the full basis. For unconstrained degrees of freedom, the corresponding row in the matrix is simply the identity, whereas for constrained degrees of freedom, the row gives its dependence on the unconstrained degrees of freedom. The computation of P will be discussed later in this section.

A discrete function $w_h \in V_h$ can equivalently be represented in the reduced basis as

$$w_h = \sum_{j=1}^{M} w_j \psi_j, \tag{5.29}$$

or in the full basis as

$$w_h = \sum_{i=1}^{N} \tilde{w}_i \tilde{\psi}_i, \tag{5.30}$$

where the coefficients are related by $\tilde{w}_i = \sum_{j=1}^{N} P_{ij} w_j$. One can also interpret P^T as a projection operator for the basis functions, since

$$w_h = \sum_{i=1}^{N} \tilde{w}_i \tilde{\psi}_i = \sum_{i=1}^{N} \tilde{\psi}_i \sum_{j=1}^{M} P_{ij} w_j = \sum_{j=1}^{M} w_j \sum_{i=1}^{N} P_{ij} \tilde{\psi}_i, \tag{5.31}$$

```
Constr = {}
for each constrained element interface c_eif {
  p_eif = parent(c_eif)
  Sp, Udof = shape_functions_and_dofs(p_eif)
  Sc, Cdof = shape_functions_and_dofs(c_eif)

  xk = equidistant_grid(eif)
  Kp = eval_shape_fun(Sp, xk)
  Kc = eval_shape_fun(Sc, xk)

  Ploc = solve(Kc, Kp)

  add_constraints(Cdof, Udof, Ploc, Constr)
}

P = reduce_constraints(Constr)
```

Figure 5.14: Outline of the constraint computation.

which by comparison with Equation (5.29) gives

$$\psi_j = \sum_{i=1}^{N} P_{ij}\tilde{\psi}_i \qquad \Longleftrightarrow \qquad \psi = P^T\tilde{\psi}. \tag{5.32}$$

In the implementation, the construction of the constraint matrix P is surprisingly complex, and we do not attempt to describe it in detail here. The overall algorithm is outlined in Figure 5.14. It uses the concept of an *element interface*, which represents an element and the collection of nodes that lies on its interface to another element. The algorithm visits all element interfaces that are constrained, meaning that they correspond to the smaller (child) element of an irregular element interface Γ^{AB} (element A in Fig. 5.13). The corresponding parent element interface, which corresponds to element B, is retrieved, as well as the set of shape functions with non-zero trace on Γ^{AB} and the corresponding global degree of freedom indices for both child and parent. Next, a grid of equidistant points is created on Γ^{AB}, and the matrices κ^A and κ^B are evaluated as described above. Since the basis is hierarchical, it is in fact possible to precompute these matrices, which do not depend on the physical coordinates of the cell, for all shape functions up to a given maximal polynomial degree, and then solve for P^{AB}, as described e. g. in [123]. Hence, the cost of this step is negligible.

The constraints computed in this way for each interface are gathered into an intermediate structure called *Constr* in Figure 5.14. This consists of a hash table which maps each constrained degree of freedom to a list of tuples of constraining degrees of freedom and the corresponding coefficients. After visiting all constrained element in-

terfaces and collecting the constraints, it is possible that some of the degrees of freedom in this list are themselves constrained. As is pointed out in [17], in which a similar approach is described, it is even possible that a constrained degree of freedom transitively has a constraint in terms of itself. To obtain a representation of the operator P as described above, on the other hand, it is required that each constrained degree of freedom is expressed in terms of unconstrained degrees of freedom only.

It is thus necessary to *reduce* the constraints in the intermediate structure to this form. The algorithm for this reduction is described in Figure 5.15, and consists of two sweeps through the structure. In a first step, the constrained degrees of freedoms are iterated from the highest-numbered to the lowest-numbered, and all dependencies on constrained degrees of freedom with higher indices are eliminated recursively. The recursion is implemented with a stack, which contains those dependencies (j, P_{ij}) of the current degree of freedom (index i) that must still be treated. Each dependency represents a constrained degree of freedom, which in turn depends on a set S_j of degrees of freedom, which may be constrained or unconstrained. The dependency on j is resolved by looping over these transitive degrees of freedom $k \in S_j$, and checking whether degree of freedom i can depend on k. In this case, the dependency on k is added to i with coefficient $P_{ij} \cdot P_{jk}$, and otherwise, it is pushed on the stack to be treated later.

When the stack is empty, degree of freedom i will only depend on degrees of freedom, which are either unconstrained, or have index $j \geq i$. At this point, the possible self-dependence is resolved by dividing all coefficients P_{ij} with $1 - P_{ii}$. Finally, a second sweep over the constraints is performed over the constraints, and the remaining dependencies on constrained degrees of freedom are eliminated. This sweep is similar to the first, except that the degrees of freedom are now traversed in order of increasing index. At the end of this step, the constrained degrees of freedom will only depend on unconstrained degrees of freedom, which yields the desired form of the operator P.

The two sweeps of this algorithm are similar to the backward and forward substitution associated with Gaussian elimination in a system of equations. Indeed the reduction is effectively a partial solution of the system of constraint equations. If we denote the unconstrained degrees of freedom with u and the constrained degrees of freedom with c, and assume that the latter have been permuted to come after the former, then the operation that we want to perform corresponds to reducing the block matrix

$$\hat{P} = \begin{pmatrix} \mathcal{I} & 0 \\ \hat{P}_{cu} & \hat{P}_{cc} \end{pmatrix} \tag{5.33}$$

```
P = reduce_constraints(Constr)
  P = unreduced_operator(Constr)

  for i = Nrows(P) .. 1
    Zi = { j : P(i,j) != 0,
           j > i, j constrained }
    s = Stack((j, P(i,j)) for all j in Zi)
    set P(i,j) = 0 for all j in Zi

    while s not empty
      (j, pij) = s.pop()
      Sj = { k : P(j,k) != 0 }
      for k in Sj
        if k <= i or k unconstrained
          P(i,k) += pij * P(j,k)
        else
          s.push((k, pij * P(j,k)))

    if (P(i,i) != 0)
      for j in Zi
        P(i,j) = P(i,j)/(1 - P(i,i))

  for i = 1 .. Nrows(P)
    Zi = { j : P(i,j) != 0, j constrained }
    s = Stack((j, P(i,j)) for all j in Zi)
    set P(i,j) = 0 for all j in Zi

    while s not empty
      (j, pij) = s.pop()
      Sj = { k : P(j,k) != 0 }
      for k in Sj
        if k unconstrained
          P(i,k) += pij * P(j,k)
        else
          s.push((k, pij * P(j,k)))
```

Figure 5.15: Reduction of constraints.

to the form

$$\tilde{P} = \begin{pmatrix} \mathcal{I} & 0 \\ \tilde{P}_{cu} & -\mathcal{I} \end{pmatrix}, \tag{5.34}$$

which can be achieved by multiplying the last row by $-\hat{P}_{cc}^{-1}$.

It should be noted that only the rows of the constraint matrix P that correspond to the constrained degrees of freedom need to be stored.

As mentioned above, the set of constraints can be interpreted as a projection operator P^T that maps the basis for the full space $\tilde{V}_h$ to a reduced basis of continuous functions for V_h. In most finite element implementations, including the one described here, it is difficult to apply this operator to the global basis functions directly, since these are not available explicitly. This is reflected in the algorithms for the assembly of matrices and vectors, which typically involve a loop over the elements, the construction of a local matrix or vector, and the addition of this object into the corresponding global structure via the mapping $\iota_K(r)$ of the local degree of freedom index r on element K to the index of the global degree of freedom ι.

There are different possibilities of incorporating the constraints into this procedure. Demkowicz et al. [37] use the concept of a *modified* element in which the constraints are applied when constructing the local matrix or vector. This has the advantage of making it possible to use *multifrontal* sparse direct solvers [43] that work with the local matrices and vectors directly. The drawback is that the local assembly functions, which contain a large part of the problem-specific logic related to the PDE, have to be modified in order to correctly apply the constraints. This reduces the genericity of the global assembly procedure, and make libraries such as HiFlow^3 that implement these procedures more difficult to use.

Consequently, it is desirable to be able to make the change of basis without modifying the existing assembly algorithm. This can be accomplished by first performing the assembly in the full basis, and then modifying the resulting matrix to act only on the reduced subspace.

Consider a variational problem with bilinear form $a : \tilde{V}_h \times \tilde{V}_h \to \mathbb{R}$ and linear form $b : \tilde{V}_h \to \mathbb{R}$. We want to restrict these operators to the domains $V_h \times V_h$ and V_h, respectively, and to compute the corresponding discrete quantities

$$A_{ij} = a\left(\psi_j, \psi_i\right) \qquad \text{and} \qquad b_i = b\left(\psi_i\right). \tag{5.35}$$

Using Equation (5.32), we obtain

$$A_{ij} = a\left(\sum_{r=1}^{N} P_{rj}\tilde{\psi}_j, \sum_{s=1}^{N} P_{si}\tilde{\psi}_i\right) = \sum_{s=1}^{N}\sum_{r=1}^{N} P_{si}a\left(\tilde{\psi}_r, \tilde{\psi}_s\right)P_{rj}, \tag{5.36}$$

$$b_i = b\left(\sum_{s=1}^{N} P_{si}\tilde{\psi}_s\right) = \sum_{s=1}^{N} P_{si}b\left(\tilde{\psi}_s\right). \tag{5.37}$$

```
A = reduce_matrix(Af, sp, P) {
  for k in sp
    i = row(k)
    j = column(k)
    if i is unconstrained
      if j is unconstrained
        A.add(i, j, Af(i, j))
      else
        for r such that P(j, r) != 0
          A.add(i, r, P(j, r) * Af(i, j))
    else
      if j is unconstrained
        for s such that P(i, s) != 0
          A.add(s, j, P(i, s) * Af(i, j))
      else
        for s such that P(i, s) != 0
          for r such that P(j, r) != 0
            A.add(s, r, P(i, s) * P(j, r)
                  * Af(i, j))
  set A(i, i) = 1 for all constrained i
}
```

Figure 5.16: Reduction of system matrix.

This can be written using matrix notation as $A = P^T \tilde{A} P$ and $b = P^T \tilde{b}$, where $\tilde{A}_{sr} = a(\tilde{\psi}_r, \tilde{\psi}_s)$ and $\tilde{b} = b(\tilde{\psi}_s)$ are the discretized operators defined on the full space, which can be computed using the standard finite element assembly algorithms without modifications.

An algorithm for computing the reduced matrix A given the full matrix $\tilde{A}$ is described in Figure 5.16. This algorithm assumes that the matrix is stored in a sparse format, and makes use of the sparsity pattern *sp* of the full matrix. The sparsity pattern of the reduced matrix can be computed by a similar algorithm.

The application of P and P^T to vectors of size M and N is straightforward, and will not be described in detail. When an iterative solver is used, the only operation required is matrix-vector multiplication with the system matrix $A = P^T \tilde{A} P$. This operation can be implemented in terms of the three matrix-vector products $\boldsymbol{x}_1 = P\boldsymbol{x}$, $\boldsymbol{x}_2 = \tilde{A}\boldsymbol{x}_1$ and $\boldsymbol{y} = P^T \boldsymbol{x}_2$, and hence the explicit reduction of the matrix as described above is not required. For direct solvers, on the other hand, the explicit reduction is necessary, and computing it once will also make the matrix-vector multiplication cheaper in the iterative context. Furthermore, with the reduced matrix explicitly at hand, it is easy to make further modifications, e.g. to apply constraints related to essential boundary conditions, as described in Section 5.4.7.

5.4.7 *Computation of Lagrange Interpolant for Boundary and Initial Conditions*

The main drawback of using a non-Lagrange basis, such as the Lobatto shape functions described in Section 5.4.2, is that the corresponding degree of freedom functionals σ_i are difficult to evaluate. With a Lagrange basis this functional is simply function evaluation at the corresponding nodal point $\boldsymbol{x}_i$: $\sigma_i(f) = f(\boldsymbol{x}_i)$. This relation is useful when dealing with essential boundary conditions and initial conditions, where a given function f has to be projected into the finite element space U_h. This is commonly done by simply evaluating f at the nodal point for each degree of freedom, and setting the corresponding unknown to the resulting value.

For simplicity, we use the same type of projection, and transform the corresponding unknowns to the Lobatto basis. The procedure is similar to that used to compute the local constraint matrix in Section 5.4.6, and it is basically identical when dealing with boundary conditions and with initial conditions. In both cases, a given function f is evaluated on each mesh entity E (boundary edge or cell) at a set of points $\boldsymbol{x}_i$, $i = 1, \ldots, N_E$, where N_E is the local number of degrees of freedom belonging to E, which yields a vector $\boldsymbol{F}_i = f(\boldsymbol{x}_i)$, and the following system of equations for the restriction of the projected function f_h to E:

$$f_h(\boldsymbol{x}_i) = \sum_{j=1}^{N_E} f_j^E \varphi_{E,j}(\boldsymbol{x}_i) = F_i, \qquad i = 1, \ldots, N_E. \tag{5.38}$$

Solving this system of equations yields the values f_j^E of the unknowns in the Lobatto basis. Since the system is solved for each entity E separately, one will obtain several values for the same unknown. If the projected function is continuous, then the value will be equal each time. In order to make the computation uniquely defined also for discontinuous f, it is possible to assign *material numbers* to each of the entities in the mesh. During the projection, the entities are traversed in order of increasing material number. The unknown values are overwritten by each new entity, so that the final value of each unknown comes from the entity with the highest material number. This makes it possible to handle functions f which are only continuous inside the set of entities having the same material number.

5.4.8 *Possible Extensions*

As should be clear from the description above, adding support for hp-FEM makes the implementation considerably more complicated than for standard FEM with a constant, low polynomial degree and no local refinement. Although fully functional in its present state, there

are several possible extensions that would increase the capabilities and the efficiency of the code.

Apart from including the possibility of handling three-dimensional elements, there is a large research interest in the use of anisotropic extensions of the solution space. In the context of hp-FEM, one can consider both anisotropic cell refinements, where a cell is not split equally in all directions; and anisotropic shape functions, which can have different degrees in each direction. Tensor-product cells naturally lend themselves better to such methods than simplices.

The use of anisotropy has the potential of improving efficiency for problems where the dominating features of the data and solution have a certain direction. A classical example are the boundary layers that occur in advection-diffusion models where the transport is dominating, but anisotropy is of interest also e. g. in electromagnetics. References that discuss the use of anisotropic hp-FEM include [48, 111, 112].

Another aspect of hp-FEM that has not been discussed very much in the literature is the possibility of using mesh coarsening and polynomial degree reductions to make the approximation space smaller. This possibility is especially desirable for time-dependent problems, where the solution, and thus the optimal approximation space, varies over time. It would, however, make the technical aspects of representing the mesh and approximation space more complex, and require a more sophisticated algorithm for controlling the adaptation.

For high polynomial degrees, especially in three dimensions, the cost of numerical quadrature tends to become very large. This is easily understood by considering that for computing the local element matrix for an element with degree p in d dimensions, $(p^d)^2$ coefficients have to be computed, each of which is a sum of values at c. p^d quadrature points, assuming exact Gauss quadrature for an integrand of degree $2p$. This yields a total cost of $\mathcal{O}(p^{3d})$ operations. For $p = 10$ and $d = 3$, this is indeed a large number. Fortunately, the costs can be reduced using a technique known as *sum factorization*, which has its roots in the spectral method. As explained in [87], it is possible to exploit the structure of tensor-product elements, and the decomposition into vertex, edge, face, and interior shape functions, to cache some redundant computation and thereby reduce the complexity to $\mathcal{O}(p^5)$ in 2D and $\mathcal{O}(p^7)$ in 3D.

Since the present work considers only 2D problems, we did not experience any significant problems with the cost of quadrature, and have therefore not used sum factorization. In 3D, the benefits of this technique are likely to be significant.

The usual bottleneck for the efficiency of the solution is the linear solver. The use of hp-FEM puts different requirements on the solver than standard FEM. With a fixed low-order approximation, the global system matrix will tend to be large but very sparse. Use of hp-FEM

will decrease the overall size of the matrix, but increase the number of nonzero elements per row. For elements with large p, there will be a large diagonal block corresponding to the interior degrees of freedom, which do not couple directly with the degrees of freedom of other elements. An efficient solver needs to deal with these blocks in an appropriate way.

A common technique, known as *static condensation*, separates the system for the interior degrees of freedom from that for the remaining degrees of freedom by forming the Schur complement. The matrix corresponding to the interior degrees of freedom will be block-diagonal, and can be solved efficiently in parallel. The remaining Schur complement system will typically be considerably smaller, and, as was mentioned in Section 5.4.2, it will be better conditioned than the original system. This makes it possible to solve it efficiently as well. An overview of this technique can be found in [134].

For iterative solvers, the use of preconditioning is essential for achieving good performance. Several approaches have been discussed in the literature, including preconditioning based on e. g. low-order approximations [29, 62, 74], domain decomposition [2], or incomplete LU-decompositions [134]. For parallel computing, the use of additive Schwarz and block-Jacobi preconditioning has been evaluated in [21]. A holistic approach involving the use of matrix-free Newton iterations and preconditioning with sub-grid linear elements is presented in [29]. The exploration of this topic is very important for the success of hp-FEM, since what finally counts in most applications is the time it takes to solve a problem, and not how optimal the discretization space is.

5.5 NUMERICAL EXAMPLES FOR THE POISSON EQUATION

In order to verify the correctness of the implementation, it is useful to test it for problems with known solutions. This section presents two such tests with boundary value problems for the Poisson equation. Both problems are well-known in the hp-FEM literature, and often used for testing purposes. They have both been included in the extensive comparison on hp-FEM methods presented in [89].

The test cases differ only with respect to the geometry, boundary conditions and force term. The common boundary value problem is formulated as follows:

$$-\Delta u = f \qquad \text{in } \Omega, \tag{5.39}$$

$$u = g \qquad \text{on } \partial\Omega. \tag{5.40}$$

We use the standard weak formulation

Seek $u \in H^1_g(\Omega)$ such that

$$\int_\Omega \nabla u \cdot \nabla \psi \, \mathrm{d}x = \int_\Omega f \psi \, \mathrm{d}x, \qquad \forall \psi \in H^1_0(\Omega). \tag{5.41}$$

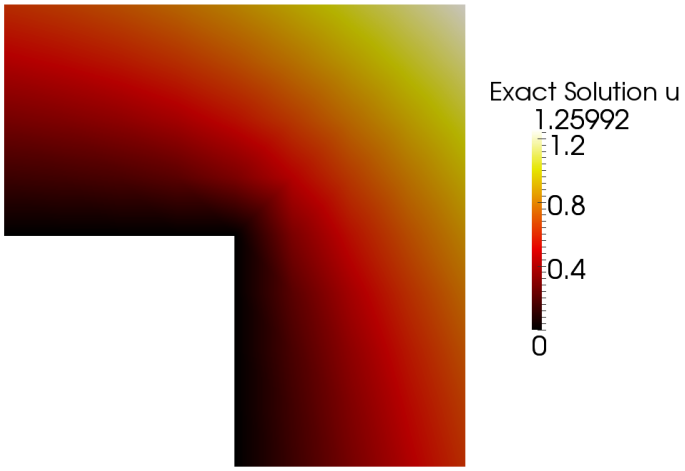


Figure 5.17: Exact solution of Problem A.

where the solution space is defined as $H^1_g(\Omega) = \{u \in H^1(\Omega) : u|_{\partial\Omega} = g\}$. In both problems we use the exact solution to compute the Dirichlet boundary conditions. In order to keep the notation simple, we express the problems in polar coordinates $r = \sqrt{x^2 + y^2}$, and $\theta = \tan^{-1}(y/x)$.

