High Performance Computing Based Methods for Simulation and Optimisation of Flow Problems

Zur Erlangung des akademischen Grades eines

DOKTORS DER NATURWISSENSCHAFTEN

von der Fakultät für Mathematik der Universität Karlsruhe (TH) genehmigte

DISSERTATION

von
Dipl.-Math. techn. Hendryk Bockelmann
aus Lüneburg

Tag der mündlichen Prüfung: 14. Juli 2010

Referent:	Prof. Dr. Vincent Heuveline
Korreferent:	Prof. Dr. Peter Wittwer
Korreferent:	Jun.-Prof. Dr. Jan-Philipp Weiß

Outline

This thesis is concerned with the study of methods in high-performance computing (HPC) for simulation and optimisation of flow problems that typically occur in the framework of microflows. We consider the adequate use of techniques in parallel computing by means of finite element based solvers for partial differential equations (PDEs) and by means of sensitivity- and adjoint-based optimisation methods. The main focus is on low Reynolds number flow described by the Navier-Stokes equations (NSE) with an additional convection-diffusion equation describing the distribution of species concentration within the fluid. Conclusively, we aim to optimise such a microfluidic system by applying an electric potential difference to influence the mixing of species using electrokinetic properties of the fluid (see Barz et al. [10, 11]).

The topic of optimisation and optimal control under constraints of partial differential equations is a very active area of research. Many authors have contributed their results, starting from the very early work of Lions [105]. For an overview of methods and perspectives of optimal control problems especially in the framework of fluid flows, we refer to the work of Gunzburger [73]. This thesis follows the classical attempts by means of sensitivity- and adjoint-based optimisation methods while the focus is set on the numerical requirements and implementation of these well-known approaches on high-performance computers. We show that the requirements with regard to computational power are quite strong when 3D instationary problems are to be solved.

In Chapter 1 we introduce the flow problems and the resulting complexity for an instationary 3D setting. The second chapter is devoted to the physical derivation and mathematical framework of weak formulation for the PDEs describing the considered setting of microflows. Furthermore, the background of optimisation with PDEs is presented. Chapters 3 and 4 show the actual capability of numerical solvers based on finite element discretisation combined with multilevel preconditioners and HPC-technologies in the framework of Domain Decomposition. Dedicated methods for optimisation of instationary problems using parallel computing techniques are presented afterwards in Chapter 5. The thesis is completed by numerical results on simulation and optimisation of microflows in Chapter 6. These results show the different requirements of presented optimisation approaches when solving PDE constrained optimisation problems and prove the effective usage of HPC-technologies in order to solve even complex 3D instationary problems.

Zusammenfassung

In dieser Arbeit werden Methoden des Hochleistungsrechnens für die Simulation und Optimierung laminarer Strömungen untersucht, wie sie typischerweise im Bereich der Mikrofluidik auftreten. Wir betrachten dabei den adäquaten Gebrauch von Methoden des parallelen Rechnens im Rahmen von Finiten Elemente basierten Lösern für partielle Differentialgleichungen sowie für sensitivitäts- und adjungierten-basierte Optimierungsverfahren. Der Schwerpunkt liegt hierbei auf Strömungsproblemen bei niedriger Reynolds Zahl, welche durch die Navier-Stokes Gleichungen beschrieben werden. Zusätzlich wird eine Konvektions-Diffusion Gleichung betrachtet um die Verteilung einer Spezienkonzentration innerhalb des Fluids erfassen zu können. Zur Optimierung solcher Systeme der Mikrofluidik wird schließlich eine Potentialdifferenz zwischen Ein- und Ausgang der Geometrie genutzt, so dass durch elektrokinetische Vorgänge die Vermischung zweier Spezien begünstigt wird (siehe Barz et al. [10, 11]).

Die Thematik der Optimierung und optimalen Kontrolle unter der Nebenbedingung partieller Differentialgleichungen ist ein sehr aktives Forschungsgebiet. Beiträge hierzu reichen von einer ersten ausführlichen Analyse von Lions [105] bis zu zahlreichen aktuellen Arbeiten. Für einen Überblick der aktuellen Methoden und Ergebnisse sowie einen Ausblick im Bereich der optimalen Kontrolle für Strömungsprobleme sei auf das Buch von Gunzburger [73] verwiesen. In dieser Arbeit verfolgen wir den klassischen Ansatz der sensitivitäts- und adjungierten-basierten Optimierung, wobei das Hauptaugenmerk auf die numerischen Voraussetzungen und die Implementierung auf Parallelrechnern gelegt ist. Insbesondere werden instationäre dreidimensionale Probleme gelöst für welche der Rechenbedarf entsprechend hoch ist.

In Kapitel 1 werden die strömungsmechanischen Beispiele der Arbeit präsentiert und die resultierende Komplexität für die numerische Behandlung abgeschätzt. Das zweite Kapitel beinhaltet die Herleitung der physikalischen und mathematischen Beschreibung - besonders die schwache Formulierung der Zustandsgleichungen sowie die Formulierung passender Randbedingungen. Weiterhin wird die Optimierung bei partiellen Differentialgleichungen eingeführt. Kapitel 3 und 4 sind der numerischen Lösung durch die Finite Elemente Methode gewidmet. Hierbei werden multilevel Vorkonditionierer präsentiert und deren Einsatzmöglichkeit unter Verwendung vom Gebietszerlegungsmethoden auf Parallelrechner erweitert. Dedizierte Methoden paralleler Hardware für die Optimierung instationärer Probleme werden in Kapitel 5 untersucht. Die Arbeit umfasst abschließend in Kapitel 6 numerische Resultate für die Simulation und Optimierung innerhalb der Mikrofluidik, welche die unterschiedlichen Anforderungen und Möglichkeiten der Numerik auf Parallelrechnern zusammenfassen.

Acknowledgements

A special thank is devoted to Prof. Dr. Vincent Heuveline who awoke my interest in PDE constrained optimisation, finite element methods and high-performance computing in the year 2005 and supported my work starting from diploma thesis up to now. I enjoyed the chance to work on the FEM code `HiFlow`^{2} [20] and to have the possibility to use HPC-systems of the Steinbuch Centre for Computing at Karlsruhe Institute of Technology as well as the supercomputer JUROPA hosted at the Forschungszentrum Jülich that was rated on rank 13 in the TOP500 list of November 2009. For the kind readiness of reading this thesis and all resulting hints I want to thank Prof. Dr. Peter Wittwer, Jun.-Prof. Dr. Jan-Philipp Weiß and PD Dr. Gudrun Thäter. Many thanks also go to my colleagues in Karlsruhe at the research group *Numerische Simulation, Optimierung und Hochleistungsrechnen* who were never averse to discuss about problems and latest computational results or bugs.

Some persons have to be mentioned in detail: first I was very pleased to find a perfect contact in Dr. Dominik Barz for the very interesting field of microflows, who also gave the possibility to verify the results with concrete physical experiments and measurements. Furthermore, I have to thank Dr. Jan Mayer for the fruitful teamwork on ILU-based preconditioners and Dipl.-Math. techn. Benedikt Galler for the flawless contributions to the programming of parallel Schur complement solvers and preconditioners. Frauke, thank you for all your patient support.

Contents

Chapter 1

Introduction

In many applications of research and development projects concerned with computational fluid dynamics (CFD), the ultimate goal nowadays is related to the optimisation and optimal control of the considered system or process. In this context, the solution process is usually much more involved than a purely forward simulation leading to a descriptive or predictive view of the treated system. Optimisation and optimal control require an iterative process involving the setting and manipulation of control variables such that the system or process gets improved. Such an improvement has finally to be measured and verified to judge the success of optimisation.

In this thesis, the focus is on applications as they typically appear in microfluidic systems, i.e. in devices with a typical scale of 100 μm. The placement of different microcomponents with several laboratory functions like pumps, valves, or detectors in combination with microchannels forms the concept of the *lab on a chip* (LOC) that deals with handling of extremely small fluid volumes and fast analysis times due to short diffusion distances. Microfluidic processes that may take place within an LOC are transport, separation or mixing of (bio)chemical species within a fluid [93]. A crucial fact for these tasks is the usually purely laminar nature of the flow fields in such devices. The absence of turbulences is beneficial with regard to numerical simulation but disadvantageous for engineering problems if a mixing of species is aimed for (e.g. to enable a chemical reaction). As a typical scenario, we investigate within a twice-folded microchannel the homogeneous and fast mixing of pure water with water in which Rhodamine B as fluorescent dye is dissolved - according to the setting of Barz et al. [10, 11]. From the engineering point of view the measurement of dye concentration by means of a micro laser-induced fluorescence method (μLIF) and of velocity field by microparticle image velocimetry (μPIV) requires a suitable experimental setup which may limit the intended experiments. Numerical simulation, on the other hand, allows to explore the system in advance and gives helpful hints for e.g. the design of microdevices. Obviously, the abstraction of physical process to numerical simulation necessitates the deployment of a mathematical model, a discretisation approach to the derived equations, as well as the entire solution process on the computer. All these steps are error-prone such that the results at the very end have to be compared to experiments in order to validate the quality of the simulation. However, an essential benefit of numerical simulation is the possibility to manipulate parameter and/or geometry in nearly arbitrary way (once a model and the according software is at hand) which then can be used to optimise the system or process under investigation.

For the considered setting of mass transport and mixing in microchannels, we have to describe the physical process in more details. The mixing of two (or more) liquids can generally be separated into two main steps (cf. [12]):

1. increasing of the contact area between liquids,

2. essential mixing by molecular diffusion across the contact area.

In flows through geometries with macroscopic dimensions, turbulence is often employed to enhance the mixing process. In microscopic geometries, the generation of a turbulent flow regime is rather difficult and, therefore, other concepts are used for mixing. Two approaches to increase the mixing are commonly used, namely active methods by employing external forces to the system (e.g. electrical forces) and passive methods that are based on a variation of the geometry. The approach

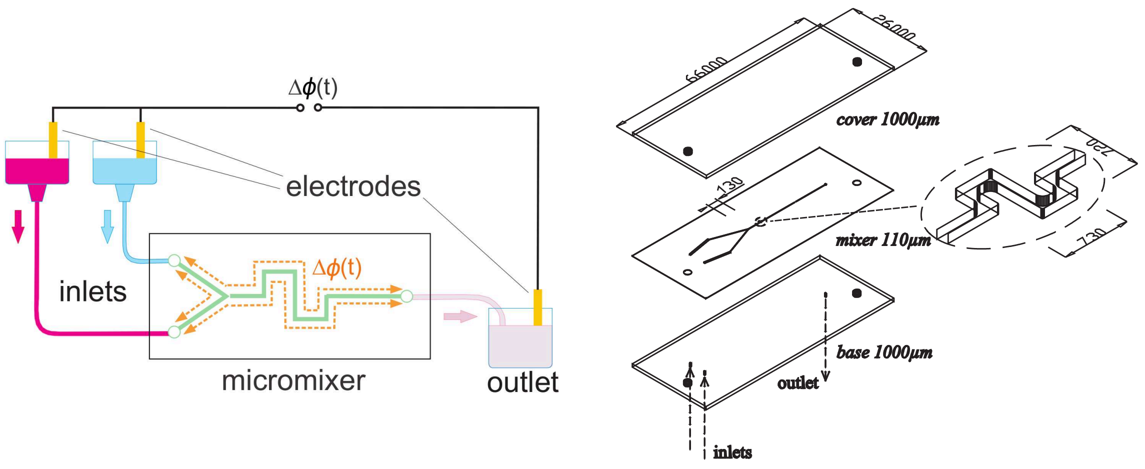


Figure 1.1: Glass layer of the microfluidic chip and the experimental setup. Pictures are kindly provided by Dominik P. J. Barz [10, 11].

for an electrically excited micromixer given by Meisel and Ehrhard [116] is based on a twice-folded channel segment in combination with an induced electroosmotic secondary flow. For the detailed experimental setting, see the description in [10, 11] and the Appendix - the main components (structure of microfluidic chip and basic setup) are depicted in Figure 1.1. The fundamental physiochemical process used within this setting is the electroosmotic excitation of the fluid related to the presence of a so-called *electrical double layer* (EDL). This layer is present adjacent to the channel walls that usually possess an electrically charged surface and therefore a certain electrical potential which induces a higher ion concentration of the - apart from that - electrically neutral fluid [12, 94]. If now electrodes are positioned within the reservoirs of the two liquids to be mixed and a specific voltage is applied to a third electrode within the outlet reservoir, the time-dependent potential difference $\Delta\phi(t)$ induces an instationary fluid flow within the EDL. This secondary flow can be used to modify the pressure-driven base flow and thus increases the contact area between the liquids.

Since the width of the EDL is usually in the order of 10^{-9}–10^{-8} m, denoted by the so-called *Debye length* ℓ_D, numerical simulation of these systems in a full three-dimensional instationary setting requires a very fine discretisation to resolve the various physical effects. The ratio of EDL size to size of the microchannel is usually $1 : 10000$ indicating that the resulting algebraic systems to be solved will be at least of order 10^5 in each space dimension. If a uniform mesh is assumed the 3D discretisation will then easily lead to 10^{15} unknowns. To reduce the numerical complexity of these systems a possible way might be the usage of locally refined discretisation and non-uniform meshes. A different approach is proposed by Barz et al. who neglect the electrical double layer near channel walls and simulate only the bulk flow within the channel using slip boundary conditions to model the influence of the EDL. Nevertheless, the resulting algebraic systems to be solved within a timestepping algorithm are still on the order of $10^5 - 10^6$ unknowns such that it is sensible to use iterative linear solvers like Krylov subspace methods on parallel computers to tackle the problem in reasonable time. A critical point for the convergence behaviour of iterative Krylov subspace methods is the presence of suitable preconditioners. To this end, we introduce Multilevel ILU preconditioners based on the `ILU++` package by Jan Mayer [112] and extend these to be used in parallel by means of Block Jacobian and Schur complement data structures.

Besides the above introduced simulation and optimisation of an electrically excited fluid flow in a microchannel, the topic of optimisation and optimal control for flow problems is also regarded for the more academical example of the well-known backward facing step flow. For these studies on parallel solver/preconditioner and optimisation routines, we treat the Navier-Stokes equations with the control objective of reduction of the recirculation/vortex area behind the step of a microchannel - see Figure 1.2 for the uncontrolled case. A common setting is given by boundary control near the edge of the step where blowing and suction of fluid is assumed. By this additional stream the re-attachment length of main fluid flow is aimed to be minimised. The choice of cost functional

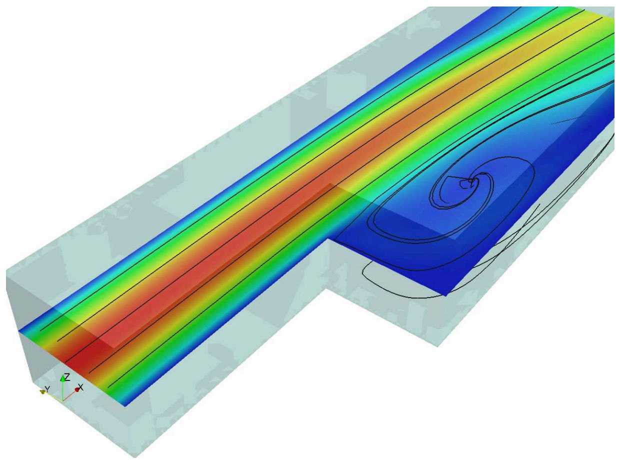


Figure 1.2: Fluid flow behind a backward facing step with streamlines indicating the recirculation.

to realise this aim is variously addressed in the literature and in this thesis only a tracking-type formulation is chosen, i.e. the difference to Stokes flow, possessing a minimal recirculation tendency, is observed.

In general the optimisation problems in this thesis base on an abstract problem: let y denote a set of state variables and u be a control variable of any kind. The system of partial differential equations describing the fluid flow is given by the equation $E(y, u) = 0$. Furthermore, a cost or objective functional $J(y, u)$ is assumed that has to be minimised under the constraint $E(y, u) = 0$. Hence, beside pure simulation tasks, we also treat problems of the form

$$\min J(y, u) \text{ such that } E(y, u) = 0.$$

Two approaches to tackle this problem are studied, namely

1. adjoint-based boundary control in conjunction with Navier-Stokes equations optimising the flow field of backward facing step problem,

2. sensitivity-based optimisation of the micromixer in Figure 1.1 by control of the potential difference $\Delta\phi(t)$.

These optimisation methods can principally be distinguished by the determination of the gradient of the objective functional. Assuming a time dependent setting, methods that rely on the solution of the adjoint equation require forward-in-time solution of the state equations and backward-in-time solution of the adjoint equations, whereas sensitivity-based methods only march forward in time. For nonlinear problems, a challenging task is related to the fact that the state variables have to be available to the adjoint equation solver. Since the adjoint equation solver marches backward in time, this means that one must store the state variables for every timestep. For large scale problems, this step, which amounts to store very large data set, may be intractable in practice or become a real bottleneck for the overall solution process. An approach combining checkpointing techniques to reduce the needed storage resources with parallel I/O is presented in this context. A main emphasis is put on the adequate use of recent technologies related to large high-performance computing (HPC) platforms like for the parallel preconditioner used for simulation tasks.

Summarising, to tackle the simulation and optimisation of an electrically excited micromixer, we first abstracted from this to a pure fluid flow setting described by the instationary Navier-Stokes equations. For this the sequential multilevel preconditioner `ILU++` was extended by parallel data structures which were implemented and tested on actual high performance computers to form an appropriate parallel preconditioner for iterative Krylov subspace methods. Afterwards also the two mentioned optimisation approaches were incorporated into the developed software. Herein, we explored a suitable combination of checkpointing schemes and parallel I/O techniques that were especially fruitful for explicit timestepping schemes with lumped mass-matrix. Extensive numerical tests were performed to validate the proposed approaches and to give an estimate on the benefits of adjoint-based optimisation in contrast to the sensitivity-based approach. While the latter one obviously possesses a much easier implementation, the first one enables a wider treatment of the system to be optimised – we comment on the ratio of effort to profit in Chapter 6. As a result of these steps, we are able to improve the three-dimensional simulation results for the electrically excited micromixer and to give useful hints in optimising the applied potential difference to increase the mixing quality.

Chapter 2

Simulation and Optimisation of Fluid Flow

As presented in the introduction, we are faced with simulation and optimisation/optimal control of problems arising in the field of fluid dynamics. In this chapter, a derivation of the governing equations of fluid dynamics and modelling results for the electrokinetic effects will be given. Afterwards, the mathematical setting is prepared that is needed for the finite element based discretisation, namely the weak formulation of underlying partial differential equations. Besides the pure simulation based framework, we also show the common approaches in optimisation with partial differential equations and apply these to get the configuration for the considered flow problems.

Consequently, this chapter is neither devoted to a rigours derivation of governing equations nor to a complete analysis of them. We only aim at providing the reader with the basics in both physics and functional analysis as they are needed for any further discussion. For a more detailed overview on the topics of this chapter, we give particular bibliographical references.

2.1 Modelling Fluid Flow

The mathematical description of fluid flow, the fluid dynamics, is a sub-discipline of continuum mechanics. In contrast to particle mechanics dealing with the equilibrium and motion of systems of point masses, continuum mechanics is a means of studying the deformation and flow of a continuous medium by ignoring its microscopic/molecular nature, i.e. it deals with mass-continua that are located in the Euclidean space $\mathbb{R}^3$. It is a simplification that makes it possible to investigate the movement of matter on scales larger compared to typical distances between molecules. Thus, in studying the movement of a fluid the fact that it is made up of molecules is ignored – but fortunately, in general one is not interested in the individual behaviour of a single molecule, but in the average motion of a large number of molecules.

A key difference between classical dynamics and fluid dynamics is the continuum hypothesis or continuum approximation. Within this the fluid velocity, density and stress tensor, that are introduced later on, are to be interpreted as appropriate averages of mechanical properties of the molecules. Furthermore, these quantities are assumed to be interpreted at each position in a volume and to be continuously spread.

Assumption 2.1.1 (Continuum hypothesis)

We assume that in the region governed by a fluid a continuous and differentiable mass-density is given by function $\rho(\mathbf{x}, t) > 0$ such that the mass $M(t)$ of an arbitrary subvolume $V(t)$ can be calculated by the integral

$$M(t) = \int\limits_{V(t)} \rho(\mathbf{x}, t) \; dV.$$

This means that any small volume element in the fluid is always supposed to be as large as needed to define a molecular average by containing a reasonable big number of molecules.

2.1.1 Basic Equations of Fluid Dynamics

The following derivation of equations is based on some fundamental assumptions on the fluid:

- relativity and quantum mechanics are ignored,
- the length scale of the flow is always taken to be large compared to the molecular mean-free path, so that the fluid can be treated as a continuum (continuum hypothesis),
- in the sense of the continuum hypothesis we label the smallest entity in the fluid as fluid-particle and assume that the position of a particle can be described by its coordinates in the Euclidean space $\mathbb{R}^3$,
- the fluid is assumed to be of uniform, homogeneous composition, i.e. diffusion and chemical reactions are not considered.

Based on the continuum hypothesis the equations of motion for fluid flow can be derived in a general format by applying the principles of mechanics, i.e. conservation of mass (continuity), balance of linear momentum (Newton's second law) and angular momentum.

Let $\Omega \subset \mathbb{R}^3$ be a fluid volume that moves in space under action of internal and external forces. Consider a testvolume $V(t) \subset \Omega$ that is open and bounded – this volume is filled with particles of the fluid. Furthermore, we define the movement of a particle $\eta \in V(t)$ by a function $\Phi(\eta, t)$ such that at time $t > 0$ the particle η is located at the point $\mathbf{x} = \Phi(\eta, t)$. We assume that Φ is as smooth as needed, i.e. Φ is invertible and $\Phi, \Phi^{-1} \in C^1(\Omega)$.

Generally two different descriptions of the motion of fluids can be considered

1. **Lagrangian description**: for fixed particle η (the Lagrangian coordinate) one follows the trajectory $t \to \Phi(\eta, t)$. Thus, the Lagrangian coordinate system moves with the fluid – this description of motion is very useful in solid mechanics.
2. **Eulerian description**: for fixed point $\mathbf{x} \in V(t)$ (the Eulerian coordinate) one observes the trajectory $t \to \Phi(\cdot, t)^{-1}(\mathbf{x})$. The Eulerian coordinate system is fixed and one studies the flow (velocity field) at a fixed point $\mathbf{x}$ as a function of time.

In the sense of the Eulerian description we define the *velocity* of a fluid particle at point $\mathbf{x} = \Phi(\eta, t) \in \mathbb{R}^3$ by

$$\mathbf{v}(\mathbf{x}, t) := \frac{\partial}{\partial t}\Phi(\eta, t).$$

This vector field can be expressed in the cartesian form

$$\mathbf{v}(\mathbf{x}, t) = \mathbf{e}_1 v_1(x_1, x_2, x_3, t) + \mathbf{e}_2 v_2(x_1, x_2, x_3, t) + \mathbf{e}_3 v_3(x_1, x_2, x_3, t)$$

with orthonormal basis vectors $\mathbf{e}_i$ and usually one seeks for knowledge of the scalar variables v_i in the observed fluid domain Ω.

Let $f(x_1, x_2, x_3, t)$ represent any property of the fluid and we ask for the temporal change for this quantity. An observation located at a fixed point in a chosen coordinate system naturally gives the local temporal derivative $\frac{\partial f}{\partial t}(x_1, x_2, x_3, t)$. On the other hand the temporal change of a particle of fixed identity (Lagrangian view) in the flow field $\mathbf{v}(x, t)$ is given by

$$\frac{df}{dt} = \lim_{\Delta t \to 0} \frac{1}{\Delta t}\Big[f(x_1 + v_1 \cdot \Delta t, x_2 + v_2 \cdot \Delta t, x_3 + v_3 \cdot \Delta t, t + \Delta t) - f(x_1, x_2, x_3, t)\Big].$$

This gives the proper expression for the total time derivative of f of a particular particle

$$\begin{aligned} \frac{df}{dt} &= \frac{\partial f}{\partial t} + v_1 \frac{\partial f}{\partial x_1} + v_2 \frac{\partial f}{\partial x_2} + v_3 \frac{\partial f}{\partial x_3} \\ &= \frac{\partial f}{\partial t} + (\mathbf{v} \cdot \nabla) f. \end{aligned} \tag{2.1}$$

The quantity $\frac{df}{dt}$ is variously termed the *substantial derivative*, *particle derivative*, *material derivative* or *Lagrangian derivative* giving the relation between local temporal derivative and so called convective derivative $(\mathbf{v} \cdot \nabla)$.

As shown in the introductory part of this chapter the continuum hypothesis allows the formulation of results not for single molecules (or particles) but for an arbitrary chosen domain of the fluid. To express the temporal change of a given quantity in such a testvolume $V(t)$, which of course itself is influenced by the motion of the fluid, we state the following fundamental theorem (cf. [50, ch. 1.4] or [154, ch. 1.5] for proof).

Theorem 2.1.1 (Reynold's transport theorem)
Let $f : \Omega \times I \to \mathbb{R}$ be a differentiable function for $\mathbf{x} \in \Omega$ and $t \in I \subset \mathbb{R}$ (time interval). For every open and bounded volume $V(t) \subset \Omega$ there holds

$$\begin{aligned} \frac{d}{dt} \int\limits_{V(t)} f(x,t)\ d\mathbf{x} &= \int\limits_{V(t)} (\frac{\partial f}{\partial t}(\mathbf{x},t) + \nabla \cdot (f(\mathbf{x},t)\mathbf{v}(\mathbf{x},t)))\ d\mathbf{x} \\ &= \int\limits_{V(t)} (\frac{\partial f}{\partial t}(\mathbf{x},t)\ d\mathbf{x} + \int\limits_{\partial V(t)} f(\mathbf{x},t)\mathbf{v}(\mathbf{x},t)\cdot\mathbf{n}\ ds \end{aligned} \tag{2.2}$$

with $\mathbf{n}$ the unit outward normal vector on $\partial V(t)$.

Conservation of Mass

Assuming a continuous and differentiable mass-density given by function $\rho(\mathbf{x},t) \in \mathbb{R}$, the mass $m(V(t))$ of an arbitrary testvolume $V(t)$ with surface $\partial V(t)$ is given by

$$m(V(t)) = \int\limits_{V(t)} \rho(x,t)\ d\mathbf{x}$$

that is constant in time (conservation of mass), i.e. $\frac{d}{dt}m(V(t)) = 0$. By equation (2.2) we deduce that

$$\frac{d}{dt} \int\limits_{V(t)} \rho(\mathbf{x},t)\ d\mathbf{x} = \int\limits_{V(t)} \left(\frac{\partial}{\partial t}\rho(\mathbf{x},t) + \nabla \cdot (\rho(\mathbf{x},t)\mathbf{v}(\mathbf{x},t))\right)\ d\mathbf{x} = 0.$$

Since the volume $V(t)$ was chosen arbitrarily the integrand must vanish. This usage of the Dubois-Reymond lemma for sufficiently smooth functions is used throughout in the sequel to transform an integral equation to a partial differential equation (see e.g. [104] for proof). Therefore, we end up with the *continuity equation*

$$\frac{\partial}{\partial t}\rho + \nabla \cdot (\rho\mathbf{v}) = 0. \tag{2.3}$$

Balance of Linear and Angular Momentum

Consider an arbitrary testvolume $V(t)$ inside the fluid with surface S and outward normal $\mathbf{n}$. Two kinds of external forces acting on this volume can be distinguished. First volume-forces, like gravity, that are of the form

$$F_{vol}(V(t)) = \int\limits_{V(t)} \rho(\mathbf{x},t)\mathbf{f}(\mathbf{x},t)\ d\mathbf{x}$$

with an exterior volume-force $\mathbf{f}$ per unit mass – these forces apply to the entire mass of the fluid element, i.e. act on each point $\mathbf{x} \in V(t)$. Second surface- or contact-forces that are given by the material outside of the volume $V(t)$ on the material interior to the fluid element $V(t)$. These forces are described by a vector force $\mathbf{s}(\mathbf{x},t)$ acting on a unit area of S, which is called the stress vector. By Cauchy's principle $\mathbf{s}(\mathbf{x},t)$ only depends on the normal $\mathbf{n}$ and by the action-reaction principle must fulfil $\mathbf{s}_n(\mathbf{x},t) = -\mathbf{s}_{-n}(\mathbf{x},t)$, where we denote the dependency on the normal by an index.

Total surface forces are therefore given by

$$F_{surf}(V(t)) = \int\limits_{\partial V(t)} \mathbf{s}(\mathbf{x},t)\ ds.$$

As we show later on the surface-forces are given by a stress tensor σ and can be written as

$$\mathbf{s}_n(x,t) = \mathbf{n}\cdot\sigma = \sigma^T\cdot\mathbf{n}, \quad [\mathbf{s}_n(x,t)]_i = \sum_{j=1}^{3}\sigma_{ji}(\mathbf{x},t)n_j,\ i=1,2,3.$$

By the principles of conservation of linear and angular momentum we have the relation of these external forces to the temporal change of momentum, i.e.

$$\frac{d}{dt}\int\limits_{V(t)} \rho(\mathbf{x},t)\mathbf{v}(\mathbf{x},t)\ d\mathbf{x} = \int\limits_{V(t)} \rho(\mathbf{x},t)\mathbf{f}(\mathbf{x},t)\ d\mathbf{x} + \int\limits_{\partial V(t)} \mathbf{s}(\mathbf{x},t)\ ds \tag{2.4}$$

$$\frac{d}{dt}\int\limits_{V(t)} \mathbf{x}\times(\rho(\mathbf{x},t)\mathbf{v}(\mathbf{x},t))\ d\mathbf{x} = \int\limits_{V(t)} \mathbf{x}\times(\rho(\mathbf{x},t)\mathbf{f}(\mathbf{x},t))\ d\mathbf{x} + \int\limits_{\partial V(t)} \mathbf{x}\times\mathbf{s}(\mathbf{x},t)\ ds \tag{2.5}$$

with arbitrary reference point $\mathbf{x}$ and the cross product denoted by $\times$.

As in the derivation of (2.3) application of Reynold's transport theorem and the Gaussian theorem to equation (2.4) yields (for $i=1,2,3$)

$$\frac{\partial}{\partial t}(\rho(\mathbf{x},t)v_i(\mathbf{x},t)) + \nabla\cdot(\rho(\mathbf{x},t)v_i(\mathbf{x},t)\mathbf{v}(\mathbf{x},t)) = \rho(\mathbf{x},t)f_i(\mathbf{x},t) + [\nabla\cdot\sigma(\mathbf{x},t)]_i.$$

Written in vector notation, one gets the so called *conservative form of the momentum equation*

$$\frac{\partial}{\partial t}(\rho\mathbf{v}) + \nabla\cdot(\rho\mathbf{v}\mathbf{v}) = \rho\mathbf{f} + \nabla\cdot\sigma. \tag{2.6}$$

Here we used the notation $\mathbf{v}\mathbf{v} = \mathbf{v}\otimes\mathbf{v} = [v_i v_j]_{i,j}$ for short to express the outer vector product. Using the chain rule for $\frac{\partial}{\partial t}(\rho\mathbf{v})$ and $\nabla\cdot(\rho\mathbf{v}\mathbf{v}) = \mathbf{v}\nabla\cdot(\rho\mathbf{v}) + \rho(\mathbf{v}\cdot\nabla)\mathbf{v}$, the equation of momentum can be written as

$$\mathbf{v}\left(\frac{\partial}{\partial t}\rho + \nabla\cdot(\rho\mathbf{v})\right) + \rho\frac{\partial}{\partial t}\mathbf{v} + \rho(\mathbf{v}\cdot\nabla)\mathbf{v} = \rho\mathbf{f} + \nabla\cdot\sigma,$$

where the first term vanishes due to the continuity equation (2.3). The resulting equation is called *non-conservative form of the momentum equation*

$$\rho\frac{\partial}{\partial t}\mathbf{v} + \rho\left(\mathbf{v}\cdot\nabla\right)\mathbf{v} = \rho\mathbf{f} + \nabla\cdot\sigma. \tag{2.7}$$

For the balance of angular momentum (2.5) a similar but tedious calculation using again Reynold's transport theorem and the Gaussian theorem yields the final result (using the Einstein summation convention)

$$\int\limits_{V(t)} \varepsilon_{ijk}\sigma_{jk}\ d\mathbf{x} = 0$$

showing that σ *is a symmetric tensor*, i.e. $\sigma = \sigma^T$. For a detailed derivation of this tensor notation in the field of continuum mechanics we refer to [17]. All in all the basic equations of fluid mechanics for a single-phase continuous medium that does not exhibit electric, magnetic or chemical effects are given by the unknowns

- density $\rho = \rho(\mathbf{x},t) \in \mathbb{R}$,
- velocity vector $\mathbf{v} = \mathbf{v}(\mathbf{x},t) \in \mathbb{R}^3$,
- stress-tensor $\sigma = \sigma(\mathbf{x},t) \in \mathbb{R}^{3,3}$.

The derived conservation equations are (with given body force per unit mass $\mathbf{f}$)

- conservation of mass (continuity equation): $\partial_t \rho + \nabla \cdot (\rho \mathbf{v}) = 0$,
- balance of linear momentum (momentum equation): $\rho \partial_t \mathbf{v} + \rho(\mathbf{v} \cdot \nabla)\mathbf{v} = \rho \mathbf{f} + \nabla \cdot \sigma$,
- balance of angular momentum: $\sigma = \sigma^T$.

The next section is devoted to close the given system of unknowns and equations by formulating constitutive equations for particular types of materials.

2.1.2 Constitutive Equations

The derived equations are very generic and in principle valid for all fluids – up to now no restriction or assumption on the underlying material has been taken. By this generality we are faced with the problem that the number of unknowns and equations do not match. Nevertheless, it is sensible that all equations are given by the fluid one wants to describe, so that in addition to the general governing equations only material specific equations should appear. Such a relation between stress tensor σ, velocity $\mathbf{v}$ and density ρ usually depends on the fluid and is given by a suitable choice according to experimental results.

Stress Tensor

Throughout derivation of the conservation equations we have tacitly assumed that the surface forces on a volume element are given by the stress vector

$$\mathbf{s}(\mathbf{x}, t) = \mathbf{s}_n(\mathbf{x}, t) = \mathbf{n} \cdot \sigma(\mathbf{x}, t) = \sigma^T(\mathbf{x}, t) \cdot \mathbf{n}$$

with a stress tensor σ[1]. We now briefly want to describe the meaning of this tensor without a detailed derivation – for this purpose we refer to [6], [104], [130] or [154].

Surface stress $\mathbf{s}(\mathbf{x}, t)$ in general is a measure of the intensity of total internal forces acting within the fluid across imaginary internal surfaces, describing the influence of material outside a considered volume $V(t)$ to the material interior $V(t)$. Let dF denote an infinitesimal part of this force and let dA be an infinitesimal surface element on $\partial V(t)$ which covers $\mathbf{x}$, then by

$$\mathbf{s}(\mathbf{x}, t) = \mathbf{s}_n(\mathbf{x}, t) = \lim_{dA \to 0} \frac{dF}{dA}$$

the stress vector acting on $\mathbf{x}$ is defined. This force not only depends on the point $\mathbf{x}$ and time t, but also on the orientation of the surface element dA, i.e. on the outward normal $\mathbf{n}$. By Cauchy's stress principle the components of the stress vector $\mathbf{s}$ acting on a surface dA are fully described by the stress tensor σ and the outward normal $\mathbf{n}$ on dA

$$[\mathbf{s}_n(\mathbf{x}, t)]_i = \sum_j \sigma_{ji}(\mathbf{x}, t) n_j.$$

This linear relation for the stress vector $\mathbf{s}$ at all positions $\mathbf{x}$ to all directions $\mathbf{n}$, as stated by the *Cauchy stress law*, can be used to show descriptively an interpretation of the Cauchy stress tensor σ at a point $\mathbf{x}$ in an orthonormal system $\mathbf{e}_i$. If we take a surface element dA perpendicular to the unit-vector $\mathbf{e}_k$ (see Figure 2.1 for the case $\mathbf{e}_3$) then the i-component of the stress vector acting on this element in a point $\mathbf{x}$ is given by

$$[\mathbf{s}_{e_k}]_i = \sum_j \sigma_{ji} \delta_{jk} = \sigma_{ki}$$

where we wrote $\mathbf{s}_{e_k}$ to emphasise the stress vector on an element with outward normal e_k. So we have the following interpretation: σ_{ij} is the magnitude of the j-component of stress vector $\mathbf{s}$ (force

[1]It should be mentioned that the notation for the stress vector is not unique in literature. There are two possible ways to define the stress tensor σ, which base on transposed meaning of the components σ_{ij}. Fortunately, the stress tensor was previously shown to be symmetric so that also the definition $\mathbf{s}_n(\mathbf{x}, t) = \sigma(\mathbf{x}, t) \cdot n$ matches.

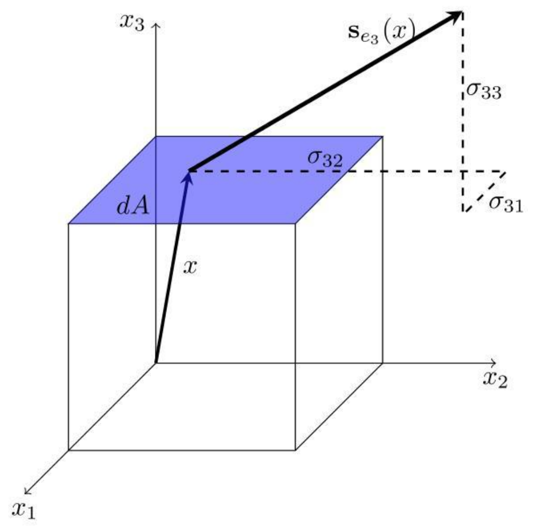


Figure 2.1: Components of the stress tensor: σ_{ij} stress in j direction on a face normal to the i axis.

per unit area) exerted across a plane surface element normal to the i-direction, at position $\mathbf{x}$ in the fluid and at time t. By this clarification we see that the diagonal elements of σ represent the *normal stresses*, i.e. normal component of surface force acting on a plane parallel to the coordinate planes, while the offdiagonal elements of σ are *tangential/shearing stresses.*

Fluids at Rest

A first observation is given for so called *Stokes' fluids*: when the fluid is in rest, only normal stresses are exerted. Furthermore, the stress-tensor is observed to be spherically symmetrical and depends directly on the thermodynamic pressure $p = p(\mathbf{x}, t)$ (for a derivation see [13]), therefore

$$\sigma|_{\mathbf{v}\equiv 0} = -pI.$$

This means that the shear stresses vanish and only normal stresses appear that are equal to the pressure. For a fluid in motion there is obviously no reason to expect this result being valid.

Stress-strain Law for Newtonian Fluids

If the fluid is in motion additional shear stresses must be considered altogether with the normal stresses. For the description of these friction-forces, which model the transport of momentum by molecular motion, the symmetric tensor τ (viscous or deviatoric stress tensor) is added

$$\sigma|_{\mathbf{v}\neq 0} = -pI + \tau.$$

The question now is how this viscous stress is related to other observable properties of common fluid motions. As a primary source of information experiments for which one could propose many such relations (called constitutive equations) can be made. But also from the theoretical point of view some approach can be done: friction in fluids generally manifests itself through shearing forces which retard the relative motion of fluid particles. A measure of the relative motion of fluid particles is given by the deformation rate tensor.

Definition 2.1.1
The tensor $D = \frac{1}{2}(\nabla\mathbf{v} + (\nabla\mathbf{v})^T)$ is called the deformation rate tensor. The tensor $W = \frac{1}{2}(\nabla\mathbf{v} - (\nabla\mathbf{v})^T)$ is called the rotation rate tensor. They build the symmetric and antisymmetric decomposition of the velocity gradient: $\nabla\mathbf{v} = D + W$. A fluid is a medium whose stress-strain law is of the form $\sigma = f(D)$.

The concrete identification of the material-function f is the subject of rheology. Here for the sake of simplicity we deal only with so-called *Newtonian fluids*, for which the hypothesis of a linear dependence of the stress tensor on the deformation rate tensor is made. The particular form of the linear function f for a Newtonian fluid is given by so-called *Stokes' Postulates* which are made on the basis of experiments and require that:

1. $\sigma = -pI + f(D)$ with a linear continuous function f.
2. The fluid is an isotropic medium, i.e. its properties are the same in all space directions. This means that the function f is invariant to orthogonal transformations, i.e. $f(SDS^T) = Sf(D)S^T$ for all transformations S with $SS^T = I,\ \det(S) = 1$.
3. If the fluid is at rest, there are no viscous stresses, i.e. $f(0) = 0$.

These conditions lead to a stress-strain law of Newtonian fluid termed the *Cauchy-Poisson law* – for proof see [6] or [50, ch. 1.8]

$$\sigma = (-p + \lambda\nabla\cdot\mathbf{v})I + 2\mu D$$

with λ (volume viscosity) and μ (shear or dynamic viscosity) being constants or scalar functions of thermodynamical quantities. Hence, we have the general equations of motion for a Newtonian fluid

$$\begin{aligned} \partial_t\rho + \nabla\cdot(\rho\mathbf{v}) &= 0,\\ \rho\partial_t\mathbf{v} + \rho(\mathbf{v}\cdot\nabla)\mathbf{v} + \nabla p - \nabla(\lambda\nabla\cdot\mathbf{v}) - \nabla\cdot(2\mu D) &= \rho\mathbf{f}. \end{aligned}$$

Incompressibility

The condition of incompressibility can be stated at least in twofold manner:

- an incompressible fluid has constant volume: using Reynolds transport theorem (2.2) with $f \equiv 1$ results in $\nabla\cdot\mathbf{v} = 0$,
- an incompressible fluid has constant density: we directly get $\nabla\cdot\mathbf{v} = 0$ from the continuity equation.

Nevertheless, these definitions aim at the same result and the stress-tensor of an incompressible Newtonian fluid simplifies to

$$\sigma = -pI + 2\mu D$$

which gives the momentum equation

$$\rho(\partial_t\mathbf{v} + (\mathbf{v}\cdot\nabla)\mathbf{v}) + \nabla p - \nabla\cdot(2\mu D) = \rho\mathbf{f}.$$

Using the notation $\nu = \mu/\rho$ for the kinematic viscosity and assuming that the shear viscosity μ is constant within the fluid, we simplify $\nabla\cdot(2\mu D) = \mu(\Delta\mathbf{v} + \nabla(\nabla\cdot\mathbf{v})) = \mu\Delta\mathbf{v}$, due to incompressibility. Therefore we end up with the *incompressible Navier-Stokes equations*

$$\begin{aligned} \partial_t\mathbf{v} - \nu\Delta\mathbf{v} + (\mathbf{v}\cdot\nabla)\mathbf{v} + \frac{1}{\rho}\nabla p &= \mathbf{f},\\ \nabla\cdot\mathbf{v} &= 0. \end{aligned} \tag{2.8}$$

Next, the general body force $\mathbf{f}$ and also suitable boundary conditions need to be derived in detail such that the system (2.8) can be defined in a strict mathematical setting.

2.1.3 Electrical Excitation of the Fluid and Mass-Transport

Besides the generation of fluid flow by the classical application of a pressure difference, e.g. by mechanical pumps or gravity force, there are also several other opportunities. One possibility in the setting of microflows is electroosmose. The fundamental effect for such an electrical excitation is related to the presence of a considerable charge density near channel walls, where ions of opposite sign are attracted to the electrical charged surface – an effect called *electric double layer (EDL)*.

The thickness of EDL is defined by the region where the electric neutrality of the fluid is violated and is usually on the order of 10^{-9} m. For a detailed description of electrical excitation within microflows we refer to [12] and references therein. To generate an electroosmotic flow, a tangential electric field is applied to the EDL which causes an ion drag that induces a fluid flow. If now the ratio of surface to volume is as large as in microchannels, the flow within the EDL effects also the entire flow field within the channel by viscous forces.

The influence of such an electric field on the momentum equation of an incompressible fluid can be considered as an external force in equation (2.8) (cf. [11])

$$\mathbf{f}_e = q\mathbf{E}. \tag{2.9}$$

This Coulomb force $\mathbf{f}_e$ describes the force exerted by the electric field $\mathbf{E}$ upon the total electric charge density q within the fluid. It was shown in [11] that under certain conditions, namely a homogeneous ion concentration and electrodes located far away from the point of interest (cf. Figure 1.1), the electrical field can be assumed as independent of flow and concentration fields, at least for a certain time. Hence, an electrostatic formulation by Gauss' law relating the distribution of electric charge to the resulting electric field seems justified, i.e. with the permittivity of the fluid ε and the electric potential ϕ we have

$$\nabla \cdot (\varepsilon \mathbf{E}) = \nabla \cdot (-\varepsilon \nabla \phi) = q. \tag{2.10}$$

It remains to describe the charge density of the fluid q to get a governing equation for the electrical situation within the fluid. Within the EDL, the charge density is $q \neq 0$ due to a excess of ions and can be expressed by

$$q = F \sum_j z_j c_j \tag{2.11}$$

with c_j denoting the concentration of the ion species j, z_j denoting the valency number of the ion species and F the Faraday constant. Outside of the EDL, within the so-called bulk region, the liquid can be approximated to be in electrical neutrality and we have

$$\sum_j z_j c_{j,bulk} \approx 0.$$

In this case, i.e. the charge density q is zero like in electrical neutral fluids, equation (2.10) reduces to the Laplace equation $\Delta\phi = 0$ assuming a spatial constant permittivity ε of the fluid. Moreover, it is obvious that electrical forces can only be induced within a small region near the channel walls, namely the EDL.

For the ion concentration within the EDL $c_{j,EDL}$ a Boltzmann distribution is used (cf. [12]) such that the electric potential ϕ within the fluid (using (2.10), (2.11) and a spatial constant permittivity) is described by the so-called Poisson-Boltzmann equation

$$\Delta\phi = -\frac{F}{\varepsilon} \sum_j z_j c_{j,bulk} \exp\left(-\frac{z_j F \phi_i}{RT}\right). \tag{2.12}$$

Here, $c_{j,bulk}$ is the (constant) concentration of the ion species j within the bulk flow and ϕ_i is the potential induced by the wall surface charge. R and T denote the gas constant and temperature. Suitable boundary conditions for the potential ϕ are given by Dirichlet values at the electrodes and no-flux condition at the channel walls.

Equation (2.12) has now to be added to the Navier-Stokes equations (2.8) to determine the electrical force for the momentum equation

$$\mathbf{f}_e = -q\nabla\phi.$$

Since the electric double layer is usually very small even compared to a microchannel width, numerical computations have to be performed on very fine meshes to resolve the effects near the channel walls resulting in very high or even prohibitive computational costs. The work of Barz et al. [11] shows that the influence of the EDL can be captured by suitable slip boundary condition for the velocity-field within the bulk region $\mathbf{v}_{bulk}$, namely

$$\mathbf{v}_{bulk} \cong -C\nabla\phi, \quad \mathbf{v}_{bulk} \cdot \mathbf{n} = 0$$

on the interface of bulk flow and EDL. The second condition is automatically fulfilled since a no-flux condition for the potential ϕ is used, i.e. $\mathbf{v}_{bulk} \cdot \mathbf{n} = -C\nabla\phi \cdot \mathbf{n} = -C\partial_n\phi = 0$. In detail, the outcome of the method of matched asymptotic expansions reveals the following set of equations

$$\begin{aligned} \partial_t\mathbf{v} - \nu\Delta\mathbf{v} + (\mathbf{v}\cdot\nabla)\mathbf{v} + \frac{1}{\rho}\nabla p &= 0, \\ \nabla\cdot\mathbf{v} &= 0, \\ \Delta\phi &= 0, \end{aligned} \tag{2.13}$$

which has to be employed to simulate the flow and electric potential in the liquid bulk outside the EDL. Comparing (2.13) to (2.8) we see that the electrical force $\mathbf{f}_e$ vanishes which is due to the electrical neutrality within the liquid bulk. The electrical double layer, in which an electroosmotic flow is induced by the electrical field, is excluded of the numerical treatment – the influence on the liquid bulk is captured by the boundary conditions.

Finally, we extend the system (2.13) by the transport of some species of concentration c within the fluid described by the convection-diffusion equation

$$\partial_t c - D\Delta c + \mathbf{v}\cdot\nabla c = 0. \tag{2.14}$$

Assuming that there are no sources or sinks within the volume V, the rate of change for a scalar quantity in an arbitrary volume is given by total flux into and out of V, i.e.

$$\frac{d}{dt}c = \partial_t c + \nabla\cdot(c\mathbf{v}) = D\Delta c.$$

Herein the right hand side $D\Delta c$ describes the diffusive flux and since we assume an incompressible Newtonian fluid it holds $\nabla\cdot(c\mathbf{v}) = \mathbf{v}\cdot\nabla c$.

Remark 2.1.1

In the derived governing equations (2.13) the bulk flow field $[\mathbf{v}, p]$ is only coupled to the potential ϕ by boundary conditions on the interface to the electric double layer. The Laplace equation describing the potential is fully decoupled from the bulk flow field. Hence, one might first solve for the potential ϕ and then solve the Navier-Stokes equations to get the flow field. In the same manner, we assume that the convection-diffusion equation (2.14) is only coupled to the bulk velocity $\mathbf{v}$ and there is no feedback to neither the flow field nor the potential (which holds only when liquids containing almost no charges are mixed). Therefore, also the convection-diffusion equation might be solved decoupled, i.e. after the flow field $\mathbf{v}$ is computed.

2.1.4 Non-dimensionalisation and Initial-/Boundary-Conditions

We previously derived the governing equations for the experimental setting in Chapter 1, namely the incompressible Navier-Stokes equations describing the flow field of a Newtonian fluid in combination with the potential equation to cover electroosmotic effects (2.13) and the convection-diffusion equation to describe the concentration field of some species/dye (2.14). The variables within this system are the physical quantities

- velocity $\mathbf{v}$ in $\frac{m}{s}$,
- pressure p in $Pa = \frac{kg}{ms^2}$,
- electric potential ϕ in $V = \frac{m^2kg}{s^3A}$,
- concentration of the dye c in $\frac{kg}{m^3}$.

Again, we emphasise that these equations describe only the bulk flow whenever electroosmotic effects are taken into account.

In order to compare fluid flow, one usually scales the equations by characteristic quantities to eliminate the physical dimensions. Common behaviour of fluid flow can hence be characterised by parameters like kinematic viscosity ν, density ρ, characteristic velocity scale v_0 and characteristic length scale d_0. Furthermore, a convective time scale $t_0 = d_0/v_0$, pressure scale $p_0 = (v_0\mu)/d_0$, concentration scale c_0 and applied potential scale ϕ_0 are defined, such that

$$X = \frac{\mathbf{x}}{d_0},\ T = \frac{tv_0}{d_0},\ \mathbf{V} = \frac{\mathbf{v}}{v_0},\ P = \frac{pd_0}{\mu v_0},\ \Phi = \frac{\phi}{\phi_0},\ C = \frac{c}{c_0}.$$

Substitution of these quantities in equations (2.13) and (2.14) results in

$$\begin{aligned} \frac{v_0^2}{d_0}\partial_T\mathbf{V} - \frac{\nu v_0}{d_0^2}\Delta\mathbf{V} + \frac{v_0^2}{d_0}(\mathbf{V}\cdot\nabla)\mathbf{V} + \frac{\nu v_0}{d_0^2}\nabla P &= 0, \\ \frac{v_0}{d_0}\nabla\cdot\mathbf{V} &= 0, \\ \frac{\varphi_0}{d_0^2}\Delta\Phi &= 0, \\ \frac{v_0 c_0}{d_0}\partial_t C - \frac{Dc_0}{d_0^2}\Delta C + \frac{v_0 c_0}{d_0}\mathbf{V}\cdot\nabla C &= 0. \end{aligned}$$

Introducing the dimensionless *Reynolds number*

$$Re = \frac{\rho v_0 d_0}{\mu} = \frac{v_0 d_0}{\nu}$$

and *Schmidt number*

$$Sc = \frac{\nu}{D}$$

we can therefore write for short

$$\begin{aligned} Re[\partial_t\mathbf{v} + (\mathbf{v}\cdot\nabla)\mathbf{v}] - \Delta\mathbf{v} + \nabla p &= 0, \\ \nabla\cdot\mathbf{v} &= 0, \\ \Delta\phi &= 0, \\ \partial_t c - \frac{1}{Re\cdot Sc}\Delta c + \mathbf{v}\cdot\nabla c &= 0, \end{aligned} \tag{2.15}$$

which is used for any further consideration[2]. The Reynolds number describes the relation between inertia and viscous forces, while the Schmidt number describes the ratio of momentum diffusivity (viscosity) and mass diffusivity. The derivation of governing equations now has to be completed with additional boundary conditions on $\partial\Omega$ and initial conditions at $t = 0$ in order to describe the physical problem entirely.

Initial Conditions

The velocity field $\mathbf{v}_0$ at $t = 0$ obviously has to be solenoidal, i.e. $\nabla\cdot\mathbf{v}_0 = 0$, and for simplicity one often chooses $\mathbf{v}_0 \equiv 0$, assuming that there is no initial flow. An initial condition for the pressure p and the potential ϕ is not needed, whereas the concentration c at $t = 0$ needs to be chosen with care. We assume that $c(\mathbf{x}, t) \in [0, 1]$ for all $\mathbf{x} \in \Omega$ and $t \geq 0$. As initial distribution of the concentration field two major possibilities are given

1. $c(\mathbf{x}, 0) = 0$ in Ω – means that the starting/inflow phase has to be simulated in which only one species concentration is initially present,
2. $c(\mathbf{x}, 0) = c_0(\mathbf{x}) \in [0, 1]$ in Ω – means that a given distribution is assumed and only the development of concentration has to be simulated.

[2]For sake of simplicity we skipped the upper-case notation. The reader will obviously identify the dimensionless system by appearance of Reynolds and/or Schmidt number.

Boundary Conditions

Let the boundary $\partial\Omega$ by subdivided into disjoint parts

- Γ_{in} describing the inflow part of the channel,
- Γ_{out} describing the outflow part of the channel,
- Γ_0 describing the rigid walls of the channel or (in case of additional electroosmotic flow) the interface between bulk flow and electric double layer.

Obvious conditions can be posed for the concentration field c. On the inflow section Γ_{in}, we assume Dirichlet-values by a given function c_{in} describing a separation in 1 and 0 to model the contact area of two inflow-channels that are filled only by one of the concentrations (cf. Figure 1.1). For $\Gamma_{out} \cup \Gamma_0$ a vanishing flux is used, i.e. $\partial_n c = 0$.

The electric potential ϕ on Γ_0 is also determined by a no flux condition $\partial_n \phi = 0$. On the inflow and outflow parts of $\partial\Omega$, prescribed inlet and outlet potentials ϕ_{in} and ϕ_{out} are given which are scaled by the applied potential difference ϕ_0 between the reservoirs. Since only the potential difference $\Delta\phi$ between Γ_{in} and Γ_{out} is of interest, we assume for the sake of simplicity $\phi_{in} = 0$ and $\phi_{out} = \Delta\phi$.

For the Navier-Stokes equations within the channel, the setting of boundary conditions is a very sophisticated topic. While the condition at rigid walls is usually determined by a so-called *no-slip condition* for viscous fluid

$$\mathbf{v}|_{\Gamma_0} = 0,$$

the conditions set on inflow and outflow boundaries are manifold like:

- Dirichlet conditions for the velocity which are based on the observation of parabolic Poiseuille flow profile (the Poiseuille flow is given by an analytical solution of the Navier-Stokes equations driven by a constant pressure gradient within cylindrical infinite long straight channels [139]),
- so called *do-nothing condition* $\partial_n \mathbf{v} - p\mathbf{n} = 0$ which is naturally given in the framework of weak formulation and is most often used to describe a free outflow boundary,
- given pressure difference, i.e. given mean pressure $\frac{1}{|S_i|} \int\limits_{S_i} p \, ds = P_i$ on each outlet S_i.

We depict these possibilities in detail when the weak formulation of the Navier-Stokes equations is introduced and refer to [18, 67] for the first two approaches and to [84] for the third.

When a pure pressure-driven flow will be simulated/optimised, the no-slip condition on rigid walls is used in combination with a given inflow velocity profile plus do-nothing condition for the outflow – alternatively also the pressure difference formulation might be used. Alternatively, when additional electroosmotic effects are taken into account, the computational domain is restricted to the bulk fluid flow region only, i.e. the very thin electrical double layer near channel walls is omitted. Hence, the no-slip condition for the velocity field cannot be used. Barz et al. [11] showed that a separation of the bulk flow and the flow within the EDL need obviously to treat the electrical force $\mathbf{f} = \mathbf{f}_e$ in (2.8) in different manner since a considerable charge density q is only given within the EDL. The asymptotic approximations and leading-order analysis carried out in [11] reveals a solution of the velocity field within the EDL that directly depends on the gradient of the applied potential ϕ. Consistently the numerical bulk solution should match the EDL solution such that a slip boundary condition for the bulk solution

$$\mathbf{v}|_{\Gamma_0} = -\Pi_2 \nabla\phi = \Pi_2 \mathbf{E}.$$

is given showing that any wall-tangential component of the applied electrical field drives the bulk flow. Indeed, since for the potential ϕ a no-flux condition $\partial_n \phi = 0$ is given at the *virtual boundary* between bulk flow and EDL, the wall-normal components of the applied electrical field do not influence the bulk flow. The dimensionless parameter (see the Appendix for used quantities)

$$\Pi_2 = \frac{\ell_D \varphi_0 q_\zeta}{v_0 d_0 \mu}$$

was derived in [11] by non-dimensionalisation of the Navier-Stokes equations within the EDL and can be interpreted as the ratio of electrical to viscous forces – it strongly depends on the electric properties of the fluid and the channel walls.

2.2 Analytical Framework

Before considering optimisation problems for fluid flow, one obviously first has to answer the question whether there is at all a solution to the underlying equations and whether this solution is unique. For this, we want to point out the weak formulation of Navier-Stokes equations and for complete electrokinetic problem, i.e. with additional potential and convection-diffusion equation. Furthermore, some existence and uniqueness results for weak Navier-Stokes equations are cited but are not presented in detail, since we only utilise the weak formulation of partial differential equations to have a starting point for the finite element based discretisation in the next chapter.

Remark 2.2.1
In the following we will almost always restrict formulations to the Navier-Stokes equations, for some reasons

- *the governing equations (2.15) of the electroosmotic driven bulk flow field, i.e. the Navier-Stokes equations, might be solved decoupled from the Laplace and convection-diffusion equation,*
- *we will also consider optimisation problems without any electrokinetic effects,*
- *the Navier-Stokes equations require a more sophisticated analysis than Laplace and convection-diffusion equation.*

For analysis of Laplace and convection diffusion equation we refer to e.g. [34, 68].

As we will show in the next section, the solvability of optimisation problems generally relies on the possibility to define an operator, called control-state operator, that assigns a unique solution of the partial differential equation to each control given. This means, e.g. for boundary control, that we have to show whether there exists a unique solution of the Navier-Stokes equations to given boundary data $\mathbf{g}$ on $\Gamma_c \subset \partial\Omega$. We therefore consider in this section the dimensionless Navier-Stokes equations with general Dirichlet boundary values

$$\begin{aligned} Re[\partial_t \mathbf{v} + (\mathbf{v}\cdot\nabla)\mathbf{v}] - \Delta\mathbf{v} + \nabla p &= \mathbf{f} && \text{in } \Omega\times(0,T) \\ \nabla\cdot\mathbf{v} &= 0 && \text{in } \Omega\times(0,T) \\ \mathbf{v} &= \mathbf{g} && \text{on } \partial\Omega\times(0,T) \\ \mathbf{v} &= \mathbf{v}_0 && \text{in } \Omega \text{ for } t=0 \end{aligned} \tag{2.16}$$

and give an overview of the present existence and uniqueness results. An extension to different boundary conditions will also be presented, as far as it is needed in the framework of considered microflows.

2.2.1 Function Spaces and Notation

We introduce the following notation for a bounded, connected, open set $\Omega \subset \mathbb{R}^d,\ d \in \mathbb{N}$ (usually $d = 2, 3$) with a Lipschitz-continuous boundary $\partial\Omega$. These definitions are standard and for more detailed accounts concerning these spaces we refer to [2, 62].

A real valued function $f : \Omega \to \mathbb{R}$ is defined in the Lebesgue-space $L^p(\Omega),\ 1 \le p \le \infty$ if it is Lebesgue-measurable und its norm

$$\|f\|_{L^p(\Omega)} = \begin{cases} \left(\int\limits_\Omega |f(x)|^p\, dx\right)^{1/p}, & 1 \le p < \infty \\ \operatorname*{esssup}_{x\in\Omega} |f(x)| = \inf\limits_{\mu(N)=0} \sup\limits_{x\in\Omega\setminus N} |f(x)|, & p = \infty \end{cases}$$

is finite. Let $(X, \|\cdot\|)$ be a Banach space of functions on Ω and let a time-interval be given as $I = (a, b) \subset \mathbb{R}$. Any so-called *abstract function* $f(t) : (a, b) \to X$ is defined in $L^p(a, b; X)$, $1 \leq p \leq \infty$ if f is Lebesgue-measurable and its norm

$$\|f\|_{L^p(a,b;X)} = \begin{cases} \left(\int\limits_a^b \|f(t)\|_X^p \, dt\right)^{1/p} & ,1 \leq p < \infty \\ \operatorname*{esssup}\limits_{a<t<b} \|f(t)\|_X & ,p = \infty \end{cases}$$

is finite. The definition of abstract functions $f \in L^p(a, b; X)$ also allows a more general definition of an integral, namely the *Bochner integral* which reduces to the Lebesgue integral for $X = \mathbb{R}$ (cf. [163]).

Next we recall the definition of Sobolev spaces $H^m(\Omega)$ or more general $W^{m,p}(\Omega)$. For a multi-index $\alpha = (\alpha_1, ..., \alpha_d)$ and $|\alpha| = \alpha_1 + ... + \alpha_d \in \mathbb{N}_0$ we have the partial derivative of f by

$$D^\alpha f(\mathbf{x}) = \frac{\partial^{|\alpha|} f}{\partial x_1^{\alpha_1} \cdots \partial x_d^{\alpha_d}}(\mathbf{x}), \quad \mathbf{x} = [x_1, \ldots, x_d] \in \Omega$$

with

$$D^{e_i} f(\mathbf{x}) = \frac{\partial f}{\partial x_i}(\mathbf{x}).$$

Let $f \in L^p(\Omega)$, α a multiindex, then $f^{(\alpha)} \in L^p(\Omega)$ is called the weak derivative of f (or derivative in distributional sense) if for all $\varphi \in C_0^\infty(\Omega)$ the following identity holds

$$\int\limits_\Omega f D^\alpha \varphi \, d\mathbf{x} = (-1)^{|\alpha|} \int\limits_\Omega f^{(\alpha)} \varphi \, d\mathbf{x}$$

then we identify: $D^\alpha f = f^{(\alpha)}$. The Sobolev space $W^{m,p}(\Omega)$ is defined as

$$W^{m,p}(\Omega) = \{f \in L^p(\Omega) : \; D^\alpha f \in L^p(\Omega), \; \forall \alpha : \; |\alpha| \leq m\}.$$

$W^{m,p}(\Omega)$ is a vectorspace and with the norm

$$\|f\|_{W^{m,p}(\Omega)} = \left(\sum_{|\alpha| \leq m} \|D^\alpha f\|_{L^p(\Omega)}^p\right)^{1/p}$$

it is a Banach-space for all $m \in \mathbb{N}_0$. Furthermore, we define $W_0^{m,p}(\Omega)$ as the closure of $C_0^\infty(\Omega)$ in the $W^{m,p}$ norm. For $p = 2$, $W^{m,2}(\Omega) =: H^m(\Omega)$ is a Hilbert-space with scalar-product

$$(f, g)_{H^m(\Omega)} := \sum_{|\alpha| \leq m} (D^\alpha f, D^\alpha g)_{L^2(\Omega)} = \sum_{|\alpha| \leq m} \int\limits_\Omega D^\alpha f D^\alpha g \, d\mathbf{x}, \quad f, g \in H^m(\Omega).$$

Obviously, for the case $m = 0$ we have that $H^0(\Omega) = L^2(\Omega)$ and we will denote the scalar-product and norm by $(\cdot, \cdot)$ resp. $\|\cdot\|$, i.e. we omit the index. Any further investigation on the weak form of Navier-Stokes equations relies mainly on the space $H^1(\Omega)$ for the velocity field $\mathbf{v}$ in which it holds[3]

$$(f, g)_1 = (f, g)_{H^1(\Omega)} = \int\limits_\Omega fg \, d\mathbf{x} + \int\limits_\Omega \nabla f \cdot \nabla g \, d\mathbf{x},$$
$$\|f\|_1 = \|f\|_{H^1(\Omega)} = (\|f\|^2 + \|\nabla f\|^2)^{\frac{1}{2}}.$$

[3]One should keep in mind that all derivatives are given in the sense of distributions, i.e. $\nabla f = [\frac{\partial f}{\partial x_1}, \ldots, \frac{\partial f}{\partial x_d}]$ with $\langle \frac{\partial f}{\partial x_i}, \varphi \rangle = -\int_\Omega f \frac{\partial \varphi}{\partial x_i} \, d\mathbf{x}, \forall \varphi \in C_0^\infty(\Omega)$.

The treatment of the pressure within (2.16) is usually handled by the space

$$L_0^2(\Omega) = \{f \in L^2(\Omega) : \int_\Omega f \; d\mathbf{x} = 0\}$$

since the formulation ∇p allows to determine the pressure only up to an additive constant.

We close the definitions by introducing dual and trace spaces (cf. [144]). A functional F defined on $H^1(\Omega)$ is in the dual space $H^{-1}(\Omega)$ if and only if

$$\|F\|_{-1} = \sup_{\|u\|_1=1} |F(u)| < \infty.$$

Evaluation of a functional $F \in H^{-1}(\Omega)$ at $u \in H^1(\Omega)$ is often also denoted by the duality pairing $\langle F, u\rangle$. Let the boundary of Ω be a C^2-boundary $\Gamma = \partial\Omega$, then a continuous linear operator $\gamma : H^1(\Omega) \to L^2(\Gamma)$ is given such that $\gamma(u)$ is the restriction of u onto Γ for all $u \in H^1(\Omega) \cap C^2(\bar{\Omega})$. $H_0^1(\Omega)$ is equal to the kernel of γ and the image $\gamma(H^1(\Omega))$ is a dense subset of $L^2(\Gamma)$ which is denoted by $H^{1/2}(\Gamma)$.

It remains to define the notation for vector valued function spaces: we suppress the dimension index and instead use a bold face notation, i.e.

$$\begin{aligned}
\mathbf{H}^k(\Omega) &= [H^k(\Omega)]^d = \{v_i \in L^2(\Omega) : \; D^\alpha v_i \in L^2(\Omega), \; \forall\alpha : |\alpha| \le k, \; i = 1, \dots, d\},\\
\mathbf{H}_g^k(\Omega) &= \{v \in \mathbf{H}^k(\Omega) : \; \mathbf{v} = \mathbf{g} \text{ on } \partial\Omega\} \subset \mathbf{H}^k(\Omega),\\
\mathbf{H}^{-1}(\Omega) &= [H^{-1}(\Omega)]^d = \{\, F : \mathbf{H}_0^1(\Omega) \;\to\; \mathbb{R} \; : \; \|F\|_{-1} < \infty \; \forall \mathbf{v} \in \mathbf{H}_0^1(\Omega)\},\\
\mathbf{H}^{1/2}(\Gamma) &= \{\mathbf{v} : \; v_i \in H^{1/2}(\Gamma), \; i = 1, \dots, d\}.
\end{aligned}$$

These spaces are equipped with the norms (denoted by $\|\cdot\|$) or semi-norm (denoted by $|\cdot|$)

$$\begin{aligned}
\|\mathbf{v}\|_k^2 &= \sum_{i=1}^d \|v_i\|_k^2, \qquad \|v_i\|_k^2 = \|v_i\|^2 + \sum_{|\alpha|=1}^k \|D^\alpha v_i\|^2, && \mathbf{v} \in \mathbf{H}^k(\Omega),\\
|\mathbf{v}|_1^2 &= \sum_{i=1}^d \Big(\sum_{j=1}^d \|\frac{\partial v_i}{\partial x_j}\|^2\Big), && \mathbf{v} \in \mathbf{H}^1(\Omega),\\
\|F\|_{-1} &= \sup_{0 \ne \mathbf{v} \in \mathbf{H}_0^1(\Omega)} \frac{\langle F, \mathbf{v}\rangle}{|\mathbf{v}|_1}, && F \in \mathbf{H}^{-1}(\Omega),
\end{aligned}$$

where in $\mathbf{H}_0^1(\Omega)$ the seminorm $|\cdot|_1$ is equivalent to $\|\cdot\|_1$.

2.2.2 Weak Formulation

The Navier-Stokes equations were derived in Chapter 2.1 on the basis of physical laws and observations. So a solution should also be kind of physically reasonable – within the official problem description of the Millennium Problem [49] this is

$$\mathbf{v}, \; p \in C^\infty(\mathbb{R}^d \times [0, \infty)) \text{ and } \int_{\mathbb{R}^d} |\mathbf{v}(x,t)|^2 \; d\mathbf{x} < C \quad \text{for all } t \ge 0.$$

This formulation appears very restrictive in the setting of regularity properties for the solution $[\mathbf{v}, p]$. A less restrictive concept for determining solutions to the Navier-Stokes equations (that is by the way also very useful for finite element based discretisation) is to weaken the spaces in which a solution is searched for. To this, the formulation (2.16) is modified to an integral mean representation, i.e. after multiplication with suitable test functions one integrates over the domain Ω. Furthermore, integration by parts based on Green's formula is used to loosen the regularity properties for solutions of the Navier-Stokes equations by reducing the order of differential operators. These steps achieve that an existence and uniqueness theory within more convenient spaces can be established which afterwards might be restored to the original problem. As long as the pair $[\mathbf{v}, p]$ in its so-called *weak formulation* is sufficiently smooth to allow for the used integration by parts, it is obviously also a *strong solution* of (2.16).

We state the fundamental theorem of partial integration in its general form using Sobolev spaces of fractional order – for the case $p = q = 2$ the trace space $H^{1/2}$ was already defined, for other cases refer to [144].

Theorem 2.2.1
Let $\Omega \subseteq \mathbb{R}^d$, $d \geq 2$, be a bounded Lipschitz-domain with boundary Γ. Further let $1 < p < \infty$ and $1/p + 1/q = 1$, then for all $u \in W^{1,p}(\Omega)$ and $\mathbf{v} \in \mathbf{W}^{1,q}(\Omega)$ holds the trace-condition

$$u|_\Gamma \in W^{1-1/p,p}(\Gamma), \;\; \mathbf{v} \cdot \mathbf{n}|_\Gamma \in W^{1-1/q,q}(\Gamma)$$

and

$$\int\limits_\Omega u \nabla \cdot \mathbf{v} \; d\mathbf{x} = \int\limits_\Gamma u \; \mathbf{v} \cdot \mathbf{n} \; dS - \int\limits_\Omega \nabla u \mathbf{v} \; d\mathbf{x}$$

with $\mathbf{n}$ being the outer normal on Γ.

Based on Theorem 2.2.1 we define the bilinear forms used in the sequel for the weak formulation of the Navier-Stokes equations.

$$a(\mathbf{u}, \mathbf{v}) = \int\limits_\Omega \nabla \mathbf{u} : \nabla \mathbf{v} \; d\mathbf{x} = \int\limits_\Omega \sum_{i,j=1}^{d} \frac{\partial u_i}{\partial x_j} \frac{\partial v_i}{\partial x_j} \; d\mathbf{x} \qquad \forall \mathbf{u}, \mathbf{v} \in \mathbf{H}^1(\Omega), \tag{2.17}$$

which is continuous on $\mathbf{H}^1(\Omega) \times \mathbf{H}^1(\Omega)$, i.e. there exists a constant $C > 0$ such that

$$|a(\mathbf{u}, \mathbf{v})| \leq C \|\mathbf{u}\|_1 \|\mathbf{v}\|_1 \quad \forall \mathbf{u}, \mathbf{v} \in \mathbf{H}^1(\Omega)$$

and coercive (or elliptic) on $\mathbf{H}^1_0(\Omega) \times \mathbf{H}^1_0(\Omega)$, i.e. there exists a constant $\alpha > 0$ such that

$$|a(\mathbf{v}, \mathbf{v})| \geq \alpha \|\mathbf{v}\|_1^2 \quad \forall \mathbf{v} \in \mathbf{H}^1(\Omega).$$

Furthermore, we define the bilinear form

$$b(\mathbf{v}, q) = -\int\limits_\Omega q \nabla \cdot \mathbf{v} \; d\mathbf{x} \qquad \forall \mathbf{v} \in \mathbf{H}^1(\Omega) \text{ and } q \in L^2(\Omega), \tag{2.18}$$

which is continuous on $\mathbf{H}^1(\Omega) \times L^2(\Omega)$ and the trilinear form

$$c(\mathbf{w}, \mathbf{u}, \mathbf{v}) = \int\limits_\Omega (\mathbf{w} \cdot \nabla)\mathbf{u} \cdot \mathbf{v} \; d\mathbf{x} = \sum_{i=1}^{d} \int\limits_\Omega (\mathbf{w} \cdot \nabla u_i) v_i \; d\mathbf{x} \qquad \forall \mathbf{u}, \mathbf{v}, \mathbf{w} \in \mathbf{H}^1(\Omega), \tag{2.19}$$

that is continuous on $\mathbf{H}^1(\Omega) \times \mathbf{H}^1(\Omega) \times \mathbf{H}^1(\Omega)$ for $d \leq 4$ and Ω bounded with Lipschitz-boundary. Proofs for continuity and coercivity are standard and can be found for example in [61].

Definition 2.2.1 (Weak solution of Navier-Stokes equations)
A pair $[\mathbf{v}, p]$ is called weak solution of (2.16), if $\mathbf{v}(\cdot, 0) = \mathbf{v}_0$ and for almost all $t \in (0, T)$ it holds: $[\mathbf{v}(\cdot, t), p(\cdot, t)] \in \mathbf{H}^1_g(\Omega) \times L^2_0(\Omega)$ is a solution of

$$\begin{aligned} Re[\langle \partial_t \mathbf{v}, \varphi \rangle + c(\mathbf{v}, \mathbf{v}, \varphi)] + a(\mathbf{v}, \varphi) + b(\varphi, p) &= \langle \mathbf{f}, \varphi \rangle \qquad && \forall \varphi \in \mathbf{H}^1_0(\Omega), \\ b(\mathbf{v}, \xi) &= 0 && \forall \xi \in L^2_0(\Omega), \end{aligned} \tag{2.20}$$

for given $\mathbf{f} \in \mathbf{H}^{-1}(\Omega)$ and $\mathbf{v}_0 \in \{\mathbf{u} \in \mathbf{H}^1(\Omega) : \; b(\mathbf{u}, q) = 0, \; \forall q \in L^2(\Omega)\}$.

For the weak formulation (2.20) the prescribed Dirichlet boundary condition $\mathbf{v} = \mathbf{g}$ on $\Gamma \times (0,T)$ is an essential one and has to be fulfilled by the chosen function space $\mathbf{H}^1_g(\Omega)$, i.e. we seek a function $\mathbf{v} \in \mathbf{v}_g + \mathbf{H}^1_0(\Omega)$ where the trace of $\mathbf{v}_g$ yields the boundary condition $\mathbf{g} = \gamma(\mathbf{v}_g)$. We will also show some natural boundary conditions that appear within the weak formulation in the next section. For any details on existence and uniqueness of solutions to the weak form of Navier-Stokes equations, we refer to the literature like [62, 101, 144, 153]. However, we present some main results for the case of stationary and instationary Navier-Stokes equations and also for the convection-diffusion equation. The Laplace equation describing the potential field is the standard elliptic problem in partial differential equation – a quite complete introduction can be found in [34].

Solution of the Navier-Stokes Equations Within Solenoidal Spaces

The dimensionless stationary Navier-Stokes equations with homogeneous Dirichlet boundary conditions

$$\begin{aligned} Re(\mathbf{v}\cdot\nabla)\mathbf{v} - \Delta\mathbf{v} + \nabla p &= \mathbf{f} && \text{in } \Omega \\ \nabla\cdot\mathbf{v} &= 0 && \text{in } \Omega \\ \mathbf{v} &= 0 && \text{on } \partial\Omega \end{aligned}$$

can (alternatively to (2.20)) be written in the weak formulation: for given $\mathbf{f} \in V^*$ seek $\mathbf{v} \in V$ such that

$$\frac{1}{Re} a(\mathbf{v},\varphi) + c(\mathbf{v},\mathbf{v},\varphi) = \langle \mathbf{f},\varphi\rangle \qquad \forall \varphi \in V. \tag{2.21}$$

This notation uses the function space

$$V = \{\mathbf{v} \in \mathbf{H}^1_0(\Omega) : \; b(\mathbf{v},q) = 0, \; \forall q \in L^2_0(\Omega)\}$$

of solenoidal vector fields and its dual space V^* with the duality pairing $\langle \mathbf{f},\mathbf{u}\rangle = \int_\Omega \mathbf{f}\cdot\mathbf{u}\, d\mathbf{x}$. Thus, neither the pressure p appears in the formulation nor the continuity equation – both are expressed by the choice of function space V. The used formulation (2.21) in the solenoidal space V can directly be deduced from the above definition, i.e. if $[\mathbf{v},p] \in \mathbf{H}^1_0(\Omega) \times L^2_0(\Omega)$ fulfils (2.20) then $\mathbf{v}$ also solves (2.21). By this reformulation elegant statements on existence and uniqueness can be made (see [35] or [126] for proof), but it remains to answer the question whether also the opposite direction holds true.

Theorem 2.2.2
For every $\mathbf{f} \in V^*$ and $d \le 3$ there exists a solution $\mathbf{v} \in V$ of the stationary Navier-Stokes problem (2.21). If $Re^2\beta\|\mathbf{f}\|_{V^*} < 1$ this solution $\mathbf{v} \in V$ is unique – here $\beta > 0$ is given by the continuity of the trilinear form (2.19), i.e. it is the smallest constant in

$$c(\mathbf{w},\mathbf{u},\mathbf{v}) \le \beta\|\mathbf{w}\|_1\|\mathbf{u}\|_1\|\mathbf{v}\|_1, \qquad \forall \mathbf{w},\mathbf{u},\mathbf{v} \in V.$$

For the instationary problem one usually defines the additional function space

$$H = \{\mathbf{v} \in \mathbf{L}^2(\Omega) : \; \nabla\cdot\mathbf{v} = 0, \; \mathbf{v}\cdot\mathbf{n}|_{\partial\Omega} = 0\}$$

which together with the solenoidal space V forms the Gelfand triple with compact embeddings (see [152] for details)

$$V \hookrightarrow H \cong H^* \hookrightarrow V^*.$$

The time-derivative has to be interpreted in a weak sense $\partial_t\mathbf{v}(t) \in V^*$ such that the problem reads: for given $\mathbf{v}_0 \in H$ and $\mathbf{f} \in L^2(0,T;V^*)$ seek $\mathbf{v} \in L^2(0,T;V)$ such that a.e. in $(0,T)$

$$\langle \partial_t\mathbf{v}(t),\varphi\rangle + \frac{1}{Re} a(\mathbf{v}(t),\varphi) + c(\mathbf{v}(t),\mathbf{v}(t),\varphi) = \langle \mathbf{f},\varphi\rangle \qquad \forall\varphi \in V \tag{2.22}$$

and $\mathbf{v}(0) = \mathbf{v}_0$. Existence for the instationary equations can be shown for $\mathbf{v} \in L^2(0,T;V) \cap L^\infty(0,T;H)$ but uniqueness is only given for $d = 2$ (see [61] or [153] for proof).

General Weak Solution of the Navier-Stokes Equations

By restating the entire saddle point structure of (2.20) as a single equation (2.21) or (2.22), we found existence and uniqueness results for the velocity field $\mathbf{v}$ in solenoidal spaces. The way back should then ideally yield a solution to the weak formulation (2.20) that contains also the pressure p. To answer the question whether there exists a unique $p \in L^2_0(\Omega)$ satisfying in the stationary case

$$(p, \nabla \cdot \varphi) = \langle \mathbf{f}, \varphi \rangle - a(\mathbf{v}, \varphi) - Re \cdot c(\mathbf{v}, \mathbf{v}, \varphi) \qquad \forall \varphi \in \mathbf{H}^1_0(\Omega)$$

we introduce a general framework for abstract variational problems (taken from [62]), that will also be used later on for the finite element discretisation. Let X, M be Hilbert spaces and X^*, M^* their dual spaces. With the continuous bilinear forms

$$a(\cdot,\cdot) : X \times X \ \to \mathbb{R}, \ \ b(\cdot,\cdot) : X \times M \ \to \mathbb{R}$$

we define the abstract saddle point problem: for given $l \in X^*$ seek $[u, \lambda] \in X \times M$ such that

$$\begin{aligned} a(u, v) + b(v, \lambda) &= \langle l, v \rangle \qquad && \forall v \in X, \\ b(u, \mu) &= 0 && \forall \mu \in M. \end{aligned} \tag{2.23}$$

Furthermore, let $V = \{v \in X : \ b(v, \mu) = 0, \ \forall \mu \in M\}$ so that the above problem can also be stated as: seek $u \in V$ such that

$$a(u, v) = \langle l, v \rangle \qquad \forall v \in V.$$

Theorem 2.2.3
Assume that the bilinear form $a(\cdot,\cdot)$ is V-elliptic, i.e. there exists a constant $\alpha > 0$ such that

$$a(v, v) \geq \alpha \|v\|^2_X \qquad \forall v \in V.$$

Then the abstract problem (2.23) possess a unique solution if and only if the bilinear form $b(\cdot,\cdot)$ satisfies the inf-sup condition

$$\inf_{\mu \in M} \sup_{v \in X} \frac{b(v, \mu)}{\|v\|_X \|\mu\|_M} \geq \beta > 0. \tag{2.24}$$

In the framework of weak Navier-Stokes equations the spaces $X = \mathbf{H}^1_0(\Omega)$, $M = L^2_0(\Omega)$ fit into this abstract setting. The ellipticity or more general the solvability of $\mathbf{v} \in V$ is given by the results above, so that it remains to prove the existence of pressure p and the inf-sup condition to ensure that a solution to (2.21) resp. (2.22) also induces a solution to (2.20). Both are given by the following lemma taken from [50] and we can conclude that the formulations within solenoidal spaces and originally chosen Sobolev spaces are equivalent.

Lemma 2.2.1
Let $l \in \mathbf{H}^{-1}(\Omega)$ such that $\langle l, \mathbf{v} \rangle = 0 \ \forall \mathbf{v} \in \{\mathbf{u} \in \mathbf{C}^\infty_0(\Omega) : \ \nabla \cdot \mathbf{u} = 0\}$. Then there exists a function $p \in L^2(\Omega)$ such that

$$\langle l, \mathbf{v} \rangle = (p, \nabla \cdot \mathbf{v}), \qquad \forall \mathbf{v} \in \mathbf{H}^1_0(\Omega).$$

For each $p \in L^2_0(\Omega)$ there exists a unique function $\mathbf{v} \in \{\mathbf{u} \in \mathbf{H}^1_0(\Omega) : \ a(\mathbf{u}, \mathbf{v}) = 0 \ \forall \mathbf{v} \in V\}$ such that

$$\nabla \cdot \mathbf{v} = p, \ \ |\mathbf{v}|_1 \leq c\|p\|$$

with a constant $c > 0$ independent of p. Therefore

$$\inf_{p \in L^2_0(\Omega)} \sup_{\mathbf{v} \in \mathbf{H}^1_0(\Omega)} \frac{(p, \nabla \cdot \mathbf{v})}{|\mathbf{v}|_1 \|p\|} = \inf_{p \in L^2_0(\Omega)} \sup_{\mathbf{v} \in \mathbf{H}^1_0(\Omega)} \frac{\|p\|}{|\mathbf{v}|_1} \geq \frac{1}{c} > 0.$$

Convection-Diffusion Equation

The convection-diffusion equation describing the transport of concentration within the microchannel was given by

$$\partial_t c - D\Delta c + \mathbf{v}\cdot\nabla c = 0.$$

We can express this equation in weak form as: for given $c_0 \in H^1(\Omega)$ seek $c \in c_{in} + H^1(\Omega)$ such that a.e. in $(0,T)$

$$\langle \partial_t c(t), \psi\rangle + D\ a(c,\psi) + (\mathbf{v}\cdot\nabla c, \psi) = 0 \qquad \forall \psi \in \{u \in H^1(\Omega) : u|_{\Gamma_{in}} = 0\}$$

and $c(0) = c_0$. Existence and uniqueness results are assured by Lax-Milgram lemma using the continuity and coercivity of the bilinear form associated with the spatial operator (see chapter 12 in [126]). The crucial point within this equation is hence not the solution itself but the numerical behaviour. It is a well-known fact that solutions on coarse meshes tend to oscillatory behaviour. This property is usually described by the (mesh) Peclet number

$$Pe = \frac{v_0 h}{D}$$

which indicates convection dominated flows whenever $Pe > 1$. To suppress numerical oscillations one might certainly use fine meshes (adjust mesh size h) or use stabilisation methods (cf. [126, 131]) like upwind methods or Streamline-Upwind Petrov-Galerkin finite element method (SUPG).

2.2.3 Boundary Conditions

The common analysis of Navier-Stokes equations is given only for homogeneous Dirichlet boundary conditions as shown before. An extension to nonhomogeneous boundary conditions can be found for example in [50] or [62], showing that for some compatability conditions on the boundary data $\mathbf{g}$ a solution $[\mathbf{v}, p]$ exists.

For our purpose it will be of major importance to have additional boundary conditions that are kind of physically motivated. Recall that we subdivided the boundary of the fluid domain Ω into disjoint parts $\partial\Omega = \Gamma_0 \cup \Gamma_{in} \cup \Gamma_{out}$. First we present the used boundary conditions for pure pressure-driven flow, i.e. without external applied potential field. Afterwards, we will take the additional boundary condition $\mathbf{v}|_{\Gamma_0} = -\Pi_2\nabla\phi$ into account to check whether this condition, that arises from the physical model, fits into the weak formulation of Navier-Stokes equations.

Suppose we first have only a no-slip boundary condition to model the solid walls of a channel, furthermore we might prescribe an inflow velocity profile. These Dirichlet boundary conditions fit into the framework of variational formulation with test functions $\varphi \in \mathbf{H}^1_0(\Omega)$ and the solution $\mathbf{v} \in \mathbf{H}^1_g(\Omega)$. To impose a boundary condition on the outflow region Γ_{out}, we choose

$$\varphi \in \mathbf{H}^1_{\Gamma_{in}\cup\Gamma_0}(\Omega) = \{\mathbf{u} \in \mathbf{H}^1(\Omega) :\ \mathbf{u}|_{\Gamma_{in}\cup\Gamma_0} = 0\}$$

so that the weak formulation of stationary Navier-Stokes equations is: seek $\mathbf{v} \in \mathbf{v}_g + \mathbf{H}^1_{\Gamma_{in}\cup\Gamma_0}(\Omega)$, $p \in L^2(\Omega)$ such that

$$\begin{aligned} a(\mathbf{v},\varphi) + Re\cdot c(\mathbf{v},\mathbf{v},\varphi) + b(\varphi,p) - \int_{\Gamma_{out}} (\partial_n\mathbf{v} - p\mathbf{n})\varphi\ ds &= 0 \qquad \forall\varphi \in \mathbf{H}^1_{\Gamma_{in}\cup\Gamma_0}(\Omega),\\ b(\mathbf{v},\xi) &= 0 \qquad \forall\xi \in L^2(\Omega). \end{aligned}$$

Since the test functions φ do not vanish on Γ_{out} we get the additional term

$$\int_{\Gamma_{out}} (\partial_n\mathbf{v} - p\mathbf{n})\varphi\ ds = 0 \qquad \forall\varphi \in \mathbf{H}^1_{\Gamma_{in}\cup\Gamma_0}(\Omega)$$

which relates the pressure to the velocity field. For channels with unknown outflow condition one often uses the additional boundary conditions (do-nothing condition)

$$\partial_n \mathbf{v} = p\mathbf{n} \qquad \text{on } \Gamma_{out}. \tag{2.25}$$

This Neumann type boundary condition is an unavoidable consequence of the fact that we work with weak formulation of the Navier-Stokes equations and also determines the pressure within the Navier-Stokes equations (initially the pressure only appears as gradient and is therefore only given up to a constant). The physical meaning of this boundary condition is yet not clear (see [18] and [67]) and gives correct numerical results only for simple Poiseuille flows in straight channels. In the following we will refer to the combination of prescribed Dirichlet values on Γ_{in} in combination with do-nothing condition in Γ_{out} as the *do-nothing formulation.*

A more physical approach to boundary conditions is given in [84] where a prescribed pressure difference between inflow and outflow boundary is required, i.e.

$$\begin{aligned} Re(\mathbf{v}\cdot\nabla)\mathbf{v} - \Delta\mathbf{v} + \nabla p &= \mathbf{f} && \text{in } \Omega \\ \nabla\cdot\mathbf{v} &= 0 && \text{in } \Omega \\ \mathbf{v} &= 0 && \text{on } \Gamma_0 \\ p &= P_{in}(t) && \text{on } \Gamma_{in} \\ p &= P_{out}(t) && \text{on } \Gamma_{out} \end{aligned}$$

The experimental setup as presented in Chapter 1 also claims the pressure-driven flow within the microchannel by a height difference between inflow and outflow region. We therefore conclude the weak formulation: seek $\mathbf{v} \in \mathbf{H}^1_{\Gamma_0}(\Omega)$, $p \in L^2(\Omega)$ such that $(i = in, out)$

$$\begin{aligned} a(\mathbf{v},\varphi) + Re\cdot c(\mathbf{v},\mathbf{v},\varphi) + b(\varphi,p) - \sum_i \int\limits_{\Gamma_i} (\partial_n\mathbf{v} - P_i\mathbf{n})\varphi\, ds &= 0 \qquad \forall\varphi \in \mathbf{H}^1_{\Gamma_0}(\Omega), \\ b(\mathbf{v},\xi) &= 0 \qquad \forall\xi \in L^2(\Omega). \end{aligned}$$

If additionally the inflow and outflow regions are perpendicular to the channel, it can be shown [84] that the normal derivative of $\mathbf{v}$ vanishes. This means that the remaining term

$$\sum_i \int\limits_{\Gamma_i} P_i\mathbf{n}\varphi\, ds$$

can be interpreted as an external force within the momentum equation driving the channel flow. In the following we will refer to this second setting of boundary conditions as the *pressure-drop formulation.* For simplicity we use $P_{out}(t) = 0$ and force the fluid flow by setting $P_{in}(t) = \Delta p$.

Even if specification of pressure difference Δp seems more physical, the problem of determining the correct value remains. To ensure the correct Reynolds number, we proceed as follows: since the mean inflow velocity

$$v_0 = \frac{1}{|\Gamma_{in}|} \int\limits_{\Gamma_{in}} v\, ds$$

was used for non-dimensionalisation of the Navier-Stokes equations, we compute this value numerically and fit the pressure-drop boundary such that the mean inflow velocity within the numerical simulation equals one. A rough estimation of this value might be given by fundamental laws of laminar flows within cylindrical pipes [55] of length L

$$\Delta p = \lambda_{lam} \frac{L\rho v_0^2}{2d_0}$$

where the parameter λ_{lam} depends on the Reynolds number and is given by experiments in form of so called Nikuradse graph (cf. [141]). For laminar flows one uses $\lambda_{lam} = 64/Re$.

We end the section on weak formulation with some remarks on the treatment of boundary conditions for the bulk velocity within electrokinetic fluid flow setting. The physical model requires a slip condition on the interface between bulk flow and flow in the electric double layer, i.e.

$$\mathbf{v}|_{\Gamma_0} = -\Pi_2 \nabla\phi.$$

First, we mentioned that this condition naturally implies the usage of pressure drop formulation for Γ_{in} and Γ_{out}, since a prescribed velocity profile $\mathbf{v}_g$ would also depend on $\nabla\phi$ and hence cannot be stated by arguments of Poiseuille like flows. Second we have to find an interpretation of this slip boundary condition in the framework of weak formulation. For a general treatment the constant $-\Pi_2$ is of minor importance so that we only consider the condition $\mathbf{v}|_{\Gamma_0} = \nabla\phi$. A weak formulation for the potential field ϕ according to the previous definitions is: seek $\phi \in \phi_{out} + H^1_{\Gamma_{in}\cup\Gamma_{out}}(\Omega)$ such that

$$(\nabla\phi, \nabla\chi) = 0 \qquad \forall\chi \in H^1_{\Gamma_{in}\cup\Gamma_{out}}(\Omega).$$

Therefore $\nabla\phi$ is only defined in the sense of distributions, whereas in the physical model we would have to require that $\phi \in C^1(\overline{\Omega})$. So the boundary condition for bulk velocity $\mathbf{v}|_{\Gamma_0} = \nabla\phi$ has to be redefined for the weak formulation. Since $\nabla\phi$ is a distribution, we propose to define the boundary condition as the limit of a convolution (Faltung) with a Dirac-sequence η_ε

$$\nabla\phi(\mathbf{x}) = \lim_{\varepsilon\to 0}(\eta_\varepsilon * \nabla\phi)(\mathbf{x}) = \lim_{\varepsilon\to 0}\int_\Omega \eta_\varepsilon(\mathbf{x}-\mathbf{y})\nabla\phi(\mathbf{y})\ d\mathbf{y}.$$

The functions $\eta_\varepsilon \in L^1_{loc}(\Omega)$ have to fulfil

i/ $\eta_\varepsilon \geq 0,\ \forall\varepsilon > 0,$

ii/ $\int_{\mathbb{R}^d} \eta_\varepsilon(\mathbf{y})\ d\mathbf{y} = 1,\ \forall\varepsilon > 0,$

iii/ $\forall r > 0: \int_{\mathbb{R}^d\setminus B_r(0)} \eta_\varepsilon(\mathbf{y})\ d\mathbf{y} \to 0$ for $\varepsilon \to 0.$

For functional analytical aspects we refer to [163] and only want to present a way to construct the functions η_ε that takes the finite element based discretisation into account. In so doing we have to forestall some notations of finite element discretisation that will be introduced in the next chapter.

To evaluate the Dirichlet boundary value $\mathbf{v}(\mathbf{x}) = \nabla\phi(\mathbf{x}),\ \mathbf{x} \in \Gamma_0$, define the Dirac-sequence η_ε as the Lagrangian basis function at $\mathbf{x}$ on the actual mesh, i.e. $\eta_{x,h} \in C_0^\infty(\Omega)$. Identifying ε with the mesh size h we have to show for $\eta_{x,h}$ the three conditions above. Obviously, $\eta_{x,h} \geq 0$ and since the support of $\eta_{x,h}$ is only given by neighbouring cells of $\mathbf{x}$, conditions i/ and iii/ are fulfilled. It remains to scale $\eta_{x,h}$ by a factor c such that

$$\int_{supp(\eta_{x,h})} c\eta_{x,h}(\mathbf{y})\ d\mathbf{y} = 1.$$

Let K_i denote the cells such that $supp(\eta_\varepsilon) = \bigcup_i K_i$. Then we can determine the factor

$$c = \left[\sum_i \int_{K_i} \eta_{x,h}(\mathbf{y})\ d\mathbf{y}\right]^{-1} = \left[\sum_i \int_{\hat{K}} \eta_{x,h}(F_i(\xi))\det(DF_i)\ d\xi\right]^{-1} = \left[\sum_i |K_i| \int_{\hat{K}} \eta_{x,h}(F_i(\xi))\ d\xi\right]^{-1}$$

where we used the transformation $F_i : \hat{K} \to K_i$ from reference cell to cell K_i and $\det(DF_i) = |K_i|$ (cf. [34, 120]).

We conclude that evaluation of Dirichlet boundary value $\nabla\phi(\mathbf{x})$ on actual mesh with mesh size h is given by

$$\nabla\phi(\mathbf{x}) = \left[\sum_i |K_i| \int_{\hat{K}} \eta_{x,h}(F_i(\xi))\ d\xi\right]^{-1} \sum_i \int_{K_i} \eta_{x,h}(\mathbf{y})\nabla\phi(\mathbf{y})\ d\mathbf{y}.$$

Finally, the integrals over K_i, $\hat{K}$ will be evaluated numerically by virtue of the finite element approximation ϕ_h. For the special quadrature rule taking only the 2^d vertices of cells K_i, $\hat{K}$ as quadrature-points, we have $\eta_{x,h}$ vanishing at all vertices except $\mathbf{x}$, so that

$$\int_{K_i} \eta_{x,h}(\mathbf{y}) \nabla\phi(\mathbf{y}) \; d\mathbf{y} \approx \frac{|K_i|}{2^d} \, \nabla\phi_h|_{K_i}(\mathbf{x}),$$

$$\int_{\hat{K}} \eta_{x,h}(F_i(\xi)) \; d\xi \approx \frac{|\hat{K}|}{2^d} = \frac{1}{2^d}.$$

All in all we get the approximation of $\nabla\phi(\mathbf{x})$ to be

$$\nabla\phi(\mathbf{x}) = \frac{\sum\limits_i |K_i| \; \nabla\phi_h|_{K_i}(\mathbf{x})}{\sum\limits_i |K_i|}$$

showing that for $\phi_h \in C^1(\overline{\Omega})$ we have $\nabla\phi(\mathbf{x}) = \nabla\phi_h(\mathbf{x})$ and for $\phi_h \in C^0(\overline{\Omega})$ that the local gradients have to be weighted by size of the cells.

2.3 Optimisation of Fluid Flow

The field of optimisation or optimal control of fluid flow problems (or more general of systems governed by partial differential equations) is recently a very active one. Contributions to the field of optimisation and optimal control of Navier-Stokes equations are manifold, whereas most consider the two dimensional case, see e.g. [85, 100, 145]. A fairly complete overview and introduction is given in the book of Gunzburger [73]. We will take a look on different approaches to optimise a system governed by partial differential equation, give a brief summary on theory of existence and uniqueness of solutions to the optimisation problem and in the end formulate the entire optimisation problems and approaches used within this thesis.

2.3.1 Optimisation with Partial Differential Equations

One of the earliest references for the analytical framework of optimal control problems for partial differential equations is the work of Lions [105]. Some general conditions for optimal solutions can also be derived from programming principles in Banach spaces which were under investigation in [111, 156] and references therein.

Our intention is now to introduce the reader to a general setting of optimisation with partial differential equation and to elaborate the concrete approach in the next subsection. A starting point is to identify the common ingredients of such optimisation problems, namely

- an *objective* that has to be optimised – this is almost always formulated as a cost functional,
- one or more *control or design parameter* that can be modified to achieve an optimisation,
- *constraints* which have to be fulfilled within the optimisation problem.

Let us denote the set of partial differential equations that have to be fulfilled by

$$A(y,u) = 0 \qquad \text{in } \Omega \tag{2.26}$$

where we assume the state variable $y \in Y$ and the control variable $u \in U$ within suitable Banach spaces Y and U. For optimal control problems, the operator $A(y,u)$ is typically given by

$$A(y,u) = \overline{A}(y,u) + C(u)$$

with a (nonlinear) operator $\overline{A}$ and a control operator C. Furthermore, the partial differential equation might be instationary, i.e.

$$\begin{aligned} \partial_t y + A(y,u) &= 0 \qquad && \text{in } \Omega \times (0,T), \\ y(\cdot,0) &= y_0 && \text{in } \Omega. \end{aligned} \tag{2.27}$$

Hence, we already have defined the control parameter and some constraints in form of the underlying partial differential equation which limits the admissible state space Y. Additional side constraints might be given for the control variable u exclusively, like bounds

$$\|u\|_U \leq \kappa.$$

In the following let us assume that the admissible spaces for state and control variables are given by Y_{ad} resp. U_{ad}. What remains is to define an objective or cost functional $J(y,u)$. Usually such a functional can be split into parts involving only the state and only the control variables

$$J(y,u) = J_1(y) + J_2(u).$$

For instationary problems a temporal averaging is common, i.e. if state and control are time dependent

$$J(y,u) = \int\limits_0^T J_1(y)\; dt + J_3(y(T)) + \int\limits_0^T J_2(u)\; dt.$$

All in all the entire optimisation problem reads:

$$\min_{(y,u)\in Y_{ad}\times U_{ad}} J(y,u) \text{ such that equation (2.26) (resp. (2.27)) is fulfilled.} \tag{2.28}$$

A crucial point within the solution process is the definition of so-called *control-to-state* mapping.

Definition 2.3.1
Consider the system of partial differential equations (2.26) resp. (2.27). The mapping $u \mapsto y$ where $y \in Y_{ad}$ is a solution of (2.26) resp. (2.27) with the control parameter $u \in U_{ad}$, is denoted by S, i.e. $y = S(u)$.

Any further statements on existence and uniqueness of a solution to the optimisation problem will strongly depend on this mapping S – we refer e.g. to [105, 59, 157] for a more detailed analysis. For sake of simplicity we assume that this mapping is well-posed, although this is not a trivial task for all problems stated before (if at all possible).

Definition 2.3.2
A control $\bar{u} \in U_{ad}$ is called optimal control and $\bar{y} = S(\bar{u}) \in Y_{ad}$ is called corresponding optimal state if

$$J(\bar{y},\bar{u}) \leq J(y,u) \qquad \forall u \in U_{ad} \text{ and } y = S(u) \in Y.$$

The existence result for the general optimisation problem in Banach spaces can be found in [88] – here we only cite the principal assumptions on which the proof bases. Let $J : Y \times U \to \mathbb{R}$ and $A : Y \times U \to Z$ be continuous, Y, U and Z Banach spaces where Y and U are reflexive. Then problem (2.28) has an optimal solution $\bar{u}$ with optimal state $\bar{y}$ under the assumptions:

- U_{ad} is convex, bounded and closed,
- Y_{ad} is convex, closed and (2.28) has a feasible point, i.e.

$$\{(y,u) \in Y_{ad} \times U_{ad} : \text{ equation (2.26) (resp. (2.27)) is fulfilled}\} \neq \emptyset,$$

- the control-to-state operator $S : U_{ad} \to Y$ is continuous and bounded,
- $(y,u) \in Y \times U \mapsto A(y,u) \in Z$ is continuous under weak convergence,

- J is weakly lower semicontinuous.

For our purpose, we recall the main setting of optimisation problems under instationary partial differential equations, that is based on [115]. To this we use the weak formulation of Navier-Stokes equations or more general instationary problems, i.e. for given control $u \in U$ seek $y \in Y$ such that for almost all $t \in (0,T)$ it holds

$$\langle \partial_t y, \varphi \rangle_{Y^*,Y} + \langle A(y,u), \varphi \rangle_{Y^*,Y} = 0 \qquad \forall \varphi \in Y$$

and $y(\cdot,0) = y_0$. The operator $A : Y \times U \rightarrow Z = Y^*$ is in general defined for suitable Hilbert spaces Y for the state and U for the control. For the special case of Navier-Stokes equations this operator is given by the spatial differential operators defining the linear forms (2.17), (2.18) and (2.19). Moreover, let H be a Hilbert space which builds together with Y a Gelfand triple $Y \hookrightarrow H \cong H^* \hookrightarrow Y^*$. A typical choice for these spaces is as before

$$Y = \{v \in H^1(\Omega) : \ v|_{\Gamma_D} = 0\} \text{ and } H = L^2(\Omega)$$

where Γ_D denotes the part of the boundary of Ω with prescribed Dirichlet boundary conditions. For a time interval $(0,T)$ we introduce the Hilbert space $W(0,T)$ defined as

$$W(0,T) = \{v : \ v \in L^2(0,T;Y) \text{ and } \partial_t v \in L^2(0,T;Y^*)\}.$$

Furthermore, we use the inner product of $L^2(0,T;H)$ given by

$$(u,v) = (u,v)_{L^2(0,T;H)} = \int_0^T (u(t),v(t))_H \, dt,$$

such that the weak formulation of state equation can be expressed as: for given control $u \in L^2(0,T;U)$ seek $y \in W(0,T)$ such that

$$(\partial_t y, \varphi) + \int_0^T \langle A(y(t),u(t)), \varphi(t) \rangle_{Y^*,Y} \, dt = 0 \qquad \forall \varphi \in W(0,T) \tag{2.29}$$

and $y(0) = y_0$. By these definition the setting of weak instationary Navier-Stokes equation fits into the general optimisation problem above, i.e. $y \in W(0,t)$, $u \in L^2(0,T;U)$. The concrete formulation of cost functional will be introduced later.

Coming back to the general setting, we assume that via the control-to-state operator $S : U \rightarrow Y$ we have a unique solution y to each control u, so that we can define the reduced cost functional $j : U \rightarrow \mathbb{R}$ by $j(u) = J(S(u),u)$. Hence, problem (2.28) can be stated as an unconstrained (at least only constrained by $u \in U_{ad}$) optimisation problem

$$\min_{u \in U_{ad}} j(u). \tag{2.30}$$

Let $U_{ad} \subset U$ be non-empty and convex, $J : Y \times U \rightarrow \mathbb{R}$ and $A : Y \times U \rightarrow Z$ be continuously differentiable. If additionally for all $u \in U_{ad}$ the state equation possesses a unique solution $y = S(u) \in Y$ and $\frac{\partial A}{\partial y}(S(u),u)$ has a bounded inverse, then $\bar{u}$ being a solution of reduced optimisation problem (2.30) satisfies the first order necessary optimality condition (see [88])

$$\langle \frac{Dj}{Du}(\bar{u}), u - \bar{u} \rangle_{U^*,U} \geq 0 \qquad \forall u \in U_{ad}. \tag{2.31}$$

For the special case $U = L^2(\Omega)$ or $U = L^2(\Gamma)$, $\Gamma \subset \partial\Omega$, and $U_{ad} = \{u \in U : a \leq u \leq b\}$, i.e. so-called box constraints, we can identify the gradient $\nabla j(u) = \frac{Dj}{Du}(u)$ by means of the Riesz representation. Then one gets the well-known equivalent representations of (2.31)

i/ $\bar{u} \in U_{ad}$, $(\nabla j(\bar{u}), u - \bar{u})_0 \geq 0$, $\forall u \in U_{ad}$,

$$\text{ii/ } \bar{u} \in U_{ad},\ \nabla j(\bar{u})(\mathbf{x}) \begin{cases} = 0 & \text{if } a(\mathbf{x}) < \bar{u}(\mathbf{x}) < b(\mathbf{x}), \\ \geq 0 & \text{if } a(\mathbf{x}) = \bar{u}(\mathbf{x}) < b(\mathbf{x}), \\ \leq 0 & \text{if } a(\mathbf{x}) < \bar{u}(\mathbf{x}) = b(\mathbf{x}), \end{cases}$$

iii/ for any $\varepsilon > 0$: $\bar{u} = P_{U_{ad}}(\bar{u} - \varepsilon \nabla j(\bar{u}))$ with $P_{U_{ad}}(u) = \min(\max(a, u), b)$.

A further simplification can be derived whenever the bounds equal $a = b = \infty$ such that $U_{ad} = U$ and the first order optimality condition is hence given by

$$\nabla j(u) = 0, \qquad \forall u \in U.$$

In order to solve the optimisation problem (2.30) we thus need a representation to evaluate at least the first derivative $\frac{Dj}{Du}$. We will present (following [88]) two approaches that are used in the sequel, namely the adjoint-based and sensitivity-based approach. The second derivative of $j(u)$ might also be used for higher order methods (see [28] or [86]), but we will not concern this point and use only Quasi-Newton methods like LBFGS method – a detailed overview on used optimisation algorithms is given in chapter 5.1. For the case of J being convex on U_{ad} the first order necessary condition (2.31) is also sufficient for global optimality.