The first problem (A) is posed on an L-shaped domain, and has a solution which is singular at the origin. The data are listed below, and the exact solution is shown in Fig. 5.17.

Problem A (Corner Singularity)**.**

$$\begin{aligned} \textit{Domain } \Omega &= [-1,1]^2 \setminus [-1,0]^2, \\ \textit{Exact solution } u(r,\theta) &= r^\alpha \sin\left(\alpha\theta + \frac{\pi}{3}\right), \quad \alpha = \frac{2}{3}, \\ \textit{Right-hand side } f &= 0. \end{aligned}$$

The second problem (B) is posed on a square domain, and has a smooth solution which exhibits a sharp gradient along a circular arc cutting through the domain (see Fig. 5.18).

Problem B (Interior Layer)**.**

$$\begin{aligned} \textit{Domain } \Omega &= \{-1.25 \le x \le -0.25, 0.25 \le y \le 1.25\}, \\ \textit{Exact solution } u(r) &= \tan^{-1}\left(\beta\,(r-1)\right), \quad \beta = 60, \\ \textit{Right-hand side } f(r) &= -\frac{\beta}{1+v^2}\left(\frac{1}{r} - \frac{2\beta v}{1+v^2}\right), \\ \textit{where } v &= \beta\,(r-1)\,. \end{aligned}$$

Both problems were solved using the implementation in HiFlow^3 described in Section 5.4. The adaptation was controlled using the Melenk-Wohlmuth hp-adaptive strategy outlined in Figure 5.1. The values of

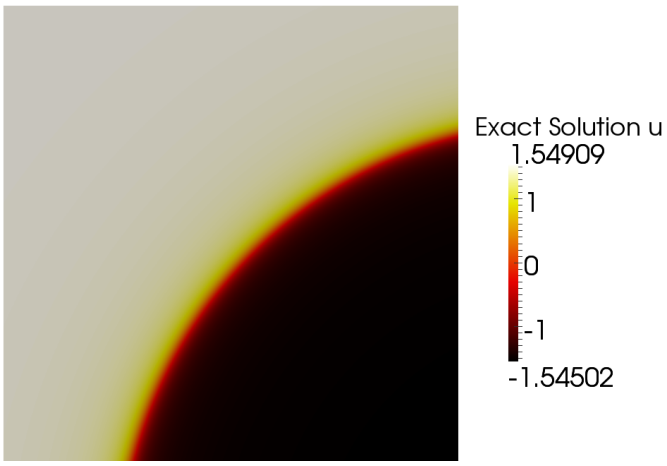


Figure 5.18: Exact solution of Problem B.

the parameters used were $\theta = 0.75$, $\gamma_h = 4$, $\gamma_p = 0.4$, and $\gamma_n = 1$, the same as used in the original article [88].

This strategy requires an a posteriori error indicator. For the Laplace operator, the following explicit error indicator was proposed in [88]:

$$\eta_K^2 = \frac{h_K^2}{p_K^2} \|f + \Delta u_h\|_{0,K}^2 + \sum_{e \in \partial K \cap \Omega} \frac{h_e}{2 \max(p_K, p_{K'})} \|[\![\partial_n u_h]\!]\|_{0,e}^2. \quad (5.42)$$

This error indicator has two terms, corresponding to a cell residual for the cell K, and an edge residual for each interior edge e of K. The cell residual is simply the L^2-norm over K of the residual expression of the PDE with the discrete solution u_h. This is weighted with a factor containing the cell width h_K and the polynomial degree p_K. The edge residual measures the jump of the solution over the edge in the L^2-norm $\|\cdot\|_{0,e}$ defined as an integral over e. Here the weighting factor involves the edge length h_e and the maximum of the polynomial degrees p_K and $p_{K'}$ of the cells K and K' that share the edge. As discussed in [88], the error indicator is reliable, and it is locally efficient in the sense that it bounds the local error from below. The constant in the lower bound depends on p_K, and so the estimation might degrade as the polynomial degree is increased.

For comparison, both problems were also solved using uniform refinement, as well as with an h-adaptive method using the Fixed Energy Fraction strategy (see Section 5.3.1). Both cases were computed with constant polynomial degrees 1 and 2. The h-adaptive method used the same error indicator η_K as the hp-method, and with the value $\theta = 0.8$ of the fraction parameter.

The results are shown in the convergence graphs for the error measured in the H^1-seminorm that are shown in Fig. 5.19 and 5.20, for Problem A and B, respectively. For Problem A, the theoretical rate of convergence has been analyzed in detail, see e. g. [15]. It can be

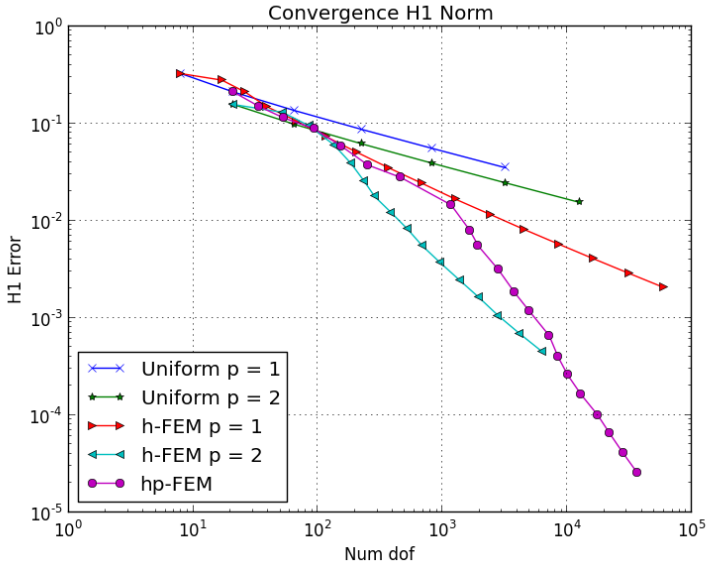


Figure 5.19: Convergence in the H^1-norm for Problem A.

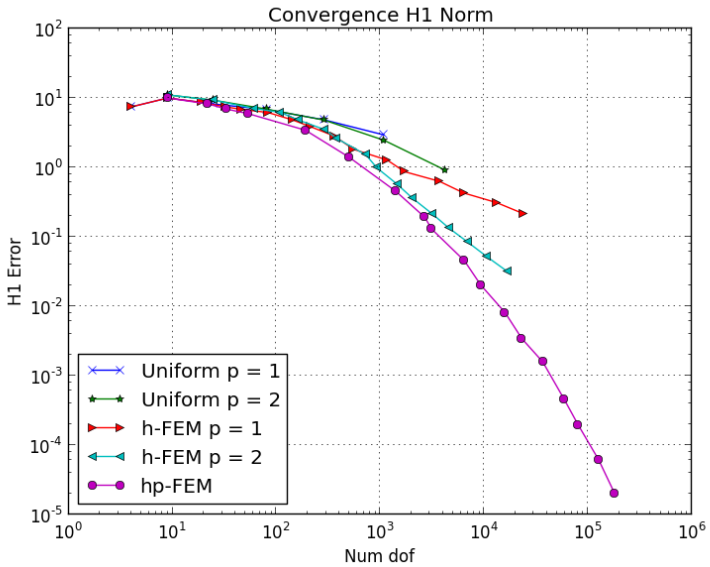


Figure 5.20: Convergence in the H^1-norm for Problem B.

Adaptation	Slope α	Experimental Convergence Rate β_e
Uniform $p = 1$	-0.37	0.74
Uniform $p = 2$	-0.36	0.72
h-adaptive $p = 1$	-0.59	1.18
h-adaptive $p = 2$	-1.16	2.32

Table 5.2: Table of fitted slopes and corresponding experimental convergence rates for Problem A.

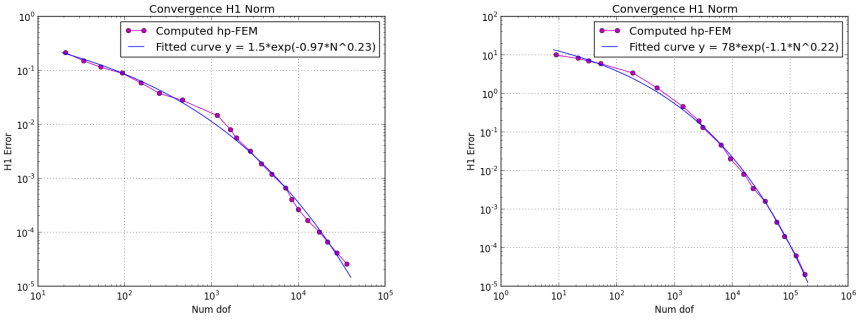
shown that the exact solution $u \in H^{\frac{5}{3}}(\Omega)$, so that according to the a priori error estimation (5.5) for uniform refinement, the error in the H^1-norm should converge as h^β, with $\beta = \frac{2}{3}$. To verify that this is indeed the case, we compare β with a *experimental convergence rate* β_e, that is computed from the slopes α of the convergence curves in Fig. 5.19 as $\beta_e = -2\alpha$. This relation comes from the fact that the number of degrees of freedom $N \propto h^{-2}$, so that $N^\alpha \propto h^{-2\alpha}$. The value of α is determined from the computed data by linear regression of $\log \|u - u_h\|_1$ against $\log N$. The results for the tests with uniform and h-adaptive refinement are summarized in Table 5.2.

For the uniform refinement, the experimentally determined rates are somewhat higher than the expected value $\frac{2}{3}$. The effect of the limited regularity, which prevents the convergence rate from being determined by the polynomial degree is clear: the rates for $p = 1$ and $p = 2$ are essentially the same. The h-adaptive methods succeed in overcoming this limitation, both attaining experimental convergence rates above their polynomial degrees.

For Problem A, the h-adaptive strategy with $p = 2$ performs better than the hp-adaptive strategy for errors down to approximately $5 \cdot 10^{-4}$. This correlates with the conclusion in [89] that the Melenk-Wohlmuth strategy tends to perform rather poorly in the preasymptotic range. It should also be noted that the algorithm is dependent on the choice of the parameters $\theta, \gamma_h, \gamma_p$, and γ_n, so that it might be possible to obtain better results with different values.

The experimental data for Problem A verifies the capacity of the hp-adaptive method to yield increasing convergence rates as the space is extended. While the convergence curve at first follows that for the h-FEM method with $p = 1$, the slope becomes steeper, but still seems more or less constant. From the numerical data, it is not clear whether the convergence rate would continue increasing as $N \to \infty$. This behavior is however clear in the results from Problem B (Fig. 5.20), where the convergence rate for the hp-strategy increases continually with N. In this test, the hp-adaptive method performs better than the uniform and h-adaptive methods for all error levels.

Given the possibility of attaining exponential convergence with hp-adaptive methods, an attempt was made to fit curves of the form $y = a \exp(-bN^c)$ to the convergence curves for the hp-adaptive method



(a) Problem A (b) Problem B

Figure 5.21: Error convergence in the H^1-norm with the hp-adaptive strategy compared to fitted exponential curves.

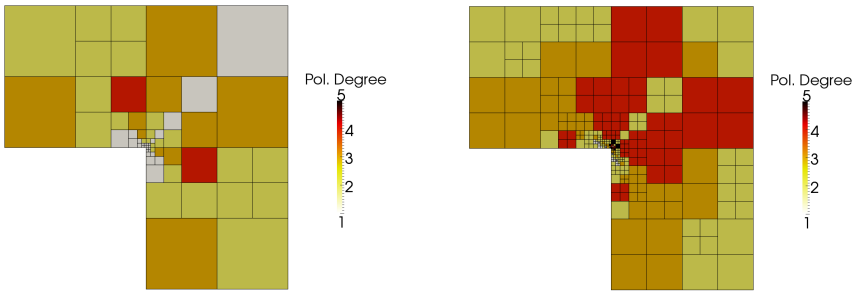
	a	b	c
Problem A	1.5	0.97	0.23
Problem B	78	1.1	0.22

Table 5.3: Table of coefficients for exponential curves $y = a\exp(-bN^c)$ fitted to the convergence curves of the H^1-error for the hp-adaptive methods.

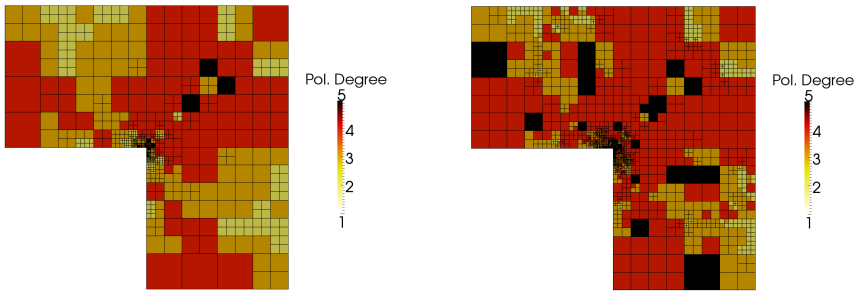
in Fig. 5.19 and 5.20. In contrast to the power laws that described the convergence of h-adaptive methods, this type of fit requires the solution of a nonlinear least-squares problem to determine the coefficients a, b, and c which minimize the error between the data points and the fitted curve. We used the Levenberg-Marquardt algorithm implemented in the function *optimize.curve_fit* of the *SciPy* numerical library [77] to compute the coefficients. Since the problem of fitting an exponential curve is ill-conditioned, the relation between the measured error y and the number of degrees of freedom N was transformed by taking the logarithm of both sides, so that $\log y$ was fitted against $\log a - bN^c$. The resulting curves are shown in Fig. 5.21, and the corresponding coefficients are listed in Table 5.3.

It should be noted that this procedure does not give conclusive information as to whether or not the method results in exponential convergence of the error. However, it provides a reference against which the experimental results can be compared.

The difference in behavior of the h-adaptive method between the two problems is illustrated in Fig. 5.22 and Fig. 5.23, which show the adaptively refined meshes and distributions of polynomial degree at four different points in the adaptation process. In the presence of the singular solution, the maximum polynomial degree used is only 5 after 20 steps, whereas for the smooth solution, degree 9 (which is the maximum degree currently allowed in the implementation) is reached already after 10 steps.

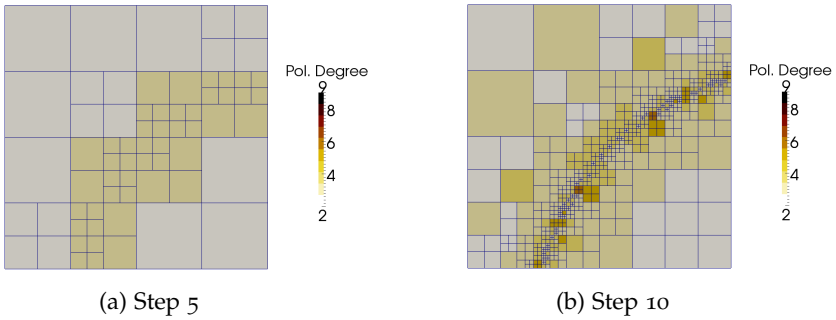


(a) Step 5 (b) Step 10

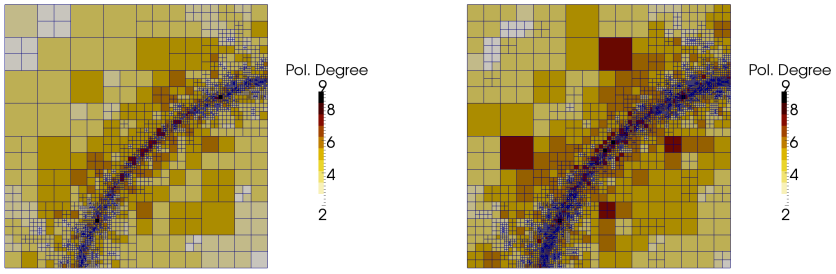


(c) Step 15 (d) Step 20

Figure 5.22: Refined meshes and degree distributions from hp-adaptive solution of Problem A.



(a) Step 5 (b) Step 10



(c) Step 15 (d) Step 17

Figure 5.23: Refined meshes and degree distributions from hp-adaptive solution of Problem B.

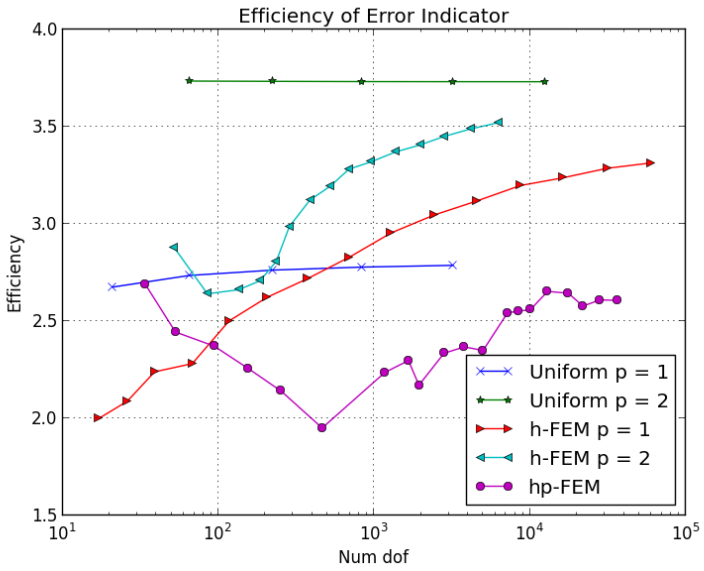


Figure 5.24: Efficiency of error indicator for Problem A. (The first data point has been removed for readability.)

It is expected that in the two-component flow problems that are the motivation of this work, one will more often encounter steep but smooth gradients, such as in Problem B, than real singularities such as the one in Problem A. Large gradients may occur for instance close to boundary layers, or in the transition between the two components of the fluid. From this perspective, the choice of hp-adaptive method is deemed to be appropriate, despite its apparent difficulties to deal with singularities.

The performance of the error indicator η_K can be evaluated based on its *efficiency*, defined as the ratio of the global estimated error to the real error measured in the H^1-seminorm:

Definition 7. *The efficiency ϵ of an error indicator η_K is*

$$\epsilon = \frac{\left(\sum_K \eta_K^2\right)^{\frac{1}{2}}}{|u - u_h|_1}.$$

Fig. 5.24 and 5.25 show the efficiency of the error indicator computed during the tested extension strategies for Problem A and B, respectively. For the former problem, the indicator overestimates the error with a factor between two and four. The efficiency grows for the h-adaptive strategies, but flattens out as the number of unknowns increases. In Problem B, there is a very large overestimation for all strategies during the first steps. For $N > 100$, the efficiencies stabilize at levels of 6, 4 and 3 for the h-FEM strategy with degree 1 and 2, and the hp-FEM strategy, respectively. Overall, the error indication is best with the hp-FEM strategy for both problems.

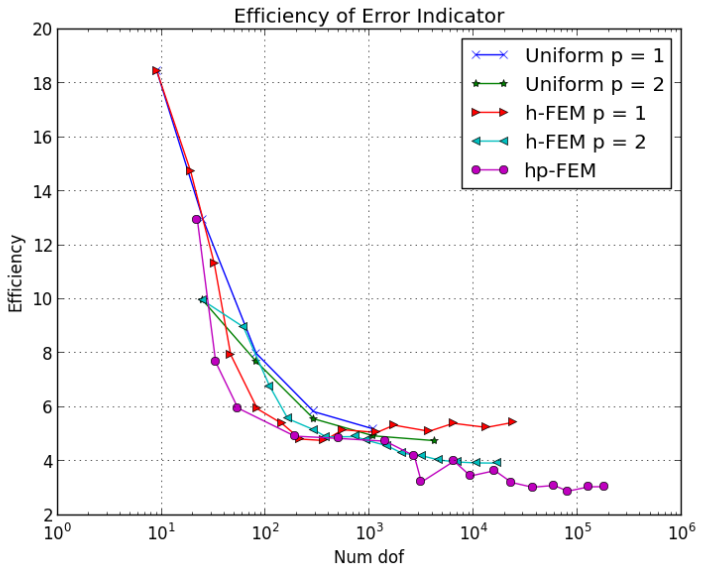


Figure 5.25: Efficiency of error indicator for Problem B. (The first data point has been removed for readability.)