The Adjoint Approach

We state the abstract optimisation problem (2.28) for the stationary case as[4]:

$$\min_{(y,u) \in Y_{ad} \times U_{ad}} J(y, u) \text{ such that } A(y, u) = 0 \text{ in } \Omega.$$

Assume that the cost functional $J : Y \times U \to \mathbb{R}$ and the state equation operator $A : Y \times U \to Z$ are continuously differentiable. Let

$$L(y, u, z) = J(y, u) + \langle z, A(y, u) \rangle_{Z^*,Z} \tag{2.32}$$

define the Lagrangian functional $L : Y \times U \times Z^* \to \mathbb{R}$ involving a Lagrange multiplier $z \in Z^*$. Then one gets the optimality system by the stationary points of L which are candidates for optimal solutions (for proof see e.g. [73, 157]).

Theorem 2.3.1
Assume that for given control $u \in U_{ad}$ the state $y \in Y_{ad}$ satisfies the state equation

$$\frac{\partial L}{\partial z}(y, u, z)\varphi = \langle \varphi, A(y, u) \rangle_{Z^*,Z} = 0 \qquad \forall \varphi \in Z^* \tag{2.33}$$

and that the adjoint state $z \in Z^*$ satisfies the adjoint equation

$$\frac{\partial L}{\partial y}(y, u, z)\varphi = \langle \frac{\partial J}{\partial y}(y, u), \varphi \rangle_{Y^*,Y} + \langle z, \frac{\partial A}{\partial y}(y, u)\varphi \rangle_{Z^*,Z} = 0 \qquad \forall \varphi \in Y \tag{2.34}$$

then the expression for the first derivative of (2.30) holds (gradient equation or optimality condition):

$$\begin{aligned} \langle \frac{Dj}{Du}(u), \varphi - u \rangle_{U^*,U} &= \langle \frac{\partial L}{\partial u}(y, u, z), \varphi - u \rangle_{U^*,U} \\ &= \langle \frac{\partial J}{\partial u}(y, u), \varphi - u \rangle_{U^*,U} + \langle z, \frac{\partial A}{\partial u}(y, u)\varphi - u \rangle_{Z^*,Z} \geq 0 \qquad \forall \varphi \in U_{ad}. \end{aligned} \tag{2.35}$$

Since $j(u) = L(S(u), u, z)$ for arbitrary $z \in Z^*$, a more convenient way to express the gradient

[4]for sake of simplicity we restrict ourselves to stationary problems and ask the reader to transfer all statements to the instationary case whenever needed

representation can be given by direct differentiation

$$\langle \frac{Dj}{Du}(u), \delta u\rangle_{U^*,U} = \langle \frac{\partial L}{\partial y}(S(u), u, z), \frac{DS}{Du}(u)\delta u\rangle_{Y^*,Y} + \langle \frac{\partial L}{\partial u}(S(u), u, z), \delta u\rangle_{U^*,U}.$$

Choosing $z = z(u)$ such that the adjoint equation (2.34)

$$\langle \frac{\partial J}{\partial y}(y, u), \varphi\rangle_{Y^*,Y} + \langle z, \frac{\partial A}{\partial y}(y, u)\varphi\rangle_{Z^*,Z} = \langle \frac{\partial J}{\partial y}(y, u), \varphi\rangle_{Y^*,Y} + \langle \left[\frac{\partial A}{\partial y}(y, u)\right]^* z, \varphi\rangle_{Y^*,Y} = 0,\ \forall \varphi \in Y$$

is fulfilled (where $[\cdot]^*$ denotes the adjoint operator), we get

$$\langle \frac{Dj}{Du}(u), \delta u\rangle_{U^*,U} = \langle \frac{\partial L}{\partial u}(S(u), u, z(u)), \delta u\rangle_{U^*,U} \qquad \forall \delta u \in U_{ad}.$$

Thus, by virtue of (2.35)

$$\frac{Dj}{Du}(u) = \frac{\partial L}{\partial u}(S(u), u, z(u)) = \frac{\partial J}{\partial u}(S(u), u) + \left[\frac{\partial A}{\partial u}(S(u), u)\right]^* z(u). \tag{2.36}$$

The Sensitivity Approach

Instead of using the adjoint approach to describe the gradient of reduced cost functional $j(u)$ one might also use the chain rule to get the sensitivity

$$\frac{Dj}{Du}(u)d = \frac{\partial J}{\partial y}(S(u), u)\frac{DS}{Du}(u)d + \frac{\partial J}{\partial u}(S(u), u)d.$$

The partial derivatives $\frac{\partial J}{\partial y}$, $\frac{\partial J}{\partial u}$ are usually not that hard to derive analytically, whereas the sensitivity/directional derivative $\frac{DS}{Du}(u)d$ is. An easy (but costly) approach would be a finite difference quotient

$$\frac{DS}{Du}(u) \approx \frac{S(u) - S(\tilde{u})}{u - \tilde{u}}.$$

Alternatively we can also differentiate the state equation $A(S(u), u) = 0$ in direction d to get the *sensitivity equation*

$$\frac{\partial A}{\partial y}(S(u), u)\frac{DS}{Du}(u)d = -\frac{\partial A}{\partial u}(S(u), u)d \tag{2.37}$$

such that the directional derivative $\frac{Dj}{Du}(u)d$ requires the solution of system (2.37) for each direction.

Adjoint Approach vs. Sensitivity Approach

Beside the two approaches (adjoint and sensitivity), which both aim at computation of derivative operator $\frac{Dj}{Du}(u)$, one might also solve the optimality system consisting of the coupled equations (2.33), (2.34) and (2.35) at once. This so called *one-shot approach* is usually not feasible due to the close coupling of equations and (after discretisation) due to the resulting huge nonlinear systems. Furthermore, for instationary problems the adjoint equation is stated *backward-in-time* leading to additional complications when standard timestepping discretisation will be used. So we restrict to the two presented approaches.

For the adjoint approach computation of $\frac{Dj}{Du}(u)$ is given in three steps

1. solve state equation[5] to get state $y = S(u)$:

$$\langle A(y, u), \varphi\rangle_{Y^*,Y} = 0,\ \forall \varphi \in Y$$

[5]we used $Z = Y^*$ and assumed that the operator A defines a symmetric linear form

2. solve adjoint equation to get adjoint state $z(u)$:

$$\langle \left[\frac{\partial A}{\partial y}(S(u),u)\right]^* z(u), \varphi\rangle_{Y^*,Y} = -\langle \frac{\partial J}{\partial y}(S(u),u), \varphi\rangle_{Y^*,Y}, \ \forall \varphi \in Y$$

3. determine $\frac{Dj}{Du}(u) = \frac{\partial J}{\partial u}(S(u),u) + \left[\frac{\partial A}{\partial u}(S(u),u)\right]^* z(u)$

to get the whole derivative $\frac{Dj}{Du}(u)$ independently of the dimension of U. On the other side the sensitivity approach only yields directional derivatives $\frac{Dj}{Du}(u)d$ in direction d such that for B being a basis of U we need to compute $\frac{Dj}{Du}(u)b, \ b \in B$. Each computation requires the steps

1. solve state equation to get state $y = S(u)$: $A(y,u) = 0$
2. solve sensitivity equation to get sensitivity $y_b = \frac{DS}{Du}(u)b$:

$$\frac{\partial A}{\partial y}(S(u),u)y_b = -\frac{\partial A}{\partial u}(S(u),u)b$$

3. determine $\frac{Dj}{Du}(u)b = \frac{\partial J}{\partial y}(S(u),u)y_b + \frac{\partial J}{\partial u}(S(u),u)b$.

This summary shows that at a first glance one should obviously use the adjoint approach to get the gradient of the reduced cost functional. Independently of the number of design parameters there is always only one (linear) adjoint system to be solved to get the gradient of the reduced cost functional, whereas the number of (linear) sensitivity systems grow linearly with the number of design parameter. But there are still some drawbacks: while the sensitivities can simply be determined by solving the linearised state equation (a step that is common within each nonlinear partial differential equation solver), the adjoint equations have to be derived analytically first. Moreover, for instationary problems the adjoint equation needs to be solved backward in time, whereas the sensitivities can be computed within each timestep of the state equation solver. A last remark on the sensitivity approach might be given by the physics of the system. In many applications the possible control u within control space U_{ad} can be expressed in the form

$$u(x,t) = \sum_{k=1}^{K} \alpha_k f_k(x,t)$$

with K being a small number. Thus the variation of control is per se limited so that sensitivity $\frac{Dj}{Du}(u)\alpha_k$ might directly be computed. In the end if the number of parameters α_k grows or if the control is even given as an infinite dimensional function, e.g. $u \in L^2(\partial\Omega)$, the adjoint based approach seems to be favourable.

2.3.2 Flow Control and Optimisation

The field of optimal control in fluid mechanics within computational science has become a large-scale research discipline starting in the 1990s with pioneering works like [1]. For an overview on functional analytic background on optimal control of the instationary Navier-Stokes equations by distributed control, we refer to [159] and references therein. A review and plenty references of the adjoint equation-based approach for flow control can be found in [71]. The use of second order methods and instantaneous control is shown in [87]. Both works give a complete background from the analytical point of view but the solved problems are related to two dimensions and no particular interest was put on the numerical requirements. The special case of Dirichlet boundary control for optimisation problems based on (stationary) Navier-Stokes equations can be found in [75].

As we presented previously we aim at using the adjoint approach as well as the sensitivity approach for optimisation. A comparison of the capabilities of both approaches will only be done by means of an academical example using the instationary Navier-Stokes equations – this should then also give hints how to tackle the considered optimisation problem of the electroosmotic micromixer. To this end let us fix the considered problem of optimising the fluid flow of backward facing step geometry first.

Pressure Driven Flow

We seek an optimal control to the instationary Navier-Stokes equations which minimises a cost functional of tracking type. For a prescribed solution $\mathbf{v}_d \in \mathbf{H}^1(\Omega)$, which might be given on the whole domain Ω or on parts of it, we want to minimise the distance between the solution $\mathbf{v}$ of Navier-Stokes equations and $\mathbf{v}_d$ – for instationary problems in the mean over time interval $(0, T)$. The observation volume $\Omega_s \subset \Omega$ might be restricted to the area of recirculation behind the step,

$$\min_{\mathbf{v},\mathbf{c}} J(\mathbf{v},\mathbf{c}) = \frac{\beta_1}{2} \int_0^T \|\mathbf{v} - \mathbf{v}_d\|^2_{L^2(\Omega_s)}\, dt + \frac{\beta_2}{2} \|\mathbf{v}(T) - \mathbf{v}_d\|^2_{L^2(\Omega_s)} + \frac{\lambda}{2} \int_0^T \|c\|^2_{L^2(\Gamma_\mathbf{c})}\, dt. \tag{2.38}$$

Instead of explicit box constraints for admissible control $\mathbf{c}$, we penalise the cost functional by an additional term involving the control c. Such an enhancement of stability of the optimisation problem by augmenting through *regularisation term* is commonly used within the literature. The velocity field $\mathbf{v}_d$ is given by the stationary solution of Stokes equations leaving all other data (Reynolds number and boundary values) within the state equation untouched. For other types of cost functionals in the framework of flow control we refer to [85, 100] and for results on existence of solutions see [75, 79, 77, 58].

For convenience of the reader we resume the setting of the problem:

- state equations are given with do-nothing boundary formulation as introduced before

$$\begin{aligned}
Re\,[\partial_t \mathbf{v} + (\mathbf{v}\cdot\nabla)\mathbf{v}] - \Delta\mathbf{v} + \nabla p &= 0 && \text{in } \Omega \times (0,T),\\
\nabla \cdot \mathbf{v} &= 0 && \text{in } \Omega \times (0,T),\\
\mathbf{v} &= 0 && \text{on } \Gamma_0 \times (0,T),\\
\mathbf{v} &= \mathbf{c} && \text{on } \Gamma_c \times (0,T),\\
\mathbf{v} &= \mathbf{v}_{in} && \text{on } \Gamma_{in} \times (0,T),\\
\partial_n \mathbf{v} - p\mathbf{n} &= 0 && \text{on } \Gamma_{out} \times (0,T),\\
\mathbf{v}(\cdot,0) &= \mathbf{v}_0 && \text{in } \Omega,
\end{aligned} \tag{2.39}$$

- initial condition $\mathbf{v}_0$ is a given function in $\mathbf{H}^1(\Omega)$ that matches the boundary conditions,
- desired state $\mathbf{v}_d$ is in $\mathbf{H}^1(\Omega)$,
- the regularisation parameter λ is a positive constant and the coefficients β_1, β_2 are non-negative constants of which at least one is positive.

The Dirichlet boundary conditions for $\mathbf{v}$ are extended by a boundary Γ_c on which the control $\mathbf{c}$ acts – a concrete definition of the space used for $\mathbf{c}$ will be given later. Up to now we only think of the control as an additional inflow/outflow where suction or injection of fluid through orifices can be applied.

To solve the optimisation problem (2.38) under constrains (2.39) by means of adjoint approach, we need to derive the adjoint equation and the gradient equation (first order optimality condition) to form the optimality system. First order necessary optimality conditions can be found in many references – the first proof of necessary conditions for the distributed optimal control problem related to the Navier-Stokes equations was given in the early work [1]. Other proofs can be found in [76, 77], [87] (where also some regularity problems are addressed) and [100]. Necessary optimality conditions for three-dimensional flow are established in [27]. A rigorous framework for boundary control of the Navier-Stokes equations can be found in [60, 86]. Thus, we skip the derivation and just present the optimality system in weak form. Define the space (with arbitrary $\Gamma \subset \partial\Omega$)

$$\mathbf{H}^1_\Gamma(\Omega) = \{\mathbf{u} \in \mathbf{H}^1(\Omega) : \ \mathbf{u} = 0 \text{ on } \Gamma\}$$

and let $\Gamma_d = \Gamma_0 \cup \Gamma_{in} \cup \Gamma_c$. To obtain a solution of the optimal control problem, we have to solve (using (2.17), (2.18), (2.19)):

- for $\mathbf{v} \in \mathbf{v}_{in} + L^2(0,T;\mathbf{H}^1_{\Gamma_0}(\Omega))$, $p \in L^2(0,T;L^2(\Omega))$ the Navier-Stokes equations with initial condition $\mathbf{v}(\cdot,0) = \mathbf{v}_0$:

$$\begin{aligned} Re\,[\langle \partial_t \mathbf{v}, \varphi\rangle + c(\mathbf{v},\mathbf{v},\varphi)] + a(\mathbf{v},\varphi) + b(\varphi,p) &= 0 \qquad \forall \varphi \in \mathbf{H}^1_{\Gamma_0 \cup \Gamma_{in}}(\Omega) \\ b(\mathbf{v},\xi) &= 0 \qquad \forall \xi \in L^2(\Omega) \\ \mathbf{v}|_{\Gamma_c} &= \mathbf{c} \end{aligned} \tag{2.40}$$

- for $\mathbf{z} \in L^2(0,T;\mathbf{H}^1_{\Gamma_d}(\Omega))$, $q \in L^2(0,T;L^2(\Omega))$ the adjoint equations with initial condition $\mathbf{z}(\cdot,T) = \beta_2(\mathbf{v}_d - \mathbf{v}(\cdot,T))$:

$$\begin{aligned} Re\,[-\langle \partial_t \mathbf{z}, \varphi\rangle + c(\varphi,\mathbf{v},\mathbf{z}) + c(\mathbf{v},\varphi,\mathbf{z})] + a(\mathbf{z},\varphi) + b(\varphi,q) &= (\beta_1(\mathbf{v}_d - \mathbf{v}),\varphi)_0 \qquad \forall \varphi \in \mathbf{H}^1_{\Gamma_d}(\Omega) \\ b(\mathbf{z},\xi) &= 0 \qquad \forall \xi \in L^2(\Omega) \end{aligned} \tag{2.41}$$

- for $\mathbf{c} \in L^2(0,T;\mathbf{L}^2(\Gamma_c))$ the gradient equation:

$$\int_0^T \int_{\Gamma_c} (\lambda \mathbf{c} + \partial_n \mathbf{z} - q\mathbf{n})\chi \; ds\; dt = 0 \quad \forall \chi \in L^2(\Gamma_c). \tag{2.42}$$

Just for completeness we denote the strong formulation of the adjoint equations (see [87] for a detailed derivation):

$$\begin{aligned} Re\left[-\partial_t \mathbf{z} - (\mathbf{v}\cdot\nabla)\mathbf{z} + (\nabla \mathbf{v})^T \cdot \mathbf{z}\right] - \Delta \mathbf{z} + \nabla q &= \beta_1(\mathbf{v}_d - \mathbf{v}) && \text{in } \Omega \times (0,T), \\ \nabla \cdot \mathbf{z} &= 0 && \text{in } \Omega \times (0,T), \\ \mathbf{z} &= 0 && \text{on } \Gamma_d \times (0,T), \\ \partial_n \mathbf{z} - q\mathbf{n} + (\mathbf{v}\cdot\mathbf{n})\mathbf{z} &= 0 && \text{on } \Gamma_{out} \times (0,T), \\ \mathbf{z}(\cdot,T) &= \beta_2(\mathbf{v}_d - \mathbf{v}(\cdot,T)) && \text{in } \Omega. \end{aligned} \tag{2.43}$$

The optimality condition can thus be stated as

$$\lambda \mathbf{c} - q\mathbf{n} + \partial_n \mathbf{z} = 0 \qquad \text{on } \Gamma_c \times (0,T). \tag{2.44}$$

We see the strongly coupled variables $\{\mathbf{v}, p, \mathbf{z}, q, \mathbf{c}\}$ within the system (2.40), (2.41) and (2.42) which is due to the nonlinear character of the Navier-Stokes equations and of the cost functional. Having the problem stated, we should state some remarks on the type of control. The Dirichlet boundary control $\mathbf{c}$ is formally defined as the trace of a function in $\mathbf{H}^1(\Omega)$ and thus $\mathbf{c} \in \mathbf{H}^{1/2}(\Gamma_c)$. Definition of the cost functional (2.38) should therefore incorporate this fact and use the $\mathbf{H}^{1/2}$-norm instead of $\mathbf{L}^2$-norm on Γ_c which means that additional space derivatives of $\mathbf{c}$ must be used. A more generic approach instead would be given by the cost functional

$$\min_{\mathbf{v},\mathbf{c}} J(\mathbf{v},c) = \tilde{J}(\mathbf{v}) + \frac{\lambda}{2}\int_0^T \int_{\Gamma_c} (|\mathbf{c}|^2 + \lambda_1 |\partial_x \mathbf{c}|^2 + \lambda_2 |\partial_t \mathbf{c}|^2)\; ds\; dt$$

leading to an optimality condition that is a boundary value problem in space-time for a partial differential equation along the control boundary Γ_c (cf. [58]). Nevertheless, also the weaker approach using the $L^2(\Gamma_c)$-norm leads at least numerically to reasonable results, if possibly the regularisation parameter λ is chosen large enough – see [87, 100]. In addition we remark that the optimality condition (2.42) and therefore the reduced gradient $\nabla j(\mathbf{c}) = \lambda \mathbf{c} - q\mathbf{n} + \partial_n \mathbf{z}$ can only be interpreted in the sense of distributions. The optimisation algorithm will obviously be based on evaluation of $\nabla j(\mathbf{c})$ to get new boundary control $\mathbf{c}_{new}$, so that we have to evaluate $\nabla j(\mathbf{c})$ to be able to set Dirichlet values – a task that was already under consideration in the last section where we showed a way to interpret the boundary condition $\mathbf{v}|_{\Gamma_0} = \nabla \phi$.

Alternatively the problem of using the correct norm for control within the cost functional can be avoided, when we use the sensitivity approach. Since we will compare the two optimisation approaches later in the numerical results, we only briefly define the backflow optimisation problem also in terms of sensitivities. Let the setting be given as before except the control $\mathbf{c}$ on Γ_c which is now defined as

$$\mathbf{v} = \mathbf{c} = \sum_{k=1}^{K} \alpha_k f_k(\mathbf{x}, t) \qquad \text{on } \Gamma_c \times (0, T) \tag{2.45}$$

with constant functions $f_k : \Gamma_c \times (0,T) \to \mathbb{R}^3$ modelling the inflow on control boundary Γ_c. This simplification using a linear combination of ansatz functions is very handy for notations, nevertheless the control $\mathbf{c}$ might also be given as a nonlinear function. Only the appearance of (design-)parameter α_k is crucial in this formulation. The cost functional will then be given as

$$\min_{\mathbf{v},\alpha_k} J(\mathbf{v}, \alpha_k) = \frac{\beta_1}{2} \int_0^T \|\mathbf{v} - \mathbf{v}_d\|^2_{L^2(\Omega_s)}\, dt + \frac{\beta_2}{2} \|\mathbf{v}(T) - \mathbf{v}_d\|^2_{L^2(\Omega_s)} + \frac{\lambda}{2} \sum_{k=1}^{K} |\alpha_k|^2. \tag{2.46}$$

Determination of the gradient by direct (formal) differentiation yields for $k = 1, \ldots, K$

$$\frac{DJ}{D\alpha_k} = \lambda \alpha_k + \beta_1 \int_0^T \int_{\Omega_s} (\mathbf{v} - \mathbf{v}_d) \mathbf{v}_{\alpha_i}\, d\mathbf{x}\, dt + \beta_2 \int_{\Omega_s} (\mathbf{v}(T) - \mathbf{v}_d) \mathbf{v}_{\alpha_k}|_{t=T}\, ds, \tag{2.47}$$

where $\mathbf{v}_{\alpha_k} = \frac{D\mathbf{v}}{D\alpha_k}$ denotes the sensitivities that are given by the sensitivity equations

$$\begin{aligned}
Re\,[\partial_t \mathbf{v}_{\alpha_k} + (\mathbf{v}_{\alpha_k} \cdot \nabla)\mathbf{v} + (\mathbf{v} \cdot \nabla)\mathbf{v}_{\alpha_k}] - \Delta \mathbf{v}_{\alpha_k} + \nabla p_{\alpha_k} &= 0 && \text{in } \Omega \times (0,T),\\
\nabla \cdot \mathbf{v}_{\alpha_k} &= 0 && \text{in } \Omega \times (0,T),\\
\mathbf{v}_{\alpha_k} &= 0 && \text{on } (\Gamma_0 \cup \Gamma_{in}) \times (0,T),\\
\mathbf{v}_{\alpha_k} &= \frac{\partial \mathbf{c}}{\partial \alpha_k} && \text{on } \Gamma_c \times (0,T),\\
\partial_n \mathbf{v}_{\alpha_k} - p_{\alpha_k} \mathbf{n} &= 0 && \text{on } \Gamma_{out} \times (0,T),\\
\mathbf{v}_{\alpha_k}(\cdot, 0) &= 0 && \text{in } \Omega.
\end{aligned} \tag{2.48}$$

We see that the sensitivity equations are nothing more than the linearised state equations (2.39).

Electrically Excited Flow

To end the section on flow control we also want to denote the sensitivity based optimisation approach for the electrically excited fluid flow. Recall the governing equations for the bulk flow within the microchannel

$$\begin{aligned}
Re[\partial_t \mathbf{v} + (\mathbf{v} \cdot \nabla)\mathbf{v}] - \Delta \mathbf{v} + \nabla p &= 0 && \text{in } \Omega \times (0,T),\\
\nabla \cdot \mathbf{v} &= 0 && \text{in } \Omega \times (0,T),\\
\Delta \phi &= 0 && \text{in } \Omega \times (0,T),\\
\partial_t c - \frac{1}{Re \cdot Sc} \Delta c + \mathbf{v} \cdot \nabla c &= 0 && \text{in } \Omega \times (0,T),
\end{aligned} \tag{2.49}$$

with suitable boundary and initial conditions as introduced before. The reader should be aware that c now denotes the concentration field $c : \Omega \to \mathbb{R}$ and no longer the control. It was already addressed that the potential ϕ was motivated in order to manipulate the flow field $\mathbf{v}$ to get a better (we will have to define what this means) distribution of concentration c. Hence, the control should be applied to the potential field ϕ. Since a physical meaningful influence on ϕ can only be given by the potential difference $\Delta\phi$ between Γ_{in} and Γ_{out}, we seek to control the boundary values $\phi|_{\Gamma_{out}}$ and tacitly assume that $\phi|_{\Gamma_{in}} = 0$.

The applied potential on Γ_{out} can be supposed to be spatially constant as we only simulate the small meander part of the whole microchannel and expect a linear behaviour of ϕ within the straight parts starting from the reservoirs. Thus the modification of $\phi|_{\Gamma_{out}}$ will only be temporally with respect to the amplitude and a possible frequency. If for any reason a periodic alternating potential difference seems justified, a Fourier series can be used to describe the temporal change, i.e. we get the control boundary condition

$$\phi(t) = \phi_c(t, \alpha_k) = \frac{\alpha_0}{2} + \sum_{k=1}^{n} \alpha_k \cos(k\omega t) + \sum_{k=n+1}^{2n} \alpha_k \sin((k-n)\omega t) \quad \text{on } \Gamma_{out} \times (0,T) \tag{2.50}$$

with the frequency $\omega = 2\pi/T$. This approach ends up with $2n+1$ parameters to be controlled, which for small n seems reasonable, since sensitivity equations are stated also forward-in-time in contrast to the adjoint equations (cf. (2.43)) – we will go into detail about this fact in Chapter 5.1. Obviously there are a lot of other attempts to describe the boundary condition (2.50) like a e.g. polynomial series. Which way is the best has to be based on physical observations and numerical simulations as we will do in Chapter 6.

The sensitivity equations can in any case be derived also independently of the choice of cost functional (we again use $\cdot_{\alpha_k} = \frac{\partial \cdot}{\partial \alpha_k}$ for the sensitivities):

$$\begin{aligned} Re\,[\partial_t \mathbf{v}_{\alpha_k} + (\mathbf{v}_{\alpha_k} \cdot \nabla)\mathbf{v} + (\mathbf{v} \cdot \nabla)\mathbf{v}_{\alpha_k}] - \Delta \mathbf{v}_{\alpha_k} + \nabla p_{\alpha_k} &= 0 \qquad \text{in } \Omega \times (0,T), \\ \nabla \cdot \mathbf{v}_{\alpha_k} &= 0 \qquad \text{in } \Omega \times (0,T), \\ \Delta \phi_{\alpha_k} &= 0 \qquad \text{in } \Omega \times (0,T), \\ \partial_t c_{\alpha_k} - \frac{1}{Re \cdot Sc} \Delta c_{\alpha_k} + \mathbf{v}_{\alpha_k} \cdot \nabla c + \mathbf{v} \cdot \nabla c_{\alpha_k} &= 0 \qquad \text{in } \Omega \times (0,T). \end{aligned} \tag{2.51}$$

Boundary conditions were also linearised and using the pressure-drop formulation are:

$$\begin{aligned} \mathbf{v}_{\alpha_k}|_{\Gamma_0} &= -\Pi_2 \nabla \Phi_{\alpha_k}, & \partial_n \mathbf{v}_{\alpha_k}|_{\Gamma_{in} \cup \Gamma_{out}} &= 0, & & \\ p_{\alpha_k}|_{\Gamma_{in} \cup \Gamma_{out}} &= 0, & & & & \\ \phi_{\alpha_k}|_{\Gamma_{in}} &= 0, & \phi_{\alpha_k}|_{\Gamma_{out}} &= \frac{\partial \phi_c}{\partial \alpha_k}, & \partial_n \phi_{\alpha_k}|_{\Gamma_0} &= 0 \\ c_{\alpha_k}|_{\Gamma_{in}} &= 0, & \partial_n c_{\alpha_k}|_{\Gamma_0 \cup \Gamma_{out}} &= 0. & & \end{aligned} \tag{2.52}$$

For the actual used setting of the problem and especially the formulation of cost functional we refer to Chapter 6.

2.4 Bibliographical Notes

In this chapter we gave an introducing and surveying view on derivation and analysis of governing equations of fluid flows as well as basic insight to optimisation and optimal control problems with partial differential equations. A detailed consideration of each of these topics would be beyond the scope of this thesis, since we will concentrate on numerical methods on high-performance computers to solve these problems. Below we list some additional references (we do not make claims of being complete) that have been useful for us and some that contain fairly extensive bibliographies.

Derivation and physical aspects of fluid flows problems

Besides the already cited [6, 13, 17, 50, 139, 154] we mention:

- Chorin, A. J. & Marsden, J. E [2000] *A mathematical introduction to fluid mechanics*, Springer
- Crowe, C. T.; Elger, D. F. & Roberson, J. A. [2005] *Engineering fluid mechanics*, Wiley
- Gurtin, M. E. [2003] *An introduction to continuum mechanics*, Acad. Press
- Kohr, M. & Pop, I. [2004] *Viscous incompressible flow for low Reynolds numbers*, WIT Press
- Landau, L. D. & Lifsic, E. M. [1987] *Fluid mechanics*, Pergamon Press

- Meyer, R. E. [1971] *Introduction to mathematical fluid dynamics*, Wiley
- Spurk, J. H. [1997] *Fluid mechanics*, Springer

Analysis of Navier-Stokes equations and partial differential equations in general

Besides the already cited [2, 35, 61, 62, 101, 144, 152, 153] we mention:

- Cuvelier, C.; Segal, A. & van Steenhoven, A. A. [1986] *Finite element methods and Navier-Stokes equations*, D. Reidel Publishing Company
- Doering, C. R. & Gibbon, J. D. [1995] *Applied analysis of the Navier-Stokes equations*, Cambridge University Press
- Evans, L. C. [2008] *Partial differential equations*, American Mathematical Society
- Galdi, G. P. [1994] *An Introduction to the Mathematical Theory of the Navier-Stokes Equations*, Springer
- Gunzburger, M. D. [1989] *Finite element methods for viscous incompressible flows: a guide to theory, practice, and algorithms*, Academic Press
- Layton, W. J. [2008] *Introduction to the numerical analysis of incompressible viscous flows*, Society for Industrial and Applied Mathematics

Optimisation with partial differential equations and in general

Besides the already cited [73, 105, 145, 157] we mention:

- Barbu, V. [1993] *Analysis and control of nonlinear infinite dimensional systems*, Academic Press
- Dennis, J. E. & Schnabel, R. B. [1996] *Numerical methods for unconstrained optimisation and nonlinear equations*, Society for Industrial and Applied Mathematics
- Fattorini, H. O. [1999] *Infinite dimensional optimisation and control theory*, Cambridge University Press
- Fletcher, R. [2004] *Practical methods of optimisation*, Wiley
- Kelley, C. T. [1999] *Iterative methods for optimisation*, Society for Industrial and Applied Mathematics
- Nocedal, J. & Wright, S. J. [2006] *Numerical optimisation*, Springer

Chapter 3

Discretisation and Sequential Solver

As shown in the previous chapter, adjoint-based optimisation as well as sensitivity-based optimisation for fluid flow problems yield systems of similar type. First there is obviously the state equation given by Navier-Stokes equations (with additional equations for potential and concentration), but also the adjoint equations and sensitivity equations are of linearised Navier-Stokes type. We pointed out that tackling the optimisation problem using an optimisation algorithm based on the reduced cost functional, necessitates the solution of both state and adjoint/sensitivity equations to derive the gradient.

The following chapter will therefore concentrate on numerical solution of these equations, i.e. discretisation of the underlying partial differential equations by means of finite element method and solution of algebraic system by suitable linear solvers. Since major issues arise from the saddle point structure of Navier-Stokes equations, we will stick to the abstract equations

$$\begin{aligned} \partial_t \mathbf{v} - \Delta \mathbf{v} + N(\mathbf{v}) + \nabla p &= 0, \\ \nabla \cdot \mathbf{v} &= 0, \end{aligned} \tag{3.1}$$

from which state equations itself and also sensitivity/adjoint equations can be recovered. Convection-diffusion and Laplace equation in the complete system might be decoupled from the Navier-Stokes equations so that this strategy seems justified. Principally we will work out numerical methods for solving the discrete system using iterative solvers and Multilevel ILU-based preconditioners - this sequential solver/preconditioner will be the basis for parallel solver/preconditioner introduced in Chapter 4.

3.1 Finite Element Method for Navier-Stokes Equations

A further simplification of the general problem (3.1) allows to treat also the Stokes problem by neglecting the nonlinear term $N(\mathbf{v})$. We will thus first show the discretisation for Stokes equations only and then mention aspects, how these results can be transferred to the equations under consideration - namely the nonlinear state equations and the linear adjoint/sensitivity equations. These steps obviously comprise the discretisation itself in space-variables but also linearisation of the nonlinear system and aspects of discretisation in time. Beside the here used discretisation by finite element method there are other approaches for the discretisation of Navier-Stokes equations or more general for fluid flow problems - these comprise finite difference or finite volume methods (cf. [125]) as well as lattice Boltzmann methods (cf. [98]).

The spatial discretisation of Navier-Stokes equations by means of finite element approximation bases on discrete representation of the weak formulation for the underlying continuous equations. For the sake of simplicity we restrict ourselves to systems with homogeneous Dirichlet boundary conditions for the velocity, i.e. $\mathbf{v}|_{\partial\Omega} = 0$ and refer to Section 2.2 for used notation.

Remark 3.1.1 (Treatment of general boundary conditions)
The weak formulation (2.20) is equipped with inhomogeneous Dirichlet boundary conditions, which are essential for this formulation, i.e. they are imposed by the function space $\mathbf{H}^1_g(\Omega)$. For the more

general case of channel flows, we also have to treat the outflow condition $\partial_n \mathbf{v} - p\mathbf{n}$ on Γ_{out} or the previously presented weak formulation of pressure-drop. In the case of inhomogeneous boundary conditions on $\Gamma_d \subseteq \partial\Omega$, we assume a function $\mathbf{g} \in \mathbf{H}^{1/2}(\Gamma_d)$ given as trace of a function in $\mathbf{H}^1(\Omega)$ which have to fulfil

$$\int_{\Gamma_d} \mathbf{g} \cdot \mathbf{n} \, ds = 0$$

whenever $\Gamma_d = \partial\Omega$ due to continuity equation. The test functions φ will always be given in $\mathbf{H}^1_{\Gamma_d}(\Omega)$, i.e. have homogeneous boundary conditions on Γ_d.

Within finite element discretisation the inhomogeneous boundary conditions are usually treated by a boundary interpolant $\mathbf{g}_h$, cf. [138]. One first chooses a function $\mathbf{g}_h$ that is in $\mathbf{V}_{h,\Gamma_d}$, the restriction of the finite element space $\mathbf{V}_h \subset \mathbf{H}^1(\Omega)$ to the boundary Γ_d, and approximates $\mathbf{g}$. Then the approximate problem is stated for $\mathbf{v}_h \in \mathbf{V}_h$ such that $\mathbf{v}_h|_{\Gamma_d} = \mathbf{g}_h$.

Using Lagrangian finite element spaces the interpolant $\mathbf{g}_h$ can simply be chosen as the function values of $\mathbf{g}$ at degrees of freedom on Γ_d. The theoretical aspects like div-stability and error-estimates carry over to the case of inhomogeneous boundary condition except for the constants - we refer to [51, 62, 78, 74] for details.

The boundary conditions for Γ_{out} on the other hand are not that complicated. As already mentioned the do-nothing condition

$$\int_{\Gamma_out} (\partial_n \mathbf{v} - p\mathbf{n}) \cdot \varphi \, ds \qquad \forall \varphi \in \mathbf{V} \subset \mathbf{H}^1(\Omega)$$

naturally appears when deriving the weak formulation of Navier-Stokes equations (2.20). The only change that has to be made is about the function space $\mathbf{V}$ for test functions φ. Instead of imposing zero boundary condition on the whole boundary $\partial\Omega$, i.e. using $\mathbf{H}^1_0(\Omega)$, we now use

$$\mathbf{V} = \{\mathbf{v} \in \mathbf{H}^1(\Omega) : \ \mathbf{v} = 0 \ \text{on} \ \partial\Omega \backslash \Gamma_{out}\}.$$

For the pressure-drop formulation we only have to impose zero boundary condition on the channel walls, for trial and test functions, since the boundary condition is incorporated into the weak formulation.

For the homogeneous weak form of (2.20) we now introduce the Galerkin formulation of finite element approximation. In the usual manner first one chooses approximating finite dimensional spaces $\mathbf{V}_h$ and S_h for the velocity and pressure, where the index h shows dependence on an associated mesh with characteristic size h. Then one replaces the infinite dimensional spaces in (2.20) by the finite ones, i.e. seek $\mathbf{v}_h(t) \in \mathbf{V}_h \subset \mathbf{H}^1_0(\Omega)$ and $p_h(t) \in S_h \subset L^2_0(\Omega)$ such that $\mathbf{v}_h(0) = I_h(\mathbf{v}_0)$ and for almost all $t \in (0,T)$ it holds[1]

$$\begin{aligned} Re\left[(\partial_t \mathbf{v}_h, \varphi_h) + c(\mathbf{v}_h, \mathbf{v}_h, \varphi_h)\right] + a(\mathbf{v}_h, \varphi_h) + b(\varphi_h, p_h) &= \langle \mathbf{f}, \varphi_h \rangle \qquad && \forall \varphi_h \in \mathbf{V}_h \\ b(\mathbf{v}_h, \xi_h) &= 0 && \forall \xi_h \in S_h \end{aligned} \tag{3.2}$$

where $I_h(\mathbf{v}_0)(x)$ is a suitable approximation to the initial data. In this work we only consider the conforming Galerkin formulation, i.e. the finite dimensional spaces $\mathbf{V}_h$ and S_h are subsets of $\mathbf{H}^1_0(\Omega)$ resp. $L^2_0(\Omega)$. The last step to get an algebraic system for the finite element approximation to Navier-Stokes equations is to identify bases for $\mathbf{V}_h$ and S_h and to express the trial and test functions as linear combinations of the basis-functions. Additionally a linearisation method must be applied since the resulting system is nonlinear - we will discuss this aspect in the following. Furthermore for the instationary case a time-discretisation has to be performed. We will use a semi-discrete approach, resulting in a nonlinear system for each timestep that is obviously similar to the stationary system. Concrete numerical methods for the linear system arising in each timestep/linearisation-step will be the subject of the next section.

The presented results concerning the finite element method are of summarising character since

[1]for sake of simplicity and readability we will use $Re = 1$ in the following

the emphasis of this work is put on implementation issues. For a more detailed overview on the finite element method for Navier-Stokes equations there exists a whole bunch of literature - for the more theoretical aspects we refer to [61, 62, 152], while the books [37, 47, 72, 126] also cover aspects of implementation and applications. The books of Gresho and Sani [68, 67] provide an extensive coverage of the finite element method related to incompressible flows, the reader will also find detailed reference tables within.

Finite Element Mesh

Since the finite element approximation spaces $\mathbf{V}_h,\ S_h$ rely on the definition of underlying meshes, we have to view the domain Ω (bounded, open and connected) as discretised into Ω_h where Ω_h does not need to be equal to Ω. Let $\mathcal{T}_h(\Omega_h)$ be a partition (also called *mesh*) of the domain Ω_h into M convex elements/cells $K_m \neq \emptyset$ such that

$$\overline{\Omega}_h = \bigcup_{m=1}^{M} K_m,\ \overset{\circ}{K}_m \cap \overset{\circ}{K}_n = \emptyset \text{ for } m \neq n$$

$$\partial K_m \text{ is Lipschitz-continuous for all } K_m \in \mathcal{T}_h$$

where $\overset{\circ}{K}_m$ is the interior of cell K_m. The condition $\overset{\circ}{K}_m \cap \overset{\circ}{K}_n = \emptyset$ means that two cells at most share edge/vertices in 2D or face/edges/vertices in 3D. We define the characteristical mesh size h to be the maximum diameter of all cells, i.e.

$$h_m = \operatorname{diam}(K_m) \leq h \qquad \forall K_m \in \mathcal{T}_h.$$

Together with the definition

$$\rho_m = \sup\{\operatorname{diam}(B) :\ B \text{ is a ball contained in } K_m\}$$

we define the regularity of a cell K_m to be

$$\sigma_m = \frac{h_m}{\rho_m}$$

and say that the mesh $\mathcal{T}_h$ is *regular* if there exists a constant $\sigma \geq 1$ independent of h and K_m, such that

$$\sigma_m \leq \sigma \qquad \forall K_m \in \mathcal{T}_h.$$

In this work we only consider polygonal domains $\Omega \subset \mathbb{R}^2$ or polyhedral domains in $\mathbb{R}^3$ which gives $\Omega = \Omega_h$ and denote the finite element mesh by $\mathcal{T}_h$. If not stated differently the boundary of Ω is denoted by Γ. Furthermore we only use finite elements that are defined on quadrilaterals in 2D or hexahedrons in 3D. The use of isoparametric elements would also allow curved boundaries, if these boundaries are smooth enough. For an overview on general construction of finite elements we refer e.g. to chapter 4 in [37] or chapter 6 in [120].

3.1.1 The Stationary Case: Space Discretisation

We will first show the discretisation with respect to the spatial variables, i.e. we restrict ourselves to the homogeneous stationary Navier-Stokes equations. Hence, after the definition of subspaces $\mathbf{V}_h \subset H_0^1(\Omega)$ and $S_h \subset L_0^2(\Omega)$ we seek for $\mathbf{v}_h \in \mathbf{V}_h,\ p_h \in S_h$ such that

$$\begin{aligned} a(\mathbf{v}_h, \varphi_h) + c(\mathbf{v}_h, \mathbf{v}_h, \varphi_h) + b(\varphi_h, p_h) &= \langle \mathbf{f}, \varphi_h \rangle & \forall \varphi_h \in \mathbf{V}_h, \\ b(\mathbf{v}_h, \xi_h) &= 0 & \forall \xi_h \in S_h. \end{aligned} \tag{3.3}$$

To be even more restrictive we first neglect the nonlinear term $c(\mathbf{v}_h, \mathbf{v}_h, \varphi_h)$ and show aspects of suitable finite element approximation for the homogeneous Stokes equations

$$\begin{aligned} a(\mathbf{v}_h, \varphi_h) + b(\varphi_h, p_h) &= \langle \mathbf{f}, \varphi_h \rangle & \forall \varphi_h \in \mathbf{V}_h, \\ b(\mathbf{v}_h, \xi_h) &= 0 & \forall \xi_h \in S_h. \end{aligned} \tag{3.4}$$

In comparison to coercive partial differential equations, we already showed in Section 2.2 that the inf-sup condition connecting the spaces $\mathbf{V}_h$ and S_h is of major importance for saddle point problems. While the inclusions $\mathbf{V}_h \subset H_0^1(\Omega)$, $S_h \subset L_0^2(\Omega)$ preserve continuity of $a(\cdot,\cdot)$, $b(\cdot,\cdot)$ and coercivity of $a(\cdot,\cdot)$ when using a conformal Galerkin method, there is in general no reason for the inf-sup condition to be valid in the discrete spaces. Hence for the discrete spaces $\mathbf{V}_h$, S_h the essential conditions are:

1. Continuity for the linear forms (2.17)-(2.19): $a \in \mathcal{L}(\mathbf{V}_h \times \mathbf{V}_h, \mathbb{R})$, $b \in \mathcal{L}(\mathbf{V}_h \times S_h, \mathbb{R})$ and $c \in \mathcal{L}(\mathbf{V}_h \times \mathbf{V}_h \times \mathbf{V}_h, \mathbb{R})$, i.e. there exist positive constant κ_a, κ_b and κ_c (independent of h) such that
$$\begin{aligned} |a(\mathbf{u}_h, \mathbf{v}_h)| &\le \kappa_a \|\mathbf{u}_h\|_1 \|\mathbf{v}_h\|_1 && \forall \mathbf{u}_h, \mathbf{v}_h \in \mathbf{V}_h, \\ |b(\mathbf{v}_h, q_h)| &\le \kappa_b \|\mathbf{v}_h\|_1 \|q_h\|_0 && \forall \mathbf{v}_h \in \mathbf{V}_h,\ q_h \in S_h, \\ |c(\mathbf{w}_h, \mathbf{u}_h, \mathbf{v}_h)| &\le \kappa_c \|\mathbf{u}_h\|_1 \|\mathbf{v}_h\|_1 \|\mathbf{w}_h\|_1 && \forall \mathbf{u}_h, \mathbf{v}_h, \mathbf{w}_h \in \mathbf{V}_h. \end{aligned}$$
These conditions are valid for the entire spaces $\mathbf{H}^1(\Omega)$, $L^2(\Omega)$ and therefore carry over to the spaces $\mathbf{V}_h$, S_h. For the first two conditions cf. [34] and the third condition is proven in Chapter 2.1 [62].

2. Coercivity condition for the linear form $a(\cdot,\cdot)$: there exists a positive constant γ_a (independent of h) such that
$$\sup_{0 \ne \mathbf{v}_h \in \mathbf{Z}_h} \frac{a(\mathbf{z}_h, \mathbf{v}_h)}{\|\mathbf{v}_h\|_1} \ge \gamma_a \|\mathbf{z}_h\|_1, \qquad \forall \mathbf{z}_h \in \mathbf{Z}_h. \tag{3.5}$$
This condition incorporates the incompressibility constraint of Navier-Stokes equations by means of the space $\mathbf{Z}_h := \{\mathbf{v}_h \in \mathbf{V}_h : \ b(\mathbf{v}_h, q_h) = 0 \ \forall q_h \in S_h\}$ of discretely divergence free functions, to the well known coercivity condition. Its validity follows from the coercivity of the linear form $a(\cdot,\cdot)$, i.e. $a(\mathbf{v},\mathbf{v}) \ge \gamma_a \|\mathbf{v}\|_1^2$ for all $\mathbf{v} \in \mathbf{H}_0^1(\Omega)$ and the inclusion $\mathbf{Z}_h \subset \mathbf{H}_0^1(\Omega)$.

3. *Ladyzhenskaya-Babuska-Brezzi* (LBB) or *inf-sup condition*: there exists a positive constant γ_b (independent of h) such that
$$\inf_{0 \ne q_h \in S_h} \sup_{0 \ne \mathbf{v}_h \in \mathbf{V}_h} \frac{b(\mathbf{v}_h, q_h)}{\|v_h\|_1 \|q_h\|_0} \ge \gamma_b. \tag{3.6}$$
Since even the conform Galerkin approach $\mathbf{V}_h \subset \mathbf{H}_0^1(\Omega)$ and $S_h \subset L_0^2(\Omega)$ does not guarantee that the inf-sup condition is fulfilled, this condition has to be proven for each choice of finite element spaces separately.

Under these conditions, similar to Section 2.2, we have the central theorem for the approximate solution $\mathbf{v}_h$, p_h of the Stokes problem (3.4) (cf. [47, 62] for proof).

Theorem 3.1.1
Assume that $\mathbf{V}_h$ and S_h are finite-dimensional subspaces of $\mathbf{H}_0^1(\Omega)$ resp. $L_0^2(\Omega)$. The discrete problem (3.4) is well-posed in the sense of Hadamard:

- *there exists a unique solution $\mathbf{v}_h$, p_h*
- *the solution continuously depends on the data $\mathbf{f}$, i.e.*
$$\|\mathbf{v}_h\|_1 \le \frac{1}{\gamma_a}\|f\|_{-1} \text{ and } \|p_h\|_0 \le \frac{1}{\gamma_b}\left(1 + \frac{\kappa_a}{\gamma_a}\right)\|f\|_{-1}$$

if and only if the discrete spaces fulfil the inf-sup condition (3.6).
In this case the solution $\mathbf{v}_h$, p_h satisfies the estimates
$$\begin{aligned} \|\mathbf{v} - \mathbf{v}_h\|_1 &\le c_1 \inf_{\mathbf{u}_h \in \mathbf{V}_h} \|\mathbf{v} - \mathbf{u}_h\|_1 + c_2 \inf_{q_h \in S_h} \|p - q_h\|_0 \\ \|p - p_h\|_0 &\le c_3 \inf_{\mathbf{u}_h \in \mathbf{V}_h} \|\mathbf{v} - \mathbf{u}_h\|_1 + c_4 \inf_{q_h \in S_h} \|p - q_h\|_0 \end{aligned} \tag{3.7}$$

where the constants c_i just depend on $\kappa_a, \kappa_b, \gamma_a$ and γ_b and are independent of h whenever γ_b is independent of h.

This theorem shows that once the inf-sup condition is satisfied, the error in the finite element approximation depends only on the ability to approximate in the chosen finite element subspaces. A detailed introduction to finite element interpolation theory can be found in [34, 120, 148] - in the following we only show approximation properties for the used Lagrange finite elements.

We have seen that for well-posedness and convergence of the discrete solution of the homogeneous Stokes equations (3.4) it is primarily the discrete inf-sup condition that has to be proved, since the continuity and coercivity conditions are given by the choice of a conform Galerkin approach. For this task we want to mention an often used criterion by Fortin [53]

Lemma 3.1.1 (Fortin criterion)
Assume that the inf-sup condition for the continuous spaces $X = \mathbf{H}_0^1(\Omega)$, $M = L_0^2(\Omega)$ is satisfied with constant $\beta^* > 0$. Furthermore assume that there exists an operator $\tau_h : X \to \mathbf{V}_h \subset X$ such that

$$\begin{aligned} b(\mathbf{u} - \tau_h(\mathbf{u}), q_h) &= 0 && \forall \mathbf{u} \in X,\ q_h \in S_h \subset M \\ \|\tau_h(\mathbf{u})\|_1 &\le C^* \|\mathbf{u}\|_1 && \forall \mathbf{u} \in X \end{aligned}$$

where $C^* > 0$ does not depend on h. Then, the inf-sup condition (3.6) is satisfied with $\gamma_b = \frac{\beta^*}{C^*}$. Also the opposite direction is valid.

Now we come back to the inhomogeneous Navier-Stokes equations (3.3) and ask for the same approximation results as in Theorem 3.1.1. Since a detailed derivation of such error-estimates is beyond the scope of this work, we just state the main results and refer to [36, 91, 61, 62, 152] for all details. First we assume that there exists a unique solution to the stationary weak Navier-Stokes equations $\mathbf{v} \in \mathbf{V} = \{\mathbf{u} \in \mathbf{H}_0^1(\Omega) : \nabla \cdot \mathbf{u} = 0\}$, $p \in L_0^2(\Omega)$, so that for some $\delta > 0$

$$Re^2 \tilde{\kappa} \|\mathbf{f}\|_{\mathbf{V}^*} < 1 - \delta, \qquad \tilde{\kappa} = \sup_{\mathbf{u},\mathbf{v},\mathbf{w} \in \mathbf{V}} \frac{|c(\mathbf{w},\mathbf{u},\mathbf{v})|}{|\mathbf{w}|_1 |\mathbf{u}|_1 |\mathbf{v}|_1}. \tag{3.8}$$

Additionally we need two approximation properties of the spaces $\mathbf{V}_h$ and S_h:

H1/ there exists a mapping $\tau_h \in \mathcal{L}(\mathbf{H}^2(\Omega) \cap \mathbf{H}_0^1(\Omega), \mathbf{V}_h)$ such that

$$\begin{aligned} (q_h, \nabla \cdot (\mathbf{u} - \tau_h(\mathbf{u}))) &= 0 && \forall q_h \in S_h,\ \mathbf{u} \in \mathbf{H}^2(\Omega) \cap \mathbf{H}_0^1(\Omega) \\ \|\tau_h(\mathbf{u}) - \mathbf{u}\|_1 &\le Ch\|\mathbf{u}\|_2 && \forall \mathbf{u} \in \mathbf{H}^2(\Omega) \cap \mathbf{H}_0^1(\Omega) \end{aligned}$$

H2/ the orthogonal projection operator ρ_h on S_h satisfies

$$\|q - \rho_h(q)\|_0 \le Ch^m \|q\|_m \qquad \forall q \in H^m(\Omega) \cap L_0^2(\Omega),\ m = 0, 1.$$

Theorem 3.1.2
Under the approximation properties H1, H2 and (3.8), the problem (3.3) for h sufficiently small has a unique solution $\mathbf{v}_h, p_h$ and

$$\lim_{h \to 0} |\mathbf{v} - \mathbf{v}_h|_1 = 0.$$

If in addition the solution $\mathbf{v}, p$ belongs to $\mathbf{H}^2(\Omega) \times (H^1(\Omega) \cap L_0^2(\Omega))$, we have the following estimate

$$|\mathbf{v} - \mathbf{v}_h|_1 \le Ch(\|\mathbf{v}\|_2 + \|p\|_1).$$

For the proof we refer to [61] and just remark that besides the condition for uniqueness this theorem relies on the inf-sup condition and the interpolation property of finite element spaces. Indeed, using Fortin criterion together with an interpolation error result like it is shown next, we see that the properties H1 and H2 are fulfilled.

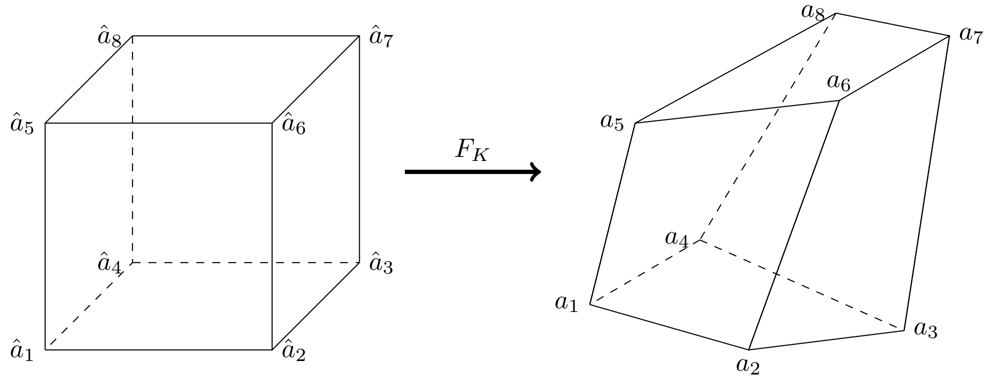


Figure 3.1: Mapping F_K from reference to arbitrary cell in 3D using 8 basis functions at the vertices.

3.1.2 Taylor-Hood Element

The inf-sup condition (3.6) was shown to be crucial for the approximation in the finite dimensional spaces $\mathbf{V}_h,\ S_h$, so that we need to introduce a pair satisfying this condition. In the following we solely use *Lagrange finite elements*, in which all degrees of freedom are point values at specific nodal points. Beside the Lagrange elements there are also so called *Hermite finite elements*, involving directional derivatives as degrees of freedom. We refer to [120] or [34] for a detailed introduction to the theory of finite elements. Following the definition of [34] a Lagrange finite element is a triplet $\{K, P, \Sigma\}$ where

i/ K is a compact, connected, Lipschitz subset of $\mathbb{R}^d$ with non-empty interior. Furthermore one defines a set of points $\{a_1, \dots, a_{n_{\text{loc}}}\}$ in K called *nodes* or *Lagrange nodes* and the subset K itself is denoted as a *cell*.

ii/ P is a vector space of functions $p : K \to \mathbb{R}$ called the *local interpolation space*.

iii/ Σ is a set of n_{loc} linear forms $\{\Phi_1, \dots, \Phi_{n_{\text{loc}}}\}$ acting on the elements of P such that $\Phi_i(p) = p(a_i),\ i = 1, \dots, n_{\text{loc}}$. Furthermore the linear mapping

$$p \in P \mapsto (\Phi_1(p), \dots, \Phi_{n_{\text{loc}}}(p)) \in \mathbb{R}^{n_{\text{loc}}}$$

is bijective - so called *P-unisolvence* of the set Σ. The linear forms Φ_i are called *local degrees of freedom* and due to the evaluating character at the nodes a_i, one often identifies the nodes with the degrees of freedom.

A direct consequence of the P-unisolvence is the existence of a basis $\{p_1, \dots, p_{n_{\text{loc}}}\}$ in P such that $\Phi_i(p_j) = p_j(a_i) = \delta_{ij},\ i, j = 1, \dots, n_{\text{loc}}$. This basis is called *local shape functions* or *nodal basis*.

Let now $\mathcal{T}_h$ be a mesh of the domain Ω. Instead of specifying a Lagrange finite element for each cell $K \in \mathcal{T}_h$ we define a reference cell $\hat{K}$ and transform the reference finite element $(\hat{K}, \hat{P}, \hat{\Sigma})$. For the two-dimensional case we define the element $[0,1]^2$ as reference cell $\hat{K}$ - in three dimensions we use $[0,1]^3$. On the reference cell basis/shape functions $\hat{p}_i$ and reference nodes $\hat{a}_i$ are assumed to be given by the chosen Lagrange finite element approach. In order to determine basis functions p_i on an arbitrary mesh cell K with nodal points a_i, we define a mapping $F_K :\ \hat{x} \in \hat{K} \mapsto x \in K$ by

$$x = F_K(\hat{x}) = \sum_{i=1}^{n} a_i \hat{p}_i(\hat{x}), \tag{3.9}$$

where the number $n \leq n_{\text{loc}}$ of basis functions depends on the local interpolation space and the intended character of the transformation. Obviously one gets $K = F_K(\hat{K})$ and $a_j = F_K(\hat{a}_j)$. Since in general the mapping F_K is nonlinear, the cells K are arbitrary straight or curved-sided elements according to the number of basis functions defining F_K in (3.9). See Figure 3.1 for the mapping in 3D using 8 basis functions at the vertices of the unit-hexahedron. Now the basis functions on a cell K are given by

$$p_i(x) = \hat{p}_i(F_K^{-1}(x)) \text{ with } x = F_K(\hat{x}).$$

This definition only holds for F_K being invertible, which is equivalent to a non-vanishing Jacobian and hence to a cell K being convex. Summing up, a generic Lagrange finite element (K, P_K, Σ_K) in the mesh $\mathcal{T}_h$ is such that

$$\begin{aligned} K &= F_K(\hat{K}) \\ P_K &= \{p : K \to \mathbb{R},\ p = \hat{p} \circ F_K^{-1},\ \hat{p} \in \hat{P}\} \\ \Sigma_K &= \{p(F_K(\hat{a}_i)),\ i = 1, \ldots, n_{\text{loc}}\}. \end{aligned}$$

Before defining the here used Lagrange finite elements in depth, we need to introduce the local and global interpolation operator. Let (K, P, Σ) be a generic finite element as shown above. Assume that there exists a normed vector space $V(K)$ such that $P \subset V(K)$ and such that the degrees of freedom $\Phi_i(v)$, $i = 1, \ldots, n_{\text{loc}}$ are well defined for all $v \in V(K)$. We define the *local interpolation operator* I_K unambiguously (because of P-unisolvence) by

$$I_K : \ v \in V(K) \mapsto I_K v = \sum_{i=1}^{n_{\text{loc}}} \Phi_i(v) p_i = \sum_{i=1}^{n_{\text{loc}}} v(a_i) p_i \in P. \tag{3.10}$$

For the Lagrange finite elements the spaces $V(K) = C^0(K)$ or $V(K) = H^s(K)$ with $s > d/2$ are suitable. The global interpolant $I_h v$ on the mesh $\mathcal{T}_h$ is then specified elementwise using the local interpolation operator (3.10)

$$(I_h v)|_K = I_K(v|_K),\ \forall K \in \mathcal{T}_h$$

and hence we have the *global interpolation operator* I_h

$$I_h : \ v \in D(I_h) \mapsto I_h v = \sum_{K \in \mathcal{T}_h} \sum_{i=1}^{n_{\text{loc}}} v(a_{K,i}) p_{K,i} \in V_h, \tag{3.11}$$

where the domain $D(I_h)$ is $C^0(\Omega_h)$ or $H^s(\Omega_h)$ with $s > d/2$. The codomain of I_h is called the *global finite element approximation space* V_h and since we are working with conformal finite elements it has to fulfil $V_h \subset H^1(\Omega)$. To ensure this conformity we need an additional condition for the mesh $\mathcal{T}_h$: any face of a cell K_m is either also a face of another cell K_n or part of the boundary $\partial\Omega$. If now two neighbouring cells K_m and K_n with sets of degrees of freedom $\{a_{K_{m/n},i}\}$, $i = 1, \ldots, n_{\text{loc}}$ fulfil

$$\mathcal{F}_{m,n} = \{a_{K_m,1}, \ldots, a_{K_m,n_{\text{loc}}}\} \cap K_n = \{a_{K_n,1}, \ldots, a_{K_n,n_{\text{loc}}}\} \cap K_m = \mathcal{F}_{n,m}$$

we have the set of global Lagrange nodes/degrees of freedom of $\mathcal{T}_h$

$$\mathcal{N}_h = \{a_1, \ldots, a_N\} = \bigcup_{K \in \mathcal{T}_h} \{a_{K,1}, \ldots, a_{K,n_{\text{loc}}}\}.$$

Then it is sufficient for V_h to be H^1-conform (cf. [47]) that functions in V_h are continuous at the nodes in $\mathcal{N}_h$, i.e. $v(a_{K_m,i}) = v(a_{K_n,i})$, $\forall i \in \mathcal{F}_{m,n}$. Furthermore the set $\mathcal{N}_h$ of global degrees of freedom allows for the definition of global basis functions in V_h via

$$\varphi_i \in V_h \text{ and } \varphi_i(a_j) = \delta_{ij},\ i, j = 1, \ldots, N$$

and hence the description of the global interpolation operator I_h (3.11) as

$$I_h : \ v \in D(I_h) \mapsto I_h v = \sum_{i=1}^{N} v(a_i) \varphi_i \in V_h. \tag{3.12}$$

Obviously the global basis functions φ_i are compositions of the local functions $p_{K,i}$.

Now we are able to introduce the *Taylor-Hood Element* and more general Q_k/Q_{k-1} elements with $k > 1$. A major reason for this choice should be mentioned beforehand: as opposed to the class

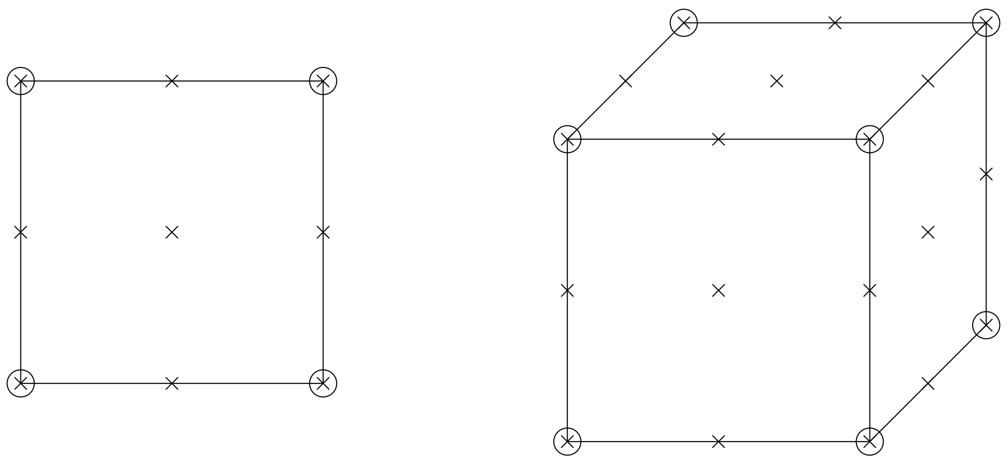


Figure 3.2: Taylor-Hood element on quadrilateral and hexahedron: bi-/tri-quadratic velocity and bi-/tri-linear pressure. Only visible degrees of freedom are shown - $\times$ velocity nodes and $\circ$ pressure nodes.

of piecewise linear or bilinear velocity fields combined with piecewise constant or linear pressure, this element is defined on the same mesh for both finite element spaces $\mathbf{V}_h$ and S_h. All other combinations of linear/constant or linear/linear elements are only inf-sup stable if the velocity field is defined on the finer mesh $\mathcal{T}_{h/2}$. For other stable elements or stabilisation techniques to circumvent the inf-sup condition, we refer to [37, 126] and references therein.

Due to the above definition of Lagrange finite elements, it is enough to specify the local interpolation space $\hat{P}$ and the set of Lagrange nodes $\hat{a}_i$. Let $\mathbb{Q}_k(\hat{K})$ denote the space of polynomials on $\hat{K}$ of degree less than or equal to k in each of the coordinate directions

$$\mathbb{Q}_k(\hat{K}) = \{\hat{q} = \sum_{i,j=0}^{k} \alpha_{ij} \hat{x_1}^i \hat{x_2}^j, \ \alpha_{ij} \in \mathbb{R}\} \qquad \text{in 2D,}$$

$$\mathbb{Q}_k(\hat{K}) = \{\hat{q} = \sum_{i,j,l=0}^{k} \alpha_{ijl} \hat{x_1}^i \hat{x_2}^j \hat{x_3}^l, \ \alpha_{ijl} \in \mathbb{R}\} \qquad \text{in 3D.}$$

The interpolation space on an arbitrary mesh cell K is then according to the previous defined mapping F_K

$$P_K = \mathbb{Q}_k(K) = \{q = \hat{q} \circ F_K^{-1} : \ \hat{q} \in \mathbb{Q}_k(\hat{K})\}$$

and the local shape functions are given by the Lagrange polynomials in the variables x_i. The set of nodes $\hat{a}_i$, which we identified with the degrees of freedom, depends on the polynomial order k and is shown for the case $k = 1, 2$ in Figure 3.2. Finally the global H^1-conform Lagrange finite element space of order k is

$$Q_k(\mathcal{T}_h) = \{u \in C^0(\overline{\Omega}) : \ u|_K \in \mathbb{Q}_k(K), \ \forall K \in \mathcal{T}_h\} \tag{3.13}$$

and the Taylor-Hood element pair is defined by (cf. Figure 3.2)

$$\begin{aligned} \mathbf{V}_h &= \{\mathbf{u}_h \in [Q_k(\mathcal{T}_h)]^d : \ \mathbf{u}_h|_\Gamma = 0\} \subset \mathbf{H}_0^1(\Omega) && \text{with } k = 2, \\ S_h &= \{q_h \in Q_l(\mathcal{T}_h) \cap L_0^2(\Omega)\} \subset H^1(\Omega) \cap L_0^2(\Omega) && \text{with } l = 1. \end{aligned} \tag{3.14}$$

Having defined the global approximation space (3.13) and the global interpolation operator I_h (3.12), we now have the following error-estimate to judge the quality of Q_k elements - for a proof we refer to [47] or [62] and remark that an analogue result holds for the 3D case.

Theorem 3.1.3 (Lagrange finite element interpolation error)
Let the Lagrange finite element space $Q_k(\mathcal{T}_h)$ be as in (3.13) and assume that $\mathcal{T}_h$ is a regular mesh of the polygonal domain $\Omega \subset \mathbb{R}^2$. Let the global interpolation operator to the space $Q_k(\mathcal{T}_h)$ be denoted by I_h^k. Then, there exists a constant c such that for all $h > 0$ and $v \in H^{k+1}(\Omega)$

$$\|v - I_h^k v\|_0 + \sum_{m=1}^{k+1} h^m \Big(\sum_{K \in \mathcal{T}_h} \|v - I_h^k v\|_m^2 \Big)^{\frac{1}{2}} \leq c h^{k+1} |v|_{k+1}.$$

In particular, since $Q_k(\mathcal{T}_h)$ is H^1-conform

$$|v - I_h^k v|_1 \leq c h^k |v|_{k+1}.$$

To state the final error-estimate for the Taylor-Hood element it is now enough to prove the inf-sup condition and use Theorem 3.1.1 in combination with the interpolation Theorem 3.1.3 above. The validity of inf-sup condition for the Taylor-Hood element can be proven in different way, e.g. by Verfürth's proof [47, 158] or usage of macro elements [62] - we just give the resulting error-estimate for the finite element interpolation of Stokes-equation.

Theorem 3.1.4 (Taylor-Hood error-estimate)
Let Ω be a bounded, plane polygon and let the solution $(\mathbf{v}, p)$ of the homogeneous Stokes problem satisfy

$$\mathbf{v} \in \mathbf{H}^{k+1}(\Omega) \cap \mathbf{H}_0^1(\Omega), \;\; p \in H^k(\Omega) \cap L_0^2(\Omega), \;\; k = 1, 2.$$

If the mesh $\mathcal{T}_h$ is regular, the solution $(\mathbf{v}_h . p_h)$ of the weak problem (3.4) with spaces $\mathbf{V}_h$, S_h defined in (3.14) satisfies the estimate:

$$|\mathbf{v} - \mathbf{v}_h|_1 + \|p - p_h\|_0 \leq c h^k (|\mathbf{v}|_{k+1} + |p|_k), \;\; k = 1, 2. \tag{3.15}$$

When Ω is convex, the additional estimate holds:

$$\|\mathbf{v} - \mathbf{v}_h\|_0 \leq c h^{k+1} (|\mathbf{v}|_{k+1} + |p|_k), \;\; k = 1, 2. \tag{3.16}$$

This inf-sup stable element was originally stated by Taylor and Hood [151] for triangles but Stenberg showed the same results also for the case of quadrilaterals [146] and hexahedrons [147]. Especially for the case of $k = 2$ the estimates (3.15) and (3.16) show one order higher accuracy than can be achieved by any element involving linear velocity, which is a further reason to prefer the Taylor-Hood element versus any combinations of linear/constant or linear/linear elements.

3.1.3 Linearisation of Navier-Stokes Equations

The finite element approximation of Navier-Stokes equations (3.3) leads to a nonlinear system of algebraic equations. These equations might be written down explicitly after identification of bases for $\mathbf{V}_h$, S_h in terms of the discrete versions of linear forms (2.17) - (2.19). For sake of simplicity we will show the linearisation by Newton's method in terms of equation (3.3) and only afterwards identify the linear algebraic system. Other linearisation methods like fixed Jacobian method, Broyden's method or Oseen linearisation were not under investigation due to the (at least locally) quadratic convergence of Newton's methods [62, 78]. The main disadvantage of Newton's method, namely the fact that at each step a new Jacobian has to be computed, can be shown to be of minor concern since this step can be parallelised perfectly on high-performance computers.

Let $\mathbf{V}_h$, S_h be a stable pair of finite element spaces and let $[\mathbf{v}_h^k, p_h^k] \in \mathbf{V}_h \times S_h$ be the current iterate, then the next iterate of Newton's method might be computed by solving the linearised system

$$\begin{aligned}
a(\mathbf{v}_h^{k+1},\varphi_h) + c(\mathbf{v}_h^{k+1},\mathbf{v}_h^k,\varphi_h) + c(\mathbf{v}_h^k,\mathbf{v}_h^{k+1},\varphi_h) + b(\varphi_h,p_h^{k+1})& \\
= \langle \mathbf{f},\varphi_h\rangle + c(\mathbf{v}_h^k,\mathbf{v}_h^k,\varphi_h) & \qquad \forall\varphi_h\in\mathbf{V}_h, \\
b(\mathbf{v}_h^{k+1},\xi_h) = 0 & \qquad \forall \xi_h \in S_h,
\end{aligned} \tag{3.17}$$

where $\mathbf{v}_h^{k+1}$ fulfils the same boundary conditions as $\mathbf{v}_h^k$. A more convenient way to perform Newton iteration is given by the defect-correction notation: for a general nonlinear problem $F(x)=0$ a formal Taylor-expansion around $\bar{x}$ yields

$$F(x) = F(\bar{x}) + DF(\bar{x})\cdot\delta x + \frac{1}{2!}D^2F(\bar{x})\cdot\delta x^2 + \ldots + \frac{1}{n!}D^nF(\bar{x})\cdot\delta x^n + R_n(x)$$

with $\delta x = x-\bar{x}$ and the remainder term $R_n(x) = \frac{1}{(n+1)!}D^{n+1}F(\xi)\cdot\delta x^{n+1}$, ξ lying on the line segment between x and $\bar{x}$. After dropping terms of order higher or equal to 2, the Newton iteration reads

$$DF(x^k)\cdot\delta x = -F(x^k),\ x^{k+1} = x^k+\delta x. \tag{3.18}$$

Using this notation allows a formulation in terms of the residual $-F(x^k)$, which will also be used in the sequel for timestepping methods. Furthermore this form prevents numerical errors for Dirichlet boundary values as we can set these values directly in the discrete vectors x^k and just set the corresponding rows in $DF(x^k)$ to identity. A special version of Newton's method (within globalisation and inexact linear solver) in terms of algorithmic procedure is given in Algorithm 2.

To end this subsection we want to present the resulting algebraic system for Newton's method. Let $[\mathbf{v}_h^k, p_h^k]$ be the solution of the last Newton step and $[\varphi_h,\xi_h]$ be the test functions, then linearisation of the weak momentum and continuity equation is given by

$$\begin{aligned}
&D_{[\mathbf{v}_h^k,p_h^k]}\big[a(\mathbf{v}_h^k,\varphi_h) + c(\mathbf{v}_h^k,\mathbf{v}_h^k,\varphi_h) + b(\varphi_h,p_h^k)\big][\delta\mathbf{v},\delta p] \\
&\qquad = a(\delta\mathbf{v},\varphi_h) + c(\delta\mathbf{v},\mathbf{v}_h^k,\varphi_h) + c(\mathbf{v}_h^k,\delta\mathbf{v},\varphi_h) + b(\varphi_h,\delta p), \\
&D_{[\mathbf{v}_h^k,p_h^k]}\big[b(\mathbf{v}_h^k,\xi_h)\big][\delta\mathbf{v},\delta p] \\
&\qquad = b(\delta\mathbf{v},\xi_h).
\end{aligned}$$

Hence, Newton iteration (3.18) for stationary Navier-Stokes equations is given by

$$\begin{pmatrix} A+L(v^k) & B^T \\ B & 0\end{pmatrix}\begin{pmatrix}\delta v\\ \delta p\end{pmatrix} = \mathrm{res}(\begin{pmatrix}v^k\\ p^k\end{pmatrix}),\ \begin{pmatrix}v^{k+1}\\ p^{k+1}\end{pmatrix} = \begin{pmatrix}v^k\\ p^k\end{pmatrix} + \begin{pmatrix}\delta v\\ \delta p\end{pmatrix}. \tag{3.19}$$

Here we already used the algebraic notation that can be derived after identification of bases in the finite element spaces $\mathbf{V}_h,\ S_h$. Let $\{\varphi_h^j\}$ be a basis of $\mathbf{V}_h$ and let $\{\xi_h^l\}$ be a basis of S_h, then we are able to formulate the approximative solution in terms of finite element coefficients $[v,p]$, i.e.

$$\begin{aligned}
\mathbf{v}_h &= \sum_{j=1}^{dn} v_j\varphi_h^j && \text{with } n \text{ trial functions per velocity component and } v_j\in\mathbb{R} \\
p_h &= \sum_{l=1}^{m} p_l\xi_h^l && \text{with } m \text{ trial functions for the pressure and } p_l\in\mathbb{R}
\end{aligned} \tag{3.20}$$

and $v = [v_j]_{j=1,\ldots,dn},\ p=[p_l]_{l=1,\ldots,m}$. We use the notation $[v,p]$ to indicate the vector representing the finite element coefficients, which should not be mixed up with the continuous functions $\mathbf{v},\ p$ for velocity and pressure. Furthermore, one gets the finite element matrices

$$\begin{aligned}
[M]_{ij} &= (\varphi_h^j,\varphi_h^i),\ [A]_{ij} = a(\varphi_h^j,\varphi_h^i) && i,j=1,\ldots,dn \\
[B]_{lj} &= b(\varphi_h^j,\xi_h^l) && l=1,\ldots,m,\ j=1,\ldots,dn \\
[C(v^k)]_{ij} &= c(\varphi_h^j,\mathbf{v}_h^k,\varphi_h^i) && i,j=1,\ldots,dn \\
[L(v^k)]_{ij} &= c(\varphi_h^j,\mathbf{v}_h^k,\varphi_h^i) + c(\mathbf{v}_h^k,\varphi_h^j,\varphi_h^i) && i,j=1,\ldots,dn
\end{aligned} \tag{3.21}$$

the external force vector

$$[F]_i = \langle \mathbf{f}, \varphi_h^i \rangle \qquad i = 1, \ldots, dn$$

and the residual vector for $i = 1, \ldots, dn$ and $l = 1, \ldots, m$

$$\begin{aligned}\mathrm{res}(\begin{pmatrix} v^k \\ p^k \end{pmatrix}) &= \begin{pmatrix} \langle \mathbf{f}, \varphi_h^i \rangle - a(\mathbf{v}_h^k, \varphi_h^i) - c(\mathbf{v}_h^k, \mathbf{v}_h^k, \varphi_h^i) - b(\varphi_h^i, p_h^k) \\ -b(\mathbf{v}_h^k, \xi_h^l) \end{pmatrix} \\ &= \begin{pmatrix} F - Av^k - C(v^k)v^k - B^T p^k \\ -Bv^k \end{pmatrix}.\end{aligned} \tag{3.22}$$

The system (3.19) shows the typical structure of a discrete saddle point problem having a stiffness matrix with a zero block in the lower right part, which again underlines the need of stable discretisation.

3.1.4 The Instationary Case: Time Discretisation

Consider now the semi-discrete weak instationary Navier-Stokes problem (3.2), for which the analysis has been provided in [83] and following papers by Heywood and Rannacher - we just remark the main result on convergence.

Remark 3.1.2 (Convergence of approximate solutions)
Assume that $\mathbf{V}_h$ and S_h is a pair of finite element spaces satisfying the inf-sup condition (3.6). Furthermore assume that for all $\mathbf{u} \in \mathbf{H}_0^1(\Omega)$ and $q \in L_0^2(\Omega)$ the finite element approximation satisfies

$$\inf_{\mathbf{u}_h \in \mathbf{V}_h} \|\mathbf{u} - \mathbf{u}_h\|_1 + \inf_{q_h \in S_h} \|q - q_h\|_0 = O(h).$$

Notice that especially the above defined Taylor-Hood element pair is suitable for this condition. Then one can prove the error-estimate for the solution of the semi-discrete scheme, assuming that the solution $\mathbf{v}, p$ has suitable regularity.

$$\|\mathbf{v}(t) - \mathbf{v}_h(t)\|_0 \le C_1(t)h^2, \ \|p(t) - p_h(t)\|_0 \le C_2(t)h$$

where the constants $C_i(t)$ might exponentially grow if the data is not small enough.

For the numerical solution of (3.2) we follow the θ-family methods as discretisation in time followed by a Newton iteration in space using inf-sup stable finite element pair as described in the preceding subsection. This way is often called *method of lines* meaning that first a discretisation of space variables is done, which formally results in a system of ordinary differential equations. Afterwards a common time discretisation scheme is used to solve the system of the form

$$(\partial_t \mathbf{v}_h, \varphi_h) = F(\mathbf{f}, \mathbf{v}_h, p_h; \varphi_h) \qquad \forall \varphi_h \in \mathbf{V}_h. \tag{3.23}$$

To be as general as needed we do not state the ODE-system, but use the discrete weak formulation (3.2) to define the timestepping methods used here.

Given a fixed number $N > 0$, the time interval $[0, T]$ is partitioned into subintervals $[t_i, t_{i+1}]$, $i = 0, \ldots, N-1$ with $\Delta t_i = t_{i+1} - t_i$ and $t_N = T$. For the sake of simplicity we chose equidistant $\Delta t_i = \Delta t = T/N$ and use the notation $t_k = k\Delta t$, $\mathbf{v}_h^k = \mathbf{v}_h(x, t_k)$ and $p_h^k = p_h(x, t_k)$ for the discrete approximation in the k-th timestep. Accordingly, we define $\mathbf{f}^k = \mathbf{f}(x, t_k)$ to be the external force evaluated at discrete steps. The θ-family of methods now determines the new solution at time $t_{k+1} = t_k + \Delta t$ by a weighted average of $\partial_t \mathbf{v}_h(t_k)$ and $\partial_t \mathbf{v}_h(t_{k+1})$ with a parameter θ in the interval $[0, 1]$:

$$\frac{\mathbf{v}_h(t_{k+1}) - \mathbf{v}_h(t_k)}{\Delta t} = \theta \partial_t \mathbf{v}_h(t_{k+1}) + (1 - \theta)\partial_t \mathbf{v}_h(t_k) + O((\frac{1}{2} - \theta)\Delta t, \Delta t^2).$$

The terms $\partial_t \mathbf{v}_h(\cdot)$ will be replaced using (3.23). This discretisation is conditionally stable for $\theta < 1/2$ and unconditionally stable for $\theta \ge 1/2$. Furthermore, the truncation error $O((1/2 - \theta)\Delta t, \Delta t^2)$ shows that second order accuracy is only given for the choice $\theta = 1/2$ which is the well known Crank-Nicolson scheme. For these classical results we refer to [26] or the more involved books

scheme	step	α_1	α_2	α_3	α_4	α_5	Δt
forward Euler	$t_{k-1} \to t_k$	0	Δt_n	Δt_n	Δt_n	0	Δt_n
backward Euler	$t_{k-1} \to t_k$	Δt_n	Δt_n	0	0	Δt_n	Δt_n
Crank-Nicolson	$t_{k-1} \to t_k$	$\Delta t_n/2$	Δt_n	$\Delta t_n/2$	$\Delta t_n/2$	$\Delta t_n/2$	Δt_n
FS (substep 0)	$t_{k-1} \to t_k^0$	$\beta\theta\Delta t_n$	$\theta\Delta t_n$	$\gamma\theta\Delta t_n$	$\theta\Delta t_n$	0	$\theta\Delta t_n$
FS (substep 1)	$t_k^0 \to t_k^1$	$\gamma\tilde{\theta}\Delta t_n$	$\tilde{\theta}\Delta t_n$	$\beta\tilde{\theta}\Delta t_n$	0	$\tilde{\theta}\Delta t_n$	$\tilde{\theta}\Delta t_n$
FS (substep 2)	$t_k^1 \to t_k^2 = t_k$	$\beta\theta\Delta t_n$	$\theta\Delta t_n$	$\gamma\theta\Delta t_n$	$\theta\Delta t_n$	0	$\theta\Delta t_n$

Table 3.1: Timestepping parameter for several one-step schemes.

[81, 162]. Other time-discretisation methods like *leap-frog scheme* or *Adams-Bashforth method* that are also second order accurate, or even higher-order *Runge-Kutta methods* are not treated here, since we concentrate on single step method, i.e. those that only need the last state $\mathbf{v}_h(t_k)$ to determine the new solution $\mathbf{v}_h(t_{k+1})$.

Hence, our timestepping scheme relies on the problem: for given initial state $\mathbf{v}_h^0 \in \mathbf{V}_h$ find $(\mathbf{v}_h^k, p_h^k) \in \mathbf{V}_h \times S_h$ for $k = 1, 2, \ldots, N$ by solving the nonlinear system

$$\begin{aligned}(\mathbf{v}_h^k, \varphi_h) + \alpha_1\, n(\mathbf{v}_h^k, \varphi_h) + \alpha_2\, b(\varphi_h, p_h^k) &= (\mathbf{v}_h^{k-1}, \varphi_h) \\ -\alpha_3\, n(\mathbf{v}_h^{k-1}, \varphi_h) &+ \alpha_4(\mathbf{f}^{k-1}, \varphi_h) + \alpha_5(\mathbf{f}^k, \varphi_h) \qquad \forall \varphi_h \in \mathbf{V}_h \\ b(\mathbf{v}_h^k, \xi_h) &= 0 \qquad \forall \xi_h \in S_h\end{aligned} \tag{3.24}$$

with $n(\mathbf{v}_h^k, \varphi_h) = a(\mathbf{v}_h^k, \varphi_h) + c(\mathbf{v}_h^k, \mathbf{v}_h^k, \varphi_h)$. The upper index now indicates the discrete timestep and no longer the index of Newton iteration. We will clarify this notation whenever confusion might arise. The chosen formulation (3.24) - see also [92] - easily allows to describe common used one-step fully implicit or explicit timestepping schemes by setting the parameters α_i appropriately - for implicit/explicit Euler, Crank-Nicolson and fractional-step θ-scheme (FS) consider Table 3.1. Since the discrete system of the Navier-Stokes-equations is usually highly stiff, often only implicit or semi-implicit schemes are chosen. Nevertheless we also want to comment on the explicit Euler, since no nonlinear system needs to be solved in each timestep. Only a kind of mass-matrix has to be factorised once and can be used throughout all timesteps by means of matrix-vector multiplication.

Crank-Nicolson Scheme

The discrete system for timestepping based on Crank-Nicolson scheme is

$$\begin{aligned}(\mathbf{v}_h^k - \mathbf{v}_h^{k-1}, \varphi_h) + \frac{\Delta t}{2}\left[n(\mathbf{v}_h^k, \varphi_h) - n(\mathbf{v}_h^{k-1}, \varphi_h)\right] + \Delta t b(\varphi_h, p_h^k) &= \frac{\Delta t}{2}\langle \mathbf{f}^k + \mathbf{f}^{k-1}\varphi_h\rangle, \quad \forall \varphi_h \in \mathbf{V}_h, \\ b(\mathbf{v}_h^k, \xi_h) &= 0 \qquad \forall \xi_h \in S_h.\end{aligned}$$

After identification of basis functions (3.20) the resulting nonlinear algebraic will be solved by Newton's method similar to the stationary case. We therefore introduce the residual at the k-th timestep and i-th Newton step as (compare to (3.22) for notation)

$$\operatorname{res}(\begin{pmatrix} v^{k,i} \\ p^{k,i} \end{pmatrix}) = \begin{pmatrix} (\mathbf{v}_h^{k-1} - \mathbf{v}_h^{k,i}, \varphi_h) - \frac{\Delta t}{2}\left[n(\mathbf{v}_h^k, \varphi_h) - n(\mathbf{v}_h^{k-1}, \varphi_h)\right] - \Delta t b(\varphi_h, p_h^k) + \frac{\Delta t}{2}\langle \mathbf{f}^k + \mathbf{f}^{k-1}\varphi_h\rangle \\ -b(\mathbf{v}_h^{k,i}, \xi_h) \end{pmatrix}$$

and solve the system via Newton's method

$$\begin{pmatrix} M + \frac{\Delta t}{2}[A + L(v^{k,i})] & \Delta t B^T \\ B & 0 \end{pmatrix} \begin{pmatrix} \delta v \\ \delta p \end{pmatrix} = \operatorname{res}(\begin{pmatrix} v^{k,i} \\ p^{k,i} \end{pmatrix}), \quad \begin{pmatrix} v^{k,i+1} \\ p^{k,i+1} \end{pmatrix} = \begin{pmatrix} v^{k,i} \\ p^{k,i} \end{pmatrix} + \begin{pmatrix} \delta v \\ \delta p \end{pmatrix}.$$

After say $i_{\max}$ Newton steps, the approximate solution at timestep k is given as

$$\begin{pmatrix} v^k \\ p^k \end{pmatrix} = \begin{pmatrix} v^{k,i_{\max}} \\ p^{k,i_{\max}} \end{pmatrix}.$$

For the initial iterate in timestep k one should use the solution of the last timestep, i.e. $[v^{k,0}, p^{k,0}] = [v^{k-1}, p^{k-1}]$ since for small timesteps Δt and/or little change in the velocity- and pressure-field the change between two successive timesteps is small.

While the implicit Euler scheme is perhaps the most classical timestepping scheme and obviously easy to implement, the Crank-Nicolson scheme attains better accuracy at the same complexity for nonlinear systems to be solved at each timestep. Both methods are moreover unconditional stable, such that the advantage of implicit Euler scheme might only be given by its numerical damping property.

Explicit/Forward Euler

For the explicit time-discretisation of Navier-Stokes equations one encounters the system

$$\begin{aligned} (\mathbf{v}_h^k, \varphi_h) + \Delta t\ b(\varphi_h, p_h^k) &= (\mathbf{v}_h^{k-1}, \varphi_h) + \Delta t\left[\langle \mathbf{f}^{k-1}, \varphi_h\rangle - n(\mathbf{v}_h^{k-1}, \varphi_h)\right] && \forall \varphi_h \in V_h \\ b(\mathbf{v}_h^k, \xi_h) &= 0 && \forall \xi_h \in S_h \end{aligned} \tag{3.25}$$

or equivalently the algebraic system

$$\begin{pmatrix} M & \Delta t B^T \\ B & 0 \end{pmatrix} \begin{pmatrix} v^k \\ p^k \end{pmatrix} = \begin{pmatrix} Mv^{k-1} + \Delta t\big[F^{k-1} - Av^{k-1} - C(v^{k-1})v^{k-1}\big] \\ 0 \end{pmatrix}. \tag{3.26}$$

We see that the saddle point structure is still present, but that the system matrix is linear and constant for all timesteps. Hence it can be factorised once and only update steps have to be performed afterwards.

Conclusively we show that this explicit scheme can be interpreted in the sense of the well known *Chorin-Temam projection method.* The principle of the projection method is a separation of velocity and pressure field operators which leads to an intermediate state for the velocity that has to be projected onto the space of solenoidal functions. It was originally stated as a fractional-step scheme in the work of Chorin [33] and Temam [152] - for details we refer there. The first step in this method only treats the continuity equation in (3.2) and aims at determining an intermediate velocity field $\mathbf{v}_{\text{int}}^{k+1}$ by solving

$$\left(\frac{\mathbf{v}_{\text{int}}^k - \mathbf{v}_h^{k-1}}{\Delta t}, \varphi_h\right) + a(\mathbf{v}^*, \varphi_h) + c(\mathbf{v}^*, \mathbf{v}^{**}, \varphi_h) = \langle \mathbf{f}^{k-1}, \varphi_h\rangle \qquad \forall \varphi_h \in \mathbf{V}_h. \tag{3.27}$$

The velocities $\mathbf{v}^*, \mathbf{v}^{**}$ can be chosen according to the desired timestepping method, i.e.

$\mathbf{v}^* = \mathbf{v}^{**} = \mathbf{v}_h^{k-1}$	results in explicit Euler method,
$\mathbf{v}^* = \mathbf{v}_h^{k-1}$ and $\mathbf{v}^{**} = \mathbf{v}_{\text{int}}^k$	results in a semi-implicit method,
$\mathbf{v}^* = \mathbf{v}^{**} = \mathbf{v}_{\text{int}}^k$	results in implicit Euler method.

The second step determines the solution $[\mathbf{v}_h^k, p_h^k]$ by projection of the intermediate velocity $\mathbf{v}_{\text{int}}^k$ solving

$$\begin{aligned} (\mathbf{v}_h^k, \varphi_h) + \Delta t\ b(\varphi_h, p_h^k) &= (\mathbf{v}_{\text{int}}^k, \varphi_h) && \forall \varphi_h \in \mathbf{V}_h, \\ b(\mathbf{v}_h^k, \xi_h) &= 0 && \forall \xi_h \in S_h. \end{aligned} \tag{3.28}$$

We see that the resulting problem for the second step has the structure of the Stokes problem and that the arising matrix equals the one in (3.25). If we now take the explicit version of first step, i.e. $\mathbf{v}^* = \mathbf{v}^{**} = \mathbf{v}_h^{k-1}$, we can directly insert the term for the intermediate velocity $(\mathbf{v}_{\text{int}}^k, \varphi_h)$ from (3.27) into (3.28) and find the proposed explicit timestepping scheme (3.26).

In contrast to the classical two step Chorin-Temam scheme, we avoid replacing the second step by a reformulation as *pressure Poisson-equation.* This can be done by using the incompressibility-condition $\nabla \cdot \mathbf{v}^{k+1} = 0$ and applying the divergence operator to the conservation of mass equation, yielding the Poisson problem for the pressure

$$\Delta p^{k+1} = \frac{1}{\Delta t} \nabla \cdot \mathbf{v}_{\text{int}}^{k+1} \qquad \text{in } \Omega.$$

This equation now needs to be equipped with boundary conditions for the pressure variable, which are not stated in the original setting and therefore lead to controversy about the right choice. For an extensive discussion to the pressure Poisson equation especially concerning the boundary conditions, we refer to [66]. The controversy about non-physical boundary conditions for the pressure is a potential drawback of the Chorin-Temam projection method, which can be prevented by using the proposed scheme (3.25).

Nevertheless the advantage of explicit schemes is the fact that a kind of constant *iteration matrix* can be stated for each timestep $t_k \to t_{k+1}$. In the case of parabolic differential equations without constraints (like the continuity equation) this matrix simply turns out to be the mass matrix. An effective way to approximate the mass matrix by a diagonal matrix is known as *lumping*. The mass-matrix $M = m_{ij} = (\varphi_h^j, \varphi_h^i)$ is replaced by the diagonal matrix $\overline{M}$ with $\overline{m}_{ii} = \sum_{j=1}^{n} m_{ij}$. It can be shown that this approximation can be interpreted as a quadrature rule only using function values on Lagrangian nodes - see [32, 118] for details and error approximations. Using the lumped mass-matrix obviously results in an explicit timestepping scheme, which is completely free of the need to invert any matrix. Hence, the scheme will possess ideal characteristics for parallel computation as we show in Chapter 5.2 for the optimisation of an instationary scalar-valued problem. A final remark on explicit timestepping is inevitable: compared to the implicit schemes, one will always be faced with a stepsize restriction of the form $\Delta t \leq Ch^2$ to ensure stability.

Fractional Step θ Scheme

Besides these classical one-step time discretisation schemes, the formulation (3.24) allows also for the *fractional step θ-scheme*, which bases on three substeps with suitable chosen parameter α_i - cf. Table 3.1. In contrast to the Euler schemes above, the fractional step θ-scheme uses three steps (with $n(\mathbf{v}_h^k, \varphi_h) = a(\mathbf{v}_h^k, \varphi_h) + c(\mathbf{v}_h^k, \mathbf{v}_h^k, \varphi_h)$):

step1

$$\begin{aligned}(\mathbf{v}_h^{k-1+\theta}, \varphi_h) + \beta\theta\Delta t\ n(\mathbf{v}_h^{k-1+\theta}, \varphi_h) + \theta\Delta t\ b(\varphi_h, p_h^{k-1+\theta}) &= \\ (\mathbf{v}_h^{k}, \varphi_h) - \gamma\theta\Delta t\ n(\mathbf{v}_h^{k-1}, \varphi_h) + \theta\Delta t(\mathbf{f}^{k-1}, \varphi_h) & \qquad \forall \varphi_h \in \mathbf{V}_h, \\ b(\mathbf{v}_h^{k-1+\theta}, \xi_h) = 0 & \qquad \forall \xi_h \in S_h.\end{aligned}$$

step2

$$\begin{aligned}(\mathbf{v}_h^{k-\theta}, \varphi_h) + \gamma\tilde{\theta}\Delta t\ n(\mathbf{v}_h^{k-\theta}, \varphi_h) + \tilde{\theta}\Delta t\ b(\varphi_h, p_h^{k-\theta}) &= \\ (\mathbf{v}_h^{k-1+\theta}, \varphi_h) - \beta\tilde{\theta}\Delta t\ n(\mathbf{v}_h^{k-1+\theta}, \varphi_h) + \tilde{\theta}\Delta t(\mathbf{f}^{k}, \varphi_h) & \qquad \forall \varphi_h \in \mathbf{V}_h, \\ b(\mathbf{v}_h^{k-\theta}, \xi_h) = 0 & \qquad \forall \xi_h \in S_h.\end{aligned}$$

step3

$$\begin{aligned}(\mathbf{v}_h^{k}, \varphi_h) + \beta\theta\Delta t\ n(\mathbf{v}_h^{k}, \varphi_h) + \theta\Delta t\ b(\varphi_h, p_h^{k}) &= \\ (\mathbf{v}_h^{k-\theta}, \varphi_h) - \gamma\theta\Delta t\ n(\mathbf{v}_h^{k-\theta}, \varphi_h) + \theta\Delta t(\mathbf{f}^{k-1}, \varphi_h) & \qquad \forall \varphi_h \in \mathbf{V}_h, \\ b(\mathbf{v}_h^{k}, \xi_h) = 0 & \qquad \forall \xi_h \in S_h.\end{aligned}$$

Originally, the scheme was proposed as an operator splitting method [25, 64, 63] separating the nonlinear convective term and the incompressibility condition. One full timestep Δt in the fractional step θ-scheme consists of a cycle of three substeps $t_{k-1} \to t_{k-1+\theta} \to t_{k-\theta} \to t_k$ and each substep is approximately of the same computational complexity as one step of the standard Crank-Nicolson-scheme. To see this we can write all three steps as nonlinear algebraic system

$$\begin{pmatrix} M + \alpha_1 [A + C(v^k)] & \alpha_2 B^T \\ B & 0 \end{pmatrix} \begin{pmatrix} v^k \\ p^k \end{pmatrix} = \begin{pmatrix} M v^{k-1} - \alpha_3 [A + C(v^{k-1})] v^{k-1} + \alpha_4 F^{k-1} + \alpha_5 F^k \\ 0 \end{pmatrix}.$$

As indicated by the choice of parameter α_i, both Crank-Nicolson-scheme and fractional step θ-scheme have $\alpha_1 \neq 0$ and $\alpha_2 \neq 0$ resulting in the same complexity for solving a nonlinear system. We want to end this section by a brief overview on analytical results concerning the fractional step θ-scheme. Following the standard approach [26] of stability analysis for numerical methods for

ODEs, we use the scalar-valued model for $t \in (0, T)$

$$\frac{du(t)}{dt} = \lambda u(t) \text{ and } u(0) = u_0$$

with solution $u(t) = e^{\lambda t} u_0$. Now the iteration $u^k = R_\theta(\Delta t \lambda) u^{k-1}$, $x = \Delta t \lambda$ yields for the fractional step θ-scheme a rational approximation to the exponential function, i.e. the *stability function*

$$R_\theta(x) = \frac{(1 + \beta\tilde{\theta}x)(1 + \gamma\theta x)^2}{(1 - \beta\theta x)^2(1 - \gamma\tilde{\theta}x)} = e^x + O(x^3).$$

The parameters are to be chosen such that $\theta \in [1/2, 1)$, $\tilde{\theta} = 1 - 2\theta$, $\beta \in [0, 1]$ and $\gamma = 1 - \beta$. For $\theta = 1 - 1/\sqrt{2}$ the scheme has second order accuracy and for $\beta > 1/2$ it is strongly A-stable - cf. [97, 117] for a proof. If one additionally chooses $\beta = (1 - 2\theta)/(1 - \theta)$ we have that $\beta\theta = \gamma\tilde{\theta}$ and hence α_1 identically for each substep, which means that the diffusion and convection operators can be treated in the same way in each substep. Compared to the previous mentioned schemes, the fractional step θ-scheme combines the advantages of implicit Euler and Crank-Nicolson-scheme, namely the second order accuracy and (by strong A-stability) the full smoothing property, which is essential in the context of rough initial or boundary data. Supplementary to these theoretical results, the fractional step θ-scheme also showed very good numerical properties [90].

3.2 Iterative Solver and Preconditioner

As shown in the previous section we use Newton's method to solve the nonlinear system arising from finite element based discretisation of the Navier-Stokes equations. In recent years especially for linear systems arising from the field of partial differential equations preconditioned Krylov subspace methods have made good progress to be competitive with sparse direct solvers. As the size of the linear system is becoming larger (due to refinement of the mesh $\mathcal{T}_h$) and its nonzero structure is becoming denser (due to more couplings in 3D discretisations and unstructured meshes) iterative methods are even superior to direct solvers in the sense of computational time and memory costs. Another advantage of iterative solvers compared with direct solvers is the possibility to control the error and therefore the effort of the solver. Since the discretisation of the underlying partial differential equation is given with an inherent discretisation error it is of minor importance to solve the arising linear system with higher accuracy than this error. Furthermore, the linear systems have to be solved in an outer/Newton loop which also allows for an inexact inner solver with possible adaptive convergence criteria. Nevertheless, it is well known that the convergence behaviour of an iterative Krylov solver strongly depends on an adequate use of preconditioners. For these reasons in the sequel we will present a multilevel ILU-based preconditioner for the iterative linear solver and also study some adaptive convergence criteria for the linear solver within Newton's method.

3.2.1 Multilevel ILU Preconditioner

Discretisation and linearisation of the Navier-Stokes equations lead to a linear saddle point problem of the abstract form - see (3.19)

$$\underbrace{\begin{pmatrix} A + L(v^k) & B^T \\ B & 0 \end{pmatrix}}_{A} \underbrace{\begin{pmatrix} \delta v \\ \delta p \end{pmatrix}}_{x} = \underbrace{\text{res}(\begin{pmatrix} v^k \\ p^k \end{pmatrix})}_{b}, \tag{3.29}$$

which has to be solved in each step of an iterative nonlinear solver, e.g. Newton's method (see Algorithm 2). A similar system also arises for the instationary equations, where only the terms $A + L(v^k)$ and B^T have to be modified. In this subsection we present an ILU-based preconditioner for iterative solvers of linear systems of the form (3.29). If not stated differently the preconditioner is incorporated into a right preconditioned restarted GMRES solver for the solution of the linear system $Ax = b$ to generalise notation[2] (see [65, 95, 134] for the algorithmic description and [46] for

[2]we will use the notation A for the entire system matrix throughout this section

a theoretical analysis). For the preconditioning step $v_{i+1} = AMv_i$ within Arnoldi-process and for the update/restart $x_k = x_0 + M_k V y_k$, we choose the matrix M to be a multilevel incomplete LDU-factorisation of the linearised system-matrix A as described in the sequel. Hence, the preconditioner M can be applied efficiently by using back substitution to solve linear systems having triangular coefficient matrices. Before going into the specifics, we summarise the approach.

First, we apply *preprocessing* to make the coefficient matrix A more suitable for incomplete factorisation. To this end, several approaches were considered. The first possibility is to normalise A, i.e. to simply scale columns and subsequently the rows of A to norm 1, which is very cheap to calculate and results in some cases in a better balanced matrix. The second possibility considered was to first normalise A and then to apply the standard PQ-reordering [135] to aim at making an initial block of A more suitable for incomplete LDU-factorisation by improving diagonal dominance and sparsity. A third possibility under consideration was to permute the rows of A and to scale both the rows and columns so that A is transformed into an I-matrix, i.e. a matrix having elements of absolute value of 1 on the diagonal and elements of at most absolute value of one elsewhere. Heuristically, it is clear that I-matrices should be more suitable for (incomplete) LDU-factorisations than general matrices and this has also been confirmed experimentally, see [15]. Note furthermore that I-matrices are preserved if rows and columns are permuted using the same permutation. Hence, also a permutation can be determined next such that an initial block of the coefficient matrix has better diagonal dominance and sparsity properties as described in [114].

One result should be mentioned in advance: normalisation of A appeared to perform better for matrices arising from the discretisation of 3D problems, while for the 2D case PQ-reordering got better performance - preprocessing to an I-matrix was not suitable at all. Especially generation of I-matrix with additional permutation, took relatively long computational time or needed large fill-in not to drop entire rows, which often led to arithmetic over- and/or underflows. For these reason we will concentrate on normalisation and PQ-reordering of A for the Navier-Stokes solver in further research and therefore only PQ-reordering will be explained in detail.

After applying the permutation, we compute an *incomplete LDU-factorisation* based on Crout's implementation of Gaussian elimination and threshold based dropping to preserve sparsity. If a pivot having an absolute value of less than `min_pivot` (usually 0.01) encounters while the elimination process, we terminate the regular factorisation phase aimed at obtaining triangular factors L and U and proceed instead in calculating an approximate Schur complement S. This completes the first level of the multilevel factorisation.

Setting $A = S$, we proceed recursively and obtain further levels until the coefficient matrix has been factored entirely. The whole preconditioning process is summarised in the flowchart 3.3. We need to emphasise that at the beginning of each level, we preprocess the new matrix just as the original coefficient matrix and that the possibility of using preprocessing between levels distinguishes this approach from single-level factorisations.

Preprocessing by PQ-Reordering

For each index $k = 1, \dots, n$, n being the dimension of A, we determined a weight w_k by

$$w_k = \frac{1}{\text{nnz}(A_{k,:})} \cdot \frac{\max\limits_{j=1,\dots,n} |A_{k,j}|}{||A_{k,:}||_1}.$$

Here, $A_{k,:}$ refers to the kth row of A. Furthermore, nnz denotes the number of non-zero elements of a vector and $||\cdot||_1$ the 1-norm. Clearly, large values for w_k indicate that the kth row is fairly sparse and/or has an element which is fairly large in absolute value when compared to the remaining elements of that row. Hence, we select the row for which w_k is largest to be the first row of the permuted matrix and we permute the column of A containing largest element by absolute value of the row selected onto the diagonal. We proceed with the remaining rows analogously. In other words, we choose the row for which w_k is second largest to be the second row of the permuted matrix and permute columns so that the largest element by absolute values is again moved onto the diagonal provided that this is possible. It certainly is possible that the largest element by absolute value of a particular row is in a column which has already been selected and permuted in a previous step. In this case, we cannot move the largest element by absolute value of this row onto

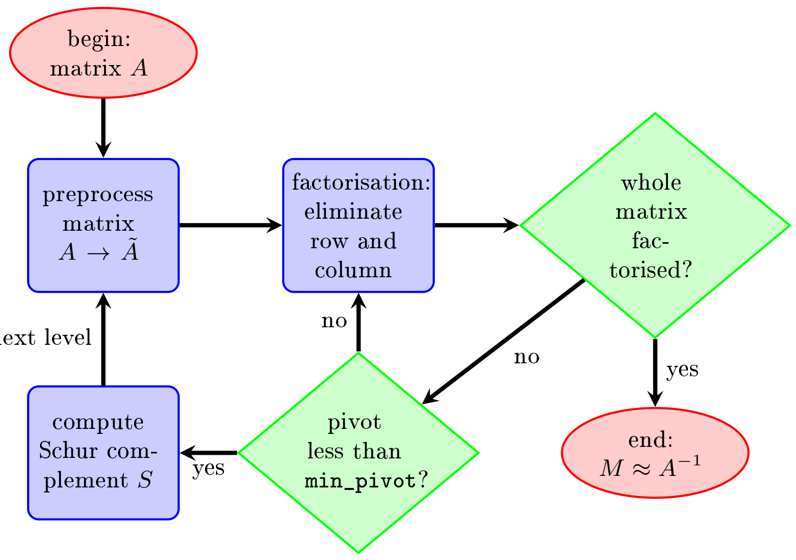

begin: matrix A
preprocess matrix $A \to \tilde{A}$
factorisation: eliminate row and column
whole matrix fac- torised?
yes
end: $M \approx A^{-1}$
no
no
pivot less than `min_pivot`?
yes
compute Schur com- plement S
$A = S$, next level

Figure 3.3: Multilevel preconditioning process.

the diagonal. Hence, we reject this row and move it to a high index arbitrarily. In this manner, we obtain a new matrix which has better properties for a large initial block and a final block of rejected rows. For details and other approaches see [135].

Multilevel Factorisations

For simplicity, we begin by describing a complete multilevel LDU factorisation as developed in [23] and [135]. We split the preprocessed square coefficient matrix $\tilde{A}$ into a block matrix

$$\tilde{A} = \begin{pmatrix} B & F \\ E & C \end{pmatrix}$$

such that the diagonal blocks B and C are square matrices. Next, we calculate an LDU factorisation $B = L_B D_B U_B$ of B and obtain

$$\begin{pmatrix} B & F \\ E & C \end{pmatrix} = \begin{pmatrix} L_B & 0 \\ E_B & I \end{pmatrix} \begin{pmatrix} D_B & 0 \\ 0 & S \end{pmatrix} \begin{pmatrix} U_B & F_B \\ 0 & I \end{pmatrix}.$$

Thus, L_B is a unit lower triangular, U_B is a unit upper triangular and D_B is a diagonal matrix. The matrices E_B and F_B are formally given by $E_B = EU_B^{-1}D_B^{-1}$ and $F_B = D_B^{-1}L_B^{-1}F$ and $S = C - E_B D_B F_B$ denotes the Schur complement. However, in practice, these matrices are calculated by Gaussian elimination and not by the formulas above. Next, we proceed recursively by setting $A = S$, preprocess to obtain a new $\tilde{A}$ and factoring once more. After a certain number of levels, we finish by completely factoring the final Schur complement.