In summary, the results of the tests with the Poisson problem support the conclusion that our hp-FEM implementation, which was described in Section 5.4, is working correctly. It was also demonstrated that the hp-adaptive strategy works. Its performance, albeit dependent on the problem and the values of the parameters, was acceptable in both cases.

5.6 HP-FEM IN THE DISCONTINUOUS GALERKIN SETTING

So far, the focus of this chapter, including the description of hp-FEM support in HiFlow^3 in Section 5.4, has been the use of hp-adaptive methods in a conforming, continuous Galerkin setting. In recent years, the use of non-conforming *discontinuous Galerkin* (DG) methods has received much attention. The main characteristic of this class of methods is that it makes use of approximation spaces which are not required to be subspaces of the test and solution spaces used in the continuous (as opposed to discrete) formulation of the problem. These approximation spaces are usually created by identifying the global basis functions with the local shape functions on each element: i. e. one does not impose that the global basis functions be continuous across the element intersections.

This is a major advantage of DG methods over continuous Galerkin (CG) methods. Since the support of the global basis functions is one element only, it is no longer necessary to impose continuity constraints as described in Section 5.4.6, and it is therefore easier e. g. to parallelize the numerical operations for this kind of space. Especially

in the context of hp-discretizations, the loosened requirements on DG approximation spaces greatly simplifies software implementations.

In order to ensure convergence toward the correct solution, it is most often necessary to modify the weak formulation that would be used in a CG setting, typically by introducing terms that penalize inter-element discontinuities of the solution. In many cases, it is possible to recover optimal asymptotic convergence rates through this type of modification. The penalizing terms often consist of integrals over the intersections between elements, in which the integrand contains jumps and averages of the solution or test functions. Several different approaches for defining the weak formulation have been discussed in the literature, e. g. the *Interior Penalty DG* methods, *Runge-Kutta DG* methods, *Local DG* methods and *Hybridized DG* methods.

The interest in DG methods for the present work is motivated by two factors. Firstly, the relatively high complexity of the implementation of support for hp-adaptivity for CG methods makes the possibly simpler and more flexible DG framework attractive, especially for extensions to parallel solvers for 3D problems. Secondly, DG methods are generally well suited to problems of hyperbolic character, such as the advection equation which describes the evolution of the level set, or Maxwell's equations, which could be used to model the optical properties of the devices that we are interested in.

Due to time constraints, it was not possible to experiment with DG methods for the two-component flow model which is the focus of this work. As a first step in this direction, however, the remainder of this section discusses the application of an interior penalty DG method to solve a class of elliptic diffusion-reaction problems. This development is based on a pair of articles [113, 114] which prove exponential convergence of certain hp-FEM extension processes for such a problem.

We shall not go into great detail regarding the implementation of the functionality that enables the discretization and solution of problems with the DG methodology. To a large part, it builds upon the functionality in version 1.2 of the existing HiFlow^3 library. Some extensions were added to allow the type of discretizations described in [114]:

- *anisotropic* Lagrange tensor product finite elements with different polynomial degrees in each coordinate direction;
- *anisotropic* refinements of hexahedral cells, i. e. splitting the cells into two or four hexahedral sub-cells instead of eight;
- customizable refinement ratios, allowing children cells to have different sizes;
- modified assembly procedure, which allows the integration of jumps and averages over element intersections.

5.6.1 *Diffusion-Reaction Problem*

We consider the following diffusion-reaction boundary value problem for an unknown variable u, which is the focus of the theoretical study in [113, 114]:

$$\begin{aligned} -\nabla \cdot (A\nabla u) + cu &= f && \text{in } \Omega \\ u &= g && \text{on } \partial\Omega. \end{aligned} \tag{5.43}$$

Here, $A \in \mathbb{R}^{3x3}$ is assumed to be a constant symmetric positive definite matrix, $c \in \mathbb{R}$ a non-negative constant, and the right-hand side function f a member of $H^{-1}(\Omega)$. $\Omega \subset \mathbb{R}^3$ is an open bounded polyhedron with Lipschitz boundary $\partial\Omega$. The solution is sought in the space $U = \{u \in H^1(\Omega) : u|_{\partial\Omega} = g\}$.

This problem (with $g = 0$) was analyzed in [113, 114], for finite element discretizations using variational formulations based on the Interior Penalty Discontinuous Galerkin framework. It was shown in [113], that under the assumption that f is analytic in $\bar{\Omega}$, it is possible to obtain exponential convergence rates with this formulation using locally varying anisotropic σ-geometric meshes and anisotropic polynomial degree distributions, for sequences of approximation subspaces that are generated using a fixed set of extensions, which are described in [114]. By the assumption of analyticity of f, the only source of irregularity of the solution is the existence of corners and edges in the domain.

5.6.2 *Interior Penalty Formulation*

In this section, we describe the hp-DG Interior Penalty discretization of the boundary value problem (5.43). We first introduce the finite element spaces. Let $\mathcal{M}$ be a mesh consisting of trilinearly mapped hexahedral cells, and let $\boldsymbol{p} : K \in \mathcal{M} \to (p_x, p_y, p_z)$ be an assignment of polynomial degrees in the $x-$, $y-$ and $z-$ directions. We denote by $\Phi_K : \hat{K} \in \mathcal{M} \to K$ the geometric mapping from the hexahedral reference cell $[0,1]^3$ to the cell K. The finite element with degree vector $\boldsymbol{p}$ is defined on $\hat{K}$ via the polynomial space

$$Q^{\boldsymbol{p}} = \{x^i y^j z^k, 1 \le i \le p_x, 1 \le j \le p_y, 1 \le k \le p_z\}. \tag{5.44}$$

The global discontinuous finite element space is then defined as

$$V_{hp} = \{v \in L^2(\Omega) : (v|_K \circ \Phi_K) \in Q^{\boldsymbol{p}(K)}(\hat{K}), \forall K \in \mathcal{M}\}. \tag{5.45}$$

This DG-formulation makes use of jump and average operators over faces of the mesh. We introduce the notation $\mathcal{F} = \mathcal{F}_I \cup \mathcal{F}_B$ for the set of faces in the mesh, which is a union of interior and boundary faces. For each interior face $f \in \mathcal{F}_I$, there are two cells K_f^1 and K_f^2

containing the face, whereas for boundary faces $f \in \mathcal{F}_B$, there is only one cell K_f. We denote by $\boldsymbol{n}_f^1$ and $\boldsymbol{n}_f^2$ the outward unit normal vectors on K_f^1 and K_f^2, respectively, for interior faces; and by $\boldsymbol{n}_f$ the outward unit normal vector for a boundary face. With this notation, we can define the jumps $[\![\cdot]\!]$ and averages $\ll \cdot \gg$ of scalars v and vectors $\boldsymbol{w}$ for an interior face as

$$[\![v]\!] = v|_{K_f^1}\, \boldsymbol{n}_f^1 + v|_{K_f^2}\, \boldsymbol{n}_f^2, \qquad [\![\boldsymbol{w}]\!] = \boldsymbol{w}|_{K_f^1} \cdot \boldsymbol{n}_f^1 + \boldsymbol{w}|_{K_f^2} \cdot \boldsymbol{n}_f^2, \tag{5.46}$$

$$\ll v \gg = \frac{1}{2}\left(v|_{K_f^1} + v|_{K_f^2} \right), \quad \ll \boldsymbol{w} \gg = \frac{1}{2}\left(\boldsymbol{w}|_{K_f^1} + \boldsymbol{w}|_{K_f^2} \right). \tag{5.47}$$

For a boundary face, these operators degenerate to

$$[\![v]\!] = v|_{K_f}\, \boldsymbol{n}_f, \qquad [\![\boldsymbol{w}]\!] = \boldsymbol{w}|_{K_f} \cdot \boldsymbol{n}_f, \tag{5.48}$$

$$\ll v \gg = v|_{K_f}, \qquad \ll \boldsymbol{w} \gg = \boldsymbol{w}|_{K_f}. \tag{5.49}$$

The variational problem then reads

$$\text{Seek } u \in V_{hp} \text{ such that} \tag{5.50}$$

$$a(u, \psi) = \int_\Omega f \psi \, \mathrm{d}x, \qquad \forall \psi \in V_{hp}, \tag{5.51}$$

with the bilinear form $a(u, \psi)$ defined by

$$\begin{aligned} a(u, \psi) = &\int_\Omega \left((A \nabla u) \cdot \nabla \psi + c u \psi \right) \mathrm{d}x \\ &- \int_{\mathcal{F}} \ll A \nabla u \gg \cdot [\![\psi]\!] \, \mathrm{d}s \\ &+ \theta \int_{\mathcal{F}} \ll A \nabla \psi \gg \cdot [\![u]\!] \, \mathrm{d}s \\ &+ \gamma \int_{\mathcal{F}} \alpha [\![\psi]\!] \cdot [\![u]\!] \, \mathrm{d}s. \end{aligned} \tag{5.52}$$

In this expression, the parameter θ makes it possible to choose between different interior penalty methods: $\theta = -1$ gives the SIP method, which is symmetric, and $\theta = 1$ gives the NIP version, which fulfills a coercivity property. $\gamma > 0$ is a stabilization parameter, which should be set to a constant value, whereas the dependence on the approximation space is controlled by the function α, defined as

$$\alpha(f) = \begin{cases} \dfrac{\max\left(p_1^\perp, p_2^\perp\right)^2}{\min\left(h_1^\perp, h_2^\perp\right)}, & f \in \mathcal{F}_I, \\ \dfrac{\left(p^\perp\right)^2}{h^\perp}, & f \in \mathcal{F}_B. \end{cases} \tag{5.53}$$

Here, $p_1^\perp, p_2^\perp$ and $p^\perp$ correspond to the polynomial degrees in the direction perpendicular to the face f in the neighboring cell(s), and $h_1^\perp, h_2^\perp$ and $h^\perp$ are the corresponding diameters of the cells in the perpendicular direction.

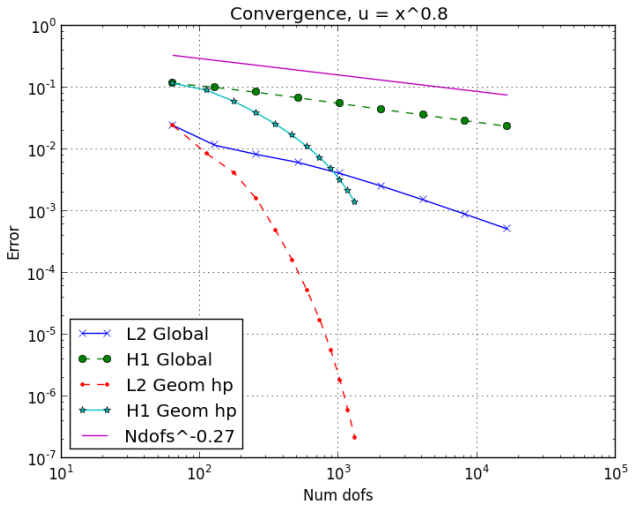


Figure 5.26: Comparison of global h-refinement ($\sigma = 0.5$) with local hp-refinement ($\sigma_x = 0.25$).

5.6.3 *Numerical Test*

As a first numerical test of this discretization, we let $\Omega = [0,1]^3$, $A = \mathcal{I}$, $c = 0$, and f and g be such that the exact solution $u = x^{0.8}$. The variation is thus entirely in the x-direction, and we therefore first consider anisotropic refinement toward the yz-plane. The initial mesh has eight cube cells, created by splitting Ω uniformly around its center.

For this test, we used the NIP ($\theta = 1$) version, and omitted the stabilization by setting $\gamma = 0$. Furthermore, the Dirichlet boundary conditions were imposed strongly on the solution and test spaces, and not via a penalty term as described in Section 5.6.2.

The gradient of the manufactured solution has a singularity on the plane $x = 0$, and is therefore not a member of $H^2(\Omega)$, but only $H^{1+4/5}(\Omega)$. With linear elements, and global refinement of the cells perpendicular to the x-axis, one expects the convergence rate of the H^1-norm of the error to be limited to $\mathcal{O}((h^\perp)^{0.8})$. It has however also been established that refining geometrically toward a singularity, combined with a linear increase of the polynomial degree away from it, will lead to exponential convergence in certain cases (see [56] for an analysis of the corresponding one-dimensional problem). Such a local refinement will lead to a *geometrically graded* mesh. The grading can be controlled by varying the bisection factor $\sigma_x \in (0,1)$, which is the position of the splitting plane on the reference cell ($\sigma_x = 0.5$ corresponds to the usual bisection in the middle of the cell, which is what is used throughout the rest of this work).

Fig. 5.26 compares the convergence of the errors measured in the L^2-norm and H^1-seminorm for global h-refinement perpendicular to

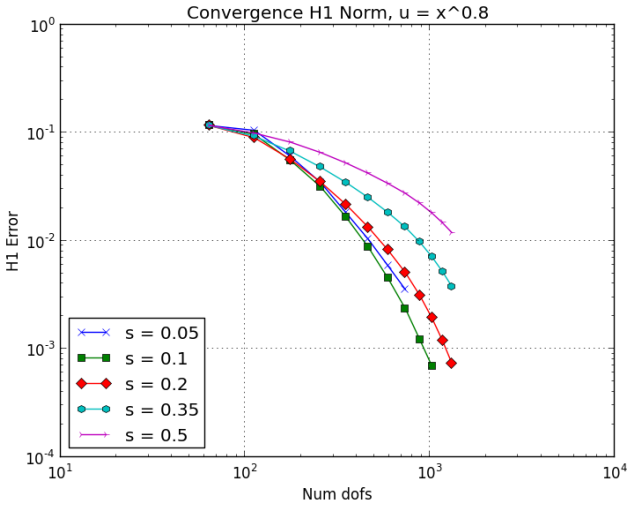


Figure 5.27: Convergence in H^1-seminorm with local hp-refinement for different values of σ_x.

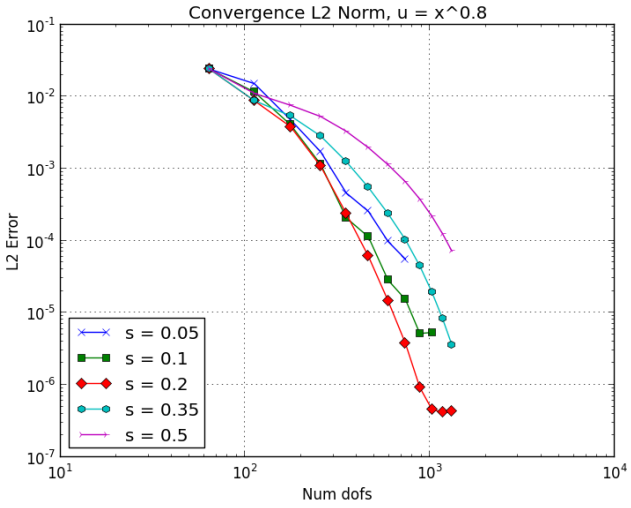


Figure 5.28: Convergence in L^2-norm with local hp-refinement for different values of σ_x.

the x-axis with $\sigma_x = 0.5$ and $\boldsymbol{p} = (1,1,1)$; and local hp-refinement, where only those cells that intersect the yz-plane were refined with $\sigma_x = 0.25$ in each step, whereas the polynomial degree p_x was increased by one for the other cells.

For comparison, the plot $N^{-0.27}$ (N number of unknown degrees of freedom) is also shown in Fig. 5.26, which shows that the slope of the H^1-seminorm convergence curve for global h-refinement corresponds to the expected value $-0.8/3 = -0.27$. This expected slope can be derived from the fact that $N \propto h^{-3}$. That the convergence for the local hp-refinement is exponential is clear from the graph.

Fig. 5.27 and 5.28 show the convergence of the error in the H^1-seminorm and L^2-norm, respectively, using the local hp-refinement with different values of the parameter σ_x. The convergence is exponential in all cases, but faster for smaller values of σ_x, down to $\sigma_x = 0.1$. For the smallest value 0.05, the trend breaks down, probably due to the increasingly large condition numbers of the system matrix. The convergence in the L^2-norm is also limited by this effect.

6

HP-ADAPTIVE FLOW SIMULATION OF A MICROFLUIDIC WAVEGUIDE

This chapter presents the application of the two-component fluid flow model and the hp-adaptive finite element method described in previous chapters to the problem of numerically simulating the flow in a microfluidic waveguide. Section 6.1 describes an example of such a device, and discusses the additional modeling choices that were made for this particular problem. Next, the details of the adaptive numerical method are presented in Section 6.2. The level set model and the correct functioning of the corresponding parts of the solver are verified for a simple test problem in Section 6.3. The numerical results for the waveguide problem are presented in Section 6.4 together with a discussion on future improvements of method. The final section gives an outlook on the possibility of combining the current work with simulation of the optics of the device, by presenting an eigenvalue problem that can be solved to compute characteristic optical properties of the waveguide.

6.1 PROBLEM DESCRIPTION AND MODELING

The problem considered in this chapter is based on the channels used for the liquid-core, liquid-cladding optical waveguides whose construction and characterization were described in [140]. In addition to creating reconfigurable waveguides, that article also describes a three-way optical switch and an evanescent coupler, both of which have functions that are controlled via the fluid flow.

One of the major advantages of using liquid-liquid interfaces in the design of optical components is the ease with which the devices can be reconfigured. The cited work demonstrates how, by modifying the relative inflow rates of the liquids in the system, the waveguide can be switched between single-mode and multi-mode operation. Furthermore, when used as an optical switch, the output port can be selected by a similar mechanism. A further advantage is that the performance of the devices is not very sensitive to manufacturing quality of the microfluidic structures that it operates in. In contrast to solid state devices, it is easy to achieve very smooth interfaces with liquids. This smoothness can be achieved even in the presence of relatively rough walls.

In this chapter, we focus on the simulation of the fluid flow in a straight waveguide. The geometry, shown in Fig. 6.1, includes a horizontal inflow channel for the core liquid, and slanted inflow channels

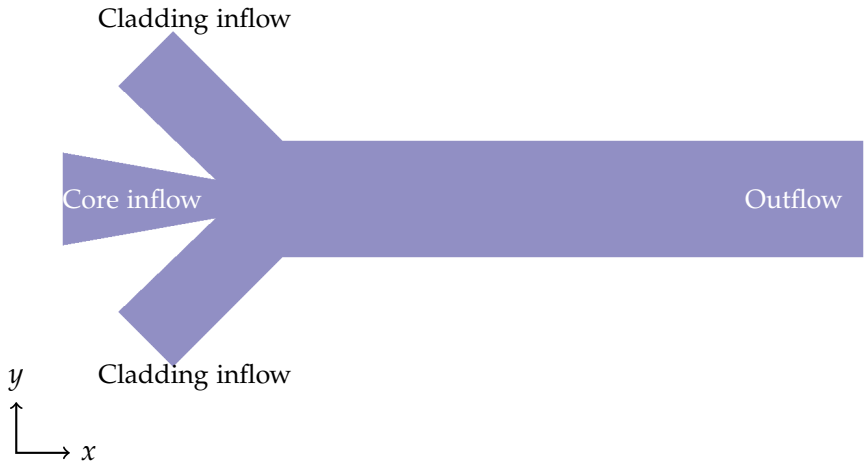


Figure 6.1: Schematic of simulated part of optofluidic waveguide.

for the upper and lower cladding liquids. These three channels join into the waveguide channel, which has a length of 1.5 mm. This represents the first part of the waveguide described in [140], which is 5 mm long. The height of the channel in the z-direction, which is neglected in our two-dimensional model, is assumed to be 100 μm.