Note that in practice, the block structure of $\tilde{A}$ does not need to be determined in advance. It is possible to begin factorisation, to terminate whenever this seems to be a good idea and to proceed in calculating the Schur complement. Hence, the block structure is determined during the course of factorisation. For the results in this work, we terminated the level whenever the absolute value of the pivot was less than 0.01.

The actual factorisation used is based on Crout's implementation of Gaussian elimination in the form presented in [103]. To use this factorisation, the matrix A needs to be available in either compressed sparse row or column format, see [133]. In a Crout factorisation, L is calculated by columns and U by rows, so that the former is stored naturally in compressed sparse column format and the latter in compressed sparse row format. However, during the kth step of elimination, we need to access both the kth row and column of A, L and U as shown in Algorithm 1. Obviously, as each matrix is only available in one format, accessing them by both by rows and columns is

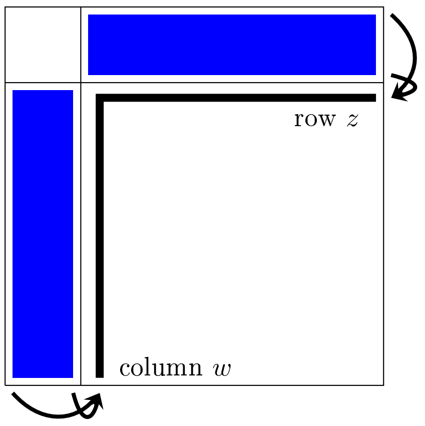


Figure 3.4: Computational pattern for Crout factorisation at k-th step where k-th row of U (denoted by z) and k-th column of L (denoted by w) are computed by means of the entries in blue area.

not possible in this form. However, rows and columns are accessed in their natural order, so it is possible to implement some additional, rather intricate, data structures of dimension $3n$ for each matrix to solve this problem, see [103] for details.

Algorithm 1 ILUC - Crout version of ILU.

```
for $k = 1, ..., n$ do
  initialise row $z$ : $z_{1:k-1} = 0,\ z_{k:n} = a_{k,k:n}$
  for $i = 1, ..., k-1$ and $l_{ki} \neq 0$ do
    $z_{k:n} = z_{k:n} - l_{ki} u_{i,k:n}$
  end for
  initialise column $w$ : $w_{1:k} = 0,\ w_{k+1:n} = a_{k+1:n,k}$
  for $i = 1, ..., k-1$ and $u_{ik} \neq 0$ do
    $w_{k+1:n} = w_{k+1:n} - u_{ik} l_{k+1:n,i}$
  end for
  apply dropping rule to row $z$ and column $w$
  $u_{k,:} = z$
  $l_{:,k} = w/u_{kk},\ l_{kk} = 1$
end for
```

The Dropping Rule

Although a larger number of different dropping rules are available to ensure sparsity such as the standard dual threshold strategy, see [133], or the inverse-based approach, see [22] and [103], we chose to use a technique which attempts to minimise the propagation of errors during the course of factorisation, see [113]. This technique is somewhat better than the standard strategy and as good as the inverse-based strategy for most problems. However, the inverse-based strategy requires more computational time, so we used the error-based approach. Let w denote the kth column of L, z^T the kth row of U and d the pivot being calculated in the kth step of elimination (see Figure 3.4 for the computational pattern). In subsequent elimination steps, the Schur complement would be modified by the matrix wdz^T if no dropping were performed. Hence, the error made in dropping element w_i in w can be estimated by $|w_i| \cdot |d| \cdot ||z||_1$. The error-based strategy drops w_i if

$$|w_i| \cdot ||z||_1 < \tau$$

holds for a given (fixed) threshold $\tau \geq 0$. Actually, it would seem more natural to drop w_i if $|w_i| \cdot |d| \cdot ||z||_1 < \tau$, however this would result in many elements being dropped whenever $|d|$ is very

Parameter	description	recommend/used value
`Threshold`	dropping tolerance $\tau = 10^{-t}$	$t \in [1, 3]$
`min_pivot`	tolerance for termination of one level	0.01
`Preprocessing_Type`	different orderings of initial matrix	0 = normalisation of A, 1 = PQ-reordering, several other I-matrix based reorderings and permutations are available
`Preconditioner_Number`	type of incomplete factorisation	1010 for multilevel ILUC factorisation
`max_levels`	maximum number of levels	10

Table 3.2: Parameters for the multilevel ILU preconditioner by ILU++.

small. Experimentally, this appears to be harmful. Similarly, an element z_i is dropped only if

$$||w||_1 \cdot |z_i| < \tau.$$

Implementation Details

The whole framework of multilevel preconditioner described above is given by the object-oriented `C++` software package `ILU++` [112]. The interface is very simple and the user can compute a solution of a linear system in a single step. The call

```
solve_with_multilevel_preconditioner(ROW,val,ja,ia,...,param)
```

computes the preconditioner, applies a linear solver and returns the solution. It is assumed that the coefficient matrix A is stored in compressed sparse row format. `val`, `ja` and `ia` are standard template library vectors containing the adjacency structure and values of A. A similar call can be used if the data are stored in traditional C-style arrays or for matrices in compressed sparse column format. `param` is an object containing the actual parameters used for setting up the preconditioner.

However, for our tests, we used an alternative which allowed us to incorporate the `ILU++` multilevel preconditioner in several iterative solvers. This approach required two steps to setup the preconditioner and single call to apply the preconditioner to a vector. First, we need to declare the preconditioner with the call

```
iluplusplus::multilevel_preconditioner Pr.
```

Next, we setup the preconditioner for a matrix A stored in compressed sparse row format by calling

```
Pr.setup(val, ja, ia, ROW, param).
```

The parameters of `ILU++` preconditioner, as indicated in the Table 3.2, can be set in the `param` field. After setting up the preconditioner, it may be applied to a standard template library vector `v` by calling

```
Pr.apply_preconditioner(v).
```

Again, a similar call exists if `v` is a C-style array.

3.2.2 Numerical Results for the Multilevel ILU Preconditioner

All calculations were done by the finite element software `HiFlow`[2] with an interface to the preconditioner `ILU++`, compiled with the GNU gcc compiler in version 4.4.3. We used a compute-server with Intel Xeon X5540 CPU at 2.53 GHz and 6 GB of main memory for all 2D calculations, whereas for the 3D calculations a main memory of 12 GB was used. Here we show the influence of different thresholds on the solution time and memory costs (fill-in ratio) for different problems of channel flows in 2D and 3D as presented in Chapter 1. Instead of solving Navier-Stokes equations,

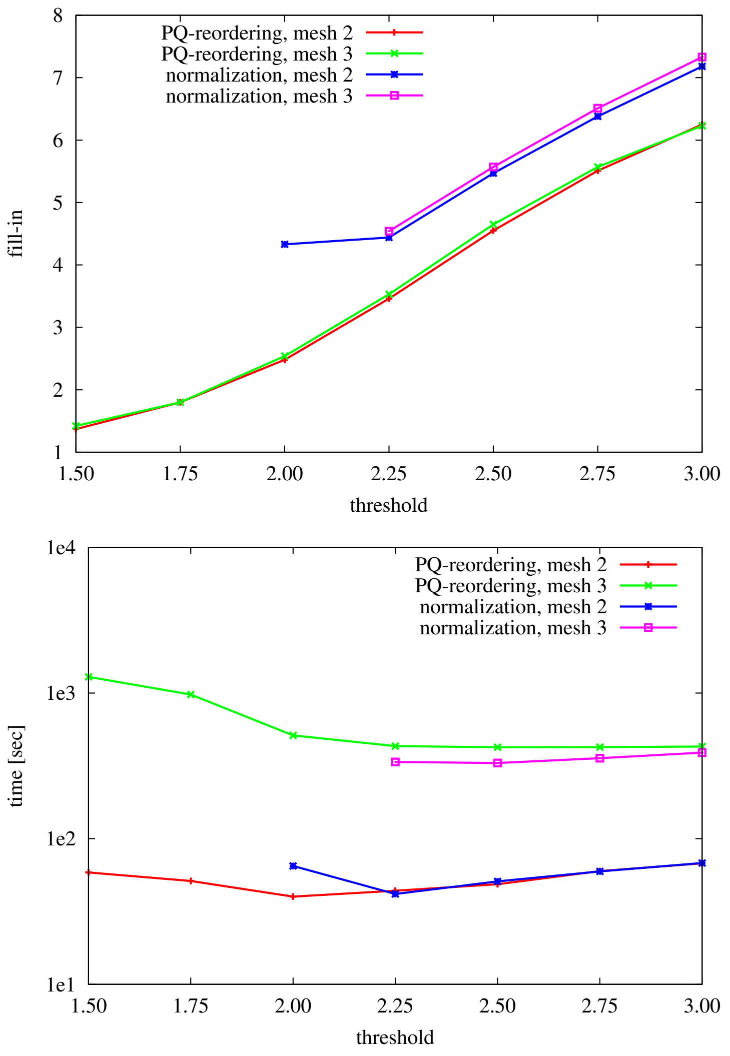


Figure 3.5: Results for the multilevel preconditioner with normalisation and PQ-reordering in the 2D case of backflow geometry using different threshold.

we first choose to solve only Stokes equations at Reynolds number 42.0 to get rid of influences of the nonlinear Newton-solver on the linear solver. Furthermore, we compared the behaviour of the solver when different preprocessing routines, i.e. normalisation and PQ-reordering, were performed on the initial matrix A. For all tests (except for the direct solver) we terminated the linear solver after 5000 GMRES-steps (with restart each 30 steps) or when the residual in k-th steps, i.e. $r_k = b - Ax_k$, fulfils

$$\|r_k\|_2 < \max(10^{-9} \cdot \|r_0\|_2, 10^{-12}).$$

The 2D Case

In the 2D case suitable preconditioners (with respect to fast computational times) were achieved with normalisation and PQ-reordering - but it should be mentioned that the PQ-reordering produces more stable solvers in the sense of arithmetic overflows. As shown in Figure 3.5 for the backflow geometry a mesh dependent threshold $t \in [2.0, 2.5]$ results in best solution time. While the number of GMRES-steps decreases the bigger the threshold is chosen, the fill-in obviously increases, where we observe a slightly higher fill-in for the normalisation compared to PQ-reordering. Dependence on the mesh, i.e. on the size of the system matrix, can be explained heuristically: at small sizes the matrix-vector multiplication is obviously fast, so that additional costs for setup of the preconditioner and additional elements within the preconditioner weight at lot. The bigger

Geometry	system matrix		`GMRES/ILU++`		`GMRES/ILU(0)`		`UMFPACK`
	size	nnz	steps	time	steps	time	time
2Dbackflow	32243	1290969	148	3.47 sec	>5000	>83 sec	0.88 sec
2Dbackflow	127843	5151529	357	40 sec	>5000	>351 sec	6.34 sec
2Dbackflow	509123	20581449	328	331 sec	>5000	>1366 sec	53 sec
2Dmeander	15443	595449	51	0.72 sec	>5000	>37 sec	0.23 sec
2Dmeander	59683	2358889	200	8.24 sec	>5000	>158 sec	1.24 sec
2Dmeander	234563	9389769	158	93.6 sec	>5000	>711 sec	7.5 sec

Table 3.3: Comparison of iterative and direct solver for the 2D case. For the GMRES results the best/fastest version of ILU++ preconditioner is used.

the matrix size the more time is spend on a simple matrix-vector multiplication and therefore a denser preconditioner is of smaller consequence for the total solver time. This observations holds true until the threshold is such high that the setup of preconditioner dominates the whole solution process. For the multilevel approach while setting up the preconditioner almost always just two or three levels are used and in the case of normalisation of A we even restrict to only one level, since after a small pivot occurs the Schur complement matrix at next level would not be permuted (just normalised) and therefore the matrix would not be in a better shape.

Finally for matrices arising from 2D-discretisations a comparison to a direct solver like `UMFPACK` [38] shows disadvantage for preconditioned iterative solvers if the absolute computational time for the linear solver is of importance - see Table 3.3 for details. Only if in addition the storage has strict limitations an iterative solver will be superior to a direct solver. Nevertheless, compared to one of the most often used preconditioners, namely ILU(0) decomposition of the system matrix, we achieve much better results since the GMRES solver with ILU(0) preconditioner throughout needed more than 5000 steps and a dedicated pivoting method due to the lower right zero-block in the stiffness matrix.

Remark 3.2.1
Besides the pure algebraic observations on solver time and fill-in, a noticeable difference for the solver can be observed, when boundary conditions are changed. In Figure 3.6 we compared the pressure-drop boundary condition to the do-nothing condition (as introduced in Chapter 2), and noticed a better behaviour for the pressure-drop condition. Thus, also the algebraic solver seems to be influenced by a more physical choice.

The 3D Case

In the 3D case we achieve quite the same results. For system matrices A as shown in Table 3.5, we found that normalisation of A results in a faster preconditioner than PQ-reordering, which is obvious since no permutation is done. Compared to the 2D case, where normalisation of A often led to arithmetic under- or overflow, the structure of 3D matrices seems to be more robust in that sense. Furthermore, the fill-in is much lower, since the entries of the matrix A are of order $O(h^3)$ instead of $O(h^2)$, with h being the size of a mesh-cell. However, a threshold about 2.0 is also enough to achieve good, i.e. fast, preconditioners. Comparisons with a direct solver in the 3D case, as shown in Table 3.5, shows the advantage of preconditioned iterative solvers over direct methods. The latter takes much more computational time or even ran out of memory, which is given due to more couplings of degrees of freedom in the 3D case and therefore a denser discretisation matrix, for which the factorisation is harder to compute and results in more fill-in. We compared the fill-in for L, U matrices of the `UMFPACK` solver and L, D, U matrices of the `ILU++` preconditioner in both 2D and 3D case of backflow geometry (cf. Table 3.4) to underline the difference of 2D and 3D discretisation matrices.

Main conclusions one can draw for the proposed multilevel preconditioner in the 3D case (we only report the results of the backflow geometry - see Figure 3.7 - these coincide with the ones for meander geometry):

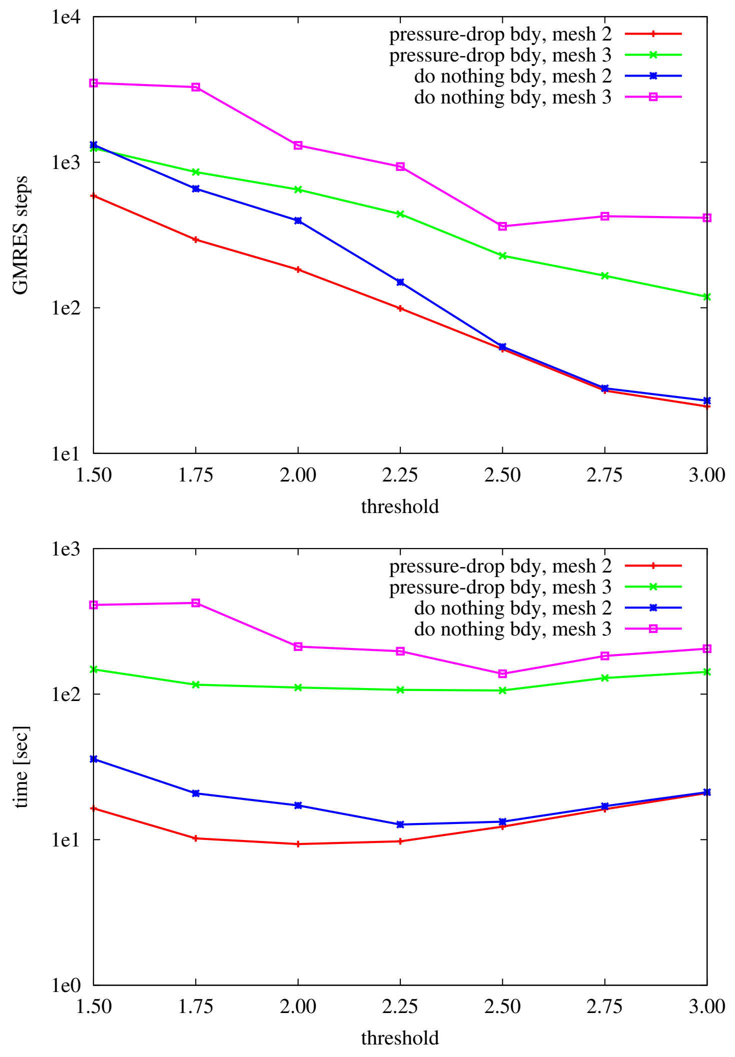


Figure 3.6: Results for the multilevel preconditioner with PQ-reordering in the 2D case of meander geometry using different threshold. Comparison of different boundary-conditions used for discretisation.

Geometry	solver/precond	size of A	nnz(A)	nnz(LU)	fill-in
2Dbackflow	`UMFPACK`	32243	1290969	7494013	5.81
2Dbackflow	`ILU++` $(t = 2.0)$	32243	1290969	3179869	2.46
3Dbackflow	`UMFPACK`	62344	12408076	111783808	9.01
3Dbackflow	`ILU++` $(t = 2.0)$	62344	12408076	6908249	0.56

Table 3.4: Fill-in for direct solver UMFPACK and ILU++ preconditioner.

Geometry	system matrix			GMRES/ILU++		UMFPACK
	size	nnz	storage	steps	time	time
3Dbackflow	8666	1595386	18.4 MB	20	0.37 sec	0.92 sec
3Dbackflow	62344	12408076	143 MB	32	5.49 sec	41.6 sec
3Dbackflow	472748	97872616	1130 MB	93	116 sec	out of memory
3Dmeander	8034	1232626	14.2 MB	13	0.22 sec	0.31 sec
3Dmeander	51368	9227116	106 MB	27	3.32 sec	5.44 sec
3Dmeander	363948	71349736	821 MB	95	84.3 sec	274 sec

Table 3.5: Numerical settings of the preconditioner tests on 3D discretisations. Additionally the results of the iterative solver is given compared to a direct solver.

- allowing slightly more fill-in (by increasing the threshold) results in a major decrease of GMRES-iterations,
- fill-in for the preconditioner is nearly negligible for thresholds beneath 2.5,
- time for solving the linear system depends directly on the threshold - more threshold results in lower solver-times but more time is spent for setup of the preconditioner, which gives higher overall solution-time,
- optimal threshold in the sense of overall solution-time is in the range of $[1.5, 2.25]$ depending on the size of the matrix, i.e. the level of mesh refinement, which is less than in the 2D case,
- while using approximately the same computational time, PQ-reordering allows less fill-in but uses more GMRES-steps compared to normalisation of A,
- for the PQ-reordering of A in most cases 2 or 3 levels of the multilevel approach are enough, while for normalisation of A restriction to one level is chosen as mentioned above.

So we conclude that a multilevel preconditioner based on preprocessing and ILU-decomposition with variable threshold dropping rule is superior to a direct solver for the discretisation of 3D Stokes and Navier-Stokes equations. For almost all testcases a threshold of $\tau = 10^{-t}$ with $t \in [1.5, 2.25]$ achieved the best performance in the sense of overall computational time for the linear solver and simultaneously uses a reasonable fill-in.

Remark 3.2.2
All results were given on the basis of discretisation of Stokes equations, such that Newton's method only needs one single step. However, the results were the same for the full Navier-Stokes equations, when the average of all Newton-steps is taken - see Figure 3.8. We almost always needed 6 Newton steps to get the desired relative residual of 10^{-9} *for the nonlinear equation, where the stopping criteria for the linear solver in each step are chosen as before - this turns out to be not that good idea, as we will see below.*

Outlook on Parallel Solver Using Multilevel ILU Preconditioner

The presented results are just based on sequential solver. For finer meshes and especially for 3D discretisations the number of degrees of freedom easily grows up to an order of 10^6, which prohibits the use of sequential solver for reasons of computational time and memory costs. A possible way to use multilevel ILU preconditioners in the framework of parallel solvers will be given in Chapter 4 where we present two approaches:

1. parallel block Jacobi preconditioner with local multilevel ILU decomposition as shown in this section,
2. inexact parallel Schur complement solver based on multilevel ILU decomposition as preconditioner for global iterative method.

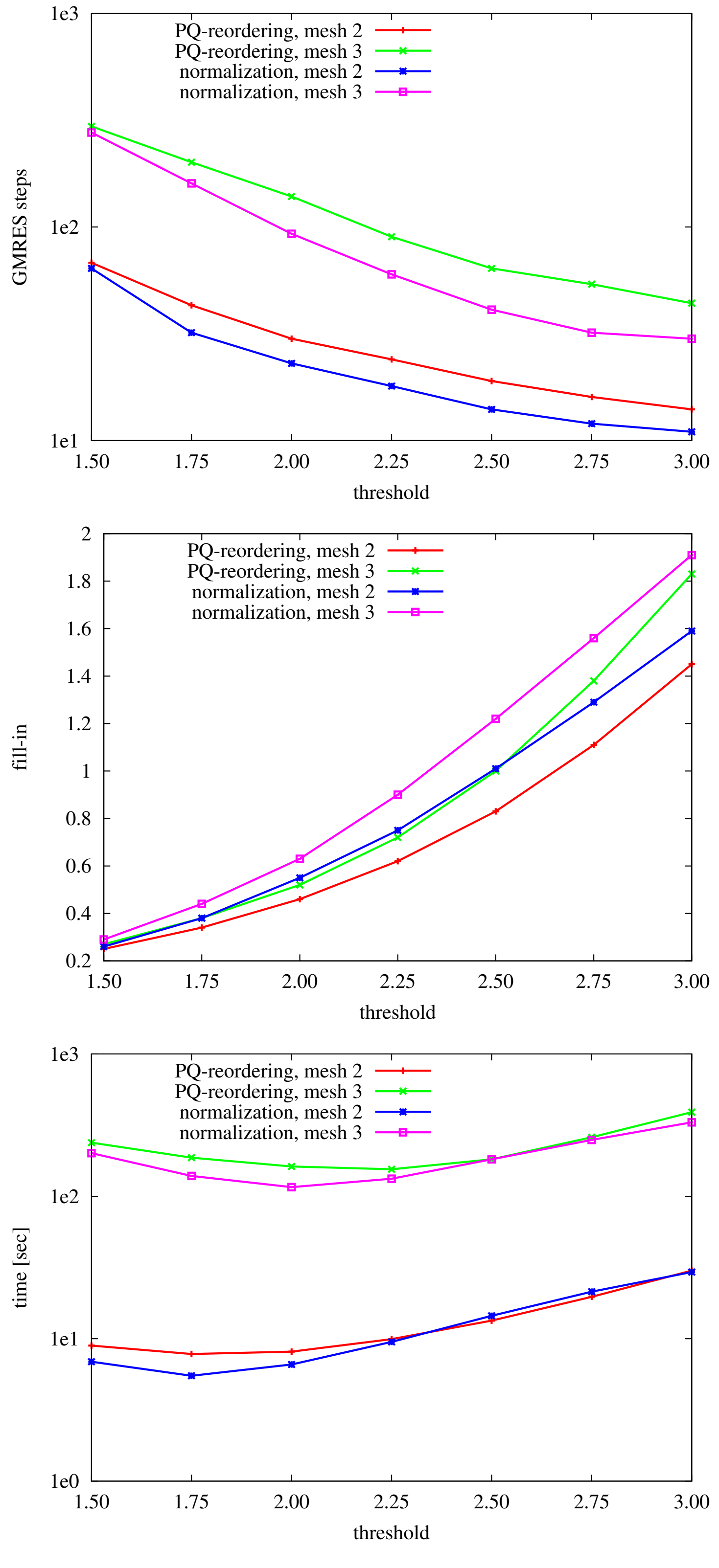


Figure 3.7: Results for the multilevel preconditioner with normalisation and PQ-reordering in the 3D case of backflow geometry using different threshold.

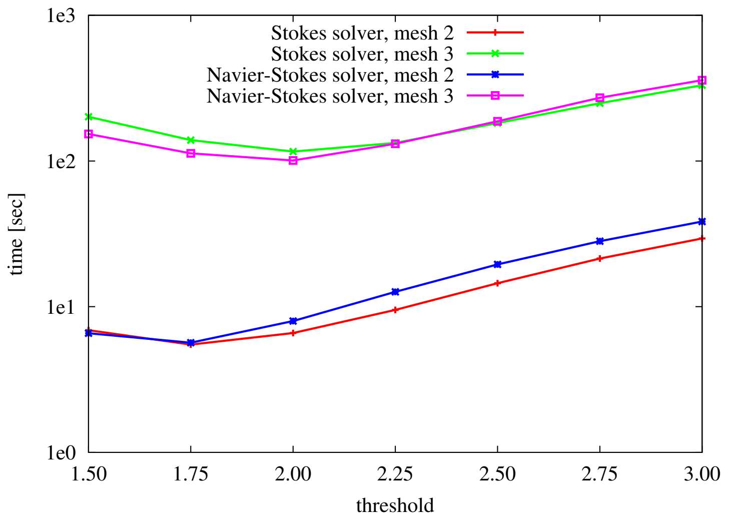


Figure 3.8: Results for the multilevel preconditioner with normalisation of A in the 3D case of backflow geometry using different threshold. Comparison of mean value for all Newton-steps of Navier-Stokes solver to Stokes solver.

3.2.3 Inexact Newton Method with Adaptive Forcing Terms

Algorithm 2 Inexact damped Newton method.

```
let $x_0,\ \eta_{max} \in [0,1),\ t \in (0,1)$ be given; set $k = 0$
while $\|F(x_k)\| > \max(atol, rtol \cdot \|F(x_0)\|)$ and $k < maxits$ do
  choose forcing term $\eta_k \in [0, \eta_{max}]$ and $\delta x_k$ such that
      $\|F(x_k) + F'(x_k)\delta x_k\| \leq \eta_k \|F(x_k)\|$
  while $\|F(x_k + \delta x_k)\| > [1 - t(1 - \eta_k)]\|F(x_k)\|$ do
    choose damping parameter $\lambda \in [0, 1]$ e.g by Armijo Backtracking rule
    $\delta x_k = \lambda \delta x_k$ and $\eta_k = 1 - t(1 - \eta_k)$
  end while
  update solution $x_{k+1} = x_k + \delta x_k$; $k = k + 1$
end while
```

In the previous subsection we worked out results for linear solver only using Stokes equations as model problem. The results for Navier-Stokes equations are the same if the average of all Newton-steps is taken. Nevertheless, there remains some doubt if we solve nonlinear systems with Newton's method and do not use a direct solver in each step. The common convergence theory of Newton's method depends on the fact, that the resulting linear systems are solved exactly, i.e. we have the classical Newton iteration for the nonlinear equation $F(x) = 0$:

$$x_{k+1} = x_k + \Delta x_k, \text{ where } F'(x_k)\Delta x_k = -F(x_k). \tag{3.30}$$

If we now replace this iteration by:

$$x_{k+1} = x_k + \delta x_k, \text{ where } F'(x_k)\delta x_k = -F(x_k) + r_k, \ \frac{\|r_k\|}{\|F(x_k)\|} \leq \eta_k, \tag{3.31}$$

which means that we solve the linear system only up to a relative residual of η_k, the question arises which tolerance η_k must be reached by the linear solver not to destroy the convergence properties of Newton's method.

In standard literature this second type of Newton iteration is called *inexact Newton method* since the arising linear systems are no longer solved exactly. By using iterative methods like

GMRES we now have to distinguish between an outer iteration (the Newton step) and an inner iteration (the GMRES step) and have to answer the questions how the inner iteration can influence the convergence of outer iteration and how the outer iteration might lead to adaptive convergence criteria for the inner iteration. In what follows we will present two ways to tackle this problem - first we briefly recall the main results of convergence theory for inexact Newton methods (based on [39] and [45]), which give an insight on the influence of inner iteration to the outer one. Second we will present results on affine invariance studies of Newton's method (following the work of Deuflhard [42]) which allow to create adaptive convergence criteria for the inner iteration based on the actual status of outer iteration. Such an adaptive way to set the tolerances η_k, also called *forcing terms*, was already presented in earlier works, e.g. [5, 40, 44, 124], but these are more heuristically motivated.

Analysis of inexact Newton method used for numerical solution of finite dimensional nonlinear systems as shown in Algorithm 2 can be found in [95, 121]. If a sufficiently good initial guess x_0 is at hand, we can skip the globalisation and get the following local convergence result (see [39] for proof), if the forcing sequence $\{\eta_k\}$ is uniformly less than one.

Theorem 3.2.1 (Local convergence of inexact Newton method)
Let $F : D \subset \mathbb{R}^n \rightarrow \mathbb{R}^n$ be a nonlinear function with and let $x^* \in \mathbb{R}^n$ such that $F(x^*) = 0$. If F is continuously differentiable in a neighbourhood of x^*, $F'(x^*)$ is regular and $\eta_k \leq \eta_{max} < t < 1$, then there exists an $\varepsilon > 0$ such that, for $\|x_0 - x^*\| \leq \varepsilon$, the sequence of inexact Newton iterates $\{x_k\}$ converges to x^*. Moreover the convergence is linear in the sense that

$$\|F'(x^*)(x_{k+1} - x^*)\| \leq t\|F'(x^*)(x_k - x^*)\|.$$

The proof, like the classical convergence of Newton's method (Newton-Kantorovich or Newton-Mysovskii theorem), bases on two main assumptions, namely

$$\begin{aligned} &F'(x)^{-1} \text{ exists and is bounded } \|F'(x)^{-1}\| \leq \beta < \infty \text{ for } x \in D, \\ &F'(x) \text{ complies a Lipschitz condition } \|F'(x) - F'(y)\| \leq \gamma\|x - y\| \text{ for } x, y \in D. \end{aligned} \tag{3.32}$$

Furthermore, the condition of x_0 being sufficiently close to x^* can be characterised by means of the so-called *Kantorovich quantity*

$$h_0 := \|\Delta x_0\|\beta\gamma, \ \Delta x_0 = x_1 - x_0,$$

which is assumed to be sufficiently small. In general this quantity cannot be computed for practical implementations, such that one cannot guarantee an initial guess to be sufficiently good. Hence, globalisation techniques are introduced to assure convergence even for bad initial guess, like the proposed Armijo damping in Algorithm 2. For this we have the following convergence theorem given by Eisenstat and Walker [45]

Theorem 3.2.2 (Global convergence of inexact Newton method)
Assume that Algorithm 2 does not break down. If x^* is a limit point of $\{x_k\}$ such that $F'(x^*)$ is invertible, then $F(x^*) = 0$ and $x_k \rightarrow x^*$. Furthermore, initial δx_k and $\lambda = 1$ will pass damping step for all sufficiently large k.

The preceding theorems guarantee at least linear convergence of the global inexact Newton method with damping but do not point out a strategy for choosing the forcing terms η_k. This task is addressed in many papers, e.g. [5, 40, 44, 124], and some choices were already under investigation in the context of solving Navier-Stokes equations, see [140]. For any further studies we will use the following somewhat heuristic proposals by Eisenstat and Walker [44]

- choice 1: given $\eta_0 \in [0, 1)$, choose

$$\eta_k = \frac{\|F(x_k) - F(x_{k-1}) - F'(x_{k-1})\delta x_{k-1}\|}{\|F(x_{k-1})\|} \qquad \text{for } k = 1, 2, ... \tag{3.33}$$

- choice 2: given $\alpha \in (1,2]$, $\gamma \in [0,1]$ and $\eta_0 \in [0,1)$, choose

$$\eta_k = \gamma\Big(\frac{\|F(x_k)\|}{\|F(x_{k-1})\|}\Big)^{\alpha} \qquad \text{for } k = 1,2,... \tag{3.34}$$

and by Dembo and Steihaug [40]

- choice 3: $\eta_k = \min\{\frac{1}{k+2}, \|F(x_k)\|\}$ for $k = 1,2,...$
- constant: $\eta_k = 10^{-1}, 10^{-3}, 10^{-5}$ for all k.

These strategies were not based on convergence analysis rather on efficiency and accuracy matching properties of inner and outer iteration, but the convergence of the inexact Newton method can be proven. A different way is proposed by the *affine invariance approach* in [42]. Herein the classical Newton-Kantorovich or Newton-Mysovskii theorems are replaced by affine invariant versions and construction of adaptive Newton algorithms is done by the paradigm to "realise affine invariant computational estimates of affine invariant Lipschitz constants that are cheaply available in the course of the algorithms".

Since we use GMRES algorithm, we restrict the affine invariance theory to the special case of *affine contravariance*, which leads to results formulated in terms of residuals $F(x_k)$ preferable in the context of GMRES. Let therefore $B \in \mathbb{R}^{n,n}$ be an arbitrary regular matrix and consider the class of problems

$$G(y) = F(By) = 0, \; x = By.$$

Observe that the assumptions (3.32) can also be stated as

$$\|(F'(\bar{x}) - F'(x))(\bar{x} - x)\| \leq \gamma\|\bar{x} - x\|^2 = \gamma\|F'(x)^{-1}F'(x)(\bar{x} - x)\|^2 \leq \gamma\beta^2\|F'(x)(\bar{x} - x)\|^2,$$

where the Lipschitz constant $\omega = \gamma\beta^2$ is affine contravariant since both sides are independent of B

$$G'(y)(\bar{y} - y) = F'(x)B(\bar{y} - y) = F'(x)(\bar{x} - x).$$

The way to derive practical Newton methods with adaptive choice of forcing terms is now:

1. identify theoretical local Lipschitz constant ω such that $\omega = \sup\limits_{x,y,z \in D} g(x,y,z)$ with $g(x,y,z)$ containing only affine invariant terms,
2. define computational local estimates $[\omega] = g(\hat{x}, \hat{y}, \hat{z})$ for specific $\hat{x}, \hat{y}, \hat{z}$ and $[\omega] \leq \omega$.

We use the notation $[h]$ as introduced in [42] to describe an estimate of the theoretical value h. Without going into details we only present the main theorem for inexact Newton method with inner GMRES solver in the sense of affine invariance theory as proven in [42].

Theorem 3.2.3 (Convergence of inexact Newton method with inner GMRES)
Let $F : D \subset \mathbb{R}^n \to \mathbb{R}^n$, $F \in C^1(D)$, D convex. Denote by $x_0 \in D$ the initial guess for inexact Newton iteration (3.31)*. Assume the affine contravariant Lipschitz condition*

$$\|(F'(y) - F'(x))(y - x)\| \leq \omega\|F'(x)(y - x)\|^2 \quad \text{for } 0 \leq \omega < \infty, \; x, y \in D.$$

Let the level set $\mathcal{L}_0 := \{x \in \mathbb{R}^n : \; \|F(x)\| \leq \|F(x_0)\|\} \subseteq D$ be compact. For each iterate $x_k \in D$ define $h_k := \omega\|F(x_k)\|$. Then the residual norm of inexact Newton iteration can be bounded as

$$\|F(x_{k+1})\| \leq (\eta_k + \frac{1}{2}(1 - \eta_k^2)h_k)\|F(x_k)\|. \tag{3.35}$$

The convergence rate can be estimated as follows:

1. *Assume that the initial guess x_0 gives rise to $h_0 < 2$. Then some $\bar{\theta}$ in the range $\frac{h_0}{2} < \bar{\theta} < 1$ can be chosen. Let the inner GMRES iteration be controlled such that $\eta_k \leq \bar{\theta} - \frac{1}{2}h_k$. Then*

the inexact Newton iterates $\{x_k\}$ converge at least linearly to some solution $x^* \in \mathcal{L}_0$ at an estimated rate

$$\|F(x_{k+1})\| \leq \bar{\theta}\|F(x_k)\|.$$

2. *If, for some $\rho > 0$, the initial guess x_0 guarantees that $h_0 < \frac{2}{1+\rho}$ and the inner iteration is controlled such that*

$$\frac{\eta_k}{1-\eta_k^2} \leq \frac{1}{2}\rho h_k, \tag{3.36}$$

then the convergence is quadratic at an estimated rate

$$\|F(x_{k+1})\| \leq \frac{1}{2}\omega(1+\rho)(1-\eta_k^2)\|F(x_k)\|^2. \tag{3.37}$$

As for the classical convergence theorem we get convergence as long as $\eta_k \leq \bar{\eta} < 1$, but additionally we are now able to use this theoretical convergence result for practical implementation of adaptive Newton method. If a user prescribed initial forcing term η_0 is given, an a-posteriori estimate of the unknown Kantorovich quantity h_k for the quadratic convergence estimate (3.37) can be done by

$$[h_k]_2 := \frac{2\frac{\|F(x_{k+1})\|}{\|F(x_k)\|}}{(1+\rho)(1-\eta_k^2)} \leq h_k$$

and also an a-priori estimate is at hand

$$[h_{k+1}] := \frac{\|F(x_{k+1})\|}{\|F(x_k)\|}[h_k]_2 \leq \frac{\|F(x_{k+1})\|}{\|F(x_k)\|}h_k = h_{k+1}. \tag{3.38}$$

Since the above theorem claims the inner iteration to be controlled via (3.36) we obtain the adaptive convergence criteria

$$\frac{\eta_k}{1-\eta_k^2} \leq \frac{1}{2}\rho[h_k]$$

based on computational available estimates. For convergence of Newton's method, i.e. $\|F(x_k)\| \to 0$ as $k \to \infty$ and therefore $h_k \to 0$, this requirement is simply $\eta_k \to 0$ which reflects the fact that the inner iteration has to be more accurate when Newton's method reaches the solution. A second estimation can be done by the inequality

$$\begin{aligned}
\|F(x_{k+1}) - r_k\| &= \|F(x_k + \delta x_k) - r_k\| = \|F(x_k) + \int_0^1 F'(x_k + t\delta x_k)\delta x_k \, dt - r_k\| \\
&\leq \int_0^1 \|[F'(x_k + t\delta x_k) - F'(x_k)]\delta x_k\| \, dt \leq \int_0^1 \frac{1}{t}\omega\|F'(x_k)t\delta x_k\|^2 \, dt \\
&= \int_0^1 t\omega\|F'(x_k)\delta x_k\|^2 \, dt = \frac{1}{2}\omega\|F(x_k) - r_k\|^2 = \frac{1}{2}\omega(1-\eta_k^2)\|F(x_k)\|^2 \\
&= \frac{1}{2}(1-\eta_k^2)h_k\|F(x_k)\|,
\end{aligned}$$

where we used the affine contravariant Lipschitz condition of theorem above and the well-known property of GMRES that

$$\|r_0 - r_k\|^2 = (1-\eta_k^2)\|r_0\|^2 \quad \text{with } r_0 = F(x_k).$$

Hence we get a second a-posteriori estimate

$$[h_k]_1 := \frac{2\|F(x_{k+1}) - r_k\|}{(1-\eta_k^2)\|F(x_k)\|} \leq h_k$$

and the according a-priori estimate

$$[h_{k+1}] := \frac{\|F(x_{k+1})\|}{\|F(x_k)\|}[h_k]_1 \leq h_{k+1}.$$

We now want to show that both estimates can also be interpreted as the choices by Eisenstat and Walker. Therefore take the convergence condition of inner iteration (3.36) first with $\rho = 1$ fixed and with the a-posteriori estimate $[h_k]_1$ which gives

$$\frac{\eta_k}{1-\eta_k^2} \leq \frac{1}{2}[h_k]_1 = \frac{\|F(x_{k+1}) - r_k\|}{(1-\eta_k^2)\|F(x_k)\|} \leq \frac{h_k}{2}$$
$$\Rightarrow \eta_k \leq \frac{\|F(x_{k+1}) - r_k\|}{\|F(x_k)\|}.$$

Obviously this condition results in what was already proposed in (3.33) when we shift the index of η_k to η_{k+1}. Let now $\rho > 0$ be arbitrary and evaluate (3.36) with the a-priori estimate (3.38) which gives

$$\frac{\eta_k}{1-\eta_k^2} \leq \frac{1}{2}\rho[h_k] = \frac{1}{2}\rho\frac{\|F(x_k)\|}{\|F(x_{k-1})\|}[h_{k-1}]_2 = \frac{\rho}{(1+\rho)(1-\eta_{k-1}^2)}\Big(\frac{\|F(x_k)\|}{\|F(x_{k-1}\|}\Big)^2 \leq \frac{\rho h_k}{2}$$

and therefore

$$\eta_k \leq \frac{\rho(1-\eta_k^2)}{(1+\rho)(1-\eta_{k-1}^2)}\Big(\frac{\|F(x_k)\|}{\|F(x_{k-1}\|}\Big)^2.$$

By an adequate choice of the parameter γ and α in (3.34) we can recover this specific estimation. Hence, the more heuristic choices of Eisenstat and Walker (3.33) and (3.34) turn out to be given as computational estimates of Kantorovich quantity if the convergence results of inexact Newton method are based on affine invariant Lipschitz condition.

Numerical Results

In order to judge the above adaptive choices of forcing terms in the framework of Navier-Stokes solver, we compared the following criteria for η_k:

- choice 1: $\eta_k = \frac{\big|\|F(x_k)\| - \|F(x_{k-1}) + F'(x_{k-1})\delta x_{k-1}\|\big|}{\|F(x_{k-1})\|}$
- choice 2: $\eta_k = \gamma(\frac{\|F(x_k)\|}{\|F(x_{k-1})\|})^\alpha$ with $\alpha = 2$ and $\gamma = 0.1, 0.5, 0.9$
- choice 3: $\eta_k = \min\{\frac{1}{k+2}, \|F(x_k)\|\}$
- constant: $\eta_k = 10^{-1}, 10^{-3}, 10^{-5}$
- affine invariance: $\frac{\eta_k}{1-\eta_k^2} \leq \frac{1}{2}\rho[h_k] = \frac{\rho}{2}\frac{\|F(x_k)\|}{\|F(x_{k-1})\|}[h_{k-1}]_2$ with $\rho = 0.9$ and a-priori estimate for Kantorovich-quantity $[h_k] \leq h_k$.

For the first 4 choices we used damped inexact Newton method as shown in Algorithm 2 with Armijo backtracking update. In the affine invariant case, we implemented the adaptive choice of damping factor as reported in [42]. The model-problem was given by sequential Navier-Stokes solver at Reynolds number 42.0 with multilevel incomplete LDU preconditioner and 3D backflow geometry, mesh-level 4 resulting in 472748 degrees of freedom (see Table 3.5 for details).

Forcing term η_k proposed as in choice 2 gave the best (in terms of overall computational time) results for the parameter $\gamma = 0.5$ so that we will compare this to all other choices. If we first compare to constant forcing values as done in Figure 3.9, we see the effects of under- and oversolving. For the very weak choice of $\eta_k = 10^{-1}$ Newton's method does not lead to local quadratic convergence and therefore requires too many steps. On the other side a strict choice of $\eta_k = 10^{-5}$ ensures local quadratic convergence but at the costs of too many GMRES steps in the first outer iterations leading to high overall solution time. Somehow a lucky and competitive choice is $\eta_k = 10^{-3}$ but this was just the case for this single example.

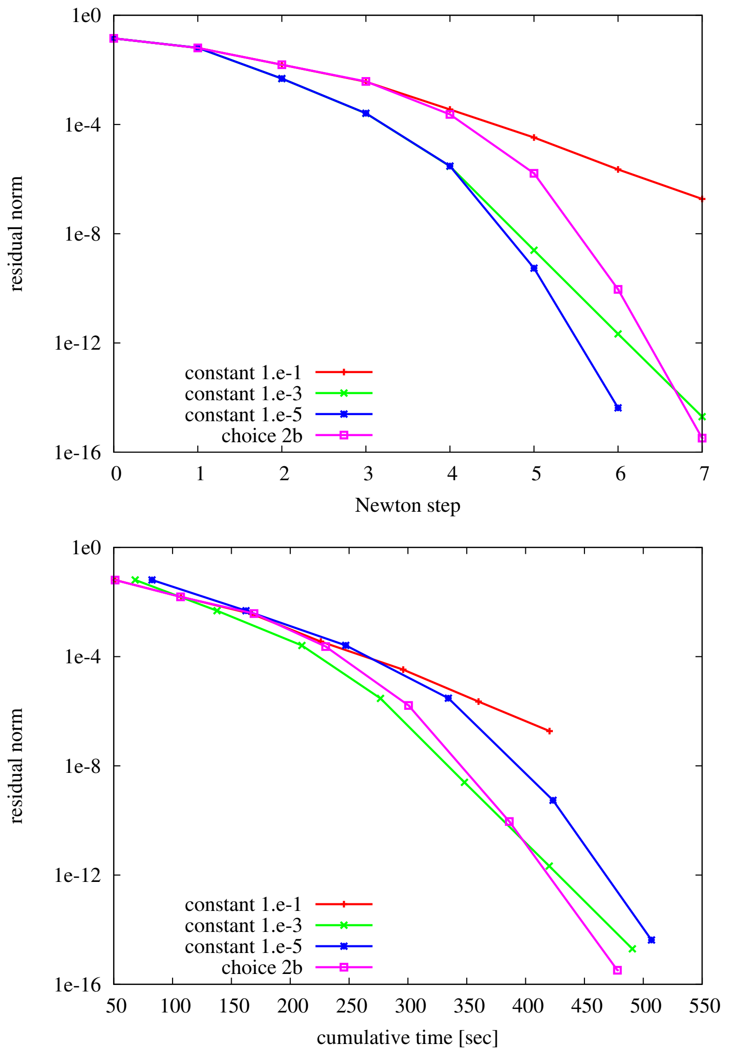


Figure 3.9: Convergence histories for inexact Newton method using constant choices of the forcing terms.

A second comparison is done among all adaptive proposals for the forcing terms η_k. As shown in Figure 3.10 the main conclusions are:

- all adaptive choices, except choice 1, behave quite the same,
- especially the affine invariance choice by Deuflhard achieves the same numerical results as the choice 2 of Eisenstat and Walker - a fact that was already mentioned above theoretically,
- the best overall results in term of fast convergence and less GMRES steps was achieved by choice 2.

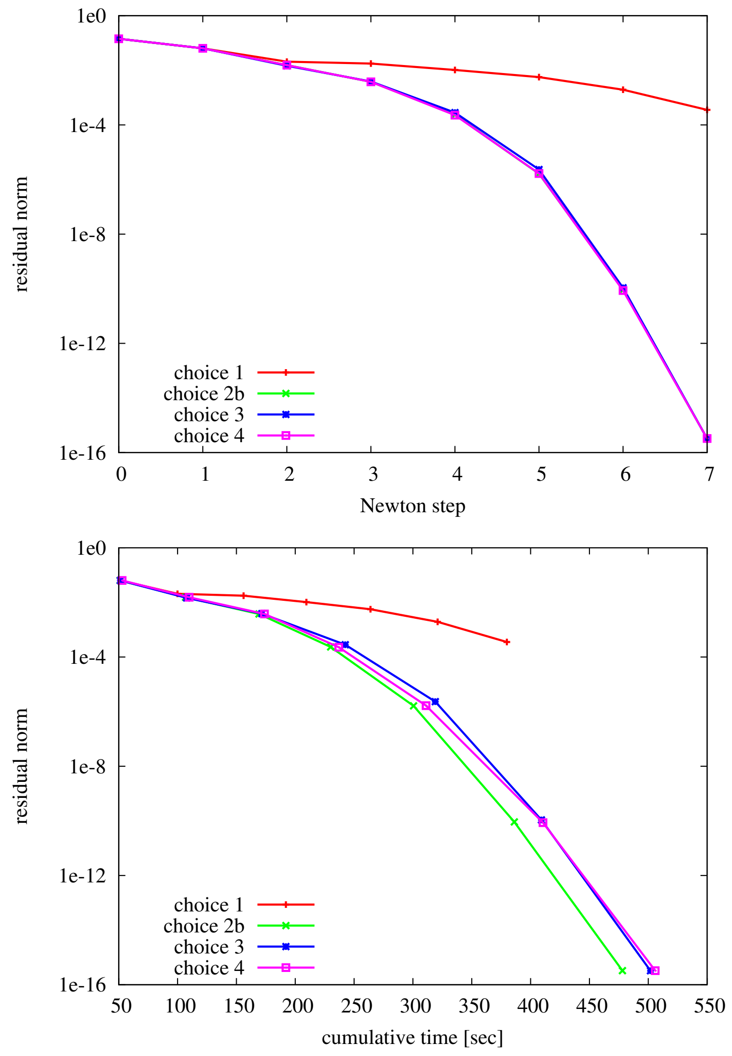


Figure 3.10: Convergence histories for inexact Newton method using adaptive choices of the forcing terms.

Chapter 4

Domain Decomposition and Parallel Solver

The algebraic systems arising from discretisation of 3D problems under investigation (cf. Section 3.2) are typically of order $10^5 - 10^6$ and already hit the wall for pure simulation using sequential algorithms. Since the introduced optimisation based on the adjoint or sensitivity equations necessitates the repeated solving of such equivalent systems, the nonlinear resp. linear algebraic solver will be the critical part of the overall solution process. A possible way out might be a model-reduction approach leading to smaller systems or an extension of the previously presented solver to parallel architectures. In this work the latter approach is chosen and we will show in this chapter how the finite element based discretisation can be used to derive parallel solver frameworks. The challenge of parallel solvers for large-scale problems is the irreducibility of the system, i.e. each degree of freedom depends upon all others - no degree of freedom may be removed and be solved for in isolation, except for the decomposition of potential, Navier-Stokes and convection diffusion equation.

First we will in general present the idea of domain decomposition techniques aimed to create smaller and/or less difficult problems that can be solved in parallel. Afterwards we will show two ways how the multilevel ILU based preconditioner introduced in the previous chapter can also be used in a parallel setting, namely within a row block data structure and within Schur complement approach in domain decomposition.

4.1 Introduction to DDM

Domain decomposition methods (DDM) sound like a relatively straightforward paradigm to parallelise the underlying problems, namely the domain, say Ω_h from Section 3.1, is decomposed into several other domains. However, the meaning of the term domain decomposition depends strongly on the underlying context, so that a first distinction seems necessary. In general domain decomposition might be referred to:

- coupling of different physical models in different regions of the underlying problem with the interface between the domains handled by various conditions,

- optimal discretisation techniques for the underlying equations that might vary in different regions,

- efficient iterative solvers and preconditioning methods for the arising system of equations that base on data distribution over several processes,

- special parallelisation techniques on modern supercomputers describing the process of distributing data from a computational model among the processes in a distributed memory system.

A clever combination of these steps in simulation and optimisation for partial differential equations is the key for fast and reliable numerical solvers. For a comprehensive introduction to the study of domain decomposition methods we refer to [110], while an overview of current work in this field of research can be found in the series *Domain Decomposition Methods in Science and Engineering* [16].

In this work we concentrate on the last two points assuming a given discretisation of the underlying equations based on conforming finite element method on regular meshes as defined in Section 3.1. Hence we claim that the discretisation and decomposition itself is somewhat optimal, but one might also use separate finite element discretisation on nonoverlapping subdomains with meshes on the subdomains that do not match on the interface. Such discretisation methods for partial differential equations are known as *mortar methods* [161] and represent a domain decomposition technique in the sense of the first two points. The challenging task for this approach is obviously the conformity of the solution over subdomain interfaces which is enforced by Lagrange multipliers. Mortar discretisation are then naturally solved by iterative domain decomposition methods such as FETI (see [48, 155] and references therein). A comprehensive presentation of domain decomposition methods for the solution of algebraic systems arising from the approximation of partial differential equations can be found in the monographs and surveys of Chan and Mathew [30], Toselli and Widlund [155], Quarteroni and Valli [127] and Smith, Bjørstad and Gropp [143].

Even if we refer to domain decomposition in the proper meaning of the word, i.e. partitioning of the computational domain, a further differentiation for the resulting solution approaches is meaningful. The variety of possible methods might be classified by the moment of decomposition in the overall solution process:

continuous: decomposition is applied to the partial differential equations,

discrete: decomposition is applied to the system of equations after discretisation.

Alternatively, one might distinguish the methods by the kind of decomposition:

overlapping: neighbouring subdomains overlap each other - leading to the class of Schwarz methods,

non-overlapping: neighbouring subdomains only share boundaries (points, lines, surfaces) - leading to the class of substructuring methods.

Finally a third classification can be derived:

direct methods: these methods involve the construction by explicit condensation of lower-dimensional systems for degrees of freedom that act as separators - for differential equations, this is the Steklov-Poincaré operator, in linear algebra, it is the Schur complement,

iterative methods: in its simplest version this involves an iteration over the subdomains, where the unknown boundary data is repeatedly updated by neighbouring domains - again this is the class of Schwarz methods.

Common to all these methods is a decomposition of the original domain into smaller subdomains. This splitting leads to overlap regions or new boundaries between two or more subdomains, called artificial boundary or sceleton. Such parts of any subdomain boundary have to be distinguished from the original boundary $\partial\Omega$, since there might be subdomains with only artificial boundaries.

Definition 4.1.1
Let $\Omega \subset \mathbb{R}^d$ by an open, connected, bounded set and define a decomposition into M subdomains by

$$\Omega = \bigcup_{i=1}^{M} \Omega_i$$

where $\Omega_i \subset \Omega$ are open, connected and bounded sets. If $\Omega_i \cap \Omega_j = \emptyset$ for $i \neq j$ and $i, j = 1, \dots, M$ the decomposition is non-overlapping, otherwise it is overlapping. In the non-overlapping case,

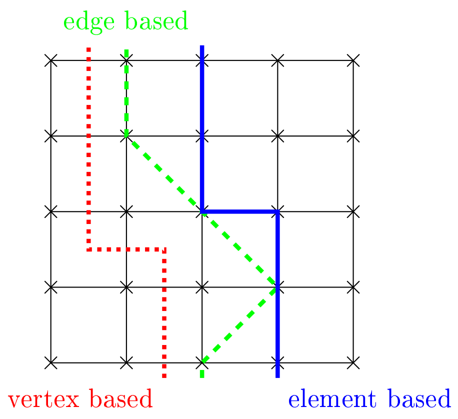


Figure 4.1: Types of domain decomposition for the mesh $\mathcal{T}_h$.

define $\Gamma_{ij} = \Gamma_{ji} = \overline{\Omega}_i \cap \overline{\Omega}_j,\ i \neq j$ *as the interface between two subdomains and*

$$\Gamma = \bigcup_{i \neq j} \Gamma_{ij}$$

as the sceleton[1]*. For the overlapping case, define* $\Gamma_{ij} = \partial\Omega_i \cap \Omega_j \neq \Gamma_{ji}$ *as the boundary of* Ω_i *that lies in* Ω_j*.*

This definition holds true for any domain Ω independent of a possible associated finite element mesh $\mathcal{T}_h$. For the case that a set of cells is given, a further differentiation can be done (cf. Figure 4.1):

vertex-based decomposition: edges and cells may straddle between subdomains, only vertices are unambiguously assigned to subdomains,

edge-based decomposition: edges are not allowed to split between different subdomains,

element-based decomposition: each cell/element is unambiguously assigned to a subdomains.

When solving problems that arise from discretisation of partial differential equations by means of finite element method, the element based decomposition offers best properties as we will show later. In the sequel the main ideas of direct and/or iterative solvers as well as preconditioners based on different kinds of domain decomposition are pointed out using a generic partial differential equation. Therefore let L be a differential operator defined on the domain Ω and we consider for simplicity only the Dirichlet boundary-value problem

$$\begin{aligned} Lu &= f && \text{in } \Omega, \\ u &= g && \text{on } \partial\Omega. \end{aligned} \tag{4.1}$$

<u>**Remark 4.1.1**</u>
We only for short denote the main concept of parallel architectures used in the sequel - for a detailed overview we refer to [128, 129]. Nowadays a clear separation of types of systems by either the memory access pattern or the instruction and data set processing seems more and more tricky due to upcoming multicore and coprocessor technologies. To our use we stick to the SIMD design meaning that a single instruction is performed in parallel by multiple processing units using multiple data. This scheme directly reflects the idea of domain decomposition as presented before and we will therefore almost always identify a subdomain Ω_i *with a processing unit* P_i*. For the sake of generality*

[1]Since in this chapter the focus is not on the boundary of $\partial\Omega$, there should be no confusion about the notation of Γ even if we used this notation previously also for $\partial\Omega$.

we do not specify any concrete meaning of a processing unit (this might e.g. be a CPU or GPU) but instead refer to general processes which work in parallel on different data. Whether each process owns its own memory or shares it with several others (like on any actual used multi-core processor) is of minor concern within this thesis. We only expect that the processing units are interconnected by a specific topology which might be a bus or any network allowing a message passing between each processing unit. As one of the standard specifications of a language-independent communication protocol used to program parallel computers we use the Message Passing Interface (MPI) [54].

4.1.1 Overlapping Methods

If the domain decomposition is applied on the continuous level of the partial differential equations and additionally a particular overlap of subdomains is assumed, the class of *alternating Schwarz methods* can be derived [137]. Let us denote by u_{ij} the values of solution u lying on Γ_{ij}, i.e. in Ω_j. We restrict the global problem (4.1) to the subdomains Ω_i and seek for a local solution u_i in Ω_i, such that

$$\begin{aligned} Lu_i &= f && \text{in } \Omega_i, \\ u_i &= g && \text{on } \partial\Omega \cap \partial\Omega_i, \\ u_i &= u_{ij} && \text{on } \Gamma_{ij}. \end{aligned} \tag{4.2}$$

Obviously, if solutions u_i and u_j of subdomains Ω_i, Ω_j with $i \neq j$ coincide in the overlapping area $\Omega_{ij} = \overline{\Omega}_i \cap \overline{\Omega}_j$, the global solution to problem (4.1) can be composed as

$$u = \bigcup_{i=1}^{M} u_i.$$

However, it is not clear how to set the boundary-values u_{ij} in (4.2) unless the solution is known. The class of Schwarz methods hence solves the local problems (4.2) alternately on different subdomains Ω_i - giving the original name *alternating Schwarz method*. Starting with an initial guess for the global solution, the problem on a subdomain is solved using the guess as part of the required boundary conditions. Two schemes can be distinguished depending on the update of values u_{ij} for next subdomain.

Multiplicative Schwarz method: Given an initial guess for the global solution u, the algorithm iterates over all subdomains Ω_i and solves the local problems (4.2) leading to a new global solution. Hence the algorithm is pure sequential and would only allow for parallelisation, if the subdomains are grouped to sets without overlap by a colouring such that no two subdomains which share common points have the same colour. Each process owning a set of non-overlapping subdomains can then solve the local problems in parallel and afterwards an exchange of local solutions will be performed building the new global iterate.

Additive Schwarz method: Here the subdomains are distributed to processes without regarding the overlap. Each process solves the local problem (4.2) using the last available solution u_{ij} from the global iterate. Only after all subdomain solutions u_i are computed an exchange between all processes is done, representing the new iterate global solution. This improvement have been introduced in [43].

The Schwarz methods are one of the first domain decomposition techniques used (cf. [106, 107]) and so a wealth of theory is available especially for the case of elliptic partial differential equations, ranging from the conditions required for convergence to estimates of the rate of convergence as a function of the amount of overlap (cf. [155]). At least the convergence of Schwarz alternating methods for the stationary Navier-Stokes equations has been proven in [109] if the Reynold number is sufficiently small. Nevertheless, nowadays the Schwarz method is more often used as a preconditioner for an iterative solver such as a Krylov subspace method. An extensive introduction to the Schwarz method used as a preconditioner in the framework of elliptic partial differential equations can be found in [143]. Beside more theoretical aspects also implementation issues are faced and an extension of Schwarz methods as multilevel preconditioners is presented. We will not prove convergence results for the Schwarz preconditioner (cf. [80] for this) but note that for many elliptic differential equations using overlapping Schwarz methods in conjunction with a Krylov subspace method the following convergence behaviour can be shown:

- number of iterations (to reduce the initial residual norm by a fixed factor) grows with the inverse of the subdomain size,
- number of iterations is independent of the mesh size, provided that the overlap region is kept proportional to the size of the subdomains,
- poor convergence is given for overlap near zero but improves rapidly as the overlap increases,
- number of iterations for the multiplicative variant is about half that for the additive algorithm.

To sum up, domain decomposition by Schwarz methods (used as solver or preconditioner) falls in the class of overlapping/iterative schemes and might be judged by the following points:

> **advantages**: original problem can be retained, one only needs a restriction to subdomains, especially the additive version allows for straightforward parallelisation,
>
> **disadvantages**: sophisticated mesh handling for overlap and management for exchange of local solutions, rate of convergence usually decreases exponentially when the amount of overlap is reduced, indicating an unavoidable computational overhead by repeated solves on the overlap regions.

To end the section on overlapping methods, we want to mention that besides the continuous point of view also the discrete way of the overlapping Schwarz methods might be used. Instead of defining explicit overlapping subdomains Ω_i, one starts from the global linear system

$$Ax = b,\ A \in \mathbb{R}^{I\times I},$$

where I is the index set of all degrees of freedom in the finite element mesh, and defines an overlapping partition of this index set

$$I_i := \{j \in I :\ \text{degree of freedom } j \in \Omega_i\},\ i = 1,\dots,M.$$

We note that the index sets I_i might also be defined without explicit knowledge of the subdomains Ω_i. Furthermore let $x \in \mathbb{R}^I$ be the global solution vector and denote by $R_i : \mathbb{R}^I \rightarrow \mathbb{R}^{I_i}$ the restriction to the i-th index set I_i, i.e.

$$[R_i x]_j = [x]_j,\ \forall j \in I_i.$$

Accordingly $R_i^T : \mathbb{R}^{I_i} \rightarrow \mathbb{R}^I$ is the projection defined by

$$[R_i^T x_i] = \begin{cases} [x_i]_j, & j \in I_i \\ 0, & \text{else} \end{cases}$$

and

$$A_i := R_i A R_i^T = A_{I_i,I_i}$$

the local system matrix on the index set I_i. Now we are able to define the iterative Schwarz method on the algebraic level:

> **Multiplicative Schwarz method**: let $e_k = x - x_k$ be the global error in k-th step and $Ae_k = b - Ax_k$ the residual. For the i-th index set $I_i \subset I$, assume that the global error is given by means of
>
> $$e_k = R_i^T v_i$$
>
> with a vector $v_i \in \mathbb{R}^{I_i}$ that has to be defined. The restricted residual in I_i is hence
>
> $$R_i A e_k = R_i A R_i^T v_i = A_i v_i = R_i(b - Ax_k)$$

with the local system matrix A_i and the new iterate for the global solution is

$$x_{k+1} = x_k + e_k = x_k + R_i^T v_i = x_k + R_i^T A_i^{-1} R_i (b - Ax_k).$$

We see that the new iterate corresponds to an update by solving the system once on the index set I_i (resp. on the subdomain Ω_i) using the old iterate x_k for boundary values. If this procedure is repeated over all index sets, one ends up with the multiplicative Schwarz method as shown before, i.e.

$$x_{k+\frac{i}{p}} = x_{k+\frac{i-1}{p}} + R_i^T A_i^{-1} R_i (b - Ax_{k+\frac{i-1}{p}}),\ i = 1, \dots, p$$

for the step from index set $i-1$ to index set i (assuming p index sets at all). The error propagation for this scheme is therefore

$$e_{k+\frac{i}{p}} = (I - R_i^T A_i^{-1} R_i A) e_{k+\frac{i-1}{p}},\ i = 1, \dots, p$$

and for a whole iteration step $x_k \to x_{k+1}$

$$e_{k+1} = (I - P_p) \cdots (I - P_2)(I - P_1) e_k,\ P_i = R_i^T A_i^{-1} R_i A$$

explaining the name multiplicative Schwarz method. One special form is given for the case that $I_i \cup I_j = \emptyset$, $\forall i \neq j$ - at this the multiplicative Schwarz methods turns out to be equivalent to the block Gauss-Seidel scheme.

Additive Schwarz method: as mentioned for the continuous case, all correction might be computed in parallel using the last global iterate x_k, i.e.

$$x_{k+1} = x_k + \sum_{i=1}^{p} R_i^T A_i^{-1} R_i (b - Ax_k).$$

Now the error propagation is given by

$$e_{k+1} = (I - \sum_{i=1}^{p} P_i) e_k.$$

Again the case $I_i \cup I_j = \emptyset$, $\forall i \neq j$ for the additive Schwarz method gives a well-known scheme, namely the block Jacobi method.

Convergence results for the algebraic form of Schwarz method can be found in Chapter 11 of [80].

4.1.2 Non-overlapping Methods

Consider the continuous global problem (4.1) and a non-overlapping decomposition of Ω, i.e. $\Omega_i \cap \Omega_j = \emptyset$ for all $i \neq j$. We formulate the local problem on subdomain Ω_i

$$\begin{aligned} Lu_i &= f && \text{in } \Omega_i, \\ u_i &= g && \text{on } \partial\Omega \cap \partial\Omega_i, \\ u_i &= U && \text{on } \Gamma_{ij}. \end{aligned} \tag{4.3}$$

The treatment of coupling between subdomains (interface coupling) is now more sophisticated. Since no overlap to the neighbouring domains is given, the definition of Dirichlet boundary-values U on the sceleton as before is no longer meaningful. These values would be preserved throughout all Schwarz type iterations and hence no gain for the global solution is given. Instead a different approach is used that can be viewed twofold:

- beside a pure Dirichlet condition, additional coupling/transmission conditions are imposed on each subdomain to ensure global interface transition,

- the original global problem is reduced to an equivalent one posed only on the interface of subdomains, i.e. an *interface operator* is introduced.

For the first point of view the method pursued by P.L. Lions [108] (in the case that L is the Laplace operator) is very natural and simple, namely that one reformulates the constraints on Γ_{ij} and requires a Robin type boundary condition (with relaxation parameter $\beta > 0$)

$$\partial_{n_i} u_i + \beta u_i = -\partial_{n_j} u_j + \beta u_j \qquad \text{on } \Gamma_{ij}.$$

Hence, not only the absolute values on the sceleton Γ_{ij} should match between two neighbouring subdomains, additionally the normal derivatives should coincide - n_i meaning the outward normal on Γ_{ij} for Ω_i towards Ω_j and so $n_i = -n_j$.

A generalisation of this transmission condition on Γ_{ij} for arbitrary partial differential operator L obviously necessitates more generic conditions to represent the relationship. We follow the notation in [127] and introduce the transmission conditions

$$\begin{aligned} \Phi(u_i) &= \Phi(u_j) \qquad && \text{on } \Gamma_{ij}, \\ \Psi(u_i) &= \Psi(u_j) \qquad && \text{on } \Gamma_{ij}, \end{aligned}$$

where the functions Φ, Ψ depend upon the nature of the underlying operator L. Having a problem specific definition of these functions at hand, one can state an iteration-by-subdomain scheme similar to the additive Schwarz method in the overlapping case. Assume that the local solution at step k is given by u_i^k for each subdomain $i = 1, \dots, M$. First average at the interface among the values of Φ using a parameter α, i.e.

$$\Phi_{ij}^{\text{av}} = \alpha \Phi(u_i^k) + (1-\alpha)\Phi(u_j^k) \qquad \text{on } \Gamma_{ij}.$$

Then on each subdomain a problem of Φ-type is solved

$$\begin{aligned} L u_i^{k+\frac{1}{2}} &= f \qquad && \text{in } \Omega_i, \\ u_i^{k+\frac{1}{2}} &= g \qquad && \text{on } \partial\Omega \cap \partial\Omega_i, \\ \Phi(u_i^{k+\frac{1}{2}}) &= \Phi_{ij}^{\text{av}} \qquad && \text{on } \Gamma_{ij}. \end{aligned}$$

Second also the values of Ψ need to be averaged (again weighted by a parameter β), i.e.

$$\Psi_{ij}^{\text{av}} = \beta \Psi(u_i^{k+\frac{1}{2}}) + (1-\beta)\Psi(u_j^{k+\frac{1}{2}}) \qquad \text{on } \Gamma_{ij}$$

and finally M independent problems of Ψ-type are to be solved

$$\begin{aligned} L u_i^{k+1} &= f \qquad && \text{in } \Omega_i, \\ u_i^{k+1} &= g \qquad && \text{on } \partial\Omega \cap \partial\Omega_i, \\ \Psi(u_i^{k+1}) &= \Psi_{ij}^{\text{av}} \qquad && \text{on } \Gamma_{ij}. \end{aligned}$$

For the Poisson problem with $\Phi(v) = v$ and $\Psi(v) = \partial_n v$ this scheme means a separation of Dirichlet and Neumann subproblems. Generally, several other combinations of the transmission conditions for the local subproblems are used like Dirichlet-Neumann, Neumann-Neumann or Robin conditions - one should notice that for each partial differential operator L the transmission conditions need to be adopted appropriately. We refer to [127] for an overview on transmission conditions for different boundary value problems including the Stokes problem. In the framework of flow problem there exist plenty of works proposing different conditions, showing that a kind of standard formulation even for one specific partial differential equation is not present.

The presented subdomain iteration approach using transmission conditions can also be interpreted as an interface equation in terms of the Steklov-Poincaré operator (cf. [3] and references therein). Common to each iterative method cycling over subdomains is an update of the values on the sceleton Γ for the transmission conditions Φ and Ψ. Without loss of generality we assume that Φ is given by $\Phi(v) = v$, i.e. a matching of pointvalues on the sceleton, and that Ψ is a linear function. If the values on Γ are known, say $\lambda = \Phi(u|_\Gamma) = u|_\Gamma$, the independent problems on each subdomain would be

$$\begin{aligned} Lu_i &= f && \text{in } \Omega_i, \\ u_i &= g && \text{on } \partial\Omega \cap \partial\Omega_i, \\ u_i &= \lambda && \text{on } \Gamma_{ij}, \\ \Psi(u_i) &= \Psi(u_j) && \text{on } \Gamma_{ij}. \end{aligned} \tag{4.4}$$

We can furthermore split the solution $u_i = u_i^0 + u_i^f$ into two parts u_i^0 and u_i^f being the solution of the Dirichlet problems

$$\begin{aligned} Lu_i^0 &= 0 && \text{in } \Omega_i, \\ u_i^0 &= g && \text{on } \partial\Omega \cap \partial\Omega_i, \\ u_i^0 &= \lambda && \text{on } \Gamma_{ij}, \end{aligned}$$

and

$$\begin{aligned} Lu_i^f &= f && \text{in } \Omega_i, \\ u_i^f &= g && \text{on } \partial\Omega \cap \partial\Omega_i, \\ u_i^f &= 0 && \text{on } \Gamma_{ij}. \end{aligned}$$

Define the solution operators by $u_i^0 = H_i\lambda$ and $u_i^f = G_i f$. Now the global problem (4.1) is formally equivalent to (4.4) if and only if $\Psi(u_i) = \Psi(u_j)$ on Γ_{ij} for all $i, j = 1, \ldots, M$. The latter condition can be reformulated using the splitting $u_i = u_i^0 + u_i^f$ into

$$\sum_{i,j=1}^{M} \Psi(u_i^0) + \Psi(u_i^f) - \Psi(u_j^0) - \Psi(u_j^f) = 0.$$

Finally we can state the Steklov-Poincaré interface equation

$$S\lambda = \chi \qquad \text{on } \Gamma \tag{4.5}$$

with

$$\chi = \sum_{i,j=1}^{M} -\Psi(G_i f) + \Psi(G_j f)$$

and the *Steklov-Poincaré operator* defined as

$$S\lambda = \sum_{i,j=1}^{M} -\Psi(H_i\lambda) + \Psi(H_j\lambda).$$

Hence, the interface equation (4.5) correspond to an equation for the unknown values on Γ using the second transmission condition posed by Ψ as defining operator. If equation (4.5) is solved by an iterative scheme one gets the correspondence to the above shown subdomain iterations, since an application of the Steklov-Poincaré operator is given by a local solution on each subdomain.

Summing up, we have the results for the non-overlapping domain decomposition based on the continuous problem (4.1):

advantages: easier mesh handling and no computational overhead due to absence of overlap,

disadvantages: transmission conditions and/or interface operators need to be established for different partial differential equations separately.

As for the overlapping case, there is obviously also a non-overlapping decomposition scheme based on the discrete equations. We will present this approach, the discrete counterpart to the Steklov-Poincaré interface equation, namely the Schur complement equation, in detail in the next section.

4.2 Parallel Data Structures and Preconditioners

Having introduced some domain decomposition approaches in the last section, we now investigate on two schemes based on non-overlapping decomposition and on finite element discretisation in detail. In contrast to the methods based on the continuous equations, the decomposition on the discrete equations (resp. the algebraic system) has the advantage that it can be used for any problem without imposing special transmission/boundary conditions. First we will work out the idea and data structures of Schur complement solver/preconditioner which is more involved than the row block decomposition that is used for comparison. In the next section the performance gained by these methods will be considered.

Let the computational domain Ω_h be equipped with a finite element mesh $\mathcal{T}_h$. We have just mentioned that a partition of $\mathcal{T}_h$ into subdomains can be done vertex-, edge- or element-based and have not explained which version should be preferred. In terms of a finite element discretisation the resulting linear algebraic structures - mainly residual vector and Jacobian matrix - are given by the weak form of the underlying partial differential equation, i.e. will be computed by means of numerical integration (cf. finite element matrices (3.21)). For sake of simplicity, in the following we assume a general bilinear form

$$a(u,v) = \int\limits_{\Omega} f(u(\mathbf{x}), v(\mathbf{x}))\ d\mathbf{x}$$

without specifying the function f in detail. Standard routines of numerical integration can then be decomposed as

$$[A]_{ij} = a(\varphi_h^j, \varphi_h^i) = \int\limits_{\Omega_h} f(\varphi_h^j, \varphi_h^i)\ d\mathbf{x} = \sum_{K \in \mathcal{T}_h} \int\limits_K f(\varphi_h^j, \varphi_h^i)\ d\mathbf{x}, \quad i,j = 1,\ldots,N$$

suggesting a decomposition of $\mathcal{T}_h$ elementwise to compute the cell-integrals in parallel. Since also the support of a trial-/test-function φ_h is locally in Ω_h the summation over all $K \in \mathcal{T}_h$ can be restricted to those cells that are in the support of φ_h^i.

In the sequel we throughout assume a non-overlapping decomposition of $\mathcal{T}_h$ into as many subdomains Ω_i as processes are available and let each subdomain be assigned to a process P_i. Such a decomposition naturally induces sets of submeshes $\mathcal{T}_{h,i}$ for each subdomain that are compatible on Γ, i.e. that they share the same edges/faces on Γ. Using Lagrange finite elements, the global degrees of freedom, whose support is entirely in Ω_i are obviously assigned to P_i. Those degrees of freedom whose support is cut by the interface between two subdomains, say Ω_i and Ω_j, are assigned either to P_i or P_j. We will comment on this choice later on when concrete data-structures are presented.

4.2.1 The Schur Complement System

As in Definition 4.1.1 we assume that M non-overlapping subdomains $\Omega_i \subset \Omega_h$ are given where each subdomain is built up by a submesh $\mathcal{T}_{h,i}$. For the global degrees of freedom in the mesh two groups are distinguished, namely internal and sceleton nodes (cf. Figure 4.2). The boundary nodes are treated as internal nodes, since they can clearly be assigned to a particular subdomain Ω_i. Sceleton nodes are those nodes belonging to the artificial boundaries created by the domain decomposition, i.e. those nodes that lie on the interface Γ. Let furthermore $\Gamma_i = \partial\Omega_i \backslash \partial\Omega$ denote the part of the sceleton Γ belonging to subdomain Ω_i and assume that the global system consists of N degrees of freedom. By N_Γ we denote the number of degrees of freedom on Γ and by N_i the number of internal degrees of freedom in Ω_i such that

$$N = N_\Gamma + \sum_{i=1}^{M} N_i.$$

The following introduced *Schur complement method* consist of eliminating internal degrees of freedom to define schemes which focus on solving a reduced system associated with only the sceleton

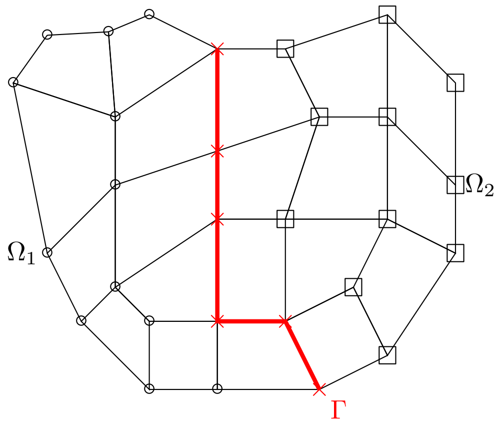


Figure 4.2: Differentiation between internal ($\circ$ in Ω_1, $\square$ in Ω_2) and sceleton (denoted by $\times$) nodes in finite element mesh $\mathcal{T}_h$.

degrees of freedom. Often also the terms *static condensation method* or *method of substructuring* are used to describe the Schur complement method. These denotations come from the engineering view of the method (cf. [99]), while the name Schur complement method bases on the mathematical definition (cf. [30, 143, 155]).

As mentioned before the elementwise non-overlapping decomposition of $\mathcal{T}_h$ allows the definition of subdomain stiffness matrices

$$A_i = \begin{pmatrix} A_{ii} & A_{i\Gamma_i} \\ A_{\Gamma_i i} & A_{\Gamma_i\Gamma_i} \end{pmatrix} \tag{4.6}$$

with

$$\begin{aligned} [A_{ii}]_{kl} &= a(\varphi_h^l, \varphi_h^k), && k, l = 1, \dots, N_i \\ [A_{i\Gamma_i}]_{lm} &= a(\tilde\varphi_h^m, \varphi_h^l), && l = 1, \dots, N_i,\ m = 1, \dots, N_{\Gamma_i} \\ [A_{\Gamma_i\Gamma_i}]_{mn} &= a(\tilde\varphi_h^n, \tilde\varphi_h^m), && m, n = 1, \dots, N_{\Gamma_i}. \end{aligned}$$

Here we denote by $\varphi_h^i,\ i = 1, \dots, N_i$ the finite element basis functions for the internal degrees of freedom on Ω_i and by $\tilde\varphi_h^i,\ i = 1, \dots, N_{\Gamma_i}$ those for the sceleton degrees of freedom on Γ_i. The stiffness matrix (4.6) refers to the finite element discretisation on Ω_i using natural boundary conditions on Γ_i and the essential boundary conditions of the global problem on $\partial\Omega_i \cap \partial\Omega$. Let now the global degrees of freedom be numbered according to:

- first number all internal degrees of freedom in subdomain Ω_1,
- proceed with all internal degrees of freedom of all other subdomains $\Omega_i,\ i = 2, \dots, M$,
- number the degrees of freedom on Γ last.

We remark, that this numbering is just used for easier notation and can be generalised as shown later. Using this specific numbering the linear system arising from finite element discretisation $Ax = b$ has the abstract form

$$\begin{pmatrix} A_{11} & 0 & \dots & 0 & A_{1\Gamma} \\ 0 & A_{22} & \dots & 0 & A_{2\Gamma} \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \dots & A_{MM} & A_{M\Gamma} \\ A_{\Gamma 1} & A_{\Gamma 2} & \dots & A_{\Gamma M} & A_{\Gamma\Gamma} \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_M \\ x_\Gamma \end{pmatrix} = \begin{pmatrix} b_1 \\ b_2 \\ \vdots \\ b_M \\ b_\Gamma \end{pmatrix}. \tag{4.7}$$

As long as the numbering of degrees of freedom is as defined above, this structure is also given for the specific linear forms stated in Section 2.2. Especially the upper left consisting of a block diagonal matrix is typical for finite element discretisation since the ordering of degrees of freedom is chosen such that there is no direct coupling between internal nodes of different subdomains. To relate the global system in (4.7) to the local matrices in (4.6), let R_i be the restriction matrix for a vector x in Ω to a local vector x_i in $\Omega_i \cup \Gamma_i$ and R_i^T be the prolongation by 0 for the degrees of freedom external to $\Omega_i \cup \Gamma_i$. Furthermore let R_{Γ_i} be the restriction matrix for a vector x_Γ of the values on the sceleton to only those values on Γ_i and $R_{\Gamma_i}^T$ be the matrix which prolongates the local vector x_{Γ_i} by 0 to the global vector on Γ. Hence the global matrix in (4.7) is given by

$$A = \sum_{i=1}^{M} R_i^T A_i R_i$$

with blocks according to sceleton degrees of freedom

$$A_{\Gamma\Gamma} = \sum_{i=1}^{M} R_{\Gamma_i}^T A_{\Gamma_i\Gamma_i} R_{\Gamma_i}, \;\; A_{i\Gamma} = A_{i\Gamma_i} R_{\Gamma_i}.$$

The structure of (4.7) is suitable for a block Gaussian elimination ending up with the Schur complement equation for the unknowns on Γ

$$S_\Gamma x_\Gamma = \chi_\Gamma. \tag{4.8}$$

Herein the global Schur complement matrix S_Γ is composed by local Schur complement matrices S_i of subdomain Ω_i

$$S_\Gamma = A_{\Gamma\Gamma} - \sum_{i=1}^{M} A_{\Gamma i} A_{ii}^{-1} A_{i\Gamma} = \sum_{i=1}^{M} R_{\Gamma_i}^T \underbrace{\left(A_{\Gamma_i\Gamma_i} - A_{\Gamma_i i} A_{ii}^{-1} A_{i\Gamma_i}\right)}_{S_i} R_{\Gamma_i} \tag{4.9}$$

and the right hand side is given by

$$\chi_\Gamma = b_\Gamma - \sum_{i=1}^{M} R_{\Gamma_i}^T A_{\Gamma_i i} A_{ii}^{-1} b_i. \tag{4.10}$$

The Schur complement equation (4.8) is the algebraic counterpart of the Steklov-Poincaré interface equation (4.5). Once x_Γ is known, a simple back substitution in (4.7) yields the internal degrees of freedom x_i.

The coefficient matrix (4.9) is obviously smaller than the global matrix in (4.7) but in general dense and very expensive to form due to the factorisation of A_{ii}, $i = 1, \ldots, M$. If nevertheless the matrix is assembled and factorised, one ends up with a *direct Schur complement solver* also called *substructuring method.* On the other side the matrix vector multiplication with the global Schur complement matrix S_Γ can be performed by a few sparse matrix vector multiplications and subdomains solves for A_{ii}^{-1}, which can be handled in parallel. Hence an iterative method might be applied to (4.8), forming the idea of *iterative Schur complement solver.* To obtain a convergence rate that is independent of, or only weakly dependent on, the mesh size h and the diameter of the subdomains H, a good preconditioner will be needed, since it is shown that at least for elliptic partial differential equations (cf. [24]) the condition number of the Schur complement matrix S_Γ satisfies

$$\kappa(S_\Gamma) = O(h^{-1}H^{-1}).$$

Due to the fact that the matrix (4.9) is actually not formed, the standard Jacobi, Gauss-Seidel, SOR and incomplete Cholesky-type preconditioners cannot be used. For a discussion of several interface preconditioners in the case of elliptic problems we refer to [30, 143] and references therein. These preconditioners are shown to be optimal for elliptic problems but need strong knowledge about the *wire-basket* of the underlying mesh, i.e. the possibility to restrict and extend algebraic vectors to only those degrees of freedom lying on the interface edges and vertices (cf. [142] and references therein).

4.2.2 Schur Complement Preconditioner

For our purpose of a generic parallel preconditioner to fluid flow problems, we want to adopt the idea of multilevel ILU-based preconditioners for the sequential case as shown in Section 3.2. In contrast to the special preconditioners for equation (4.8) that are available for elliptic problem, we state an abstract version just based on the algebraic form of the Schur complement equation. To this recall that the global Schur complement S_Γ is given by

$$S_\Gamma = \sum_{i=1}^{M} R_{\Gamma_i}^T S_i R_{\Gamma_i}.$$

Furthermore we perform a formal LU factorisation of the subdomain stiffness matrix A_i defined in (4.6)

$$A_i = \begin{pmatrix} A_{ii} & A_{i\Gamma_i} \\ A_{\Gamma_i i} & A_{\Gamma_i\Gamma_i} \end{pmatrix} = \begin{pmatrix} L_{A_{ii}} & 0 \\ A_{\Gamma_i i}U_{A_{ii}}^{-1} & L_{S_i} \end{pmatrix} \begin{pmatrix} U_{A_{ii}} & L_{A_{ii}}^{-1}A_{i\Gamma_i} \\ 0 & U_{S_i} \end{pmatrix}$$

yielding the LU factorisation of the local Schur complement $S_i = L_{S_i}U_{S_i}$. Hence an (approximate) factorisation of A_i naturally implies an (approximate) factorisation of S_i and so we are able to use the incomplete factorisation from Section 3.2 on each subdomain to get a parallel preconditioner for the problem (4.8), namely

$$S_\Gamma^{-1} \approx P_\Gamma := \sum_{i=1}^{M} R_{\Gamma_i}^T ILU(S_i) R_{\Gamma_i}. \tag{4.11}$$

Up to now we tacitly assumed that the local matrices A_{ii} are factorised for both the direct Schur complement solver and the iterative one. If this condition is losen, it can no longer be guaranteed that a solver for the Schur complement equation (4.8) also solves the entire problem $Ax = b$. Nevertheless this case (Schur complement equation with inexact local solver for A_{ii}) is still suitable as preconditioner for the global system $Ax = b$. To see this, we define $A_{\Gamma\Gamma}$ as before and

$$\begin{aligned} A_{II} &= \text{blockdiag}(A_{ii}), \\ A_{I\Gamma} &= \text{blockmat}(A_{i\Gamma}) = \begin{pmatrix} A_{1\Gamma} \\ \vdots \\ A_{M\Gamma} \end{pmatrix}, \\ A_{\Gamma I} &= \text{blockmat}(A_{\Gamma i}) = \begin{pmatrix} A_{\Gamma 1} & \dots & A_{\Gamma M} \end{pmatrix}, \end{aligned}$$

allowing to write (4.7) as

$$\begin{pmatrix} A_{II} & A_{I\Gamma} \\ A_{\Gamma I} & A_{\Gamma\Gamma} \end{pmatrix} \begin{pmatrix} x_I \\ x_\Gamma \end{pmatrix} = \begin{pmatrix} b_I \\ b_\Gamma \end{pmatrix}. \tag{4.12}$$

A formal factorisation gives (Id being the identity matrix in appropriate dimension)

$$A = \begin{pmatrix} Id & 0 \\ A_{\Gamma I}A_{II}^{-1} & Id \end{pmatrix} \begin{pmatrix} A_{II} & 0 \\ 0 & S_\Gamma \end{pmatrix} \begin{pmatrix} Id & A_{II}^{-1}A_{I\Gamma} \\ 0 & Id \end{pmatrix}$$

and the inverse

$$A^{-1} = \begin{pmatrix} Id & -A_{II}^{-1}A_{I\Gamma} \\ 0 & Id \end{pmatrix} \begin{pmatrix} A_{II}^{-1} & 0 \\ 0 & S_\Gamma^{-1} \end{pmatrix} \begin{pmatrix} Id & 0 \\ -A_{\Gamma I}A_{II}^{-1} & Id \end{pmatrix}. \tag{4.13}$$

Assume that an interface preconditioner for S_Γ, say P_Γ, and also for A_{II} (in form of M independent preconditioners for A_{ii}) is at hand, say $P_{II} \approx A_{II}^{-1}$. Then replacing the corresponding block matrices in (4.13) forms a preconditioner (also called *induced preconditioner* [136]) for the global system matrix A, say P_A

$$A^{-1} \approx P_A = \begin{pmatrix} Id & -P_{II}A_{I\Gamma} \\ 0 & Id \end{pmatrix} \begin{pmatrix} P_{II} & 0 \\ 0 & P_\Gamma \end{pmatrix} \begin{pmatrix} Id & 0 \\ -A_{\Gamma I}P_{II} & Id \end{pmatrix}. \tag{4.14}$$

This preconditioner needs three applications of P_{II} - parallel performed on each subdomain Ω_i - and one application of P_Γ. Such an approach to use interface preconditioners incorporated into a preconditioner for the global system has already successfully been used in the framework of fluid flow problems, see [29, 70].

Algorithm 3 Schur complement preconditioner P_{SC} for global system $Ax = b$.

```
1. given actual global iterate
       $x^k = [x_I^k, x_\Gamma^k]$
2. assemble right hand side
       $\chi_\Gamma = x_\Gamma^k - A_{\Gamma I} P_{II} x_I^k$
3. iterative solver with $m_{SC}$ steps for $\tilde{x}_\Gamma^k$ using Schur complement matrix
       $S_\Gamma = A_{\Gamma\Gamma} - A_{\Gamma I} P_{II} A_{I\Gamma}$
4. solve for internal degrees of freedom
       $\tilde{x}_I^k = P_{II}(x_I^k - A_{I\Gamma}\tilde{x}_\Gamma^k)$
5. end with preconditioned vector
       $\tilde{x}^k = [\tilde{x}_I^k, \tilde{x}_\Gamma^k] = P_{SC} x^k$
```

We now want to present a slightly different preconditioner for the global system (4.12), namely an inexact solver for the Schur complement equation (4.8) itself. As it will turn out this proposal is a generalisation of P_A in (4.14). The idea is straightforward: since a direct Schur complement solver acts like A^{-1}, an inexact version obviously inherits this behaviour. With inexact we mean two points. First equation (4.8) is solved by a Krylov subspace iteration, say GMRES, second the factorisation of internal matrices A_{ii} is replaced by an incomplete multilevel ILU decomposition as proposed in Section 3.2. Hence the parallel preconditioner contains two main parameters

- number of iterations for the Schur complement solver,
- all options for the local ILU decomposition of A_{ii}, in principal the threshold/fill-in.

In the following we denote by *Schur complement preconditioner* the proposed preconditioner for the global system $Ax = b$, which should not be mixed up with the preconditioner for the Schur complement equation (4.8) denoted by P_Γ. To highlight the connection to (4.14) we sum up the steps needed for the Schur complement preconditioner. Assume that $x^k = [x_I^k, x_\Gamma^k]$ is the global iterate for equation (4.12) in the k-th step, then x^k is preconditioned by the inexact Schur complement preconditioner to $\tilde{x}^k = [\tilde{x}_I^k, \tilde{x}_\Gamma^k]$ performing Algorithm 3. Main aspects are the replacement of exact factorisation A_{II}^{-1} by incomplete multilevel ILU decomposition which we denote by $P_{II} \approx A_{II}^{-1}$ as in (4.14) and the possibility to perform an arbitrary number of iteration steps for the Schur complement equation within step 3. Additionally one might use the preconditioner (4.11) in step 3.

Comparing Algorithm 3 to an application of preconditioner P_A (4.14) on x^k, i.e. $\tilde{x}^k = [\tilde{x}_I^k, \tilde{x}_\Gamma^k] = P_A x^k$, gives

$$\begin{aligned}\begin{pmatrix}\tilde{x}_I^k\\ \tilde{x}_\Gamma^k\end{pmatrix} &= \begin{pmatrix}Id & -P_{II}A_{I\Gamma}\\ 0 & Id\end{pmatrix}\begin{pmatrix}P_{II} & 0\\ 0 & P_\Gamma\end{pmatrix}\begin{pmatrix}Id & 0\\ -A_{\Gamma I}P_{II} & Id\end{pmatrix}\begin{pmatrix}x_I^k\\ x_\Gamma^k\end{pmatrix}\\ &= \begin{pmatrix}Id & -P_{II}A_{I\Gamma}\\ 0 & Id\end{pmatrix}\begin{pmatrix}P_{II}x_I^k\\ P_\Gamma(x_\Gamma^k - A_{\Gamma I}P_{II}x_I^k)\end{pmatrix}\\ &= \begin{pmatrix}P_{II}x_I^k - P_{II}A_{I\Gamma}P_\Gamma(x_\Gamma^k - A_{\Gamma I}P_{II}x_I^k)\\ P_\Gamma(x_\Gamma^k - A_{\Gamma I}P_{II}x_I^k)\end{pmatrix}.\end{aligned}$$

Since $P_{II} \approx A_{II}^{-1}$ and $P_\Gamma \approx S_\Gamma^{-1}$ we have that the global preconditioner P_A equals the proposed inexact Schur complement preconditioner P_{SC} if in step 3 only one iteration step on the sceleton unknowns is used.

Summing up, the proposed preconditioner allows to recover well known global preconditioners but also offers the possibility to enhance them by substitution of P_Γ through an iterative solver. Additionally, if there is no special preconditioner for S_Γ at hand, the global preconditioner can still be used. One might say, that the preconditioner for the Schur complement equation (4.8) has been replaced by a Schur complement preconditioner for the global equation (4.12).

4.2.3 Data Structures for Non-overlapping Domain Decomposition

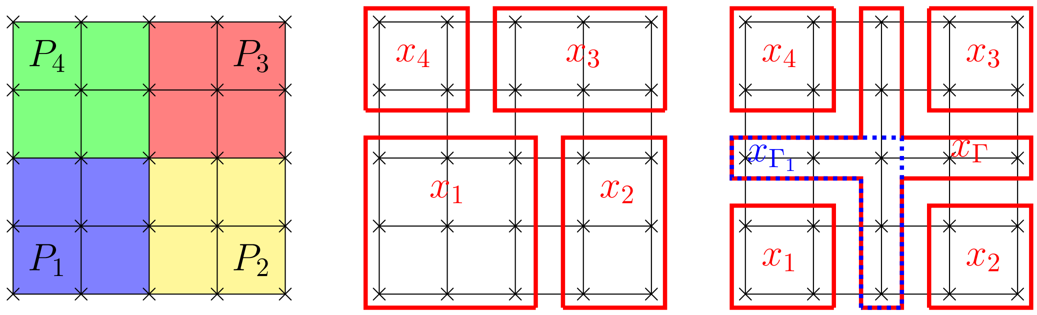


Figure 4.3: Non-overlapping elementwise mesh decomposition (left) and according partition of degrees of freedom (middle and left) - denoted by $\times$ for Q1 elements.

Given a non-overlapping domain decomposition allows to define solvers (both direct and iterative) for the Schur complement equation as well as to use the Schur complement equation as preconditioner for the global system. We now address the question of underlying data structures which will lead to a second class of preconditioners, namely the block Jacobian preconditioner. Afterwards in the next section we will show numerical results for the presented preconditioners using multilevel ILU decomposition and compare these to other standard implementations.

Important components for a parallel iterative solver based on domain decomposition methods within a finite element package are:

- parallel vector assemblies and basic algebraic operations on parallel vector,
- parallel matrix assemblies (also setup of nonzero/sparsity pattern),
- parallel matrix-vector product,
- parallel preconditioners for Krylov subspace methods.

Assume that we have a non-overlapping elementwise mesh decomposition as shown in the left part of Figure 4.3. If one uses Lagrange finite elements (or any other type of finite elements that possesses degrees of freedom on the intersection of subdomains) the definition of a parallel data structure brings up the question, which process should own the nodes on the sceleton. An elementwise decomposition only induces the belonging of global degrees of freedom whose support is entirely in a subdomain Ω_i. One might disregard the underlying mesh decomposition and distribute matrices and/or vectors by rows among the processes, resulting in a *row block storage scheme*

$$A = \left(\begin{array}{ccccc} A_{11} & A_{12} & A_{13} & \cdots & A_{1M} \\ \hline A_{21} & A_{22} & A_{23} & \cdots & A_{2M} \\ \hline \vdots & \vdots & \vdots & \ddots & \vdots \\ \hline A_{M1} & A_{M2} & A_{M3} & \cdots & A_{MM} \end{array}\right), \quad x = \left(\begin{array}{c} x_1 \\ \hline x_2 \\ \hline \vdots \\ \hline x_M \end{array}\right) \tag{4.15}$$

with $i = 1, \ldots, M$ row block submatrices consisting of

- A_{ii} local sequential matrix, called *diagonal-block*,
- $A_{i*} \backslash A_{ii}$ local sequential matrix, called *offdiagonal-block*.

Such a distribution of parallel data assigns each degree of freedom uniquely to a process and therefore to a subdomain. The underlying domain decomposition is hence no longer elementwise but vertex-oriented, see middle part of Figure 4.3. Since the matrix A is the finite element stiffness matrix, the offdiagonal-blocks $A_{ij},\ i \neq j$ represent the coupling between subdomains and are usually sparse or even empty, when two subdomains do not share a common interface. Standard algebraic operations with this data structure are straightforward. Vector addition and multiplication with a scalar can be performed in parallel without any communication, only the scalar-product involves a global reduce communication. For the matrix-vector product $Ax = b$ using (4.15) each process P_i needs to receive from P_j the vector entries x_j whenever $A_{ij} \neq 0$. Hence a sophisticated message-passing between processes is needed. For the row block partitioning of parallel matrix and vector storage, we use the well established PETSc library [9], which delivers "a suite of data structures and routines for the scalable (parallel) solution of scientific applications modelled by partial differential equations".

In contrast to the row block partitioning, we also present parallel data structures suitable for the elementwise domain decomposition in combination with Schur complement approaches. Recall that the global stiffness matrix and global algebraic system can be written as (cf. (4.12))

$$\begin{pmatrix} A_{II} & A_{I\Gamma} \\ A_{\Gamma I} & A_{\Gamma\Gamma} \end{pmatrix} \begin{pmatrix} x_I \\ x_\Gamma \end{pmatrix} = \begin{pmatrix} b_I \\ b_\Gamma \end{pmatrix},$$

where the block matrices are composed by independent local matrices

$$A_i = \begin{pmatrix} A_{ii} & A_{i\Gamma_i} \\ A_{\Gamma_i i} & A_{\Gamma_i\Gamma_i} \end{pmatrix}.$$

Furthermore we define a generic algorithm to assemble vector and matrix structures within an elementwise domain decomposition in a finite element code. Algorithm 4 shows a basic loop to compute entries of the residual vector and/or Jacobian matrix, needed in Newton's method as shown in Section 3.1.3. This algorithm is performed by each process on its local subdomain Ω_i in parallel. A closer look on Algorithm 4 shows that step iv/ requires communication only when the

Algorithm 4 Generic assembly loop for residual vector or Jacobian matrix entries.

```
given a set of cells $\{K_l\}$ in local subdomain $\Omega_i$
clear global data structure for residual vector or Jacobian matrix
for all cells $K_l$ in local subdomain $\Omega_i$ do
  i/ get degrees of freedom of actual cell: dof_ind
  ii/get values of solution vector $x$ according to dof_ind: local_sol
  iii/ compute local integral corresponding to dof_ind, e.g. for Jacobian matrix using solution
  of last Newton step local_sol

      $A_{i,j} = \int_{K_l} f(\texttt{local\_sol}; \varphi_h^j, \varphi_h^i)\, d\mathbf{x}$

  iv/ add local integral value to global data structure
end for
```

row block matrix/vector (4.15) is used, namely whenever one degree of freedom in `dof_ind` is not owned by the calculating process, this value has to be communicated. In contrast the pure local representation of global stiffness matrix A in form (4.6) allows to hold every data computed in step iii/ on the specific process.

Now we are able to define the *Schur complement storage scheme*: parallel matrix storage is organised by local subdomain block matrices corresponding to internal and sceleton dofs as in (4.6). For the parallel vector storage the internal degrees of freedom are obviously stored by the process that owns the related subdomain. In addition each process stores all sceleton degrees of freedom, that lie on $\partial\Omega_i \cap \Gamma$, which means that all sceleton degrees of freedom are stored at least two times. This vector storage necessitates a further data structure to manage the storage of

Geometry	degrees of freedom
2Dbackflow	2032003
2Dmeander	929923
3Dbackflow	3681748
3Dmeander	2732756

Table 4.1: Setting of testcases for the comparison of parallel data structures in 2D and 3D.

sceleton degrees of freedom - e.g. an assignment which process stores the original sceleton degree of freedom and which processes hold copies. Compare the right with the middle part of Figure 4.3 for the difference to the row block storage scheme.

Coming back to Algorithm 4. For Jacobian assembly we saw that the Schur complement storage scheme does not require any communication compared to the row block storage scheme. To compute the residual vector, step ii/ and iv/ involve communication for the row block storage scheme whenever an entry of `dof_ind` is not local. Since the Schur complement storage scheme stores all locally needed data (at least as a copy), step ii/ again does not need any communication. Only step iv/ is of interest in this case. After all local computations where performed, a single routine to update the sceleton degrees of freedom is inevitable - we will call this routine `sceleton_refresh`.

Remark 4.2.1
To use the introduced Schur complement storage scheme, it is not necessary to number the degrees of freedom as in (4.7)*, on the contrary any numbering might be used. One only needs data structures to identify the sceleton degrees of freedom and mappings from global numbering to local numbering of matrices* (4.6)*. These can cheaply be organised as local tables, while the global numbering is done.*

Summing up, the row block storage scheme needs communication whenever a degree of freedom on a cell is used that is not hosted by the process that holds this cell - this is the case for assembling routines as well as for algebraic routines, such as matrix-vector product. The Schur complement storage scheme on the other side only relies on one single synchronisation routine for the sceleton degrees of freedom. Especially the algebraic operations on a vector are of the same complexity for both schemes, i.e. only the scalar product requires communication between processes. At last we mention that also the matrix-vector product for Schur complement storage scheme can be performed in parallel and no a-priori communication is needed (in contrast to those for the offdiagonal-blocks in row block storage scheme). However the communication is postponed to the `sceleton_refresh` routine.

Numerical Results

We end this subsection with some numerical results indicating the capabilities of the presented parallel data structures. To judge the scalability in parallel algorithms one usually defines the *speedup* of an operation or an entire algorithm by the ratio of running time on one process (the sequential version) T_1 to the running time on n parallel processes T_n, i.e.

$$S = \frac{T_1}{T_n}.$$

Ideally the speedup S should be equal to the number of used processes n indicating perfect scalability. Since the presented parallel data structures (at least the Schur complement structure) are not meaningful in the sequential case, we adapt the definition of speedup by scaling with a factor of 2. By this we adjust the result to a pure comparison of parallel case and skip the sequential case while the usual notation can still be used.

In detail we compared the typical operations needed within Algorithm 4, namely

- parallel Jacobian matrix assemble - divided into `InitJac` for the initialisation of nonzero structure and `ComputeJac` for filling the matrix by integration of local parts,

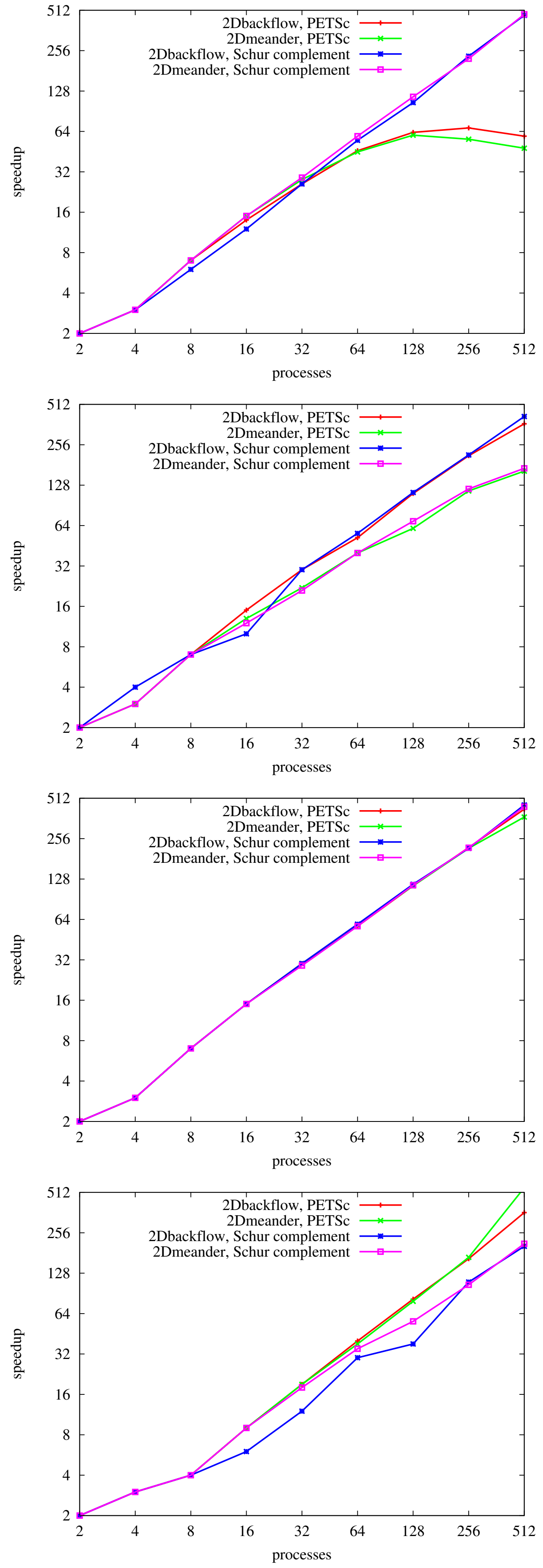


Figure 4.4: Scalability of row block storage and Schur complement storage approach in the 2D case - from top to bottom: `InitJac`, `ComputeRes`, `ComputeJac` and `MVmult`

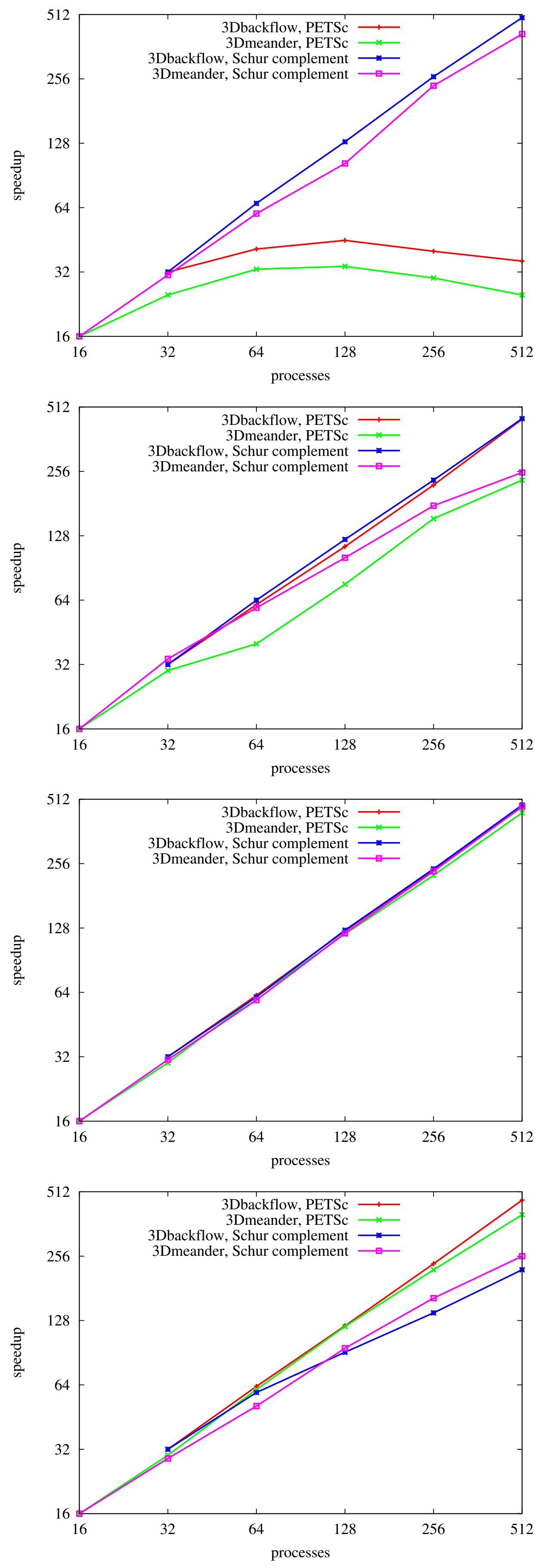


Figure 4.5: Scalability of row block storage and Schur complement storage approach in the 3D case - from top to bottom: `InitJac`, `ComputeRes`, `ComputeJac` and `MVmult`

- parallel residual vector assemble `ComputeRes` - local integration and communication of non-local entries resp. `sceleton_refresh`,
- parallel matrix-vector multiplication `MVmult`.