In order to make use of floating point numbers accurately, we introduced rescaled units of length that are better suited to simulations at the micrometer scale. The basic quantities that are part of the model are length, mass and time, for which we introduce the units e_L, e_M, e_T, respectively. These are defined in terms of the SI units as follows:

$$e_L = 10^{-4}\ \mathrm{m}, \qquad e_M = 10^{-9}\ \mathrm{kg}, \qquad e_T = 10^{-2}\ \mathrm{s}. \tag{6.1}$$

The units were chosen to ensure that the values of the input data is neither very small nor very large in magnitude. We have the following conversion rules for density, dynamic viscosity, and speed:

$$1\ \mathrm{g}\cdot\mathrm{cm}^{-3} = 1\ e_M e_L^{-3}, \tag{6.2}$$

$$1\ \mathrm{Pa}\cdot\mathrm{s} = 10^3\ e_M e_L^{-1} e_T^{-1}, \tag{6.3}$$

$$1\ \mathrm{m}\cdot\mathrm{s}^{-1} = 10^2\ e_L e_T^{-1}. \tag{6.4}$$

In the literature on optofluidic devices, it is common that the flow rate, i. e. the volume of fluid that passes a given point per unit of time, is reported instead of the speed, since the former can be measured directly. To determine the inflow boundary condition, it is necessary to convert between the flow rate and the inflow speed. To this end, we assume that the flow profile at the inflow has a three-dimensional parabolic profile, such as would be obtained with a linear pressure distribution in a long pipe with a circular cross-section

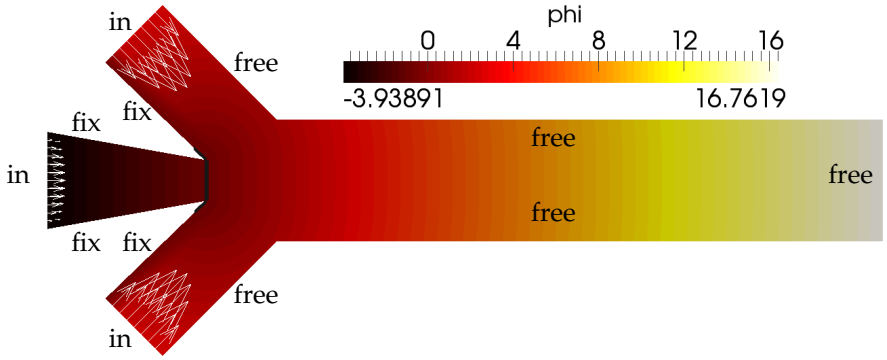


Figure 6.2: The velocity field $\boldsymbol{u}$ (arrows), level set function ϕ (color), and interface Γ (solid line) at $t = 0$.

(Hagen-Poiseuille flow, see e. g. [116]). One can then derive the following relation between the inflow rate R and the maximum speed U_m:

$$R = \frac{4WH}{9} U_m \qquad \Longleftrightarrow \qquad U_m = \frac{9R}{4WH}, \tag{6.5}$$

where W is the width of the channel in the xy-plane, and H its size in the z-direction. We assume $H = 1\, e_L = 10^{-4}$ m everywhere, whereas W varies between the different inflow channels.

The actual boundary inflow velocity was chosen to have a two-dimensional parabolic profile with maximum speed U_m and to be directed orthogonally to the corresponding inflow boundary. At the outflow boundary, the natural boundary condition (4.33) was used, and at the remainder of the boundary $\boldsymbol{u} = 0$ was imposed. The effect of the wetting properties of the materials was not included in the model. As initial condition, we simply extended the velocity field at the boundary by zero into the rest of the domain, as shown in Fig. 6.2.

For the level set function ϕ, the initial condition was chosen based on the initial interface $\Gamma(0)$, which is shown in Fig. 6.2. The initial interface is situated at the end of the core inflow section of the channel, and $\phi^0(\boldsymbol{x})$ is defined as the shortest distance from $\boldsymbol{x}$ to $\Gamma(0)$.

We used the initial condition also to define the boundary condition for ϕ, by simply imposing the function to be constant in time on a part of the boundary. According to the model formulated in previous chapters, the value of ϕ only has to be fixed on the inflow boundaries, which are marked with *in* in Fig. 6.2. During the computations, we found that expanding the part of the boundary where ϕ was kept at a prescribed value to also include the sections marked with *fix* significantly improved the solution quality, by helping to suppress oscillations which would appear close at these boundaries. These artifacts were likely related to the sharp re-entrant corners where the inflow channels meet. On the rest of the boundary, marked with *free* in Fig. 6.2, no boundary conditions were imposed.

	Dyn. Viscosity μ	**Density** ρ	**Refractive Index** n
Core	$8.85 \cdot 10^{-3}\,\mathrm{Pa} \cdot \mathrm{s}$	$1.39\,\mathrm{g} \cdot \mathrm{cm}^{-3}$	1.445
Cladding	$8.9 \cdot 10^{-4}\,\mathrm{Pa} \cdot \mathrm{s}$	$1.0\,\mathrm{g} \cdot \mathrm{cm}^{-3}$	1.335

Table 6.1: Values of material parameters of the core and cladding fluids.

The material parameters for the fluids were chosen in accordance to the fluids used in [140], see Table 6.1. In that work, the cladding fluid was de-ionized water (H_2O), and the core fluid was a 5M aqueous solution of $CaCl_2$. The refractive index is listed for reference, but not used in the computation.

In contrast to the assumption made for the two-component flow model described in previous chapters, these two fluids are not immiscible. This is a drawback for their use as medium for the liquid-liquid waveguide and related devices described in [140], since mixing of the fluids will lead to the broadening and eventual disappearance of the interface. It also degrades the optical characteristics of the devices. To deal with this drawback, the speed of the flow is chosen large enough, so that the interface remains reasonably well-defined over that part of the interface which is utilized for the optical functionality.

From this perspective, the immiscible two-component flow model is an idealization in which it is assumed that mixing effects do not play a major role. By setting the coefficient $\tau = 0$, the introduction of an interfacial tension force which would not be present between the actual fluids is avoided. On the other hand, with the current model, it is also possible to evaluate the potential advantages of using immiscible fluids, by choosing $\tau > 0$. In this case, the interface would remain sharp in reality, but one would have to deal with the undesirable formations of bubbles and other instabilities of the interface due to the additional forces.

6.2 HP-ADAPTIVE COMPUTATION

In order to obtain an accurate representation of the interface Γ at a moderate computational cost, we used an hp-adaptive strategy to successively extend the approximation spaces. There are several variables involved in the coupled flow-interface problem, and ideally one would like to find a strategy that is efficient for all of them. Such a strategy is likely to be quite complex, however, and therefore some simplifications were made in the present work.

Firstly, we used the same mesh for the approximation spaces for all variables. This simplifies the coupling between the flow and interface models, by avoiding the need for transfer of solutions between meshes. Since the evolution of the flow is not of primary interest, we furthermore restricted the extension possibilities by using fixed polynomial degrees which correspond to the classic Taylor-Hood el-

ement (quadratic elements for the velocity, and linear elements for the pressure). As mentioned at the end of Section 4.4, this combination is known to be stable, something that would have to ascertained explicitly if arbitrary degrees were allowed.

Secondly, in order to be usable as an optical device, the distribution of the core and cladding fluids should reach a stationary or *steady* state after which the interface stops evolving in time. Since we are using a time-stepping method to approach the steady state successively, a criterion is needed to determine when the iteration should be stopped.

During the computations, we observed that convergence to the stationary state proceeded much more slowly close to the boundary of the channel than in the middle. This can be explained by the non-uniform distribution of flow speeds in the channel. Due to the quadratic inflow profile, the flow is slower close to the boundary than in the middle, and the corresponding propagation of the information related to the boundary conditions from the inflow boundaries to the outflow therefore takes longer. Since it is mainly the evolution close to the interface in the middle of the channel which is of interest, the condition of convergence to the stationary state was relaxed by solving instead until a fixed time T, and then taking this as an approximation of the stationary solution. It was verified manually that the shape of the interface did not vary significantly over the last time-steps.

Finally, we restricted ourselves to an error indicator which is only connected to the error of the interface problem. By virtue of using a least-squares formulation, there is a simple error indicator associated to this problem, namely the cell integral of the residual. At a given time-step n, we define the error indicator for each cell $K \in \mathcal{M}$ to be

$$\eta_{K,n}^2 = \int_K R(\phi^n, \phi^{n-1}, \boldsymbol{u}^n, \boldsymbol{u}^{n-1})^2 \,\mathrm{d}x, \tag{6.6}$$

where

$$R(\phi^n, \phi^{n-1}, \boldsymbol{u}^n, \boldsymbol{u}^{n-1}) = \mathcal{L}\phi^n - f = \\ (\phi^n + \theta\Delta t \boldsymbol{u}^n \cdot \nabla\phi^n) - \left(\phi^{n-1} - (1-\theta)\Delta t \boldsymbol{u}^{n-1} \cdot \nabla\phi^{n-1}\right)$$

is the residual of the semi-discretized advection equation. Under the assumption that the norm of the linear part $\|\mathcal{L}\phi^n\|_0$ is equivalent to the solution norm $\|\cdot\|_1$, the error indicator η_K^2 will be both reliable and locally efficient, making it a good candidate for controlling the adaptive method. The use of the cell residual as a posteriori error estimator for least-squares FEM is discussed e. g. in [1] and [25, Chapter 12].

The adaptation was performed at the end of each instationary computation, when the solution was considered to have converged to a stationary state. The error measure used in the adaptation process

was the value of the error indicator (6.6) at the last step of the time-stepping iteration. The residual R, and hence also the error indicator $\eta_{K,n}^2$, can be interpreted as having two parts. Rewriting R as

$$R(\phi^n, \phi^{n-1}, \boldsymbol{u}^n, \boldsymbol{u}^{n-1}) = \left(\phi^n - \phi^{n-1}\right) + \left(\theta \Delta t \boldsymbol{u}^n \cdot \nabla \phi^n + (1-\theta)\Delta t \boldsymbol{u}^{n-1} \cdot \nabla \phi^{n-1}\right), \quad (6.7)$$

separates the instationary contribution $(\phi^n - \phi^{n-1})$ from the stationary contribution $\left(\theta \Delta t \boldsymbol{u}^n \cdot \nabla \phi^n + (1-\theta)\Delta t \boldsymbol{u}^{n-1} \cdot \nabla \phi^{n-1}\right)$. At a stationary state, $\phi^n = \phi^{n-1} = \phi$ and $\boldsymbol{u}^n = \boldsymbol{u}^{n-1} = \boldsymbol{u}$, in which case the residual reduces to $R_{\text{stat}}(\phi, \boldsymbol{u}) = \Delta t \boldsymbol{u} \cdot \nabla \phi$. This residual corresponds to the part of the error which arises from the discretization in space alone. It is weighted compared to the instationary term $R_{\text{instat}} = (\phi^n - \phi^{n-1})$ in the full residual with the time-step Δt. In accordance with this separation of the residual, we introduce the instationary and stationary error indicators

$$(\eta_{K,n}^{\text{instat}})^2 = \int_K \left(\phi^n - \phi^{n-1}\right)^2 \, \mathrm{d}x, \quad (6.8)$$

and

$$(\eta_{K,n}^{\text{stat}})^2 = \Delta t^2 \int_K (\boldsymbol{u}^n \cdot \nabla \phi^n)^2 \, \mathrm{d}x, \quad (6.9)$$

respectively. These quantities can be used to determine whether the source of the error is mainly the temporal or the spatial discretization. In the numerical results which will follow we adapt only the spatial discretization, and hence the stationary error indicator will be the most interesting. It is however important to keep in mind also the instationary contributions, since in the results which will be presented, the solution was not allowed to converge to a stationary state everywhere in the domain.

In its entirety, this choice of discretization and adaptive process cannot be expected to yield the full benefits of hp-adaptive FEM, in particular the very high rates of convergence that is expected for this class of methods. Here, the time-discretization is second order for both parts of the model, and the spatial discretization of the velocity is also restricted to second order. Since stationary solutions are sought, the first constraint is not of great concern, whereas the second is indeed a potential limiting factor for the efficiency of the method.

The use of an hp-FEM for the level set function is motivated in this case by the fact that this function only has to be approximated well in one part of the domain, namely close to the interface. By combining cell refinement and locally varying polynomial degrees, it should be possible to construct approximation spaces that are well suited to the requirements of each particular problem with the help of an adaptive algorithm.

Extending the current approach to include hp-adaptivity also for the flow model, and using more sophisticated error control for the time evolution, is an obvious direction for future work. The use of hp-adaptive methods with strongly coupled models such as the one considered in this work is still an area that needs to be explored in more detail, and the simplified method proposed here should be seen as a step in this direction.

6.3 VERIFICATION OF THE LEVEL SET COMPUTATION FOR A TEST PROBLEM

The program that was developed for the waveguide simulation is quite complex, and it is therefore important to try to verify that it functions correctly. We use the well-established method of solving a problem with a known, manufactured solution, and verifying that the analytically predicted rates of convergence of the error are achieved.

In this case, we did not manage to construct an analytic solution of the full model, and therefore restricted the investigation to the advection equation which governs the evolution of the level set function. The test problem consisted of solving a variation of this PDE with a uniform, constant velocity field $\boldsymbol{u} = (1, 0.75)$ on the domain $\Omega = [0,1]^2$. An extra right hand side function f was introduced to force the level set to approach a steady state solution chosen as

$$\phi^s(\boldsymbol{x}) = \sin(4\pi x)\sin(4\pi y), \qquad \boldsymbol{x} \in \Omega. \tag{6.10}$$

The test problem was then posed as follows:

$$\frac{\partial \phi(\boldsymbol{x},t)}{\partial t} + \boldsymbol{u}\cdot\nabla\phi(\boldsymbol{x},t) = f(\boldsymbol{x}), \qquad \boldsymbol{x} \in \Omega, \tag{6.11}$$

$$\phi(\boldsymbol{x},t) = 0, \qquad \boldsymbol{x} \in \partial\Omega_{\text{in}}, \tag{6.12}$$

$$\phi(\boldsymbol{x},0) = 0, \qquad \boldsymbol{x} \in \Omega, \tag{6.13}$$

where $f(\boldsymbol{x}) = \boldsymbol{u}\cdot\nabla\phi^s(\boldsymbol{x})$.

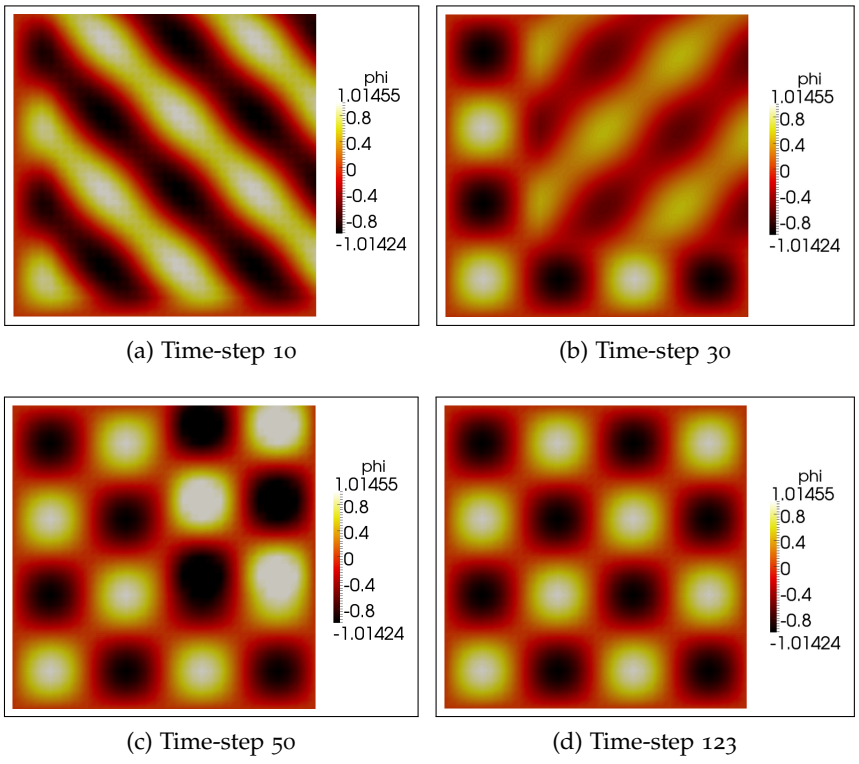
The inflow boundary $\partial\Omega_{\text{in}} = \{\boldsymbol{x} \in \partial\Omega : \boldsymbol{u}\cdot\boldsymbol{n}(\boldsymbol{x}) < 0\}$, where $\boldsymbol{n}(\boldsymbol{x})$ is the outward normal to the boundary $\partial\Omega$.

The problem was discretized with the method described in Section 4.3. The force term f results in this case in an extra contribution

$$\int_\Omega \left(\Delta t f(\boldsymbol{x})\left(\xi_i + \theta\Delta t\boldsymbol{u}\cdot\nabla\xi_i\right)\right)\,\mathrm{d}x, \tag{6.14}$$

which is added to the load vector b_i. If the stationary state is attained at time t_s, then $\frac{\partial\phi(\boldsymbol{x},t)}{\partial t} = 0$, and $\boldsymbol{u}\cdot\nabla\phi(\boldsymbol{x},t) = \boldsymbol{u}\cdot\nabla\phi^s(\boldsymbol{x})$ for $t \geq t_s$.

The program performing the simulation of the waveguide with the full model was adapted so that the test problem could be solved instead. The code responsible for advancing the level set was not modified, in order to ensure that the test was effective. Using a time-step



(a) Time-step 10 (b) Time-step 30

(c) Time-step 50 (d) Time-step 123

Figure 6.3: Evolution toward steady state of the solution to the test problem.

of $\Delta t = 0.01$, the time-stepping procedure was iterated until the solution had reached a steady state. This was determined by two stopping criteria, by which the relative changes in the ℓ_2-norm of the vector of degrees of freedom as well as the computed H^1-seminorm of the error between the current and the previous time-steps were both required to be smaller than 10^{-6}.

The problem was solved on a set of uniformly refined meshes with polynomial degrees $p = 1, \ldots, 4$, and the L^2-norm and H^1-seminorm of the error $\phi^s - \phi_h$ were computed. The coarsest mesh had 64 cells, and the finest 2048. Fig. 6.3 shows the evolution of the solution toward steady state at refinement level 4 with $p = 2$.

Optimal error estimates can be proven for this case, (see [91, Section 9.3]), meaning that the error should decrease as $\mathcal{O}(h^{p+1})$ in the L^2-norm, and $\mathcal{O}(h^p)$ in the H^1-seminorm, respectively, since the manufactured solution is smooth.

Fig. 6.4 and 6.5 show the convergence of the error measured in the L^2-norm and H^1-seminorm, respectively, as a function of the cell width h. The corresponding experimental convergence rates that are listed in Table 6.2 were computed by fitting the logarithm of the computed error against the logarithm of the cell width h. As can be seen, the theoretically predicted convergence rate is attained in each case, which is a strong indication that the implementation of the part of the solver that is responsible for the level set computation is correct.