The row block storage scheme was implemented by means of the PETSc library [7, 8], while the Schur complement storage was implemented autonomous within the software `HiFlow`^{2}. Results presented in Figures 4.4 and 4.5 are based on the geometries and according discretisation as for the sequential solver, except for the level of refinement - here we used finer meshes as depicted in Table 4.1. Partitioning of data is based on the entire decomposition of the finite element mesh, which was generated by a graph partition of the cell-neighbourhood graph provided by Scotch library [122, 123]. The block partition of data is realised by *first-come-first-serve* approach (cf. Figure 4.3): we assign the degrees of freedom on the sceleton to the process with lowest rank, which obviously produces a small imbalance of data distribution on the processes but allows for easier handling of degrees of freedom.

Remark 4.2.2 (Hardware configuration for numerical tests)
All numerical tests ran on the distributed memory parallel computer HP XC3000 at Steinbuch Centre for Computing (Karlsruhe Institute of Technology) using the GNU gcc compiler in version 4.4.3. On the HP XC3000 we used a varying number of the 288 compute nodes to test for scalability. Each node contains two Quad-core Intel Xeon X5540 CPU (Nehalem) which run at a clock speed of 2.53 GHz and has 24 GB of main memory. Setup of scalability tests was such that each CPU was equipped with 3 GB of main memory. The interconnect is an Infiniband 4X QDR with latency of about 2 microseconds and point to point bandwidth between nodes of more than 3100 MB/s.

As one can see in Figure 4.4 initialisation and computation of Jacobian matrix scale perfectly for the Schur complement approach due to the pure sequential setup of this data type. For the row block version we lose scalability within the initialisation of nonzero structure of Jacobian matrix, since communication is needed to detect the couplings for the offdiagonal block. These effects are even worse in the 3D case: having the number of global degrees of freedom of the same order of magnitude as in the 2D case, the ratio of interior to sceleton degrees of freedom is smaller resulting in more communication. An improvement of routines for determining the neighbours of one subdomain by using ghost layers of the mesh would allow for better scalability of the `InitJac` part, but there will always be communication needed when offblock entries are assembled.

For the assembling of the residual vector, i.e. `ComputeRes`, we again have a slight benefit for the Schur complement structure. All local integrals can be computed in parallel and only afterwards an exchange within the `sceleton_refresh` routine is needed, for which the communication can be tuned using subcommunicators of neighbouring subdomains. The row block structure on the other side necessitates communication before local integrals can be computed (to get all needed degrees of freedom) and afterwards (to sum up the local parts) - nevertheless the difference due to overhead of communication is not that noticeable in Figures 4.4 and 4.5. It should be mentioned that the two communication steps for the row block structure might also be condensed into one, if additional ghost layers of degrees of freedom are used such that each process knows about all needed local degrees of freedom (cf. [7] for an idea on this). By this approach one would get the analogue to the fact that in the Schur complement structure we store the local sceleton for each subdomain - hence the speedup of `ComputeRes` should be of the same order for both data structures.

Last point within the comparison of data structures is given by the matrix-vector multiplication. Here, the tuned PETSc routines are superior to the Schur complement approach. But, as we will present in the next section, the latter one enables more efficient preconditioners such that the total number of needed matrix-vector multiplications can drastically be reduced. In summary it can be said, therefore, that both data structures allow for scalable results of basic linear algebra routines within an finite element framework - with a slightly advantage in matrix assembling for the Schur complement approach.

Remark 4.2.3
From an algebraic point of view the block partitioning of parallel matrix and vector is a common

choice. Assuming no geometrical information linked to the sparse global matrix A, the data distribution to processes is usually done by means of the adjacency graph. Each process will hold a set of rows, which has been determined e.g. by minimising the intersection of adjacency edges [57] - such a partition is equivalent to a vertex-based decomposition of the underlying mesh. Its main drawback for finite element discretisation is obvious: couplings of degrees of freedom within a cell will be disrupted and the distinction between interior and sceleton nodes is no longer given. Nevertheless a preconditioning by inexact Schur complement solver can still be used at the costs of additional data structure management for the offdiagonal blocks in (4.15) *- see [136].*

4.3 Performance of Parallel Solvers/Preconditioners

In this section the performance of row block and Schur complement based preconditioners is investigated in conjunctions with the presented sequential Multilevel ILU preconditioner. Furthermore, we compare the possibility to use the ILU preconditioner within both data structures to some basic preconditioners provided by the PETSc library and also to parallel direct solver. All numerical test are based on the discretisation of two- and three-dimensional Stokes equations on the backflow geometry since this example turned out to be representative as it was also before. Additionally, we used the hardware configuration as depicted in Remark 4.2.2.

4.3.1 Usage of Multilevel ILU Preconditioner in Parallel

For the presented Schur complement data structures we already specified how the sequential multilevel ILU decomposition can be used to form a parallel preconditioner (and even a solver). In addition also the row block partition of data allows to use the results of Chapter 3.2, namely by means of a parallel block Jacobian preconditioner. We summarise the three major cases:

Global Block Jacobi Preconditioner

The preconditioner P_{BJ} can be used upon the parallel data structure (4.15) by skipping the couplings $A_{ij},\ i \neq j$ and using the ILU decomposition of A_{ii}. Doing so the preconditioner operation $y \leftarrow P_{BJ}x$ is purely sequential and will therefore scale perfectly, but by skipping the offdiagonal coupling one will loose information the more processes are used. Threshold and used preprocessing for the local matrices A_{ii} are the parameters to be set like in the sequential case of Chapter 3.2. Application of the block Jacobian ILU preconditioner P_{BJ} necessitates a modification in standard `PETSc` routines since the diagonal block cannot be accessed directly – fortunately these steps turned out to be not that expensive. Nevertheless, an implementation using own data structures for the global matrix might allow for even better results.

Global Schur Complement Preconditioner

Solving the Schur complement equation (4.8) with an inexact local solver for A_{ii}^{-1}, namely an ILU decomposition, gives a preconditioner P_{SC} for the global system $Ax = b$ – see Algorithm 3. This preconditioner provides a variety of parameter to tune the performance, i.e.

- threshold and preprocessing for the ILU decomposition of A_{ii},
- convergence criteria for the iterative solver of Schur complement equation $S_\Gamma x_\Gamma = \chi_\Gamma$,
- additional preconditioner for S_Γ (this would be a preconditioner for the preconditioner).

Compared to the block Jacobian preconditioner P_{BJ} this preconditioner preserves the couplings between subdomains by approximatively solving the Schur complement equation, such that the application $y \leftarrow P_{SC}x$ will be costlier due to needed communication routines. A comparable approach was already used in [52] for the compressible Euler equations. Therein the local system matrices A_{ii} were only approximated by means of ILU(0) decomposition for which we already showed the inferior behaviour compared to Multilevel ILU preconditioner.

Preconditioner for Schur Complement Solver

In (4.11) we defined a preconditioner P_Γ for the Schur complement equation

$$S_\Gamma x_\Gamma = \chi_\Gamma.$$

Again one can use all results on basis of the sequential preconditioner for local matrices A_{ii}. We will combine this preconditioner with an exact solver for A_{ii}^{-1} to get a preconditioned Schur-complement solver.

The last preconditioner is, in contrast to the first two preconditioners, not applied to the global system (4.7) resp. (4.15). Since especially the global Schur complement preconditioner might differ in each utilisation, we cannot use standard `GMRES(m)` implementation any more. Instead we throughout use the Flexible-GMRES `FGMRES(m)` with restart in the parallel case, which allows to vary the preconditioner – described in detail in [56, 132].

4.3.2 Numerical Results

Results obtained for the Schur complement approach and Multilevel ILU preconditioner (in the following denoted by `ILU++` due to the used software) were compared to some established solvers and preconditioners. We judge the results in two ways: first the absolute time for solving the linear system, second the scalability. The results for standard preconditioners were directly obtained using the PETSc library [7, 8] together with the block matrix storage scheme presented before. Based on the literature and extensive numerical tests we compared against Block Jacobi and Additive Schwarz (with overlap of 4) preconditioner – both with local `ILU(0)` factorisation in the 3D case and local `ILU(1)` factorisation in the 2D case. These preconditioners were reported to be standard in the PETSc library and already showed good performance for *hp* finite element discretisation of incompressible flows [14].

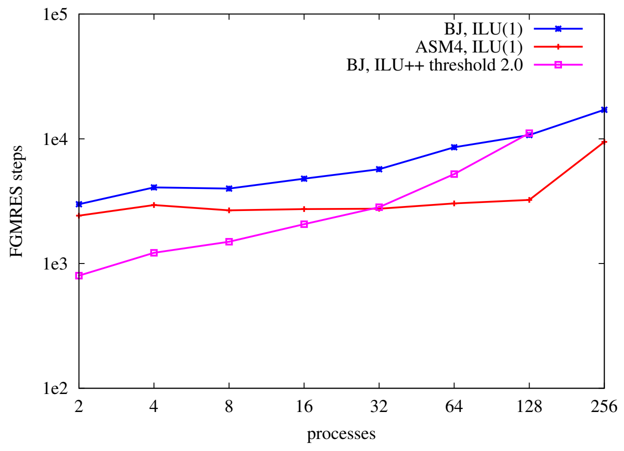
Global Block Jacobi Preconditioner

As one can see in Figure 4.6 for the 2D case and in upper row of Figure 4.7 for the 3D case, the results for preconditioner P_{BJ} with local `ILU++` factorisation are superior to the ones with local `ILU(k)` factorisation – this fact was already observed for the sequential case in Chapter 3.2. Also the choice of threshold for best, i.e. fastest, performance is quite similar to the sequential case (lower row in Figure 4.7) where a threshold of $\tau \approx 2.0$ gave good overall results. But the outstanding result is given by the fact that Block Jacobian preconditioner nearly throughout fails (and therefore we skip the graphical evaluation) for problems of size of order larger than $5 \cdot 10^5$, i.e. needed `FGMRES`-steps are beyond 10^5 and overall solution is not worth discussing. Especially comparison to the Distributed Multifrontal Solver MUMPS [4] shows the poor performance of iterative solvers in the 2D case (Figure 4.6) – only scalability of iterative solver is somewhat better but still not optimal. In terms of absolute time to solve the linear system, the direct solver, as in the sequential case, is superior for two dimensional problems but worse in three dimensions (Figure 4.7). Nevertheless, the direct solver suffers from its large memory requirement (that might not be present when the number of processes is small) and bad scalability.

In terms of scalability the Additive Schwarz preconditioner was chosen to establish some exchange of couplings which is not present for the Block Jacobian preconditioners. This advantage can be seen in the left part of Figure 4.6 and 4.7 where the number of `FGMRES` steps does not increase that much when the number of processes increases (compared to Block Jacobian preconditioners). Summing up, we found that for the global Block Jacobian preconditioner the same results regarding local ILU decomposition are given as for the sequential case. But the entire approach by block preconditioning suffer from the skipping of couplings in offdiagonal block which then has to be compensated by additional global `FGMRES` steps.

Preconditioner for Schur Complement Solver

Motivated by the absolute solver time of the direct solver MUMPS, we investigated on the performance of the iterative Schur complement solver. Equation (4.8) is solved via a `GMRES` iteration



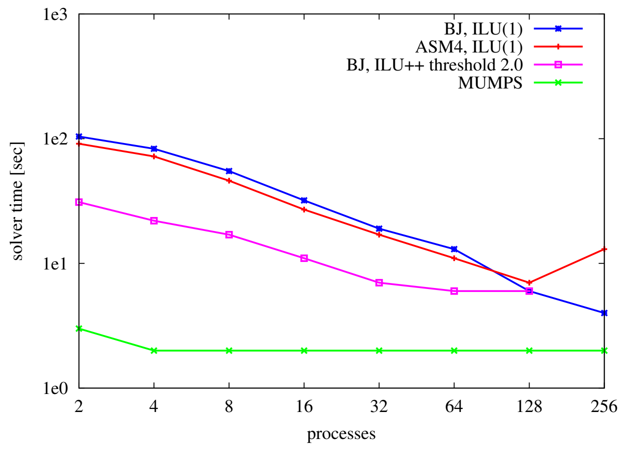


Figure 4.6: Performance of row block data structure based preconditioner compared to direct solver MUMPS – 2Dbackflow problem with 127843 degrees of freedom, global `FGMRES` steps top, solver time bottom.

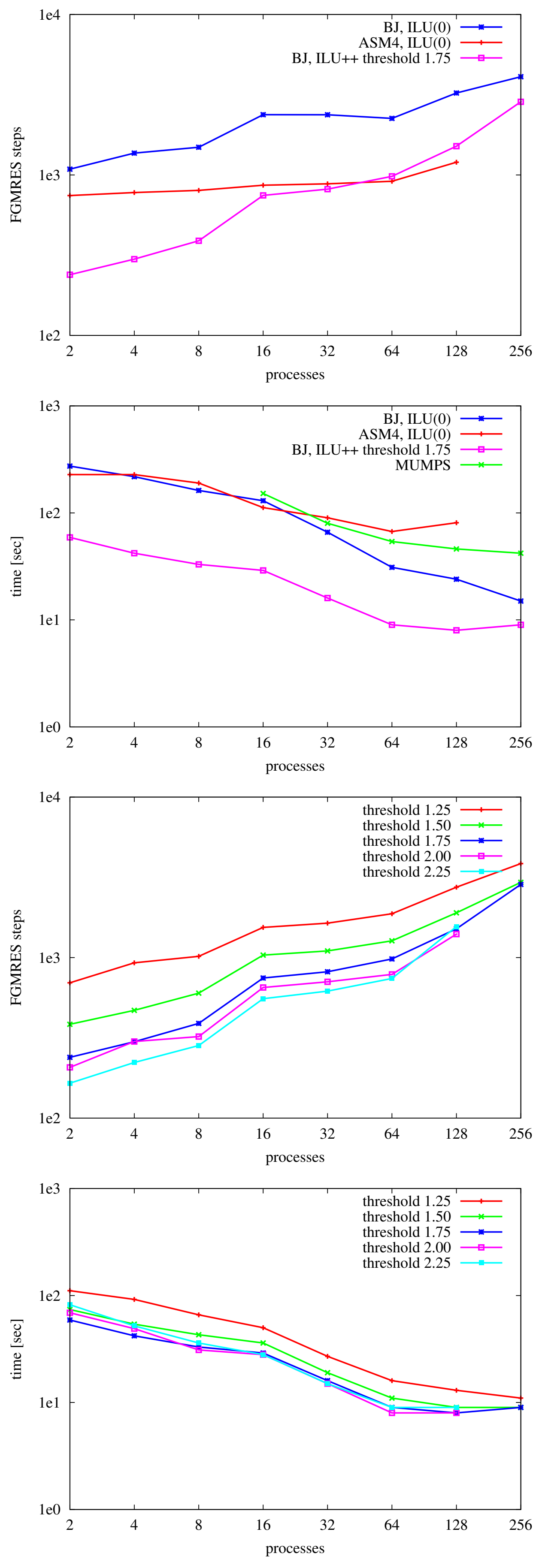


Figure 4.7: Performance of row block data structure based preconditioner with local Multilevel ILU factorisation and comparison to direct solver MUMPS – 3Dbackflow problem with 472748 degrees of freedom.

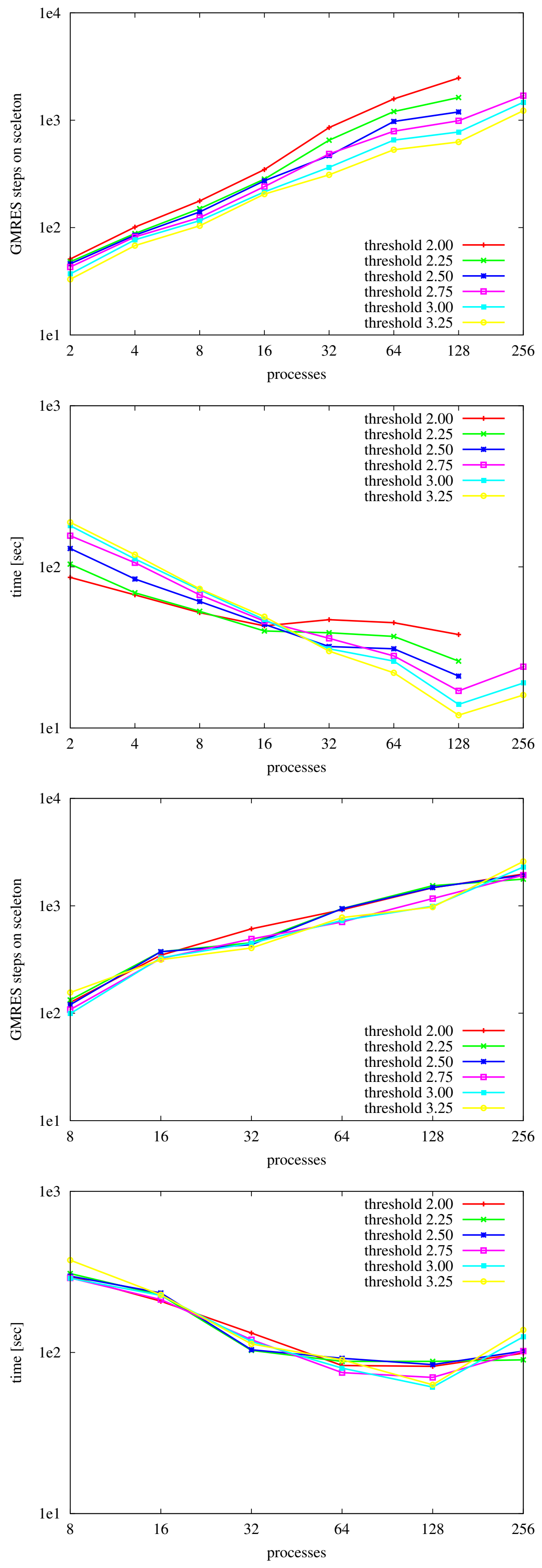
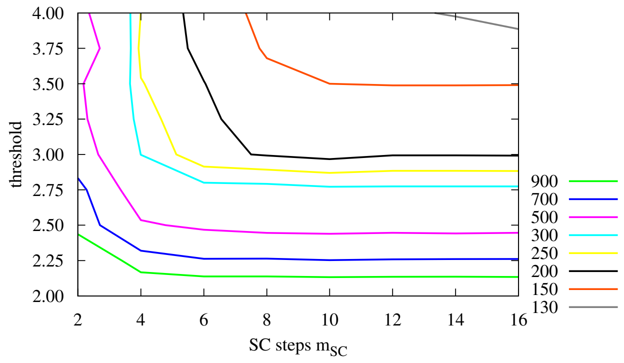


Figure 4.8: Performance of Schur complement solver using preconditioner P_Γ with different threshold – two top plots: 2Dbackflow problem with 509123 degrees of freedom – two bottom plots: 3Dbackflow problem with 472748 degrees of freedom.



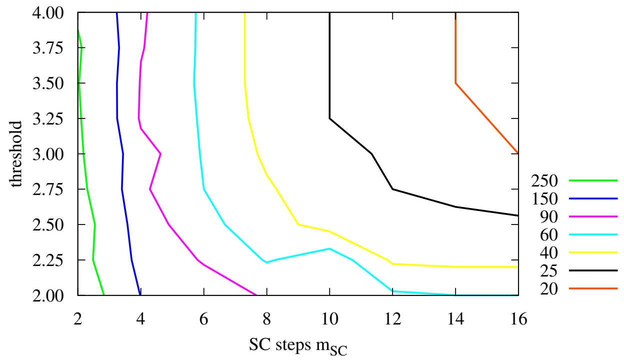


Figure 4.9: Contour map of global `FGMRES` steps for different parameter settings of Schur complement preconditioner P_{SC} on 16 processes. 2Dbackflow problem with 509123 degrees of freedom (top) and 3Dbackflow problem with 472748 degrees of freedom (bottom).

on the sceleton x_Γ. To this end, the local problems (i.e. A_{ii}^{-1} for the local Schur complement S_i) within equations (4.9) and (4.10) are treated by the direct solver UMFPACK [38]. Furthermore, we use the preconditioner P_Γ as in (4.11) with PQ-reordering for 2D problems, normalisation for 3D problems and different size of threshold for the incomplete multilevel factorisation $ILU(S_i)$. The results for 2D and 3D problems are shown in Figure 4.8: one can see that the more processes are used, the bigger the threshold should be. Whereas in the 3D case the influence of threshold for the preconditioner P_Γ is not that notable for different numbers of processes, we have in the 2D case a clear turning point when one uses more than 16 processes. All in all the iterative Schur complement solver with local exact solver for A_{ii}^{-1} was not competitive to the presented solver so far, neither in terms of absolute time to solve the system nor in terms of scalability (see overall comparison in Figure 4.12). Especially for the scalability the imbalance of sceleton degrees of freedom to interior degrees of freedom is important the more processes are used.

Global Schur Complement Preconditioner

At last we study the global Schur complement preconditioner P_{SC} as given in Algorithm 3 – again we use a global `FGMRES` iteration as accelerator. This preconditioner allows to modify a variety of parameters and we only inspected the behaviour when the number of iterations m_{SC} within the inexact Schur complement solver and also the threshold for incomplete factorisation of A_{ii} are changed. The preconditioner for the Schur complement operator S_Γ was not under investigation (see Figure 4.8 for this). Instead we chose a fixed setting that allows cheaply usage, namely a threshold of $\tau = 1.75$ throughout.

Testing the wide range of suitable parameters would go beyond the scope of this thesis and we only present the possibilities for setting on 4, 16 and 64 processes both for the 2D and 3D case as before. To get a feeling for the behaviour of the solver for different parameter we show the results for global `FGMRES` steps and needed time to solve the system in contour plots. Results for global `FGMRES` steps are similar for every number of used processes (see Figure 4.9 for the case of 16 processes): the more precise the Schur complement equation is solved, the less global `FGMRES` steps are needed. This can be achieved either by increasing the threshold for incomplete factorisation of A_{ii} or by increasing the number of iterations m_{SC} for the iterative Schur complement solver.

For the results in terms of absolute solver time, we did not get such clear statements. As depicted in Figures 4.10 and 4.11 the best setting of parameters threshold τ and steps m_{SC} strongly depends on the number of used processes. A rule of thumb might be given by: the more processes are used, the more threshold τ and the more iterative steps m_{SC} should be used. Nevertheless, the performance of preconditioner P_{SC} is very sensitive to choice of parameters – already a modification of e.g. threshold by 0.5 might cause a doubling of solver time. Heuristical observations were given by the fact that the ratio of iterative steps m_{SC} to number of sceleton degrees of freedom N_Γ should remain constant for each number of processes. It will be a future work to adjust the parameters of P_{SC} automatically using the information given by partition of finite element mesh.

4.3.3 Comparison of all Solvers

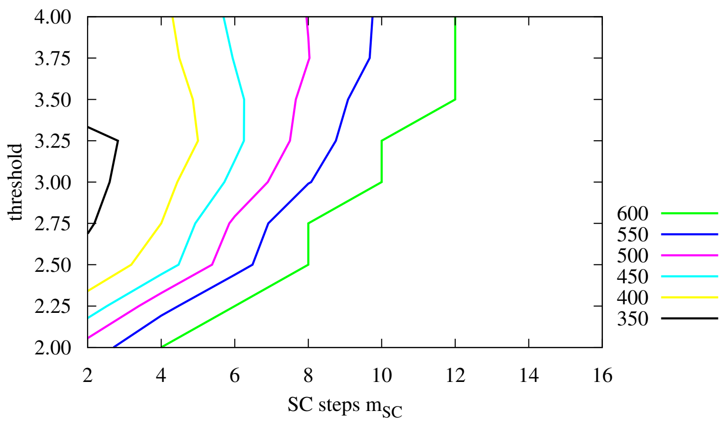
Finally we gather the results achieved for the presented parallel solver/preconditioner, i.e. we compare the direct solver MUMPS, Schur Complement solver, global `FGMRES` solver with Additive Schwarz preconditioner, with Block Jacobian preconditioner P_{BJ} and with Schur complement preconditioner P_{SC}. The outcomes for absolute solver times differ a lot for the 2D and 3D case – see Figure 4.12. In the 2D case the best performance is given by direct solver based methods, i.e. the MUMPS and Schur complement solver. Results for iterative solvers can be separated into two classes: whereas the solver with preconditioners P_{SC} and ASM at least solved the problem, the usage of Block Jacobian preconditioner P_{BJ} took throughout more than $1.e5$ iterations and did not reach the solution in reasonable time. But it has to be mentioned that the P_{SC} preconditioner showed a fatal performance for more than 64 processes – number of `FGMRES` steps and solution time are worth discussing. A possible explanation is given by the worse ratio of interior to sceleton degrees of freedom. The costs for inexact Schur complement solver are then dominated by the restriction and prolongation operations that need increasing communication.

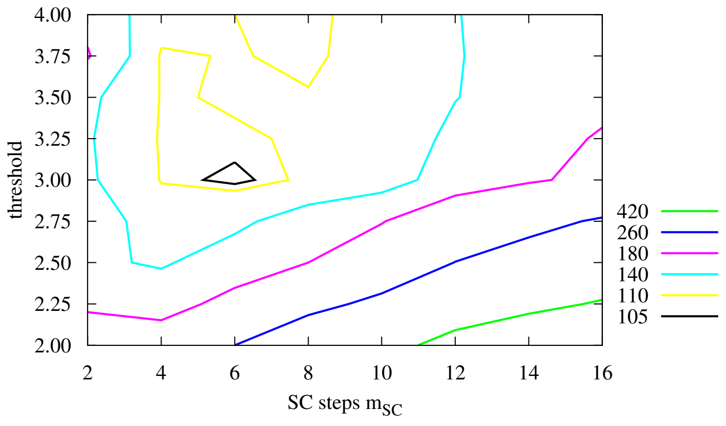
In the 3D case it is the other way around. Here the iterative `FGMRES` solver with block `ILU++` preconditioner gave the best results. This discrepancy to the two dimensional case was already observed in Chapter 3.2 for the sequential solver where the bigger fill-in in three dimensions was found to be crucial for direct solvers. Iterative solvers obtain a slightly better scalability and need definitely less memory compared to direct solvers which fit to actual supercomputer architectures. It should be mentioned that the Block Jacobian preconditioner P_{BJ} yields better results than both preconditioners P_{SC} and ASM although only the latter ones take the couplings of subdomains into account. These couplings also come into play by global matrix-vector multiplications as can be seen in left of Figure 4.12. For the pure sequential preconditioner P_{BJ} one obviously needs a higher number of `FGMRES` steps, whereas especially the Schur complement preconditioner P_{SC} allows for a nearly constant number of `FGMRES` steps independent of the number of processes.

Summing up, we conclude that for considered three dimensional problems iterative solvers are more suitable than direct solvers. Both approaches, row block partition and Schur complement approach, yield suitable parallel data structures and the possibility to use preconditioners based on Multilevel ILU decomposition, which showed a better performance than standard preconditioners. However, the scalability is only weak and the setting of appropriate parameters, especially for the P_{SC} preconditioner, is tricky. But the presented results cause to consider in a future work an optimised implementation of data structures and related preconditioners to obtain better results.

4.3.4 Comparison of Complete Parallel Approach

As a very last result on parallel data structure and parallel preconditioner we show the capabilities of preconditioners P_{BJ} and P_{SC} in combination with according data structures for large scale





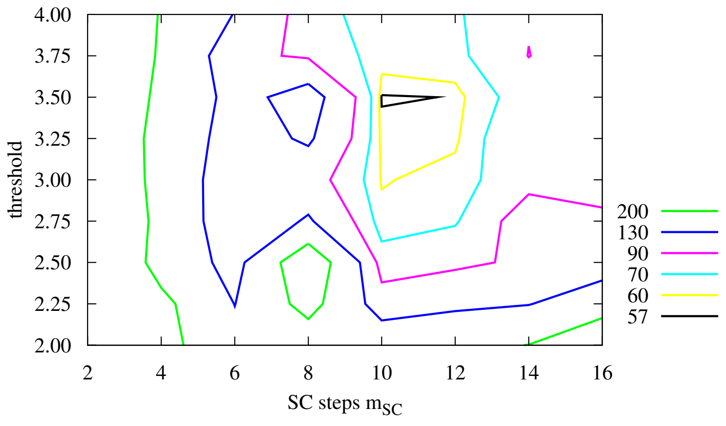
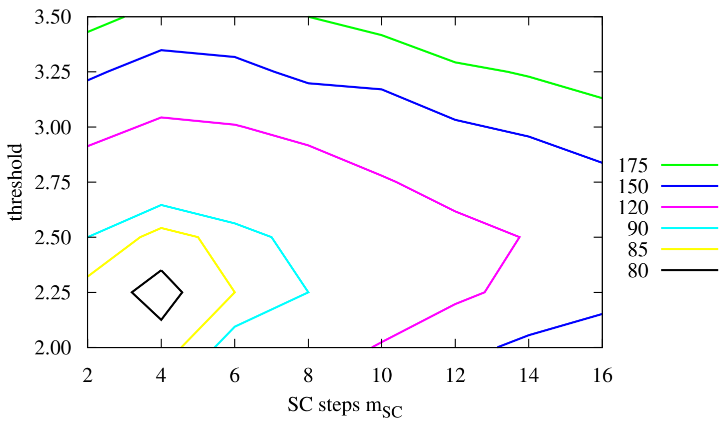
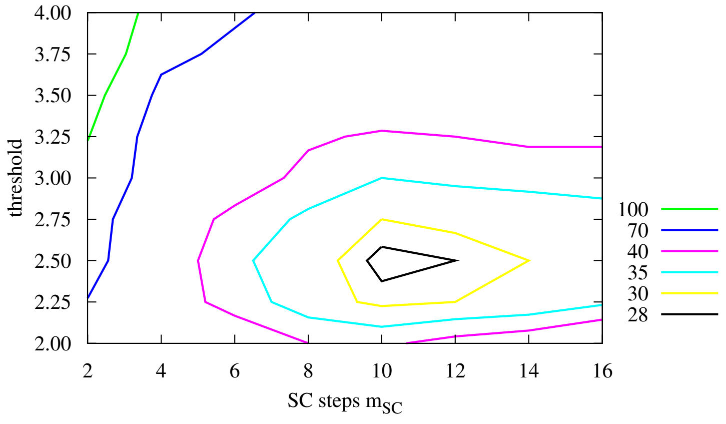


Figure 4.10: Contour map of absolute solver time for different parameter settings of Schur complement preconditioner P_{SC} on 2Dbackflow problem with 509123 degrees of freedom – 4 processes (top), 16 processes (middle) and 64 processes (bottom).





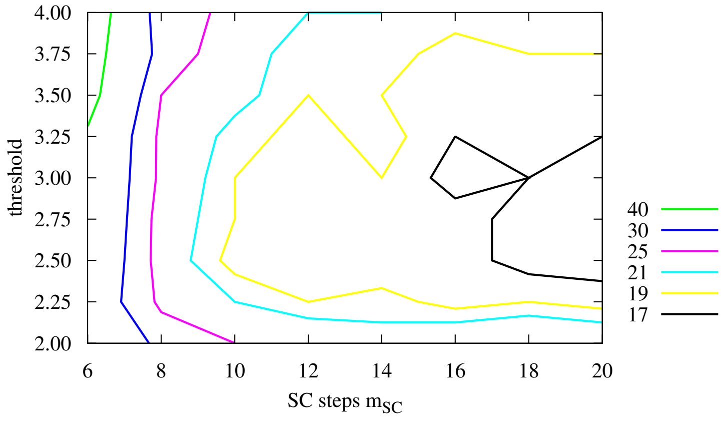


Figure 4.11: Contour map of absolute solver time for different parameter settings of Schur complement preconditioner P_{SC} on 3Dbackflow problem with 472748 degrees of freedom – 4 processes (top), 16 processes (middle) and 64 processes (bottom).

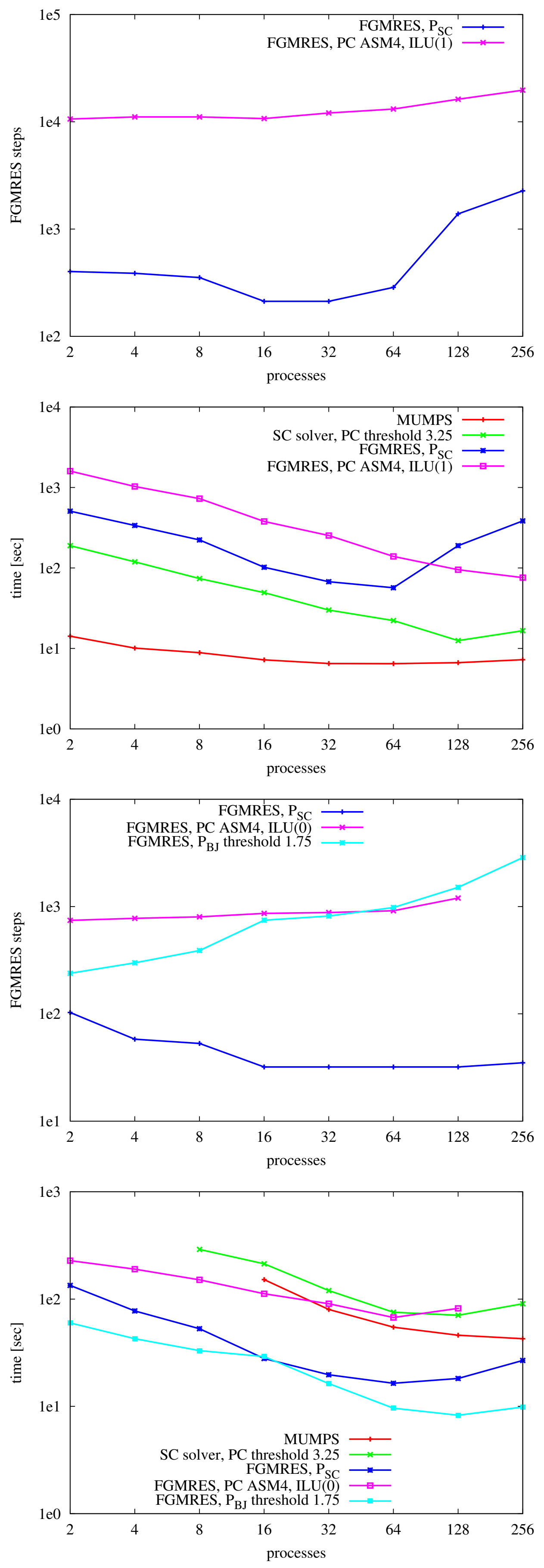


Figure 4.12: Comparison of `FGMRES` steps and absolute time of several linear solvers for 2Dbackflow problem with 509123 degrees of freedom (two top plots) and 3Dbackflow problem with 472748 degrees of freedom (two bottom plots).

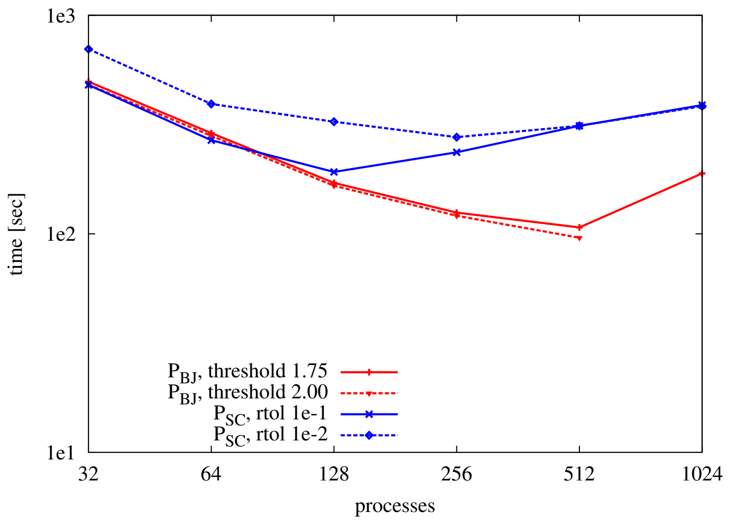


Figure 4.13: Comparison of absolute time to solve Stokes equations using P_{SC} and P_{BJ} for 3Dbackflow problem with 3681748 degrees of freedom. Missing values are due to reached maximum number of iterations.

problems. We aim at comparing the overall solution time for Navier-Stokes equations on the three dimensional backflow geometry, including assembly of Jacobian matrix as well as the linear solver. Furthermore, we make use of the adaptive forcing strategy shown in Chapter 3.2. For these tests on scalability the high-performance computer JUROPA-JSC at Forschungszentrum Jülich equipped with 2208 nodes was used. Each node contains two Quad-core Intel Xeon X5570 CPU (Nehalem-EP) which run at a clock speed of 2.93 GHz and has 24 GB of main memory. Setup of scalability tests was such that each CPU was equipped with 3 GB of main memory. The interconnect is an Infiniband QDR with non-blocking Fat Tree topology. In contrast to the previous test on HP XC3000 we no longer used GNU gcc compiler but the Intel C++ compiler in version 11.1.059 as recommended by the JUROPA user guide.

Discretisation of the 3D problem on a finite element mesh with 143360 cells results in a total number of 3681748 degrees of freedom. Therefore the parallel solver needed at least 32 processes – a lower number of processes resulted in memory allocation errors, showing that problems of this complexity necessitate the use of modern parallel systems. First we tested the parameter setting of preconditioner P_{BJ} and P_{SC} solving Stokes equations. The threshold for block Jacobian ILU decomposition was 1.75 and 2.00, while for the inexact Schur complement solver we used a local threshold of 3.0. Instead of specifying an upper bound m_{SC} on the iterative steps for Schur complement equation solver (as done before), we used a relative tolerance of $1e-1$ and $1e-2$ for the residual norm on the sceleton as termination criterion. Figure 4.13 shows that the Block Jacobian preconditioner possesses better scalability properties than the Schur complement preconditioner. Especially the sensitivity of Schur complement preconditioner to parameter setting is visible. In contrast to P_{BJ} preconditioner, a good setting of parameter for P_{SC} preconditioner is essential and might even result in best absolute time.

We then used this setting for the linear solver to solve the Navier-Stokes equations. Results are presented in Table 4.2 and Figure 4.14 indicating that the linear solver is the crucial point of scalability for the entire nonlinear solver. While the assembling of Jacobian matrix shows good scalability properties, the linear solver avoids a good overall result. Nevertheless, the adaptive forcing allows to reduce the time spent for linear solver to a minimum at costs of extra Jacobian matrix assembling, which facilitates scalability of overall solution process.

	number of processes					
	32	64	128	256	512	1024
Block Jacobian preconditioner P_{BJ}						
final residual norm	8.35e-12	1.66e-11	1.0e-12	9.98e-13	1.0e-12	9.96e-13
overall solution time	999	532	306	189	131	211
mean Jacobian assemble time	92.9	47.8	24.4	12.5	6.5	4.0
mean FGMRES steps	269	332	455	653	822	1649
mean linear solver time	68	37	23	17	13	28
Schur Complement preconditioner P_{SC}						
final residual norm	8.66e-13	2.35e-12	5.79e-13	6.83e-13	7.41e-13	
overall solution time	1752	981	776	730	1303	>1800
mean Jacobian assemble time	93.4	46.7	23.5	11.8	6.5	3.0
mean FGMRES steps	16	13	9	9	15	
mean linear solver time	194	114	105	109	210	

Table 4.2: Results of row block and Schur complement approach for Navier-Stokes solver within 3Dback-flow geometry – obtained on supercomputer JUROPA.

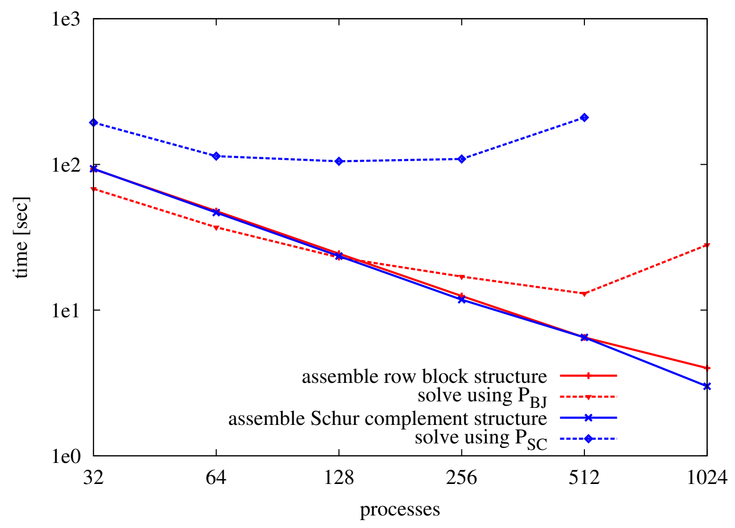


Figure 4.14: Comparison of mean absolute time to assemble and solve Jacobian of Navier-Stokes equations using P_{SC}, P_{BJ} and according parallel data structures.

Chapter 5

Optimisation Approach on HPC Systems

After detailed derivation of generic sequential and parallel solver for the primal and adjoint/sensitivity equation, we now come back to the entire optimisation problem. Since both optimisation approaches described in Chapter 2.3 are based on determining the gradient of the reduced cost functional, i.e. $\frac{Dj}{Du}(u) = \nabla j(u)$, at this stage we can work out a framework on the routines for evaluation of cost functional $j(u) = J(S(u), u)$ and gradient $\nabla j(u) = \frac{DJ}{Du}(S(u), u)$.

In the following we emphasise consequences for the special case of instationary PDE-constrained optimisation implemented on HPC-systems. This work bases on the contribution [21] of the author for the PARA2008 workshop. For the adjoint based optimisation approach we are faced with the discrepancy that implicit solvers allow for less timesteps but are costly (see Chapter 4.3). On the other hand explicit solvers need lots of timesteps that are very cheap but require the overall backup of state solution, which motivates the usage of checkpointing schemes and parallel I/O. For the sensitivity based optimisation approach there is no need to compute the adjoint solution backward in time. Instead the partial derivative for each control parameter can be computed within the forward sweep.

5.1 Gradient-based Optimisation Algorithm

We already showed in Chapter 2.3 that a gradient-based optimisation algorithm for PDE-constrained problems requires the solution of state- and adjoint-/sensitivity-equations to determine the gradient (first order optimality condition). This procedure is shown in Figure 5.1 for the case of instationary flow optimisation by boundary control - the reader might refer to Chapter 2.3.2 for a detailed notation. The main difference within the adjoint and sensitivity approach for determination of gradient ∇j is the number of linear/nonlinear systems to be solved as well as the storage requirements to be fulfiled. Before we report on the used optimisation algorithm to determine the update for control variables, we compare the needs to compute the gradient for stationary and instationary problems. Let's assume that the sensitivity-based representation of control u is given by design parameter $\alpha_k,\ k = 1, \dots, n$.

For a stationary problem one iteration step of the schematic procedure 5.1 requires for an adjoint and also for a sensitivity-based method:

i/ one nonlinear system of the state equations to be solved,

ii/ backup of state solution,

iii/ for adjoint method: one linear system of the adjoint equations to be solved using backup of state solution,

iii/ for sensitivity method: n linear systems of sensitivity equations to be solved using backup of state solution,

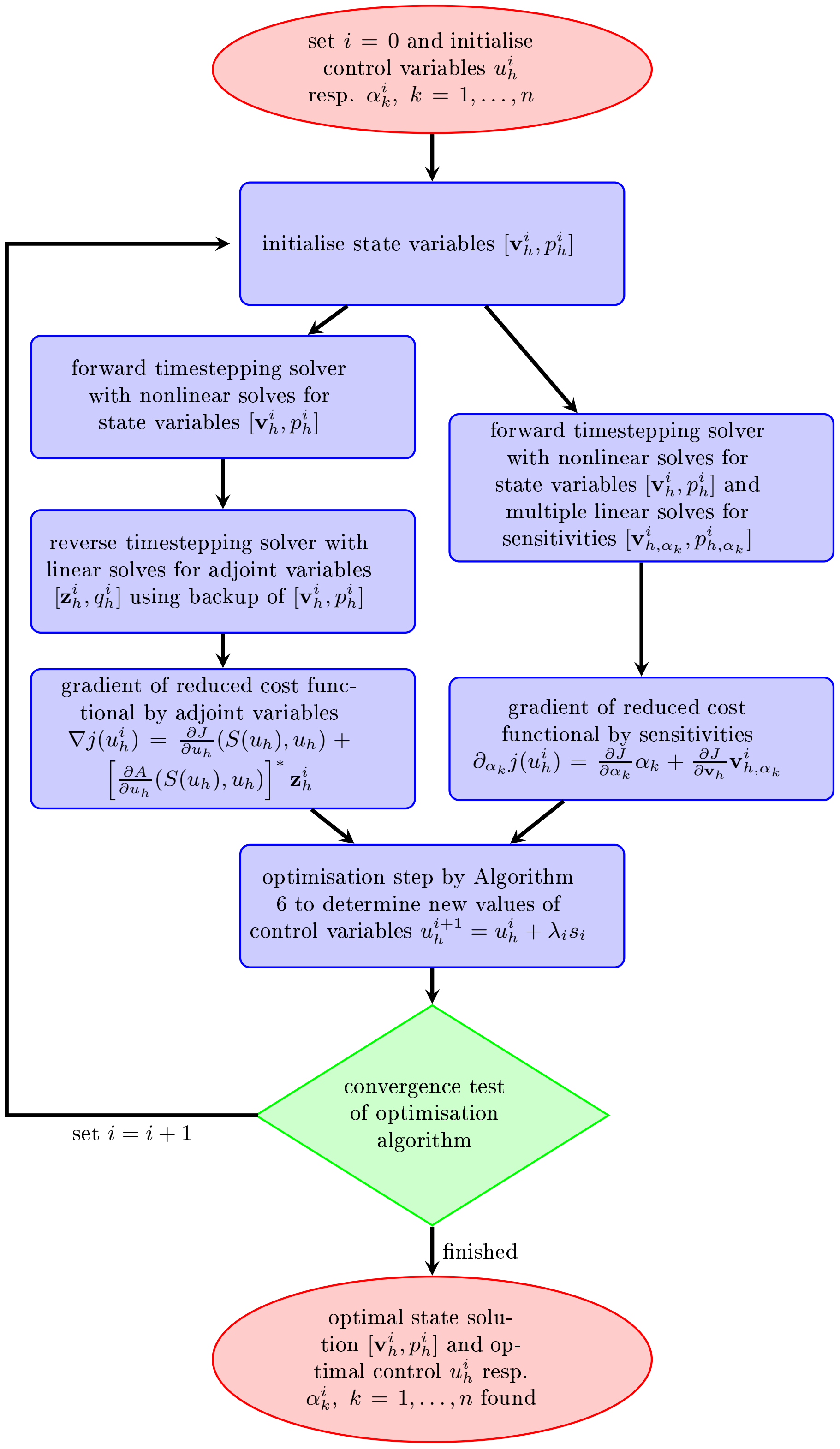

set $i = 0$ and initialise control variables u_h^i resp. α_k^i, $k = 1, \dots, n$
initialise state variables $[\mathbf{v}_h^i, p_h^i]$
forward timestepping solver with nonlinear solves for state variables $[\mathbf{v}_h^i, p_h^i]$
reverse timestepping solver with linear solves for adjoint variables $[\mathbf{z}_h^i, q_h^i]$ using backup of $[\mathbf{v}_h^i, p_h^i]$
gradient of reduced cost functional by adjoint variables $\nabla j(u_h^i) = \frac{\partial J}{\partial u_h}(S(u_h), u_h) + \left[\frac{\partial A}{\partial u_h}(S(u_h), u_h)\right]^* \mathbf{z}_h^i$
forward timestepping solver with nonlinear solves for state variables $[\mathbf{v}_h^i, p_h^i]$ and multiple linear solves for sensitivities $[\mathbf{v}_{h,\alpha_k}^i, p_{h,\alpha_k}^i]$
gradient of reduced cost functional by sensitivities $\partial_{\alpha_k} j(u_h^i) = \frac{\partial J}{\partial \alpha_k}\alpha_k + \frac{\partial J}{\partial \mathbf{v}_h}\mathbf{v}_{h,\alpha_k}^i$
optimisation step by Algorithm 6 to determine new values of control variables $u_h^{i+1} = u_h^i + \lambda_i s_i$
convergence test of optimisation algorithm
set $i = i + 1$
finished
optimal state solution $[\mathbf{v}_h^i, p_h^i]$ and optimal control u_h^i resp. α_k^i, $k = 1, \dots, n$ found

Figure 5.1: Schematic prodecure of adjoint- and sensitivity-based optimisation algorithm for discretised instationary flow problems.

iv/ evaluation of gradient by standard algebraic operations.

We see that the adjoint-based method in case of stationary problems always requires 1 nonlinear system and 1 linear system to be solved, while the sensitivity-based method requires the more linear system solves the more design parameter α_k are used. This fact obviously advocates for the adjoint-based approach, but for instationary problem the position turns over. For an instationary problem using a temporal discretisation of m timesteps, one iteration step of the schematic procedure 5.1 requires for an adjoint-based method:

i/ m nonlinear systems of the state equations to be solved forward in time,

ii/ backup of m state solutions at all timesteps,

iii/ m linear systems of the adjoint equations to be solved backward in time using backup of state solutions,

iv/ evaluation of gradient by standard algebraic operations.

Whereas for the sensitivity-based method we need:

i/ one nonlinear system of the state equations to be solved per timestep (totally m for all timesteps),

ii/ backup of actual state solutions at actual timestep,

iii/ n linear systems of sensitivity equations to be solved at actual timestep using backup of actual state solutions,

iv/ evaluation of gradient by standard algebraic operations.

Hence, the sensitivity-based method does not require the separation in forward and backward solver and does not need to backup all m state solutions - instead only the actual state solution is needed in every timestep. For the solver we count for both methods m nonlinear system solves and for the adjoint based method m linear system solves compared to $m \cdot n$ linear system solves for the sensitivity based method. To overcome the problem of full backup of state solution for adjoint based method, we will introduce checkpointing schemes in the sequel which lead to additional nonlinear system solves for the state equations. Since this problem is not given for the sensitivity based method, we can treat this approach like a pure simulation task having more than one system to be solved in each timestep - so in the sequel we concentrate on techniques for the adjoint-based method only.

Parallelism within the Scheme 5.1 can be implied for all solver steps by means of parallel solvers as presented in Chapter 4. Furthermore the optimisation algorithm, which will be figured out next, possesses opportunities for parallelism since mainly basic algebraic operations are to be peformed like matrix-vector product or dot-product. The optimisation algorithm might use the parallel data structures of Chapter 4.2, but for most cases the data distribution of the control variable is already given by the domain decomposition approach. A reordering of control data is often not gainful and for the sensitivity-based optimisation the number of control variables is such small that even a sequential optimisation algorithm can be used.

Quasi-Newton Methods

The rough estimation on storage requirements done before for instationary PDE-constrained optimisation problems needs to be extended by an estimation of the size of the entire optimisation problem. In case of sensitivity-based optimisation, the size of optimisation problem, i.e. the size of discretised control u and also of gradient $\nabla j(u)$, equals $m \cdot n$. When explicit timestepping is used such that $m \approx 10^5$, these problems are of size 10^6 and have to be regarded as *large scale optimisation problems.* In case of adjoint-based optimisation, the optimisation problem will also be of size $m \cdot n$, but now n denotes the spatial discretisation of control u. For the finite element based discretisation of boundary control problems under consideration this equals the number of degrees of freedom of control u_h on boundary Γ_c - on 3D meshes one easily ends up with $n \approx 10^3$.

Even when implicit timestepping is used to keep down the number of timesteps, say $m \approx 10^2$, the optimisation problem is of size 10^5 and therefore again a *large scale optimisation problem.*

The first order optimality condition for the reduced cost functional was given by (cf. (2.31))

$$\langle \nabla j(\overline{u}), u - \overline{u} \rangle_{U^*,U} \geq 0 \qquad \forall u \in U_{ad}$$

and can by rewritten as

$$\nabla j(u) = 0 \qquad \forall u \in U$$

whenever $U_{ad} = U$, i.e. no additional side constrains are given for the admissible controls. How to compute the gradient $\nabla j(u)$ was addressed by adjoint and sensitivity equations such that we now need to concentrate on large scale optimisation algorithm. Since by the reduced cost functional we can view the problem as an unconstrained optimisation problem, we use a general approach by means of Newton's method in several variables for unconstrained optimisation. Furthermore the problem is treated as already discretised, i.e. we identify the control u_h by the values of degrees of freedom. Find the control $u_h = [u_1, ..., u_{m \cdot n}] \in \mathbb{R}^{m \cdot n}$ such that the cost functional $j(u_h) = J(S(u_h), u_h)$ is a minimum. Standard approximation of $j(u_h)$ by the first three terms in a Taylor series expansion at actual iterate u_h^i yields (see e.g. [96]):

$$j(\overline{u}_h) = j(u_h^i) + \nabla j(u_h^i)(\overline{u}_h - u_h^i) + \frac{1}{2} B(u_h^i)[\overline{u}_h - u_h^i, \overline{u}_h - u_h^i]$$

with the symmetric Hessian matrix

$$B(u_h^i) = \nabla^2 j(u_h^i).$$

Define the search-direction $s = \overline{u}_h - u_h^i$ and rewrite the expansion as

$$j(\overline{u}_h) = j(u_h^i) + \nabla j(u_h^i)s + \frac{1}{2} B(u_h^i)[s, s] \tag{5.1}$$

where the term $\nabla j(u_h^i)s$ is the directional derivative along s and the term $H(u_h^i)[s, s]$ is called curvature or second directional derivative in the direction s. For the minimum u_h^* one has the following observations (see [41])

$$\begin{aligned} j(\overline{u}_h) &> j(u_h^*) && \forall \overline{u}_h \in \mathbb{R}^{m \cdot n}, \\ \nabla j(u_h^*)s &= 0 && \forall s \in \mathbb{R}^{m \cdot n}, \\ B(u_h^*)[s, s] &> 0 && \forall s \in \mathbb{R}^{m \cdot n}. \end{aligned}$$

For Newton's method one choses the search direction

$$s_i = -B(u_h^i)^{-1} \nabla j(u_h^i),$$

which is the minimiser of the quadratic model (5.1) to get the new approximation

$$u_h^{i+1} = u_h^i + s_i.$$

This method is based on finding a zero of the gradient vector (first order neccessary optimality condition) and there is no guarantee that the step will move towards a local minimum rather than a stationary point or maximum. To preclude this, we must insist that the step be downhill, i.e. $\nabla j(u_h^i)s_i < 0$, which is the positive definiteness of Hessian matrix when the Newton step is chosen.