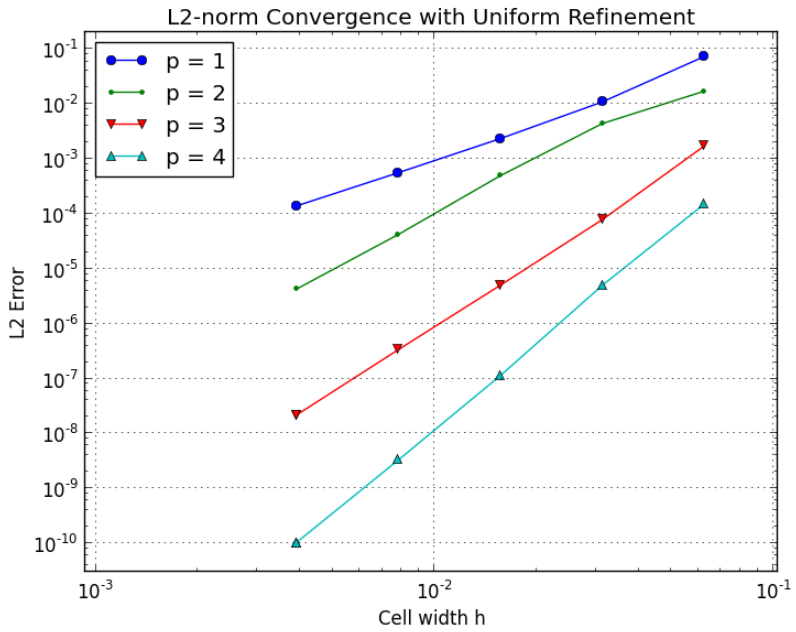


Figure 6.4: L^2-norm of the error as a function of cell width h for the stationary solution of the test problem.

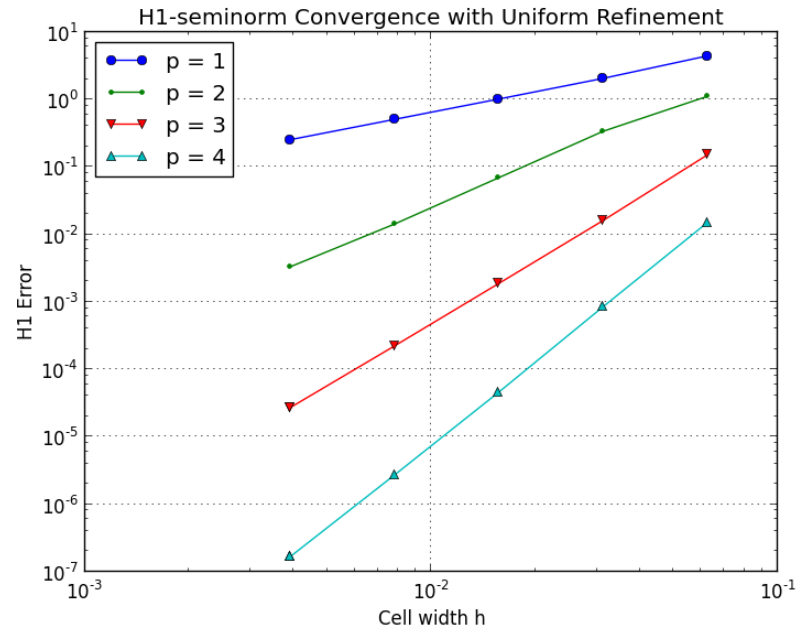


Figure 6.5: H^1-seminorm of the error as a function of cell width h for the stationary solution of the test problem.

Polynomial degree	Convergence Rate L^2	Convergence Rate H^1
1	2.24	1.03
2	3.07	2.14
3	4.03	3.10
4	5.14	4.11

Table 6.2: Experimentally determined convergence rates for the test problem with smooth solution.

6.4 NUMERICAL RESULTS FOR WAVEGUIDE MODEL

We now turn to the numerical simulation of the waveguide, presenting results from computations first with uniform refinement, and then with the hp-adaptive method. The scenario considered has inflow rates $R_{\text{core}} = 5\,\mu\text{L}\cdot\text{min}^{-1}$ and $R_{\text{clad}} = 20\,\mu\text{L}\cdot\text{min}^{-1}$ for the core and cladding, respectively. The corresponding inflow speeds are $U_{\text{core}} = 0.785\,e_L e_T^{-1}$ and $U_{\text{clad}} = 3.75\,e_L e_T^{-1}$. No interfacial tension force is applied in this case.

Fig. 6.6 gives an impression of the evolution of the solution from shortly after the initial state, until the final steady state.

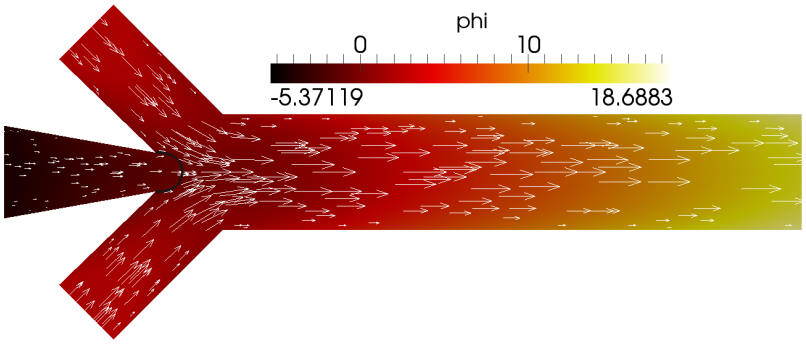
6.4.1 *Uniform Refinement*

As a first experiment, we solved the waveguide problem on a set of uniformly refined meshes, with different uniform polynomial degree distributions for the level set function. The coarsest mesh (refinement level 1) had 220 cells, and each further refinement multiplied this number by 4. The instationary problem was solved in each case until the fixed final time $T = 12$, with time-step size $\Delta t = 0.1$.

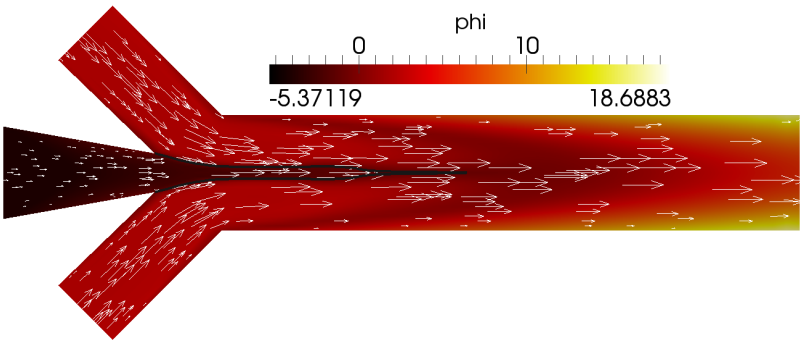
At the final time, the solution had not yet completely converged to steady state in the vicinity of the upper and lower no-slip boundaries in the main part of the channel. This is reflected in Fig. 6.7, which shows the distribution of the error indicator $\eta_{K,n}$ at the final time for refinement level 3 and $p = 1$. Toward the corners of the outflow boundary, the indicated error is large, since the solution has not completely converged there. The rest of the error is concentrated at the part of the channel where the three inflow streams meet. Here the most complex flow interaction takes place, so it is to be expected that the error is large.

Fig. 6.8 shows the distribution of the instationary error indicator $\eta_{K,n}^{\text{instat}}$. As was pointed out earlier, this part of the error is concentrated to the boundary of the domain.

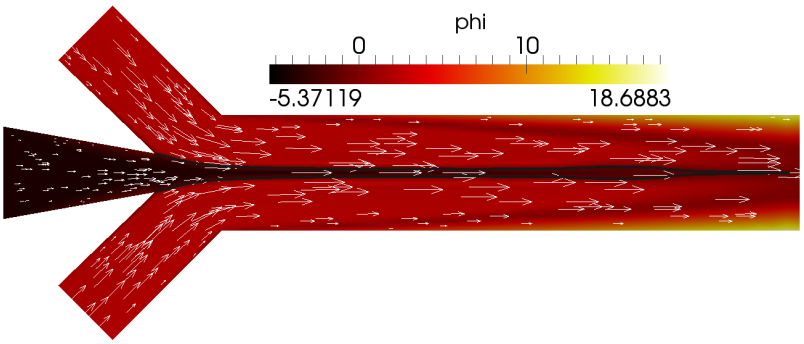
We investigated the convergence of the simulation on uniformly refined meshes, with different polynomial degrees. The problem was solved for refinement levels 1 – 5 with $p = 1,\ldots,5$, and the global error indicator $\eta = \sqrt{\sum_{K\in\mathcal{M}} \eta_{K,n}^2}$ was computed at the final time-



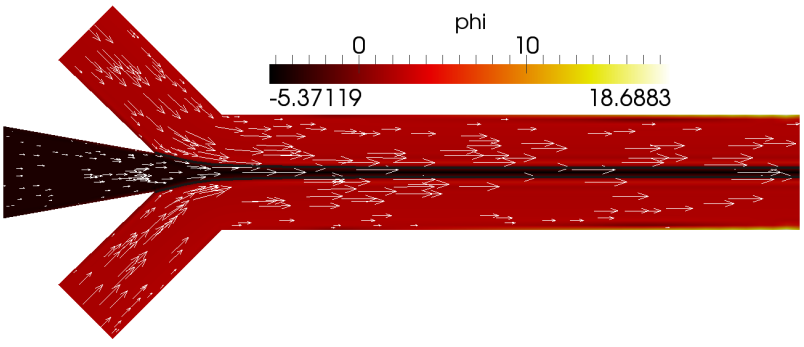
(a) Time-step 5



(b) Time-step 20



(c) Time-step 35



(d) Time-step 120

Figure 6.6: Solution of computed waveguide scenario at four different time-steps, showing the evolution toward steady state. The color scale corresponds to ϕ, the arrows to $\boldsymbol{u}$ and the solid line to the interface Γ.

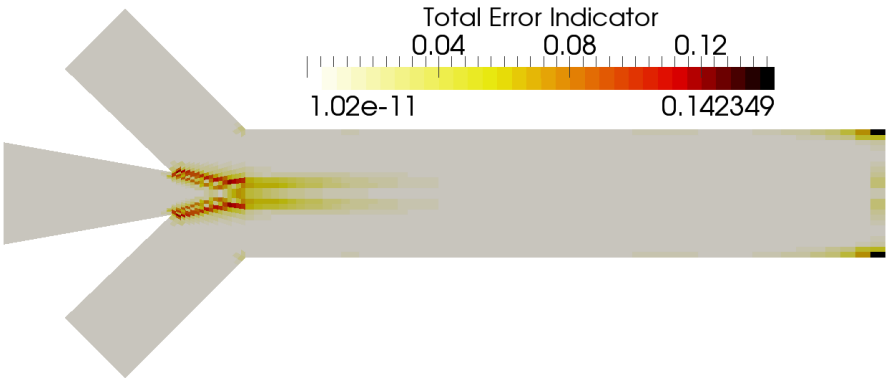


Figure 6.7: Distribution of error indicator $\eta_{K,n}$ after the final time-step for refinement level 3 with $p = 1$.

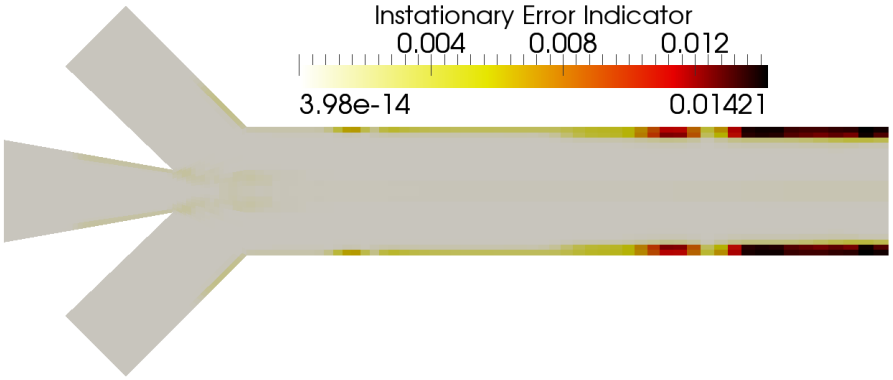


Figure 6.8: Distribution of instationary error indicator $\eta_{K,n}^{\text{instat}}$ after the final time-step for refinement level 3 with $p = 1$.

Polynomial degree	Experimental Convergence Rate
1	0.27
2	0.43
3	0.45
4	0.52

Table 6.3: Experimentally determined convergence rates of the error indicators with uniform refinement.

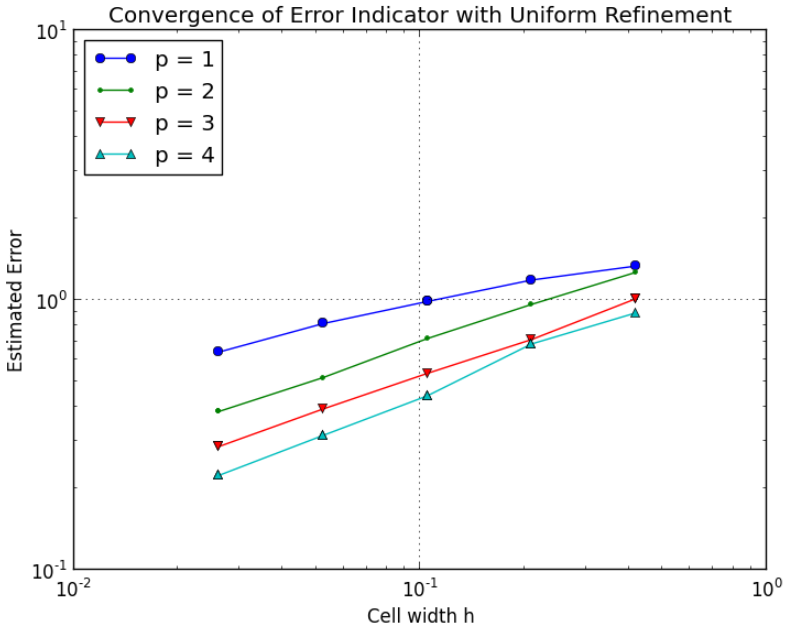


Figure 6.9: Convergence of the error indicators for uniform refinement with different polynomial degrees.

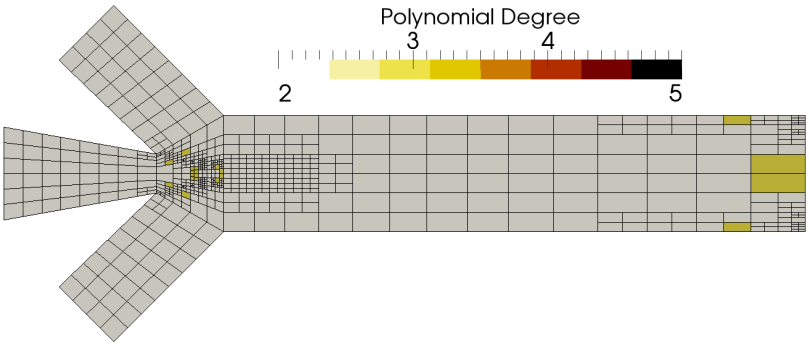
step. Fig. 6.9 shows η as a function of the maximum cell width h. The corresponding experimentally determined convergence rates r were computed as the slope of a linear fit of $\log \eta$ against $\log h$, to determine r in the expression $\eta = C \cdot h^r$. The results are listed in Table 6.3.

The convergence rates are very low in this case, even when increasing the polynomial degree. As can be seen in Fig. 6.7, the error contributions are localized, mainly to the interface between the fluids, as well as close to the outflow boundary. This motivates the use of an hp-adaptive method, with which one can hope to detect those parts of the domain where the solution is so irregular that refinement is required, while achieving fast convergence through use of high-order elements away from these parts.

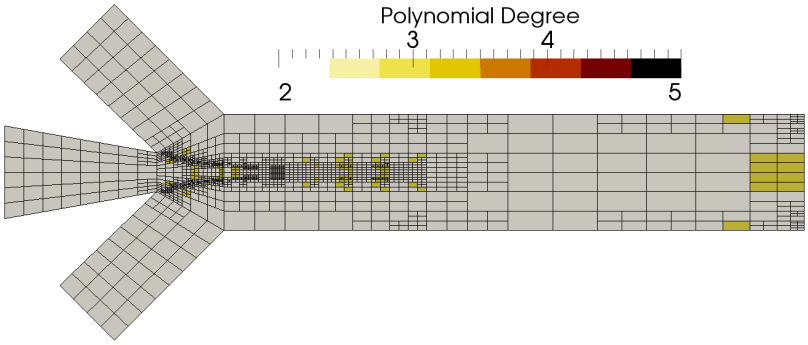
6.4.2 *hp-Adaptive Refinement*

In the first hp-adaptive computations, we used the Melenk-Wohlmuth adaptation strategy described in Chapter 5 (see Figure 5.1) together with the error indicator (6.6). The simulation time was chosen to be $T = 7.5$ and the time-step size was $\Delta t = 0.1$.

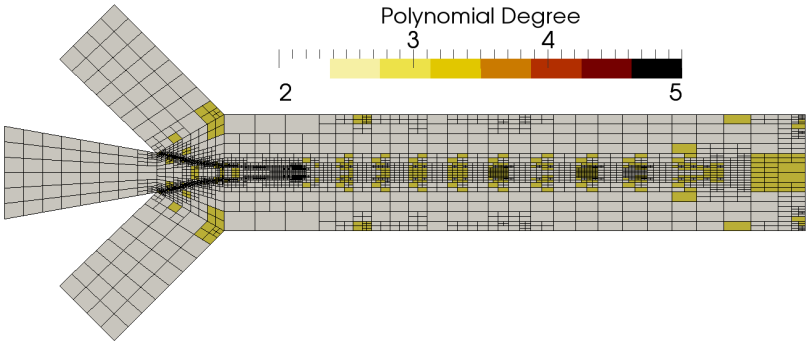
The result of the adaptation process is illustrated in Fig. 6.10, which shows the mesh and the distribution of polynomial degrees at different steps adaptation steps. As is to be expected, strong refinement occurs near to the interface, as well as in the region where the flows from the inflow channels meet. There is also much refinement to-



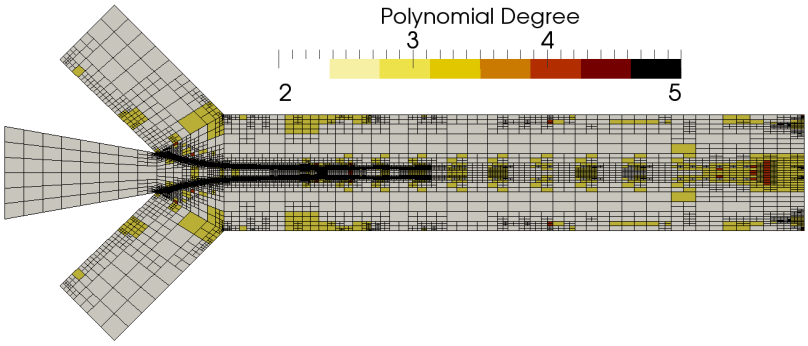
(a) Adaptive step 4



(b) Adaptive step 8



(c) Adaptive step 16



(d) Adaptive step 20

Figure 6.10: Evolution of the mesh and distribution of the polynomial degrees of ϕ for selected steps in the adaptive process, using the original Melenk-Wohlmuth strategy. Note that the degree scale has been adjusted: the actual maximum degree is 4.

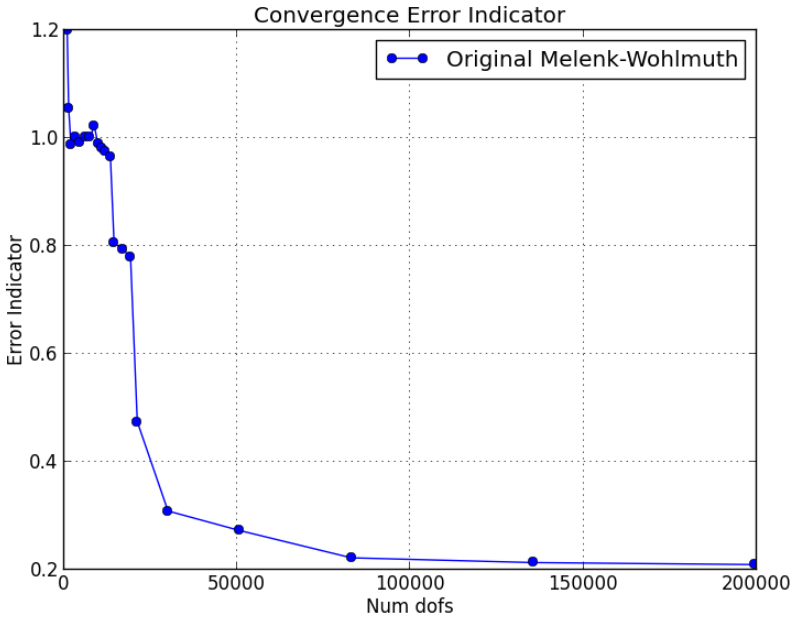


Figure 6.11: Evolution of error indicator with the number of degrees of freedom for the original Melenk-Wohlmuth adaptive strategy.

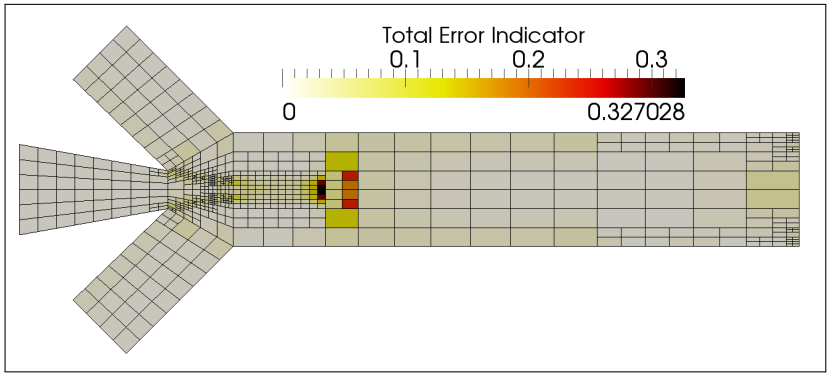
ward the boundaries. The polynomial degrees are not increased very much: starting from an initial degree of 2, the maximum attained in the computation is 4.

The evolution of the error indicator at the last time-step as a function of the number of degrees of freedom used for the hp-adapted finite element space for the level set function is shown in Fig. 6.11.

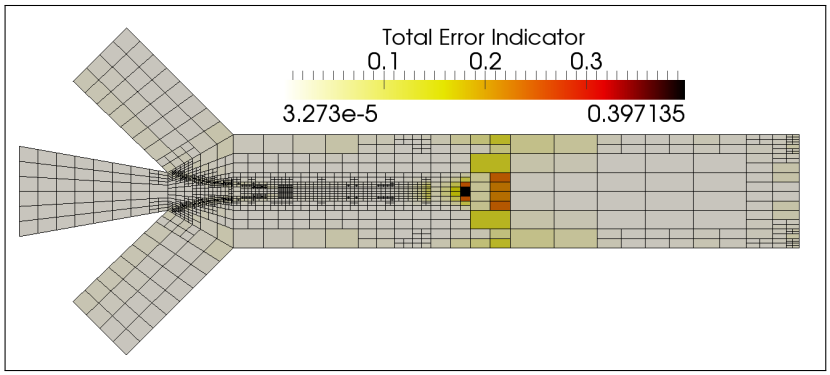
It is clear from the graph that the adaptation process is not working optimally. The error decrease is not completely monotonous, and flattens out when the number of degrees of freedom grows large. Furthermore, between adaptive steps 3 and 12, very little progress is made in reducing the error.

Looking at the distribution of the local error indicators, shown for selected adaptive steps in Fig. 6.12, reveals the source of the problem. At step 4, the error is concentrated around a transition from a region of more refinement to one of less refinement, with several irregular edges present. Four steps later, this configuration has only been transported further along the channel, without any decrease of the maximum error. Only after 16 adaptive steps, when the peak in the error is almost at the end of the channel, is there a substantial decrease in the error. As can be seen in the last image, the same problem returns at finer levels.

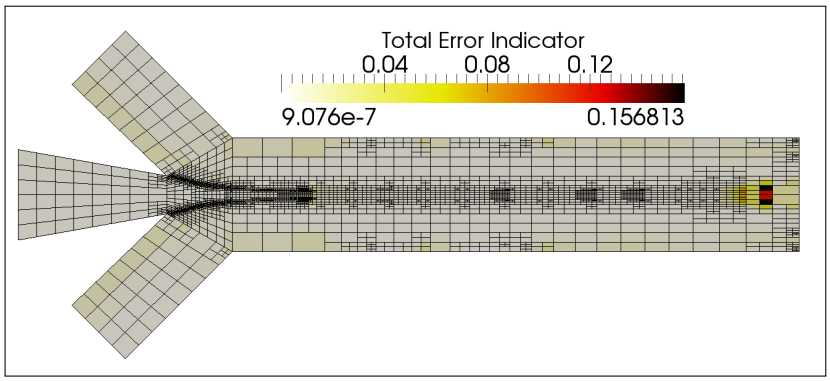
Hence, it seems that the cause of the peaks in the error distribution can be traced to the enforcement of continuity constraints over the vertical irregular edges. We interpret this phenomenon as follows. The effect of enforcing such a constraint is that the values of the de-



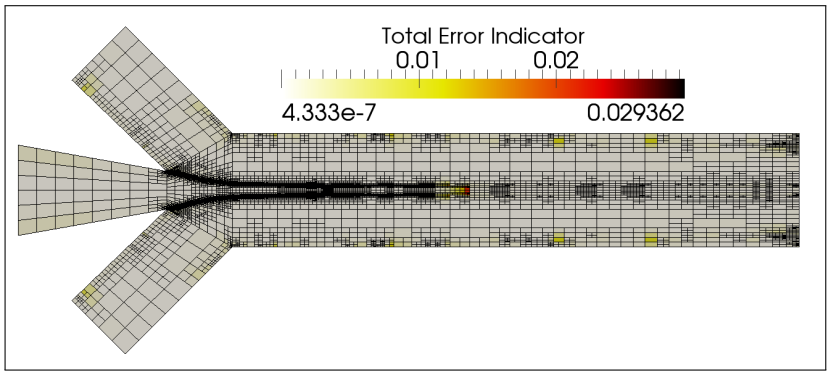
(a) Step 4



(b) Step 8



(c) Step 16



(d) Step 20

Figure 6.12: Distribution of cell error indicators at steady-state for selected adaptive steps.

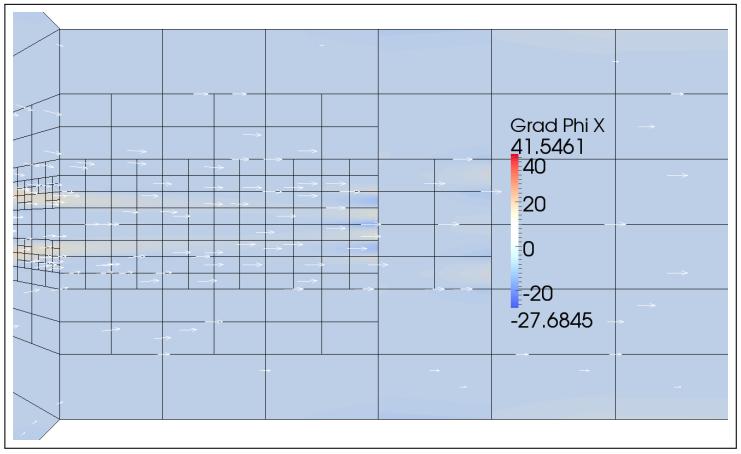


Figure 6.13: The gradient of the level set is prevented from becoming orthogonal to the horizontal velocity field due to the continuity constraints. The scale shows the horizontal component of $\nabla\phi$.

grees of freedom on the smaller side are fixed to weighted averages of those on the larger side. This causes a transfer of information in the direction parallel to the irregular edge. For hyperbolic equations, the flow of information when the exact solution evolves follows the characteristics of the equation, but due to the continuity constraints, this property is not conserved for the approximate solution. This manifests itself in components of $\nabla\phi$ which are not orthogonal to the velocity. Fig. 6.13 shows x-component of $\nabla\phi$ that appears on the smaller side of the irregular edge. This yields a large contribution to the error indicator through the term $\boldsymbol{u} \cdot \nabla\phi$, especially in the middle of the channel, where $\|\boldsymbol{u}\|$ is large.

From another perspective, this interpretation can be understood by considering the error as arising from the use of an approximation space that is H^1-conforming, and not allowing approximations in $S \setminus H^1$. An ideal approximation space would only impose continuity in the direction of $\boldsymbol{u}(\boldsymbol{x}, t)$, and not in all directions. At the technical level, however, working with such an approximation space would be very challenging.

Although the error indicator correctly captures the occurrence of these artifacts, the enrichment algorithm is unable to avoid them in an effective way. As is clear from Fig. 6.12, the mesh regularization necessary to resolve the irregularities only takes place indirectly: as the mesh is further refined, and the one-irregularity rule is imposed, large error concentrations can be pushed toward the end of the channel. When the refinement along the entire middle of the channel has reached a certain level, the artifacts reappear with smaller magnitude at a finer level.

In an attempt to improve the speed of convergence, the adaptive strategy was modified slightly. In order to avoid excessive refinement during the phases where little error decrease takes place, the thresh-

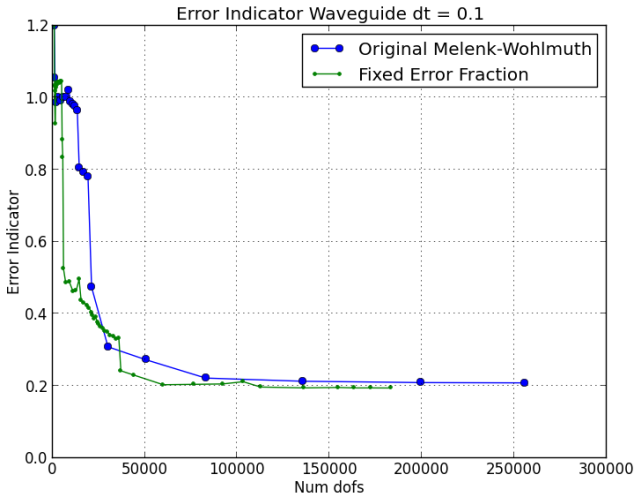


Figure 6.14: Comparison of the original and the modified adaptive strategy, and their performance with respect to convergence of the error indicator.

old for enrichment was changed to be based on a fixed total error fraction instead of the mean cell error in the original strategy. This approach is essentially the same as the *Fixed Energy Fraction* method for h-adaptivity which was described in Section 5.3.1. In this case, the corresponding minimal set $\mathcal{R}$ was computed such that

$$\sum_{K \in \mathcal{R}} \eta_{K,n}^2 \geq \theta^2 \sum_{K \in \mathcal{M}_h} \eta_{K,n}^2. \tag{6.15}$$

The first condition in Figure 5.1 was then replaced with a threshold equal to the smallest value of the error indicators for the cells in $\mathcal{R}$. The Melenk-Wohlmuth mechanism for deciding between h- and p-refinement was retained. The performance of the two strategies are compared in Fig. 6.14. The modified strategy seems to be more economical at the beginning of the adaptation process, but later the two methods yield similar results. Furthermore, the modified strategy uses a very large number of steps, which limits its efficiency in terms of computation time.

Despite the fact that the adaptive strategies are hindered by the artifacts, which limit the convergence of the error indicator, they still outperform the use of uniform refinement, which was investigated in Section 6.4.1. Fig. 6.15 compares the convergence of the error indicators in terms of the number of degrees of freedom. Except for the plateau where the error level is around 1.0, the adaptive methods require much fewer degrees of freedom to obtain a solution with a given error level. The most accurate solutions attained ($\eta \approx 0.2$) required c. $6 \cdot 10^4$ and $8 \cdot 10^4$ for the modified and original Melenk-Wohlmuth strategies, respectively, which can be compared to $9 \cdot 10^5$ for uniform refinement with $p = 4$. It is thus clear, that the use of

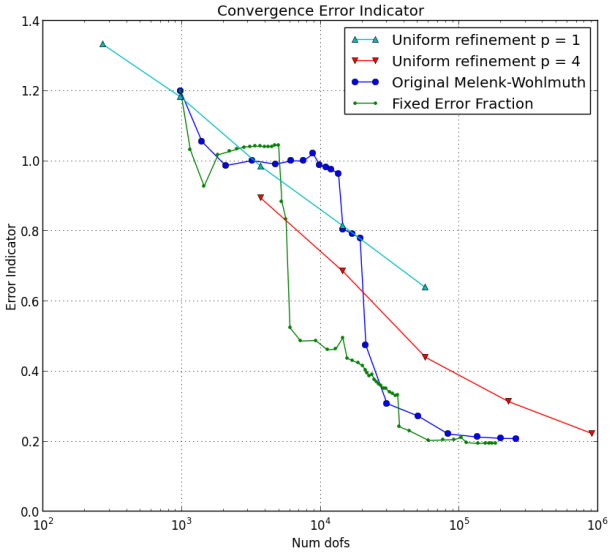


Figure 6.15: Convergence of error indicator for uniform and hp-adaptive refinements. The uniform refinements correspond to the results in Section 6.4.1 with polynomial degrees 1 and 4. A logarithmic scale was chosen for the horizontal axis to make the graph more readable.

hp-adaptivity is beneficial in this case, albeit the limitations that were discussed earlier.

6.4.3 *Future Directions for the Adaptive Computation*

Several different improvements to address the shortcomings of the hp-adaptive computation are possible, but could not be made in the context of the present work due to time constraints.

Since the major source of errors is the occurrence of hanging nodes and the related constraints, working with more regular meshes could yield a significant improvement. A first possibility would be to use triangular meshes or meshes with cells of mixed type together with a regularizing procedure. One would then have to add support for triangular elements in the hierarchical basis framework described in Section 5.4. Another method would be to modify the existing refinement procedure to allow for elimination of hanging nodes in a larger domain in each step. Since the problem only occurs at certain irregular edges, a specialized procedure that incorporates a priori information about the main flow direction can ensure that refinements orthogonal to this direction are regularized. In this context, the use of anisotropic refinements could also be beneficial.

Another possible approach is to apply post-processing to the solution for the purpose of computing the error indicator. One way to do

this would be to smoothen the function, with the goal of filtering out its high-frequency components. This could help reduce the discontinuities of the gradients shown in Fig. 6.13, and thereby decrease the impact of these artifacts. The smoothing could be accomplished using e. g. the Gauss-Seidel method. The error indicator would then be computed from the filtered function, which could then be discarded.

The choice of boundary conditions should also be investigated further. As can be seen in Fig. 6.12, the adaptation yields quite strong refinement close to the boundaries. What happens at the horizontal boundaries of the main channel should not have a great impact on the shape of the core fluid, since the flow is essentially parallel to these boundaries; therefore refinements in these parts of the domain should be considered unwanted. Effectively, the approximation of the level set function is only required to be accurate close to the interface between the fluids, whereas further away it suffices that its sign is correct.

The outflow boundary requires more careful consideration. Here, there is also strong refinement, especially toward the corners. This is possibly related to the chosen natural boundary condition for the flow, which produces a velocity field with a non-zero vertical component. With a different boundary condition it might be possible to obtain an outflow condition which is more similar to the flow in the interior of the domain, and thereby reduce the need for fine mesh resolution at the end of the channel.

Finally, an aspect which is often discussed in the context of the Level Set method is the use of *re-distancing* methods, which periodically modify the level set function to ensure that it is approximately equal to the distance function with respect to the interface. As discussed shortly in Section 3.3, the two most prominent methods for re-distancing are the solution of the *reinitialization equation* (3.15), and the *fast marching method*. The use of a re-distancing method would have the benefit of providing a more regular level set function, which could be expected to yield better results.

For one, it would likely be possible to use higher polynomial degrees instead of strong refinement in the vicinity of the interface, and the material parameters could be computed more accurately. Furthermore, re-distancing provides a natural way of updating the boundary conditions for the level set function as the interface involves in time, by computing them based on the current position of the interface. This could alleviate some of the difficulties discussed above.

Re-distancing also has some drawbacks, mainly related to the fact that the computed solution is modified in a strong way. Apart from possibly introducing additional approximation errors which are difficult to estimate and control, it is not clear how the modification of the function interacts with the error estimation and the adaptive

process. Furthermore, it incurs an additional cost, and increases the complexity of the solution process and the corresponding software.

6.5 OPTICAL CHARACTERIZATION OF THE WAVEGUIDE

In this section, we shortly outline a possible procedure for determining the optical characteristics of the microfluidic waveguide. Given the material properties of the two liquids, and the shape of the interface between them, the propagation of electromagnetic waves in the waveguide is governed by Maxwell's equations [84]. The full set of equations involves time-dependent PDEs for four complex vector-valued field variables, which have to be completed with constitutive relations. Here, we will make a number of simplifying assumptions, and derive a simple eigenvalue equation that can be used to approximately quantify the optical behavior of the waveguide. This type of analysis is commonly performed in electrical and optical engineering. A through treatment of the theory of electromagnetic waves and waveguides can be found in [81].

The assumptions are listed below, along with the simplifications that they imply:

- The fluids are assumed to be *linear*, *homogeneous*, and *isotropic* with respect to their optical properties. The corresponding material laws can then be used to reduce the number of independent field variables to two: the *electric* field $\tilde{\boldsymbol{E}}(\boldsymbol{x}, t) \in \mathbb{C}$ and the *magnetic* field $\tilde{\boldsymbol{H}}(\boldsymbol{x}, t) \in \mathbb{C}$. Furthermore, the material properties can be described with two scalar variables: the *electric permittivity* ϵ and the *magnetic permeability* μ. These are assumed to be constant in time.

- We restrict the setting to one where the waveguide does not contain any electric charges or currents.

- The time-dependence is taken to be *harmonic* with a single, fixed frequency ω, which corresponds to monochromatic light. The fields can then be written on the form $\tilde{\boldsymbol{E}}(\boldsymbol{x}, t) = \boldsymbol{E}(\boldsymbol{x})e^{-\imath\omega t}$ and $\tilde{\boldsymbol{H}}(\boldsymbol{x}, t) = \boldsymbol{H}(\boldsymbol{x})e^{-\imath\omega t}$. Since Maxwell's equations are linear, the full fields can be reconstructed from the time-harmonic solutions with a Fourier representation.

- Only solutions which propagate along the main axis of the waveguide, so-called *guided modes*, will be considered. The coordinate system is chosen so that this coincides with the x-direction. Furthermore, it is assumed that the waveguide is translation-invariant, i. e. its cross-section is the same for all values of x. This enables a further separation of the fields with respect to the spatial variables: $\tilde{\boldsymbol{E}}(\boldsymbol{x}, t) = \boldsymbol{E}(y, z)e^{-\imath\omega t+\imath\beta x}$ and $\tilde{\boldsymbol{H}}(\boldsymbol{x}, t) = \boldsymbol{H}(y, z)e^{-\imath\omega t+\imath\beta x}$.

Under these assumptions, Maxwell's equations for $\tilde{\boldsymbol{E}}$ and $\tilde{\boldsymbol{H}}$ can be formulated as follows:

$$\nabla \times \tilde{\boldsymbol{E}} = -\mu \frac{\partial \tilde{\boldsymbol{H}}}{\partial t}, \qquad \nabla \cdot \tilde{\boldsymbol{E}} = 0, \tag{6.16}$$

$$\nabla \times \tilde{\boldsymbol{H}} = \epsilon \frac{\partial \tilde{\boldsymbol{E}}}{\partial t}, \qquad \nabla \cdot \tilde{\boldsymbol{H}} = 0. \tag{6.17}$$

By taking the curl of the first equation, and substituting the third, we can derive a wave equation for $\tilde{\boldsymbol{E}}$:

$$\nabla \times (\nabla \times \tilde{\boldsymbol{E}}) = -\mu \frac{\partial}{\partial t} (\nabla \times \tilde{\boldsymbol{H}}) \tag{6.18}$$

$$= -\mu\epsilon \frac{\partial^2 \tilde{\boldsymbol{E}}}{\partial t^2} \tag{6.19}$$

$$\Leftrightarrow \Delta \tilde{\boldsymbol{E}} - \mu\epsilon \frac{\partial^2 \tilde{\boldsymbol{E}}}{\partial t^2} = 0 \tag{6.20}$$

where we have used that $\nabla \cdot \tilde{\boldsymbol{E}} = 0$ when reducing the curl-curl operator to the Laplace operator. A similar wave equation can be derived for $\tilde{\boldsymbol{H}}$ analogously.

We now take into account the last assumption above, and consider only guided mode solutions to this equation. Thanks to the special dependence on t and x, the partial derivatives with respect to the corresponding variables can be substituted with multiplication by $-\imath\omega$ and $\imath\beta$, respectively. The wave equation can hence be reduced to a homogeneous Helmholtz equation for the amplitude $\boldsymbol{E}(y, z)$ of the field:

$$\Delta_T \boldsymbol{E} + \left(\omega^2 \mu\epsilon - \beta^2\right) \boldsymbol{E} = 0, \tag{6.21}$$

where $\Delta_T = \frac{\partial^2}{\partial y^2} + \frac{\partial^2}{\partial z^2}$.

Again, an analogous equation can be derived for $\boldsymbol{H}$. It can be shown (see e. g. [81, Section 3.6]) that it suffices to solve for the longitudinal components E_x, H_x, whereafter the transverse components of $\boldsymbol{E}, \boldsymbol{H}$ can be obtained through explicit formulas involving their derivatives. The equation of interest is therefore

$$\Delta_T E_x(y, z) + \left(\omega^2 \mu\epsilon - \beta^2\right) E_x(y, z) = 0. \tag{6.22}$$

The domain for this equation is a cross-section through the waveguide orthogonal to x. Depending on the shape of the microchannel in the third dimension z, the distribution of the two fluids in this plane will be different, which leads to different solutions of (6.22). At the interfacial surface between the fluids, the tangential component of $\tilde{\boldsymbol{E}}$ and $\tilde{\boldsymbol{H}}$, and hence also $\boldsymbol{E}$ and $\boldsymbol{H}$, must be continuous, whereas the normal component can be discontinuous. These conditions, together

	$E_x = 0$	$E_x \neq 0$
$H_x = 0$	TEM: Trans. Electromagnetic	TM: Trans. Magnetic
$H_x \neq 0$	TE: Trans. Electric	Hybrid

Table 6.4: Classification of guided modes in a waveguide.

with fact that the field must decay exponentially away from the interface in the cladding fluid in order for guidance to take place, restricts the solutions of (6.22) that yield guided waves.

The guided mode solutions are commonly classified according to whether E_x or H_x, or both, are identically zero. When this is the case, the corresponding amplitude vector $\boldsymbol{E}$ or $\boldsymbol{H}$ is transverse to the direction of propagation, which yields the nomenclature in Table 6.4.

For particular waveguide geometries, such as rectangular or circular cylinders, it is possible to find the solutions of (6.22) analytically, and determine what types of modes can be transmitted. In general, only a discrete set of modes are admissible, which can be characterized through *guidance conditions* that relate the dimensions of the waveguide and the material parameters to the modes that can propagate at a given frequency. In many cases, there is a *cutoff* frequency, below which a given mode cannot propagate at all.

Solutions to (6.4) can form the basis for evaluating the optical characteristics of the waveguide. From the simulation of the two-component fluid flow in two dimensions, the width of the core fluid can be extracted and used as the geometrical parameter in a model where either a rectangular or a circular distribution of the material is assumed. As mentioned above, this type of model can be investigated analytically. A more realistic characterization would require a three-dimensional simulation of the flow, to determine the correct material distribution, after which the guided modes can be determined by solving (6.4) numerically.

Coupling the flow simulation with optical characterization opens the door to further applications of this thesis. Of particular interest is the possibility to optimize the inflow velocity of the channel with respect to the existence of guided modes, in order to be able to control the optical device in a precise way.

7 CONCLUSION

In this final chapter of the thesis, we first summarize the main contributions of the work, and discuss its wider significance. This is followed by an outlook on the perspectives for future developments building on what has been presented here.

7.1 SUMMARY OF CONTRIBUTIONS

As described in the introduction to this work, there are several levels of abstraction in the methodology of Computational Science. The contributions of this thesis cover all of these levels, as shown in Fig. 7.1.

The motivation of the thesis was the desire to solve problems in the domain of optofluidics. We considered in particular the use of liquid-liquid interfaces to create optical devices. To be able to describe such devices mathematically, a model describing incompressible flow for immiscible fluids was created, building on the work of [54]. This model couples the incompressible Navier-Stokes equations with an interface description based on a level set function, and has the form of an initial boundary value problem for a pair of PDE:s posed on the entire domain under consideration. It takes into account the interfacial tension force, but neglects other effects, for instance those related to the wetting properties of the walls.

The mathematical model was discretized using an adaptive finite element method, that applies the hp-FEM technique to the level set variable in an attempt to efficiently obtain a highly accurate representation of the shape of the interface, which is considered to be the main quantity of interest. A least-squares variational formulation was used for the stabilization of the advection equation which describes the evolution of the level set function. The flow equations were discretized using standard Q_2/Q_1 finite elements and linearization by the Newton method. The flow variables were approximated on the same mesh as the level set function. For the latter, a successive enlargement of the approximation space based on the Melenk-Wohlmuth algorithm was used, with an error indicator derived from the least squares formulation. To the author's knowledge, the application of hp-adaptive FEM using the least-squares formalism has not been explored previously for two-component flow problems. In general, there are not many examples where hp-adaptive methods have been used for coupled models, and many challenges remain in this area.

Finally, the design and implementation of support for hp-adaptive techniques in a generic and reusable software framework made up

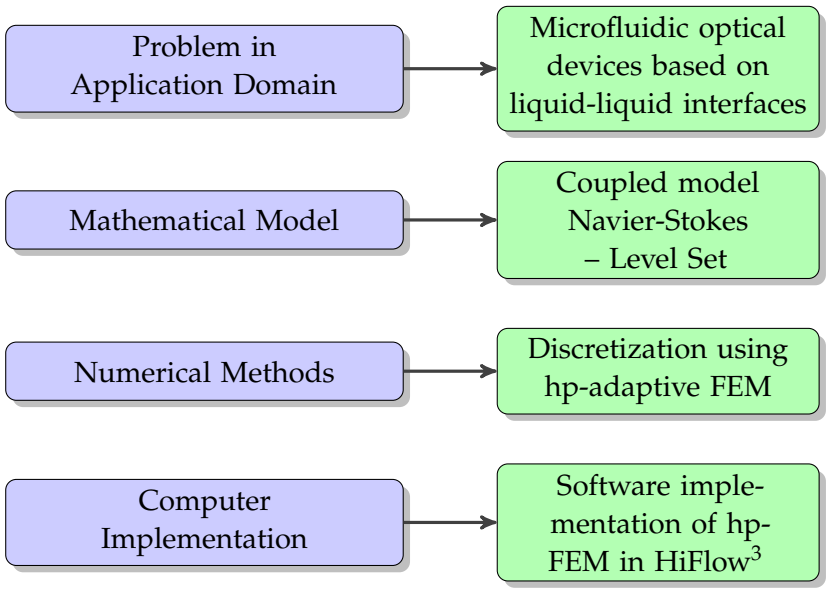


Figure 7.1: Contributions of thesis at the different levels of abstraction.

a major part of this work. The implementation combines techniques from several previous works, for instance the use of the hierarchical Lobatto basis advocated in [37, 38, 123], and the treatment of constraints presented in [17]. The resulting code provides, in our opinion, an appropriate balance between complexity of its use and maintenance on the one hand, and configurability and performance on the other. Although the extension is at present somewhat limited in functionality, being restricted to quadrilateral meshes and sequential programs, it uses an explicitly extensible design that allows it to be adapted to future use cases. An additional advantage of this implementation is the close integration with the other tools of the HiFlow^3 library, which allows users easy access to e. g. GPU-accelerated linear solvers.

An important aspect of any scientific work is its wider significance and applicability to other situations. Although this thesis was mainly motivated by the problem of simulating microfluidic optical devices based on liquid-liquid interfaces, a large part of its outcome at the lower three abstraction levels in Fig. 7.1 can be transferred to other situations. The model for two-component immiscible fluids is valid for many situations at macroscopic scale. The hp-adaptive techniques, and the corresponding software support provided in HiFlow^3, can also be used for problems arising in completely different domains of application. This is the main benefit of being able to work at the different levels of abstraction presented above.

7.2 OUTLOOK

The work presented in this thesis naturally leads to new challenging questions, which the author hopes to be able to pursue in the future. Furthermore, as with any major project, there were some aspects which could not be treated due to time constraints.

Starting with the first level in Fig. 7.1, the results achieved for the waveguide example in this work open the path to applying the same method to several other microfluidic optical devices, for instance the optical switch described in [140] or the lens presented in [129].

As for the model, an important step toward a more complete understanding of the optical devices would of course be to add a description of the optical phenomena in addition to the fluid flow. As was briefly outlined in Section 6.5, this could take the form of either a simplified wave model or the complete set of Maxwell's equations being solved using the geometry of the device computed with the model presented in this work. With this model at hand, the level set description of the geometry of the optical structures provides a good basis for solving numerical optimization problems with respect to the shape and topology of the devices. A further intriguing possibility would be to include a model for the impact of the electromagnetic field on the fluid flow, which has several important applications in Optofluidics.

The numerical methods presented in this work could also be developed in a variety of directions. Apart from the specific improvements for the adaptive computation that were discussed in Section 6.4.3, it is probable that the adaptive strategy and the error indicator for the level set employed in the current solver can be improved upon. A first step in this direction would be to include hp-FEM discretizations of the flow variables as well, and perhaps allow different approximation spaces for these and the level set function. The use of discontinuous Galerkin methods with hp-discretizations would also be an interesting alternative in this context.

Finally, the current software implementation could be extended in various aspects. In the author's opinion, the most intriguing question concerns the development of linear solvers and preconditioners that can made efficient on modern hardware by exploiting the high degree of parallelism that is achievable as long as the data can be made available close to the compute elements. Here, methods employing highly unstructured and dynamic data structures are at a clear disadvantage over those that use more predictable memory accesses. The challenges associated with the gap between increasing compute power and data access capabilities of modern hardware have to be met if finite element methods in general, and hp-adaptive methods in particular, are to remain competitive against other approaches.

BIBLIOGRAPHY

[1] J. H. Adler et al. "Efficiency Based Adaptive Local Refinement for First-Order System Least-Squares Formulations." In: *SIAM Journal on Scientific Computing* 33.1 (2011), pp. 1–24.

[2] M. Ainsworth. "A Preconditioner Based on Domain Decomposition for h-p Finite-Element Approximation on Quasi-uniform Meshes." In: *SIAM Journal on Numerical Analysis* 33.4 (1996), pp. 1358–1376.

[3] M. Ainsworth and J. T. Oden. *A Posteriori Error Estimation in Finite Element Analysis*. Pure and Applied Mathematics. New York: John Wiley & Sons, Inc., 2000.

[4] M. Ainsworth and B. Senior. "An adaptive refinement strategy for hp-finite element computations." In: *Applied Numerical Mathematics* 26.1-2 (1998), pp. 165–178.

[5] D. M. Anderson, G. B. McFadden, and A. A. Wheeler. "Diffuse-Interface Methods in Fluid Mechanics." In: *Annual Review of Fluid Mechanics* 30.1 (1998), pp. 139–165.

[6] H. Anzt et al. "HiFlow3 - A Flexible and Hardware-Aware Parallel Finite Element Package." In: *Proceedings of the 9th Workshop on Parallel/High-Performance Object-Oriented Scientific Computing (POOSC '10)*. ACM, 2010, pp. 1–6.

[7] H. Anzt et al. "HiFlow3 - A Multi-Purpose and Flexible Parallel Finite Element Package." In: *Open Source CFD International Conference*. 2011, pp. 1–15.

[8] D. N. Arnold, F. Brezzi, and B. Cockburn. "Discontinuous Galerkin Methods for Elliptic Problems." In: *Discontinuous Galerkin Methods*. Vol. 11. Lecture Notes in Computational Science and Engineering. Springer, 2000, pp. 89–101.

[9] I. Babuška and B. Q. Guo. "Regularity of the Solution of Elliptic Problems with Piecewise Analytic Data. Part I. Boundary Value Problems for Linear Elliptic Equation of Second Order." In: *SIAM Journal on Mathematical Analysis* 19.1 (1988), pp. 172–203.

[10] I. Babuška and W. C. Rheinboldt. "A-posteriori error estimates for the finite element method." In: *International Journal for Numerical Methods in Engineering* 12.10 (1978), pp. 1597–1615.

[11] I. Babuška and W. C. Rheinboldt. "Error Estimates for Adaptive Finite Element Computations." In: *SIAM Journal on Numerical Analysis* 15.4 (1978), pp. 736–754.

[12] I. Babuška and M. Suri. "The h-p version of the finite element method with quasiuniform meshes." In: *Mathematical Modelling and Numerical Analysis* 21.2 (1987), pp. 199–238.

[13] I. Babuška, B. A. Szabó, and I. N. Katz. "The p-Version of the Finite Element Method." In: *SIAM Journal on Numerical Analysis* 18.3 (1981), pp. 515–545.

[14] I. Babuška and M. Suri. "Locking effects in the finite element approximation of elasticity problems." In: *Numerische Mathematik* 62.1 (1992), pp. 439–463.

[15] I. Babuška and M. Suri. "The p and h-p versions of the finite element method, basic principles and properties." In: *SIAM Rev.* 36.4 (1994), pp. 578–632.

[16] W. Bangerth, R. Hartmann, and G. Kanschat. "deal.II — a General Purpose Object Oriented Finite Element Library." In: *ACM Trans. Math. Softw.* 33.4 (2007).

[17] W. Bangerth and O. Kayser-Herold. "Data Structures and Requirements for *hp* Finite Element Software." In: *ACM Trans. Math. Softw.* 36.1 (2009), pp. 1–31.

[18] R. E. Bank, A. H. Sherman, and A. Weiser. "Refinement Algorithms and Data Structures for Regular Local Mesh Refinement." In: *Scientific Computing Volume 1 IMACS Transactions on Scientific Computation*. Ed. by R. Stepleman et al. North-Holland, 1983, pp. 3–17.

[19] R. E. Bank and A. Weiser. "Some a posteriori error estimators for elliptic partial differential equations." In: *Math. Comp* 44 (1985), pp. 283–301.

[20] E. Bänsch. "Finite element discretization of the Navier-Stokes equations with a free capillary surface." In: *Numerische Mathematik* 88.2 (2001), pp. 203–235.

[21] A. C. Bauer and A. K. Patra. "Performance of parallel preconditioners for adaptive hp FEM discretization of incompressible flows." In: *Communications in Numerical Methods in Engineering* 18.5 (2002), pp. 305–313.

[22] M. Baumann. "Numerical Simulation of Tropical Cyclones using Goal-Oriented Adaptivity." PhD thesis. Karlsruhe Institute of Technology, 2012.

[23] R. Becker and S. Mao. "Quasi-Optimality of Adaptive Nonconforming Finite Element Methods for the Stokes Equations." In: *SIAM Journal on Numerical Analysis* 49.3 (2011), pp. 970–991.

[24] R. Becker and R. Rannacher. "An Optimal Control Approach to A Posteriori Error Estimation in Finite Element Methods." In: *Acta Numerica* (2001), pp. 1–102.

[25] P. Bochev and M. Gunzburger. *Least-Squares Finite Element Methods*. Applied Mathematical Sciences. Springer Science+Business Media, 2009.

[26] G. Böhme. *Strömungsmechanik nichtnewtonscher Fluide*. 2nd Edition. B. G. Teubner, 2000.

[27] L. Bos, M. A. Taylor, and B. A. Wingate. "Tensor Product Gauss-Lobatto Points Are Fekete Points for the Cube." In: *Mathematics of Computation* 70.236 (2001), pp. 1543–1547.

[28] S. Brenner and L. Scott. *The Mathematical Theory of Finite Element Methods*. Texts in Applied Mathematics. Springer, 2008.

[29] J. Brown. "Efficient Nonlinear Solvers for Nodal High-Order Finite Elements in 3D." In: *Journal of Scientific Computing* 45 (2010), pp. 48–63.

[30] W. Cao, H. W., and R. Russell. "An r-Adaptive Finite Element Method Based upon Moving Mesh PDEs." In: *Journal of Computational Physics* 149.2 (1999), pp. 221–244.

[31] H. Chen, X. Xu, and R. Hoppe. "Convergence and quasi-optimality of adaptive nonconforming finite element methods for some nonsymmetric and indefinite problems." In: *Numerische Mathematik* 116.3 (2010), pp. 383–419.

[32] H. G. Choi. "A least-square weighted residual method for level set formulation." In: *International Journal for Numerical Methods in Fluids* (2011).

[33] M. H. Cho, H. G. Choi, and J. Y. Yoo. "A direct reinitialization approach of level-set/splitting finite element method for simulating incompressible two-phase flows." In: *International Journal for Numerical Methods in Fluids* (2010).

[34] S. E. Chung et al. "Lab-on-a-Chip." In: *Handbook of Optofluidics*. Ed. by A. R. Hawkins and H. Schmidt. Boca Raton: CRC Press, 2010. Chap. 7.

[35] P. Ciarlet. *The Finite Element Method for Elliptic Problems*. Classics in Applied Mathematics. Society for Industrial and Applied Mathematics, 2002.

[36] A. Dedner, R. Klöfkorn, and D. Kröner. "Higher Order Adaptive and Parallel Simulations Including Dynamic Load Balancing with the Software Package DUNE." In: *High Performance Computing in Science and Engineering '09*. Ed. by W. E. Nagel, D. B. Kröner, and M. M. Resch. Springer-Verlag Berlin Heidelberg, 2010, pp. 229–239.

[37] L. Demkowicz. *Computing with hp-Adaptive Finite Elements, Vol. 1. One and Two Dimensional Elliptic and Maxwell Problems*. Chapman & Hall/CRC, 2007.

[38] L. Demkowicz et al. *Computing with hp-Adaptive Finite Elements, Vol. 2. Frontiers: Three Dimensional Elliptic and Maxwell Problems with Applications*. Chapman & Hall/CRC, 2008.

[39] L. Demkowicz, D. Pardo, and W. Rachowicz. *3D hp-Adaptive Finite Element Package (3Dhp90). Version 2.0.* Technical Report 14. TICAM, 2002.

[40] L. Demkowicz, W. Rachowicz, and P. Devloo. "A Fully Automatic hp-Adaptivity." In: *Journal of Scientific Computing* 17.1 (2002), pp. 117–142.

[41] L. Demkowicz and L. Vardapetyan. "Modeling of electromagnetic absorption/scattering problems using hp-adaptive finite elements." In: *Computer Methods in Applied Mechanics and Engineering* 152.1-2 (1998), pp. 103–124.

[42] W. Dörfler. "A Convergent Adaptive Algorithm for Poisson's Equation." In: *SIAM Journal on Numerical Analysis* 33.3 (1996), pp. 1106–1124.

[43] I. S. Duff and J. K. Reid. "The Multifrontal Solution of Indefinite Sparse Symmetric Linear." In: *ACM Trans. Math. Softw.* 9.3 (1983), pp. 302–325.

[44] J. Eaton, D. Bateman, and S. Hauberg. *GNU Octave Manual Version 3*. Network Theory Limited, 2008.

[45] K. Eriksson et al. "Introduction to Adaptive Methods for Differential Equations." In: *Acta Numerica* 4 (1995), pp. 105–158.

[46] A. Ern and J.-L. Guermond. *Theory and Practice of Finite Elements*. Springer-Verlag New York, 2004.

[47] Y. Fainman et al., eds. *Optofluidics. Fundamentals, Devices and Applications*. McGraw-Hill, 2010.

[48] P. Frauenfelder. "hp-Finite Element Methods on Anisotropically, Locally Refined Meshes in Three Dimensions with Stochastic Data." PhD thesis. Eidgenösische Technische Hochschule Zürich, 2004.

[49] P. Frauenfelder and C. Lage. "Concepts – An Object-Oriented Software Package for Partial Differential Equations." In: *ESAIM: Mathematical Modelling and Numerical Analysis* 36.5 (2002), pp. 937–951.

[50] M. Fried. "A level set based finite element algorithm for the simulation of dendritic growth." In: *Computing and Visualization in Science* 7.2 (2004), pp. 97–110.

[51] E. Gamma et al. *Design Patterns: Elements of Reusable Object-Oriented Software*. 1st ed. Boston: Addison-Wesley, 10, 1994.

[52] J. Glimm et al. "A Critical Analysis of Rayleigh-Taylor Growth Rates." In: *Journal of Computational Physics* 169.2 (2001), pp. 652–677.

[53] J. Glimm et al. "Three-Dimensional Front Tracking." In: *SIAM Journal on Scientific Computing* 19.3 (1998), pp. 703–727.

[54] S. Gross and A. Reusken. *Numerical Methods for Two-Phase Incompressible Flows*. Springer Series in Computational Mathematics. Springer-Verlag Berlin Heidelberg, 2011.

[55] W. Gui and I. Babuška. "The h, p and h-p Versions of the Finite Element Method in 1 dimension. Part III : The Adaptive h-p Version." In: *Numerische Mathematik* 49 (1986), pp. 659–683.

[56] W. Gui and I. Babuška. "The h, p and h-p Versions of the Finite Element Method in 1 dimension. Part II : The Error Analysis of the h-p Versions." In: *Numerische Mathematik* 49 (1986), pp. 613–657.

[57] W. Gui and I. Babuška. "The h, p and h-p Versions of the Finite Element Method in 1 dimension. Part I : The Error Analysis of the p-Version." In: *Numerische Mathematik* 49 (1986), pp. 577–612.

[58] B. Guo and I. Babuška. "The h-p version of the finite element method. Part 1: the basic approximation results." In: *Computational Mechanics* 1 (1986), pp. 21–41.

[59] B. Guo and I. Babuška. "The h-p version of the finite element method. Part 2: General results and applications." In: *Computational Mechanics* 1 (1986), pp. 203–220.

[60] M. E. Gurtin, D. Polignone, and J. Vinals. "Two-phase binary fluids and immiscible fluids described by an order parameter." In: *Mathematical Models Methods in Applied Sciences* 6.6 (1995), pp. 815–831.

[61] F. H. Harlow and J. E. Welsh. "Numerical Calculation of Time-Dependent Viscous Incompressible Flow of Fluid with Free Surface." In: *The Physics of Fluids* 8 (1965), pp. 2182–2189.

[62] R. Hartmann, M. Lukáčová-Medvidóvá, and F. Prill. "Efficient preconditioning for the discontinuous Galerkin finite element method by low-order elements." In: *Applied Numerical Mathematics* 59.8 (2009), pp. 1737–1753.

[63] M. Hashimoto et al. "Flowing Lattices of Bubbles as Tunable, Self-Assembled Diffraction Gratings." In: *Small* 2.11 (2006), pp. 1292–1298.

[64] A. R. Hawkins and H. Schmidt, eds. *Handbook of Optofluidics*. Boca Raton: CRC Press, 2010.

[65] V. Heuveline, D. Lukarski, and J.-P. Weiß. "Scalable Multi-Coloring Preconditioning for Multicore CPUs and GPUs." In: *Euro-Par 2010 Parallel Processing Workshops*. Ed. by M. Guarracino et al. Vol. 6586. Lecture Notes in Computer Science. Springer-Verlag Berlin Heidelberg, 2011, pp. 389–397.

[66] V. Heuveline and F. Schieweck. "On the Inf-Sup Condition for Higher Order Mixed FEM on Meshes with Hanging Nodes." In: *Mathematical Modelling and Numerical Analysis* 41.1 (2007), pp. 1–20.

[67] V. Heuveline et al. "A Multi-Platform Linear Algebra Toolbox for Finite Element Solvers on Heterogeneous Clusters." In: *PPAAC'10, IEEE Cluster 2010 Workshops*. 2010.

[68] J. G. Heywood, R. Rannacher, and S. Turek. "Artificial Boundaries and Flux and Pressure Conditions for the Incompressible Navier-Stokes Equations." In: *International Journal for Numerical Methods in Fluids* 22.5 (1996), pp. 325–352.

[69] C. W. Hirt and B. D. Nichols. "Volume of Fluid (VOF) Method for the Dynamics of Free Boundaries." In: *Journal of Computational Physics* 39.1 (1981), pp. 201–225.

[70] P. Houston, C. Schwab, and E. Suli. "Discontinuous hp-Finite Element Methods for Advection-Diffusion-Reaction Problems." In: *SIAM Journal on Numerical Analysis* 39.6 (2002), pp. 2133–2163.

[71] P. Houston and E. Süli. "A note on the design of hp-adaptive finite element methods for elliptic partial differential equations." In: *Computer Methods in Applied Mechanics and Engineering* 194.2-5 (2005), pp. 229–243.

[72] T. Hughes, J. Cottrell, and Y. Bazilevs. "Isogeometric analysis: CAD, finite elements, NURBS, exact geometry and mesh refinement." In: *Computer Methods in Applied Mechanics and Engineering* 194.39-41 (2005), pp. 4135–4195.

[73] N. Hu, X.-Z. Guo, and I. N. Katz. "Bounds for Eigenvalues and Condition Numbers in the p-Version of the Finite Element Method." In: *Mathematics of Computation* 67.224 (1998), pp. 1423–1450.

[74] N. Hu, X.-Z. Guo, and I. N. Katz. "Multi-p Preconditioners." In: *SIAM Journal on Scientific Computing* 18.6 (1997), pp. 1676–1697.

[75] A. Iserles. *A First Course in the Numerical Analysis of Differential Equations*. Cambridge: Cambridge University Press, 1996.

[76] J. Johnsen. "Simple Proofs of Nowhere-Differentiability for Weierstrass's Function and Cases of Slow Growth." In: *Journal of Fourier Analysis and Applications* 16.1 (2010), pp. 17–33.

[77] E. Jones, T. Oliphant, P. Peterson, et al. *SciPy: Open source scientific tools for Python*.

[78] G. E. Karniadakis, A. Beskok, and N. Aluru. *Microflows and Nanoflows : Fundamentals and Simulation*. Interdisciplinary Applied Mathematics. New York: Springer Science+Business Media, 2005.

[79] G. Karniadakis and S. Sherwin. *Spectral/hp Element Methods For Computational Fluid Dynamics*. Numerical Mathematics and Scientific Computation. Oxford: Oxford University Press, 2005.

[80] B. Kirk et al. "libMesh: a C++ library for parallel adaptive mesh refinement/coarsening simulations." In: *Engineering with Computers* 22.3 (2006), pp. 237–254.

[81] J. Kong. *Electromagnetic Wave Theory*. New York: John Wiley & Sons, Inc., 1990.

[82] W. R. Longley and R. G. Van Name, eds. *The Collected Works of J Williard Gibbs*. Longmans, Green and Co., 1928.

[83] C. Mavriplis. "Adaptive mesh strategies for the spectral element method." In: *Computer Methods in Applied Mechanics and Engineering* 116 (1994), pp. 77–86.

[84] J. C. Maxwell. "A Dynamical Theory of the Electromagnetic Field." In: *Philosophical Transactions of the Royal Society of London* 155 (1865), pp. 459–512.

[85] S. McKee et al. "The MAC method." In: *Computers & Fluids* 37.8 (2008), pp. 907–930.

[86] J. Melenk. "On condition numbers in hp-FEM with Gauss-Lobatto-based shape functions." In: *Journal of Computational and Applied Mathematics* 139.1 (2002), pp. 21–48.

[87] J. Melenk, K. Gerdes, and C. Schwab. "Fully discrete hp-finite elements: fast quadrature." In: *Computer Methods in Applied Mechanics and Engineering* 190 (2001), pp. 4339–4364.

[88] J. Melenk and B. Wohlmuth. "On residual-based a-posteriori error estimation in hp-FEM." In: *Advances in Computational Mathematics* 15 (2001), pp. 311–331.

[89] W. F. Mitchell and M. A. McClain. *A Comparison of hp-adaptive Strategies for Elliptic Partial Differential Equations (long version)*. NIST Interagency/Internal Report (NISTIR) 7824. National Institute of Standards and Technology, 2011.

[90] W. F. Mitchell and M. A. McClain. "A Survey of hp-Adaptive Strategies for Elliptic Partial Differential Equations." In: *Recent Advances in Computational and Applied Mathematics*. Ed. by T. E. Simos. Springer Netherlands, 2011, pp. 227–258.

[91] Bo-nan Jiang. *The Least-Squares Finite Element Method*. Springer-Verlag Berlin Heidelberg, 1998.

[92] W. Noh and P. Woodward. "SLIC (Simple Line Interface Calculation)." In: *Proceedings of the Fifth International Conference on Numerical Methods in Fluid Dynamics*. Ed. by A. van de Vooren and P. Zandbergen. Vol. 59. Lecture Notes in Physics. Springer-Verlag Berlin Heidelberg, 1976, pp. 330–340.

[93] J. T. Oden. "Some historic comments on finite elements." In: *Proceedings of the ACM conference on History of scientific and numeric computation*. HSNC '87. New York: ACM, 1987, pp. 125–130.

[94] E. T. Olsen and J. J. Douglas. "Bounds on spectral condition numbers of matrices arising in the p-version of the finite element method." In: *Numerische Mathematik* 69.3 (1995), pp. 333–352.

[95] E. Olsson and G. Kreiss. "A conservative level set method for two phase flow." In: *Journal of Computational Physics* 210.1 (2005), pp. 225–246.

[96] E. Olsson, G. Kreiss, and S. Zahedi. "A conservative level set method for two phase flow II." In: *Journal of Computational Physics* 225.1 (2007), pp. 785–807.

[97] S. A. Orszag. "Numerical Methods for the Simulation of Turbulence." In: *Physics of Fluids* 12.12 (1969), pp. 250–257.

[98] S. Osher and R. Fedkiw. *Level Set Methods and Dynamic Implicit Surfaces*. New York: Springer-Verlag New York, 2003.

[99] S. Osher and R. P. Fedkiw. "Level Set Methods: An Overview and Some Recent Results." In: *Journal of Computational Physics* 169.2 (2001), pp. 463–502.

[100] S. Osher and N. Paragios, eds. *Geometric Level Set Methods in Imaging, Vision and Graphics*. Springer-Verlag, 2003.

[101] S. Osher and J. A. Sethian. "Fronts propagating with curvature-dependent speed: Algorithms based on Hamilton-Jacobi formulations." In: *Journal of Computational Physics* 79.1 (1988), pp. 12–49.

[102] A. T. Patera. "A spectral element method for fluid dynamics: Laminar flow in a channel expansion." In: *Journal of Computational Physics* 54.3 (1984), pp. 468–488.

[103] L. Pesch et al. "hpGEM — A Software Framework for Discontinuous Galerkin Finite Element Methods." In: *ACM Transactions on Mathematical Software* 33.4 (2007), pp. 1–25.

[104] L. Rayleigh. "On the theory of surface forces - II. compressible forces." In: *Philosophical Magazine* 30 (1892), pp. 209–220.

[105] M. Richter. "Optimization of Photonic Band Structures." PhD thesis. Karlsruhe Institute of Technology, 2010.

[106] W. J. Rider and D. B. Kothe. "Reconstructing Volume Tracking." In: *Journal of Computational Physics* 141.2 (1998), pp. 112–152.

[107] R. Rieben, D. White, and G. Rodrigue. "Improved conditioning of finite element matrices using new high-order interpolatory bases." In: *IEEE Transactions on Antennas and Propagation* 52.10 (2004), pp. 2675–2683.

[108] S. Ronnas et al. "Design and Implementation of Distributed Meshes in HiFlow3." In: *Competence in High Performance Computing 2010*. Ed. by C. Bischof et al. Springer-Verlag Berlin Heidelberg, 2012, pp. 61–71.

[109] A. Schmidt and K. G. Siebert. "A posteriori estimators for the h-p version of the finite element method in 1D." In: *Applied Numerical Mathematics* 35.1 (2000), pp. 43–66.

[110] J. Schöberl and S. Zaglmayr. "High Order Nédélec Elements with local complete sequence properties." In: *COMPEL: The International Journal for Computation and Mathematics in Electrical and Electronic Engineering*. Vol. 24. 2. 2005, pp. 374–384.

[111] D. Schötzau and C. Schwab. "Mixed hp-FEM on Anisotropic Meshes." In: *Mathematical Modelling and Methods in Applied Sciences* 8.5 (1998), pp. 787–820.

[112] D. Schötzau, C. Schwab, and R. Stenberg. "Mixed hp-FEM on Anisotropic Meshes II: Hanging Nodes and Tensor Products of Boundary Layer Meshes." In: *Numerische Mathematik* 83 (1999), pp. 667–697.

[113] D. Schötzau, C. Schwab, and T. Wihler. *hp-dGFEM for second-order elliptic problems in polyhedra. II: Exponential convergence*. Tech. rep. 29. Seminar for Applied Mathematics, ETH Zürich, 2009.

[114] D. Schötzau, C. Schwab, and T. Wihler. *hp-dGFEM for second-order elliptic problems in polyhedra. I: Stability and quasioptimality on geometric meshes*. Tech. rep. 28. Seminar for Applied Mathematics, ETH Zürich, 2009.

[115] C. Schwab. *p- and hp-Finite Element Methods*. Oxford: Oxford Science Publications, 1998.

[116] L. Segel and G. Handelman. *Mathematics Applied to Continuum Mechanics*. Classics in Applied Mathematics. New York: Society for Industrial and Applied Mathematics, 2007.

[117] J. A. Sethian. "A Fast Marching Level Set Method for Monotonically Advancing Fronts." In: *Proc. Nat. Acad. Sci*. 1995, pp. 1591–1595.

[118] J. A. Sethian. *Level Set Methods and Fast Marching Methods: Evolving interfaces in computational geometry, fluid mechanics, computer vision, and materials science*. 2nd edition. Cambridge: Cambridge University Press, 1999.

[119] S. Shin, I. Yoon, and D. Juric. "The Local Front Reconstruction Method for direct simulation of two- and three-dimensional multiphase flows." In: *Journal of Computational Physics* 230.17 (2011), pp. 6605–6646.

[120] J. C. Slattery, L. Sagis, and E.-S. Oh. *Interfacial Transport Phenomena*. 2nd edition. New York: Springer Science+Business Media, 2007.

[121] P. Šolín, J. Červený, and I. Doležel. "Arbitrary-level hanging nodes and automatic adaptivity in the hp-FEM." In: *Mathematics and Computers in Simulation* 77.1 (2008), pp. 117–132.

[122] P. Šolín, L. Dubcova, and I. Doležel. "Adaptive hp-FEM with Arbitrary-Level Hanging Nodes for Maxwell's Equations." In: *Advances in Applied Mathematics and Mechanics* 2.4 (2010), pp. 518–532.

[123] P. Šolín, K. Segeth, and I. Doležel. *Higher-Order Finite Element Methods*. Boca Raton: Chapman & Hall/CRC, 2003.

[124] P. Šolín et al. "PDE-independent adaptive hp-FEM based on hierarchic extension of finite element spaces." In: *Journal of Computational and Applied Mathematics* 233.12 (2010), pp. 3086–3094.

[125] J. H. Spurk and N. Aksel. *Fluid Mechanics*. 2nd edition. Springer-Verlag Berlin Heidelberg, 2008.

[126] R. Stevenson. "Optimality of a Standard Adaptive Finite Element Method." In: *Foundations of Computational Mathematics* 7.2 (2007), pp. 245–269.

[127] M. Sussman, P. Smereka, and S. Osher. "A Level Set Approach for Computing Solutions to Incompressible Two-Phase Flow." In: *Journal of Computational Physics* 114.1 (1994), pp. 146–159.

[128] B. Szabó. "Some recent developments in finite element analysis." In: *Computers & Mathematics with Applications* 5.2 (1979), pp. 99–115.

[129] S. K. Y. Tang, C. A. Stan, and G. M. Whitesides. "Dynamically reconfigurable liquid-core liquid-cladding lens in a microfluidic channel." In: *Lab on a Chip* 8.3 (2008), pp. 395–401.

[130] R. Temam and A. Miranville. *Mathematical Modeling in Continuum Mechanics*. Cambridge: Cambridge University Press, 2001.

[131] A.-K. Tornberg and B. Engquist. "A finite element based level-set method for multiphase flow applications." In: *Computing and Visualization in Science* 3.1 (2000), pp. 93–101.

[132] G. Tryggvason et al. "A Front-Tracking Method for the Computations of Multiphase Flow." In: *Journal of Computational Physics* 169.2 (2001), pp. 708–759.

[133] S. O. Unverdi and G. Tryggvason. "A front-tracking method for viscous, incompressible, multi-fluid flows." In: *Journal of Computational Physics* 100.1 (1992), pp. 25–37.

[134] T. Vejchodský and P. Šolín. "Static condensation, partial orthogonalization of basis functions, and ILU preconditioning in the hp-FEM." In: *Journal of Computational and Applied Mathematics* 218.1 (2008), pp. 192–200.

[135] R. Verfürth. "A Posteriori Error Estimators for the Stokes Equations." In: *Numerische Mathematik* 55 (1989), pp. 309–325.

[136] R. Verfürth. *A Review of a Posteriori Error Estimation and Adaptive Mesh-Refinement Techniques*. Chichester: Wiley-Teubner, 1996.

[137] D. Vezenov et al. "A low-threshold, high-efficiency microfluidic waveguide laser." In: *Journal of the American Chemical Society* 127.25 (2005), pp. 8952–8953.

[138] P. E. Vos, S. J. Sherwin, and R. M. Kirby. "From h to p efficiently: Implementing finite and spectral/hp element methods to achieve optimal performance for low- and high-order discretisations." In: *Journal of Computational Physics* 229.13 (2010), pp. 5161–5181.

[139] J. V. D. Waals. "The thermodynamic theory of capillarity under the hypothesis of a continuous density variation." In: *Journal of Statistical Physics* 20 (1893), pp. 197–244.

[140] D. Wolfe et al. "Dynamic control of liquid-core/liquid-cladding optical waveguides." In: *Proceedings of the National Academy of Sciences of the United States of America* 101.34 (2004), pp. 12434–12438.

[141] X. Yang et al. "An adaptive coupled level-set/volume-of-fluid interface capturing method for unstructured triangular grids." In: *Journal of Computational Physics* 217.2 (2006), pp. 364–394.