For the considered large scale problems the computation and storage of the Hessian matrix is often beyond reach. Furthermore the factorisation of the Hessian matrix, i.e. computation of

$$H(u_h^i) = B^{-1}(u_h^i)$$

might be very costly. The most simple way would be to estimate the inverse Hessian matrix $H(u_h^i)$ by the identity, resulting in search direction $s_i = -\nabla j(u_h^i)$ and the wellknown steepest descent method. A further improvement of inverse Hessian approximation is given by recursive update formulae like SR1, BFGS or DFP. These Quasi-Newton methods base on the update of the inverse Hessian approximation

$$H_{i+1} \approx H(u_h^{i+1})$$

in each iteration step of the optimisation algorithm using the former derived iterates $s_i = u_h^{i+1} - u_h^i$ and $y_i = \nabla j(u_h^{i+1}) - \nabla j(u_h^i)$ - see [41, 119] for a complete discussion. All update formulae avoid the possibly costly computation and factorisation of the Hessian matrix, but still need lots of storage - to overcome this problem *limited memory updates* are usually used.

We will concentrate on the usage of LBFGS method in combination with a line-search approach for globalisation of Quasi-Newton method. The main idea of limited memory BFGS is the replacement of full update formula for the inverse Hessian matrix[1]

$$H_{i+1} = \left(Id - \frac{s_i \otimes y_i}{y_i \cdot s_i}\right) H_i \left(Id - \frac{y_i \otimes s_i}{y_i \cdot s_i}\right) + \frac{s_i \otimes s_i}{y_i \cdot s_i}$$

by a version that only uses a specific number of vectors. Using say m vectors to store the history of s_i and y_i, the fundamental operation

$$s_i = -H_i \nabla j(u_h^i)$$

within a Quasi-Newton optimisation algorithm can be expressed by a two loop recursion 5 - refer to Chapter 7 in [119]. For the entire optimisation Algorithm 6 within the optimisation procedure

Algorithm 5 LBFGS two loop recursion to determine search direction s_i.

```
$q = \nabla j(u_h^i)$
for $j = i-1, \dots, i-m$ do
  $a_j = \frac{s_j \cdot q}{y_j \cdot s_j}$
  $q = q - a_j y_j$
end for
$r = H_i^0 q = \frac{s_{i-1} \cdot y_{i-1}}{y_{i-1} \cdot y_{i-1}} q$
for $j = i-m, \dots, i-1$ do
  $b = \frac{y_j \cdot r}{y_j \cdot s_j}$
  $r = r + s_j(a_j - b)$
end for
$s_i = -r = -H_i \nabla j(u_h^i)$
```

depicted in Figure 5.1 we have to add some globalisation strategy to the Algorithm 5 - here we use a simple line-search approach based on the Armijo update rule. It has to be mentioned that the 1D line-search step to determine the step length λ_i neccesitates the evaluation of cost functional $j(u)$. This step requires for a nonlinear cost functional and nonlinear state equations the solution of the full instationary partial differential equations such that the determination of step length might become the most costly part of Algorithm 6. To avoid the step length computation one might use trust-region methods instead of line-search methods - see [119] for details.

Algorithm 6 Line-search optimisation step with limited Quasi-Newton update - LBFGS method.

```
given control $u_h^i$ and gradient $\nabla j(u_h^i)$
compute $s_i = -H_i \nabla j(u_h^i)$ by Algorithm 5
update control $u_h^{i+1} = u_h^i + \lambda_i s_i$, step length $\lambda_i$ fulfilling sufficient decrease condition for cost functional $j(u_h)$
if $i > m$ then
  discard $[s_{i-m}, y_{i-m}]$ from storage
end if
store $s_i = u_h^{i+1} - u_h^i$ and $y_i = \nabla j(u_h^{i+1}) - \nabla j(u_h^i)$
```

[1] $a \otimes b = [a_i b_j]_{i,j}$ denotes the outer product of two vectors $a, b \in \mathbb{R}^n$

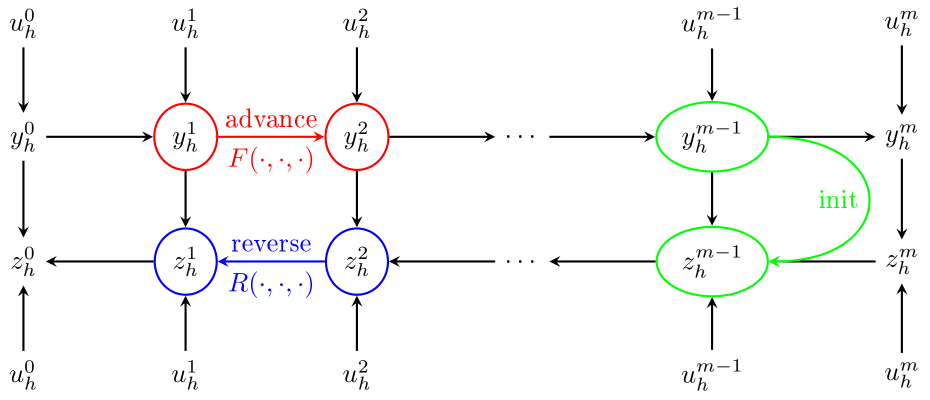

$$\begin{tikzcd}
u_h^0 \arrow[d] & u_h^1 \arrow[d] & u_h^2 \arrow[d] & & u_h^{m-1} \arrow[d] & u_h^m \arrow[d] \\
y_h^0 \arrow[r] \arrow[d] & y_h^1 \arrow[r, "\text{advance}", "F(\cdot,\cdot,\cdot)"'] \arrow[d] & y_h^2 \arrow[r] \arrow[d] & \cdots \arrow[r] & y_h^{m-1} \arrow[r] \arrow[d] \arrow[d, bend left=60, "\text{init}"] & y_h^m \arrow[d] \\
z_h^0 & z_h^1 \arrow[l] & z_h^2 \arrow[l, "\text{reverse}"', "R(\cdot,\cdot,\cdot)"] & \cdots \arrow[l] & z_h^{m-1} \arrow[l] & z_h^m \arrow[l] \\
u_h^0 \arrow[u] & u_h^1 \arrow[u] & u_h^2 \arrow[u] & & u_h^{m-1} \arrow[u] & u_h^m \arrow[u]
\end{tikzcd}$$

Figure 5.2: Typical forward-reverse scheme for optimisation of instationary problems.

5.2 Parallel I/O and Checkpointing Strategies

For the presentation of parallel hardware and software techniques used for PDE-constrained optimisation in combination with an adjoint based optimisation approach, we consider an instationary problem of the general form (cf. Chapter 2.3)

$$\min_{(y,u)\in Y\times U_{ad}} J(y,u) \tag{5.2}$$

subject to

$$\langle \partial_t y + A(y,u), \varphi\rangle_{Y^*,Y} = 0 \qquad \forall \varphi \in Y \tag{5.3}$$

where y (resp. u) describes the state (resp. control) variable. Further Y (resp. U_{ad}) describes the state (resp. control) space and we assume $J : Y \times U_{ad} \to \mathbb{R}$ for the objective functional and $A : Y \times U_{ad} \to Z = Y^*$ for the state equation to be nonlinear. As model problem the user might refer to the setting of fluid flow optimisation in Chapter 2.3 where the adjoint problem associated to (2.33) was found to be

$$\langle \frac{\partial J}{\partial y}(y,u), \varphi\rangle_{Y^*,Y} - \langle \partial_t z + \left[\frac{\partial A}{\partial y}(y,u)\right]^* z, \varphi\rangle_{Y^*,Y} = 0 \qquad \forall \varphi \in Y \tag{5.4}$$

and the optimality condition was given by

$$\langle \frac{\partial J}{\partial u}(y,u) + \left[\frac{\partial A}{\partial u}(y,u)\right]^* z, \varphi - u\rangle_{U^*,U} \geq 0 \qquad \forall\ u \in U_{ad}. \tag{5.5}$$

Considering finite element discretisation of the continuous problems (5.3) and (5.4) as depicted in Chapter 3.1, we derive the short forms

$$\begin{aligned} y_h^{i+1} &= F(y_h^{i+1}, y_h^i, u_h^{i+1}) \qquad && i = 0,1,\ldots,m-1 \\ z_h^i &= R(z_h^{i+1}, y_h^i, u_h^i) \qquad && i = m-1,\ldots,0 \end{aligned} \tag{5.6}$$

where we assume y_h^i and z_h^i to be the discrete solution of the primal resp. adjoint equation at timestep i. The discrete solution operators associated to the operators in (5.3) and (5.4) are denoted by $F(\cdot,\cdot,\cdot)$ and $R(\cdot,\cdot,\cdot)$ showing the forward and reverse character in time. Beside these operators often a combined advance and reverse step, called init-step, can be figured out that also comprises the initialisation of adjoint solution z_h^m. It is important to note that the adjoint problem is always linear, i.e. z_h^i only depends on z_h^{i+1}, but has to be solved *backward-in-time*. Furthermore assuming a nonlinear primal problem (or a nonlinear cost functional) each step of the adjoint solution is directly coupled to the corresponding step of the primal one (see Figure 5.2).

This means that the solution of the adjoint equation relies on the knowledge of the complete

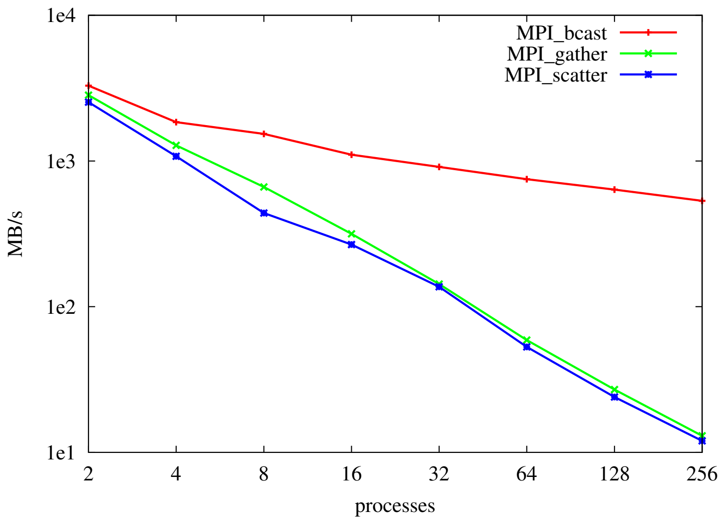


Figure 5.3: Benchmarking results of collective MPI-routines on HP XC3000 (Steinbuch Centre for Computing).

forward solution y_h^i, $i = 0, \dots, m-1$. In order to reduce the needed storage resources, different checkpointing strategies have been introduced:

- uniform checkpoint distribution (see [31] and references therein),
- binomial checkpoint distribution (see [160] and references therein),
- adaptive (online) checkpointing (see [82] and references therein).

Typically for complex instationary problems a full backup of states easily grows up to some TByte and cannot be stored directly in memory. Further on many large HPC platforms one has to face the problem that data I/O is being handled on a unique master node. The needed gather and scatter routines are very expensive on typical clusters due to the fact that performance of the network decreases dramatically the more processes are used in a collective operation.

Benchmarking for gather and scatter routines on the HP XC3000 cluster at Steinbuch Centre for Computing (Karlsruhe Institute of Technology) with an Infiniband 4X QDR interconnect showed that the related performance of such collective operations decrease dramatically for increasing the number of processes. While the standard point-to-point communication shows a performance of more than 3100 MB/s the bandwidth reduces by more than 90% for collective communication (see Figure 5.3).

Hence, one has to look for a solution that on the one hand reduces the amount of data to be stored, i.e. does not store complete backup, and on the other hand uses sophisticated HPC-techniques to enable scalable data I/O.

5.2.1 Solution Process on HPC-platform

For practical problems under consideration we mainly encounter large scale discretisation and it turns out that the time needed to store one forward solution step may exceed the computational time needed to obtain this step. This is especially the case when explicit timestepping schemes are used. We will show this effect by an academical example of adjoint based optimisation, namely: given an instationary partial differential equation for the state-solution y and a stationary/desired solution $y^* = y^*(\mathbf{x})$. Formulate a tracking type cost functional to control the PDE (here by means of boundary control) such that the stationary solution is achieved within a fixed time interval $(0, T)$. This principle can be found in the following parabolic example of boundary control but also in the examples of Chapter 2.3.

The entire problem reads

$$\min_{(y,u)\in H^1(\Omega)\times L^2(\Gamma_c)} J(y,u) = \frac{1}{2}\int_0^T \|y-y^*\|^2_{L^2(\Omega)}\,dt + \frac{\lambda}{2}\int_0^T \|u\|^2_{L^2(\Gamma_c)}\,dt \tag{5.7}$$

under state equation: $y(\cdot,0) = y_0 \neq 0$ and for $t \in (0,T)$ it holds $y(\cdot,t) \in \{v \in H^1(\Omega) : \ v = u \text{ on } \Gamma_c\}$ such that

$$\langle \partial_t y, \varphi\rangle + (\nabla y, \nabla\varphi)_0 + \alpha(y,\varphi)_0 = 0 \qquad \forall\varphi \in H^1_0(\Omega). \tag{5.8}$$

We use the parameter α to modify the behaviour of the system. Choosing $\alpha < -\mu_{min}$, where μ_{min} is the smallest eigenvalue of the Laplace operator (e.g. $\mu_{min} = 2\pi^2$ in two dimensions), the system becomes unstable and the solution blows up, i.e. the norm $\|y_h\|$ tends to infinity. Hence, the boundary control u on Γ_c can be viewed as a stabilisation of the system, see [19].

The adjoint equation to (5.8) can directly be stated as: $z(\cdot,T) = 0$ and for $t \in (0,T)$ it holds $z(\cdot,t) \in H^1_0(\Omega)$ such that

$$-\langle \partial_t z, \varphi\rangle + (\nabla z, \nabla\varphi)_0 + \alpha(z,\varphi)_0 = (y^*-y,\varphi)_0 \qquad \forall\varphi \in H^1_0(\Omega) \tag{5.9}$$

whereas the optimality condition (having no additional bounds on control u) is given by: for $t \in (0,T)$ it holds

$$(\lambda u + \partial_n z, \chi)_{L^2(\Gamma_c)} = 0 \qquad \forall\chi \in L^2(\Gamma_c). \tag{5.10}$$

We refer to [157] for a derivation and detailed analysis of such parabolic problems. Next the discretisation of state and adjoint equation is done by conform spatial finite element method (using finite element space $V_h \subset H^1_0(\Omega)$) in combination with explicit timestepping (cf. Chapter 3.1) and one gets the discrete system for (5.8)

$$(y_h^k,\varphi_h)_0 + \Delta t\left[(\nabla y_h^{k-1},\nabla\varphi_h)_0 + \alpha(y_h^{k-1},\varphi)_0\right] = (y_h^{k-1},\varphi_h)_0 \qquad \forall\varphi_h \in V_h,\ k=1,\dots,m$$

With standard notation of stiffness- and mass-matrix

$$\begin{aligned} A = [a_{ij}] &= \Delta t\left[(\nabla\varphi_{h,j},\nabla\varphi_{h,i})_0 + \alpha(\varphi_{h,j},\varphi_{h,i})_0\right] \qquad & i,j=1,\dots,n \\ M = [m_{ij}] &= (\varphi_{h,j},\varphi_{h,i})_0 & i,j=1,\dots,n \end{aligned}$$

this can be written as the algebraic system, where we use $y_h^k = \sum_{i=1}^n Y_i^k\varphi_{h,i}$ and $Y^k = [Y_1^k,\dots,Y_n^k]^T$,

$$MY^k + AY^{k-1} = MY^{k-1} \qquad k=1,\dots,m$$

or equivalently

$$Y^k = Y^{k-1} - M^{-1}AY^{k-1} \qquad k=1,\dots,m. \tag{5.11}$$

If finally the mass-matrix $M = m_{ij} = (\varphi_{h,j},\varphi_{h,i})_0$ is replaced by the lumped diagonal matrix $\overline{M} = \overline{m}_{ii} = \sum_{j=1}^n m_{ij}$, equation (5.11) can be used as explicit update scheme for the discrete solution y_h^k. Since this scheme mainly uses matrix-vector multiplication (plus additional componentwise vector operations), it is a prominent candidate for parallelisation as we will shown later.

With respect to the previously described problems of storage space/time and in order to solve the general adjoint problem (2.34) resp. the concrete problem (5.9), we consider an hybrid approach combining two steps:

- checkpointing-strategy: store few dedicated timesteps instead of a complete state variable backup and recompute missing states for the backward problem if needed,
- parallel I/O: allows every node in a cluster to write/read data simultaneously.

These two approaches result in different benefits. In our framework checkpointing strategies rely on a reduced storage by means of a dedicated choice of the state solution forward-in-time. Each of the stored timesteps correspond to a checkpoint. This kind of technique aims on the one hand

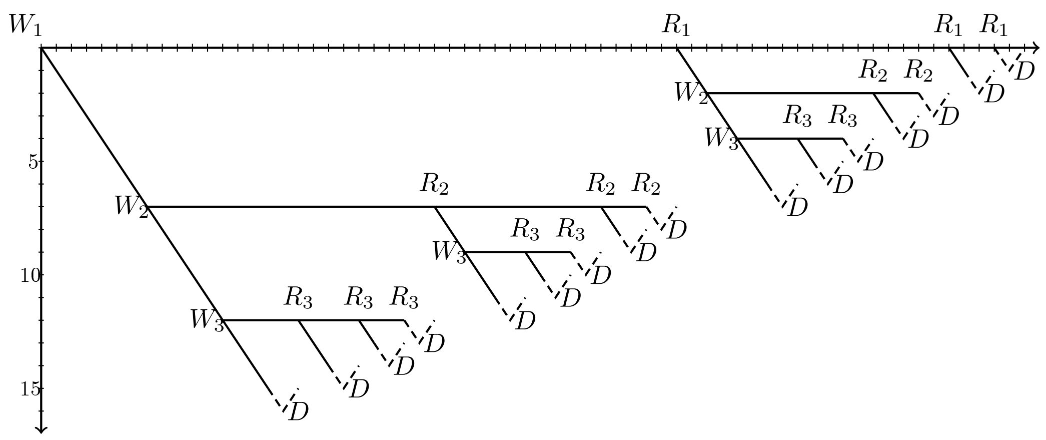


Figure 5.4: Binomial checkpoint distribution in case of 16 timesteps and 3 checkpoints. R_i means *read i-th checkpoint*, W_i means *write i-th checkpoint* and D stands for *derive adjoint state.*

at strongly reducing the needed storage resources. On the other hand the checkpoints are chosen such that the needed recomputing efforts for the missing timesteps are kept minimal. A typical situation is depicted in Figure 5.4 assuming binomial checkpoint distribution (cf. [69, 160]).

Parallel I/O techniques lead to additional benefits. These technologies allow data to be transmitted in a fully parallel way between clients, on which they are produced and used, and central dedicated storage server, on which they should be stored. In practice a significant speedup for data-access can be gained as compared to typical cluster with serial file systems.

Hardware Configuration

Development of supercomputers nowadays more and more tends to systems which are well-resourced with processors but with less main memory per core. In the November 2009 list of top500 supercomputers no system in top 30 is equipped with less than 10000 cores and the ratio of main memory to total number of cores shows a clear tendency, e.g.

- top 1: Jaguar (Cray XT5-HE Opteron Six Core) at Oak Ridge National Laboratory (United States) with 224162 cores. Each of compute node contains two hex-core AMD Opteron processors and 16 GB main memory, resulting in 1.33 GB/core,
- top 4: JUGENE (Blue Gene/P Solution) at Forschungszentrum Jülich (Germany) with 294912 cores. The 72 racks with 1024 compute nodes host 4-way SMP processor and 2 GB main memory per compute node, resulting in 0.5 GB/core,
- top 13: JUROPA (Sun Constellation, NovaScale R422-E2, Intel Xeon X5570) at Forschungszentrum Jülich (Germany) with 26304 cores. 2208 compute nodes with 2 Intel Xeon X5570 quad-core processors and 24 GB main memory, resulting in 3 GB/core.

Furthermore in the last years the memory and interconnect bandwidth did not increase that much compared to the computational power of modern CPUs. These *hardware facts* underline the necessity to develop solutions that rely on less memory access but more computations to be done with the data. Explicit timestepping schemes might therefore become more and more popular especially in combination with parallel data I/O.

For the investigation presented subsequently we have considered the HPC platform HP XC3000 cluster at Steinbuch Centre for Computing (the system layout is similar to the JUROPA, see Remark 4.2.2) for which the parallel file system bases on the Lustre file system developed by Sun Microsystems [150]. A main concept underlying this system is the separation between metadata (directories or file attributes like name and user rights) and real data (the information/content of a file). Another important point is a sophisticated system for management of locks that prevents simultaneous write from different clients on the same dataset by which consistence of the file system is guaranteed. Figure 5.5 shows the main components and protocols in a Lustre system.

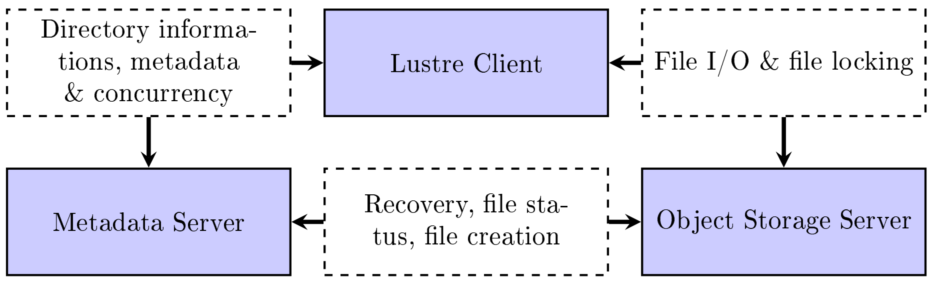
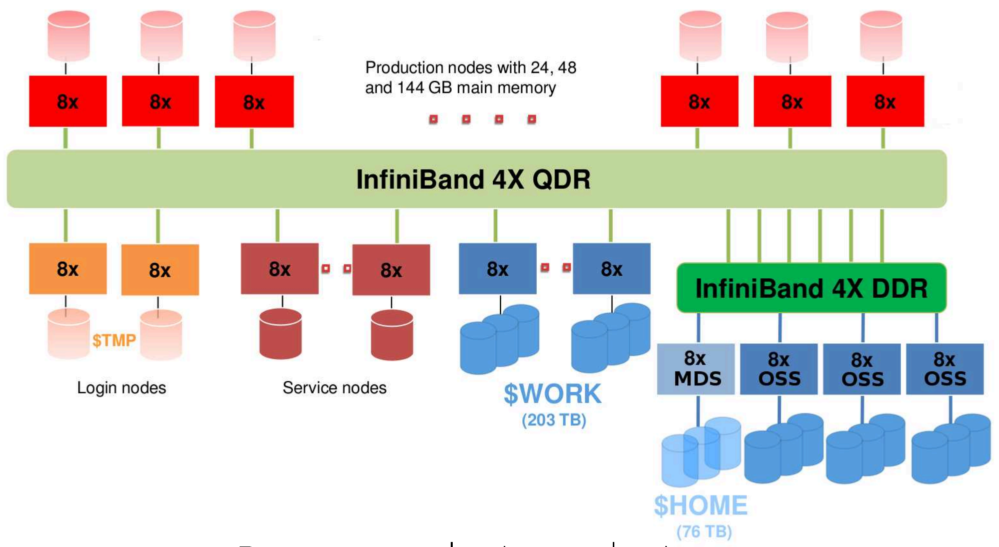

Directory informations, metadata & concurrency
Lustre Client
File I/O & file locking
Metadata Server
Recovery, file status, file creation
Object Storage Server

Figure 5.5: Components and protocols in a Lustre system.



Property	$HOME	$WORK
Disk space	76 TByte	203 TByte
Read perf./node	600 MB/s	1800 MB/s
Write perf./node	700 MB/s	1800 MB/s
Total read perf.	1700 MB/s	4800 MB/s
Total write perf.	1500 MB/s	4800 MB/s

Figure 5.6: System architecture of parallel file system Lustre on HP XC3000 (Steinbuch Centre for Computing).

The overall architecture and performance of the Lustre system on the HP XC3000 platform differs for the directories `$HOME` and `$WORK` due to a different number of servers and different interconnect. All servers are configured as failover pairs, i.e. if the metadata server (MDS) fails the admin server adopts his part, while the object storage server (OSS) actually hold the data that are distributed by striping (see Figure 5.6). In order to store a distributed vector (for example the i-th timestep of forward solution) these parallel I/O techniques does not rely any more a gather instruction on a master-processor.

Usage of Checkpointing Schemes

The topic of checkpointing in the context of PDE constrained optimisation has been the object of intensive research efforts (see e.g. [89] and references therein). In our context we consider the approach based on the algorithm REVOLVE described in [69]. This procedure aims at minimising the number of needed additional forward-step computations assuming restriction associated to the limited storage capacity.

It is important to note that such checkpointing strategies assume the number of available checkpoints to be given and do not prescribe this parameter on their own. In order to define the optimal number of checkpoints for a given problem a trade-off taking into account the I/O

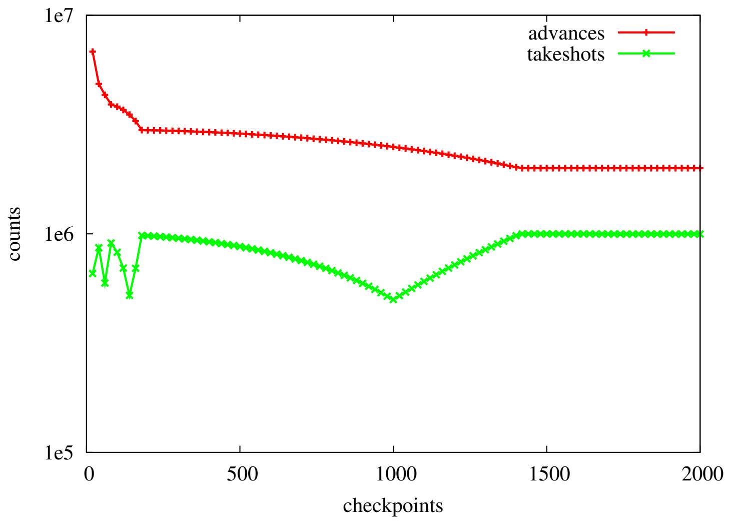


Figure 5.7: Lower bound for number of checkpoints by desired costs for optimisation algorithm and local minimum of number of storage accesses (takeshots) – assuming 10^6 timesteps to be reversed.

bandwidth, latency as well as the CPU-costs needed for the solution of a forward step has to be considered (see e.g. [149]). A standard approach usually bases on minimising the number of storage access, while at the same time the additional forward-step computations should be bounded. It consists of three steps:

1. **Determine the upper bound** $n_{ch,max}$: the limited storage capacity leads to a maximal number of checkpoints for a given problem size.

2. **Determine the lower bound** $n_{ch,min}$: a lower bound is obtained from the condition that the ratio between the computational effort for the optimisation and the CPU costs associated to the forward simulation should be bounded and independent of the discretisation level of the considered PDEs.

3. **Minimise number of storage access in** $[n_{ch,min}, n_{ch,max}]$: in the previous determined interval one determines the number of checkpoints leading to a minimal number of storage access (takeshots). This minimum can easily be determined on the basis of the function describing the dependency between the number of checkpoints and takeshots which is known a priori (see Figure 5.7).

This approach bases on the observation that forward-step computation is faster than storage access. In the context of parallel solvers combined with parallel data I/O this kind of argumentation may be wrong from the sense that the costs for I/O are not a constant any more but show similar scalability properties as compared to the costs related to the forward simulation. Our observation on the considered HPC platform is that minimising the number of takeshots may be contraproductive. In contrast one should consider the maximal number of possible checkpoints leading to an optimal overall computational time as depicted in the following numerical results.

For a typical example of instationary problem optimisation (10^6 timesteps, 10^6 degrees of freedom (dof) in each timestep assuming double precision, resulting in ≈ 7.3 TByte for complete backup) the proposed standard strategy results in – see Figure 5.7:

- upper bound $n_{ch,max} = 13744$ due to prescribed capacity of ~ 0.1 TByte,

- lower bound $n_{ch,min} = 180$ due to maximal extra costs of factor 3, i.e. maximum $3 \cdot 10^6$ forward-steps (advances),

- optimal $n_{ch,opt} = 1000$ (local minimum of takeshots) resulting in 500500 takeshots and 2497497 forward-steps (overhead factor of ~ 2.5).

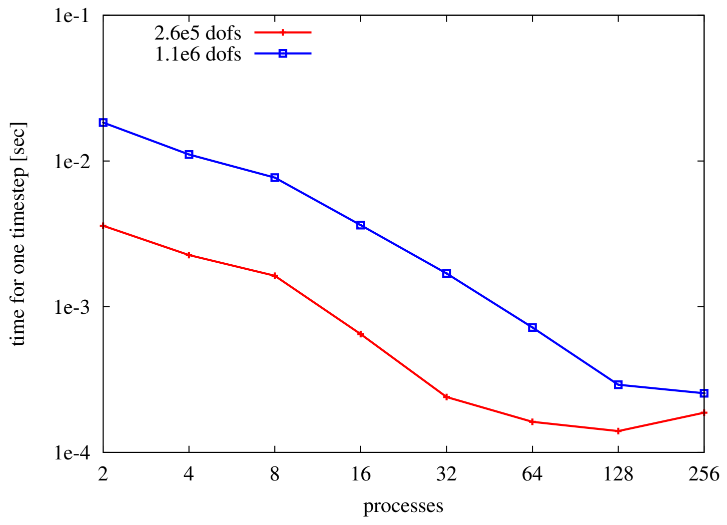


Figure 5.8: Absolute times for update scheme (5.11) operation $Y^{k-1} \to Y^k$ on different discretisation levels.

Using parallel I/O techniques enables to push the number of used checkpoints to upper bound $n_{ch,max}$ resulting in 986254 takeshots and 1986253 forward-steps. The overhead factor reduces to ~ 1.98 while the number of takeshots is nearly doubled which is completely buffered by the parallel I/O improvement of factor 10 compared to standard storage approach as shown in the sequel.

5.2.2 Numerical Results

To show the capabilities of parallel hardware, we again consider the academical optimisation example (5.7) under constraint (5.8). The explicit update scheme (5.11) is first being tested for scalability properties, while afterwards the parallel I/O performance is presented. Conclusively the proposed approach of combining checkpointing schemes with parallel I/O techniques is evaluated.

Scalability of Explicit Timestepping

The update scheme (5.11) was implemented by means of the parallel row block data structure that provided a very good scalability of matrix-vector product (see Chapter 4.2). We tested the pure forward simulation of state equation (5.8) for different number of degrees of freedom and for the fixed time interval $(0, 1)$. The number of timesteps m, i.e. the temporal discretisation Δt, was chosen to a fixed value such that the solution at $t = 1$ remains suitable with respect to the CFL-condition – a comparison to solution gained by implicit timestepping has been passed.

Figure 5.8 shows that the update scheme (5.11) possesses a good weak scalability, i.e. the bigger the problem is (finer discretisation) the more processes can efficiently be used. But in contrast to the optimal scalability of pure matrix-vector multiplication (see Figure 4.4 and 4.5), we encounter for one timestep $Y^{k-1} \to Y^k$ additional parallel vector operation (including global reduction operations to compute norms) such that the scalability is slightly worse.

Parallel I/O Performance

We now consider the performance of parallel I/O for the `$WORK` partition of HP XC3000 cluster assuming the configuration described before. The underlying optimisation problem is at this stage of minor concern, since the results hold true for general data storage. For the implementation we consider the following three different setups:

1. PETSc-binary and -ASCII format using functions `PetscViewerBinaryOpen` and `VecView` resp. `PetscViewerASCIIOpen` and `VecView` (see [7]),

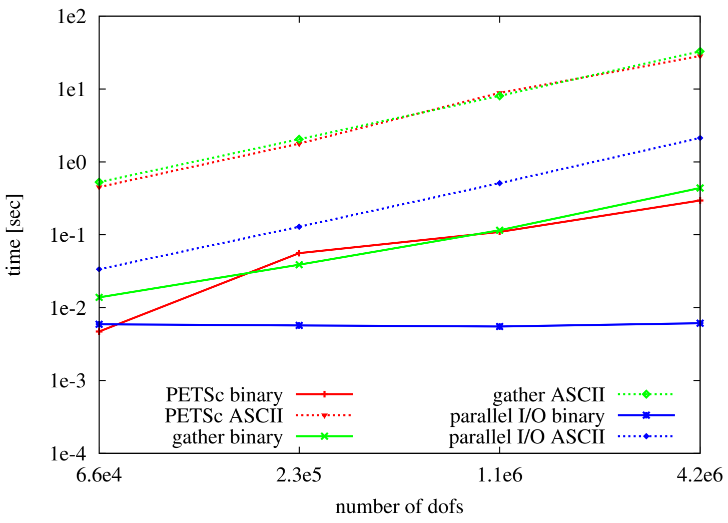


Figure 5.9: Time needed to store distributed vector Y^k of scheme (5.11) distributed over 32 processes.

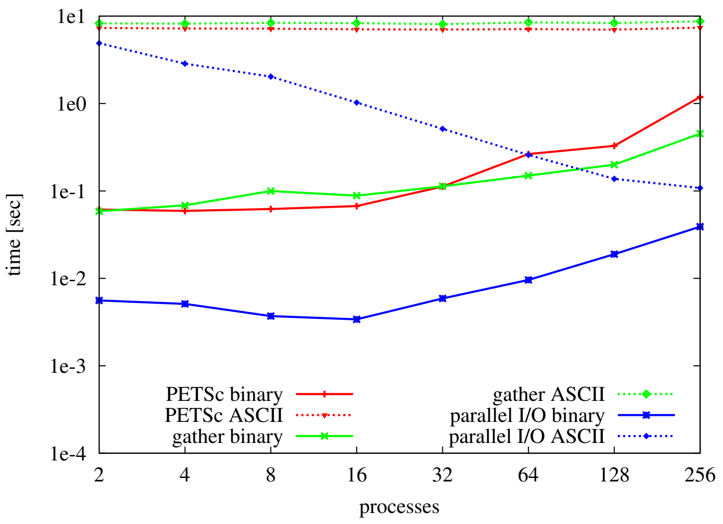


Figure 5.10: Time needed to store distributed vector Y^k of scheme (5.11) with 1050625 degrees of freedom distributed over different number of processes.

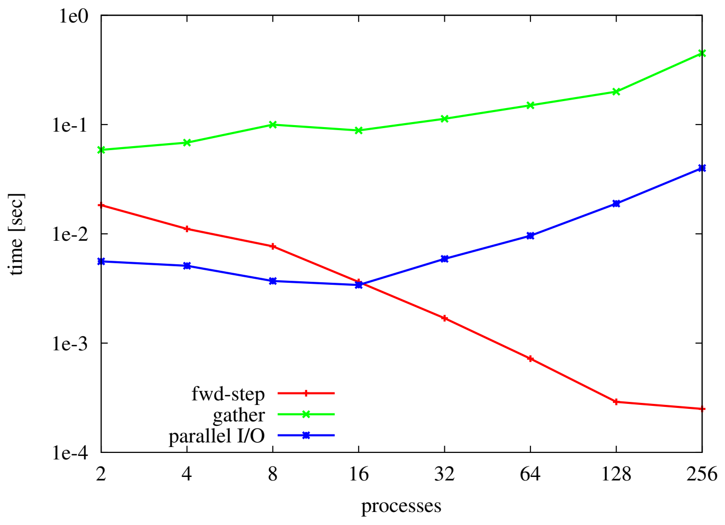


Figure 5.11: Time needed for fwd-step and storage operations using 1050625 degrees of freedom distributed over different number of processes.

2. gather parallel row block vector into sequential vector and save in binary resp. ASCII format using STL data structures and collective MPI-routines,

3. parallel I/O saving binary resp. ASCII format using STL data structures and parallel file system based on Lustre.

The first two approaches represent the gather/scatter approach. Furthermore we used both ASCII and binary format to get more illustrative results and to be able to check files easier – for the actual production modus obviously only binary format should be used. Time for saving the vector grows proportionally to the number of components the vector has using both standard gather/scatter technique and parallel I/O. Quantitatively however the results obtained by parallel I/O give an improvement by ca. 90% (see Figure 5.9 for the case of 32 processes).

As depicted in Figure 5.10 the performance for gather/scatter storage operations is almost independent of the number of processes. Only if more than 32 processes are used there is a noticeable increase in storage time – this is due to the fast decaying MPI-performance for collective routines (see Figure 5.3). The gained efficiency and scalability for parallel I/O depend obviously directly on the number of available OSS (see Figure 5.6 and [102] for more details). For ASCII file storage one encounters almost a speedup in storage access that can be explained by the bigger amount of data to be transmitted at once. For much smaller binary files instead the latency of network becomes recognisable using more than 16 processes.

Combining Parallel I/O and Checkpointing Strategy

The efficiency of combination of parallel I/O and checkpointing schemes as described before is evaluated in the sequel using the academical example above. We considered computation of one gradient (5.10) by means of adjoint approach using Algorithm 6 and the adjoint equation (5.9). That is, we aim at one complete reversal as shown in Figure 5.2. For the spatial discretisation with $1.1e6$ degrees of freedom and $\Delta t = 1.0e-7$, $T = 0.1$, a full backup of state solution gained by (5.8) results in 7.64 TByte to be stored, which is obviously not to be handled.

In order to find the optimal number of checkpoints $n_{ch,opt}$ used for this problem, we compared the standard approach by minimising the number of takeshots (as described before) to the possible storage of $n_{ch,opt} = n_{ch,max}$ checkpoints. For the reversal scheme 5.2 in combination with REVOLVE algorithm [69] one encounters the general observations (see Figure 5.4 for example) which are independent of the number of checkpoints used:

- number of reverse timesteps (i.e. adjoint solution solver) is $m-1$,

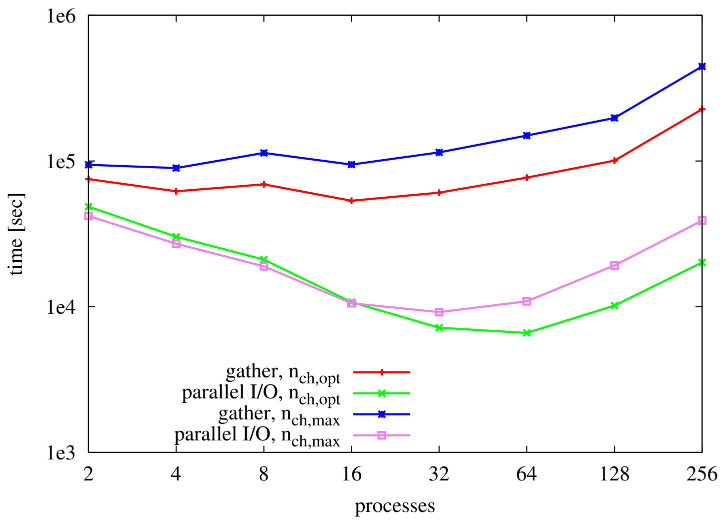


Figure 5.12: Time needed for compute gradient of model problem using different storage approaches.

- number of restores of checkpoints is $m - 1$.

Hence, these two steps can be neglected within the comparison. The difference in runtime to compute the gradient (5.10) will only be given by number of storage access (takeshots) and number of forward-steps (advance). Benchmarking of these basic operations showed the scalability of forward-steps and also the lack of scalability for storage operations (see Figure 5.11). As already mentioned the storage time by parallel I/O would also scale or at least stay constant if more OSS are used, whereas the gather-operation per se cannot scale.

The comparison of used storage approach and number of used checkpoints in Figure 5.12 clearly indicates two points. First, the usage of gather/scatter operations to store a checkpoint prevents any scalability of the entire computation. Second, optimal number of checkpoints in the sense of minimal time to compute gradient by reversal scheme 5.2 depends on the ratio of time for one forward-step to one storage operation. If storing of a checkpoint can be performed faster than one forward-step, one should use maximal number of checkpoints available by memory limitation. If on the other side a forward-step is computed faster, choice of $n_{ch,opt}$ checkpoints as described before results in optimal time – this turning point can clearly be seen in Figure 5.12. One also notices that once the storage operation is the to costly compared to one forward-step (in this example at 128 processes with a ratio of ~ 65) no benefit of additional processes is given.

Conclusively we showed that parallel I/O enables a much faster gradient computation as standard gather/scatter operations and that the often used checkpointing schemes need to be reconsidered. For future work one also have to take into account a possible distinction of different memory stages, i.e.

- main memory for each core,
- local disk space on each node accessible for some cores,
- global disk space on cluster accessible for all cores,

to get scalability even for storage operation. These stages give additional opportunities for memory access and layout of checkpoints – e.g. the most often restored/used checkpoints should be stored within the fastest accessible memory. We refer to [149] for investigations on this active field of research.

Chapter 6

Numerical Results

Within this chapter, we investigate the capabilities of the adjoint- and sensitivity-based control methods developed in Chapter 2 in combination with the high performance computing techniques described in Chapter 4 and 5. We apply the proposed methods to the Navier-Stokes equations with control of the velocity boundary values as well as to the electroosmotic micromixer with control of the applied potential. In the first section, a comparison of the adjoint- and sensitivity-based control method is adapted to the Navier-Stokes equations using the well-known optimisation problem of vortex reduction behind a backward facing step. This model problem serves as feasibility study for the presented techniques and provides the way to more physical related problems. Section 6.2 contains an extension to the simulation of the basis electroosmotic micromixer as presented in [10, 11], while at last in Section 6.3 this setting is additionally aimed to be optimised.

6.1 Vortex Reduction Behind a Backward Facing Step

We consider the control of a backward facing step flow as model problem for the adjoint- and sensitivity-based approach in optimisation – the used computational domain Ω is presented in the Appendix, for results of the two-dimensional case see [87]. The flow field within Ω is described by the instationary three-dimensional Navier-Stokes equations with suitable boundary conditions,

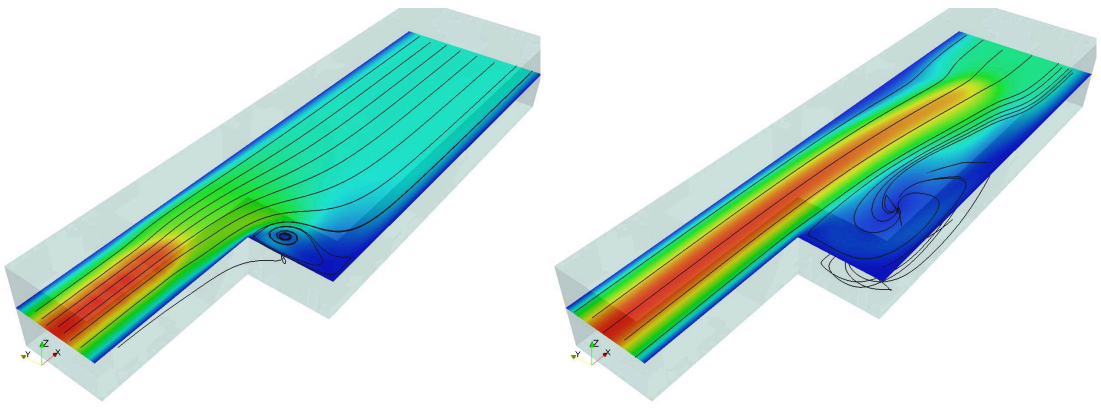


Figure 6.1: Startsolution $\mathbf{v}_0$ at $t = 1$ (left) and uncontrolled solution at $t = 4.5$ (right) for optimisation of backward facing step example showing the absolute velocity and streamlines within cutplane $z = 0.5$.

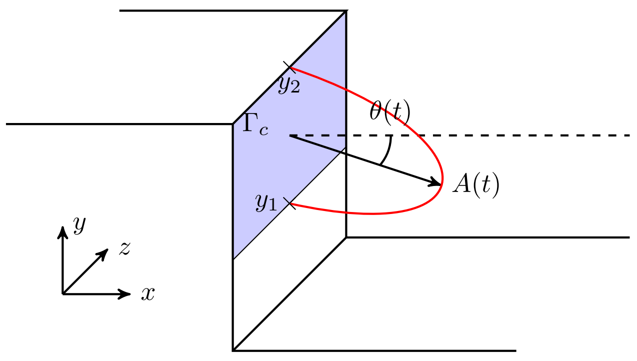


Figure 6.2: Schematic illustration of boundary control c on upper part of the back wall – allowed variation of angle $\theta(t)$ and amplitude $A(t)$ for parabolic inflow profile are shown.

namely

$$
\begin{aligned}
Re\,[\partial_t \mathbf{v} + (\mathbf{v}\cdot\nabla)\mathbf{v}] - \Delta\mathbf{v} + \nabla p &= 0 && \text{in } \Omega\times(0,T),\\
\nabla\cdot\mathbf{v} &= 0 && \text{in } \Omega\times(0,T),\\
\mathbf{v} &= 0 && \text{on } \Gamma_0\times(0,T),\\
\mathbf{v} &= \mathbf{c} && \text{on } \Gamma_c\times(0,T),\\
\mathbf{v} &= \mathbf{v}_{in} && \text{on } \Gamma_{in}\times(0,T),\\
\partial_n\mathbf{v} - p\mathbf{n} &= 0 && \text{on } \Gamma_{out}\times(0,T),\\
\mathbf{v}(\cdot,0) &= \mathbf{v}_0 && \text{in } \Omega.
\end{aligned}
$$

The boundary $\partial\Omega$ is separated into disjoint parts: the channel walls Γ_0 where a no slip condition is assumed, the inflow and outflow boundary Γ_{in}, Γ_{out} and the boundary where a control is possible. Inflow and outflow boundary conditions are discussed in Section 2.2.3, here we use a steady parabolic inflow profile $\mathbf{v}_{in}$ in combination with the do-nothing outflow condition. As control boundary, the upper part of the back wall $\Gamma_c = [2,2]\times[\frac{2}{3},1]\times[0,1]$ is chosen. Furthermore, the startsolution $\mathbf{v}_0$ of the velocity field is determined by simulation of the system without control for $t\in[0,1]$ – see left plot in Figure 6.1. Hence, the control appears delayed but affects the flow field right before the vortex behind the step develops, i.e. we investigate the optimisation within the time interval $t\in[1,5]$ which, for the sake of simplicity, is shifted and denoted by $[0,T]$.

Specific kind of control depends on the optimisation approach. For the adjoint-based optimisation we assume that $\mathbf{c}\in L^2(0,T;\mathbf{L}^2(\Gamma_c))$ and the first-order optimality conditions based on the adjoint equations are already stated in Chapter 2.3.2. We only recall the cost functional used in the sequel as

$$
\min_{\mathbf{v},\mathbf{c}} J_a(\mathbf{v},\mathbf{c}) = \frac{1}{2}\int_0^T \left[\int_{\Omega_s} |\mathbf{v}-\mathbf{v}_d|^2\,d\mathbf{x} + \lambda\int_{\Gamma_c} \mathbf{c}^2\,ds\right]\,dt, \tag{6.1}
$$

i.e. we aim at a reduction of the recirculation behind the step by minimising the L^2-difference of velocity field to a given $\mathbf{v}_d\in\mathbf{L}^2(\Omega)$. Here $\Omega_s\subset\Omega$ denotes the observation area which is located to measure the recirculation behind step, in particular $\Omega_s = [2,5]\times[0,1]\times[0,1]$. The desired flow field $\mathbf{v}_d$ is chosen to be the solution of the stationary Stokes equations at Reynolds number $Re = 200$.

We compare the adjoint-based approach as given by the system (2.40), (2.41) and (2.42) to the sensitivity-based optimisation. To this, we define the control acting on Γ_c by an inflow stream modelled via a parabolic profile to imitate a flow similar to the Poiseuille flow. Using the spatial constant function ($y_1 = \frac{2}{3},\ y_2 = 1,\ z_1 = 0,\ z_2 = 1$) defined on Γ_c

$$
f(y,z) = (y_2-y)(y-y_1)\left(\frac{y_2-y_1}{2}\right)^{-2}(z_2-z)(z-z_1)\left(\frac{z_2-z_1}{2}\right)^{-2},
$$

we set the orientation $\theta(t)$ and amplitude $A(t)$ of the stream by (see Figure 6.2 for a schematic illustration)

$$\begin{aligned} v_0|_{\Gamma_c} &= A(t)\cos(\theta(t))f(y,z),\\ v_1|_{\Gamma_c} &= -A(t)\sin(\theta(t))f(y,z),\\ v_2|_{\Gamma_c} &= 0. \end{aligned} \tag{6.2}$$

The time dependent angle $\theta(t)$ and amplitude $A(t)$ are described by a polynomial ansatz

$$A(t) = \sum_{k=0}^{n} \alpha_k t^k, \quad \theta(t) = \sum_{k=0}^{n} \beta_k t^k,$$

with design parameters α_k and β_k such that the sensitivity-based optimisation relies on $2(n+1)$ variables. This choice is motivated by the results of the two-dimensional problem presented in [87] and the fact that at least a constant co-inflow should reduce the recirculation area. The cost functional in this case is given as before but with a modified part for the control

$$\min_{\mathbf{v},\alpha_k,\beta_k} J_s(\mathbf{v},\alpha_k,\beta_k) = \frac{1}{2}\int_0^T\int_{\Omega_s} |\mathbf{v}-\mathbf{v}_d|^2 \, d\mathbf{x}\, dt + \frac{\lambda}{2}\sum_{k=0}^{n}\left(|\alpha_k|^2+|\beta_k|^2\right). \tag{6.3}$$

Computation of the gradient and sensitivity equations are in general presented in (2.47), (2.48) and read in the concrete case for $\alpha_k,\ \beta_k,\ k=0,\dots,n$

$$\begin{aligned} \frac{DJ_s}{D\alpha_k} &= \lambda\alpha_k + \int_0^T\int_{\Omega_s} (\mathbf{v}-\mathbf{v}_d)\mathbf{v}_{\alpha_k}\, d\mathbf{x}\, dt,\\ \frac{DJ_s}{D\beta_k} &= \lambda\beta_k + \int_0^T\int_{\Omega_s} (\mathbf{v}-\mathbf{v}_d)\mathbf{v}_{\beta_k}\, d\mathbf{x}\, dt, \end{aligned}$$

with the sensitivities $\mathbf{v}_{\alpha_k} = \frac{D\mathbf{v}}{D\alpha_k}$ and $\mathbf{v}_{\beta_k} = \frac{D\mathbf{v}}{D\beta_k}$ given by (2.48). Only the boundary conditions for $\mathbf{v}_{\alpha_k}$ and $\mathbf{v}_{\beta_k}$ on Γ_c need to be adapted to the partial derivatives of (6.2), i.e.

$$\begin{aligned} v_{\alpha_k,0}|_{\Gamma_c} &= t^k\cos(\theta(t))f(y,z)\\ v_{\alpha_k,1}|_{\Gamma_c} &= -t^k\sin(\theta(t))f(y,z)\\ v_{\alpha_k,2}|_{\Gamma_c} &= 0\\ v_{\beta_k,0}|_{\Gamma_c} &= -t^kA(t)\sin(\theta(t))f(y,z)\\ v_{\beta_k,1}|_{\Gamma_c} &= -t^kA(t)\cos(\theta(t))f(y,z)\\ v_{\beta_k,2}|_{\Gamma_c} &= 0. \end{aligned}$$

Figure 6.3 shows the results of optimisation where we denote by *costs* the evolution of cost functional $J_a(\mathbf{v},\mathbf{c})$ resp. $J_s(\mathbf{v},\alpha_k,\beta_k)$. Since the cost functional (6.3) does not contain an explicit time dependent part for the control, we uniformly distribute the costs related to the control by

$$J_s(\mathbf{v},\alpha_k,\beta_k) = \frac{1}{2}\int_0^T\left[\int_{\Omega_s}|\mathbf{v}-\mathbf{v}_d|^2\, d\mathbf{x} + \frac{\lambda}{T}\sum_{k=0}^{n}\left(|\alpha_k|^2+|\beta_k|^2\right)\right] dt \tag{6.4}$$

to get an evolution of the costfunctional that can directly be compared to (6.1). For the adjoint-based approach we use the regularisation $\lambda = 10^{-1}$ while for the sensitivity-based approach $\lambda = 10^{-2}$ is sufficient. The Reynolds number is in all cases $Re = 200$. The iteration of optimisation Algorithm 6 in combination with the Scheme 5.1 is stopped if the ℓ^2 difference of two successive iterates is less than 10^{-3} or if the step length within line-search step is less than 10^{-6}. In comparison to the uncontrolled case, we see that both optimisation approaches result in nearly constant values for the cost functional indicating that the development of the vortex behind the step is reduced.

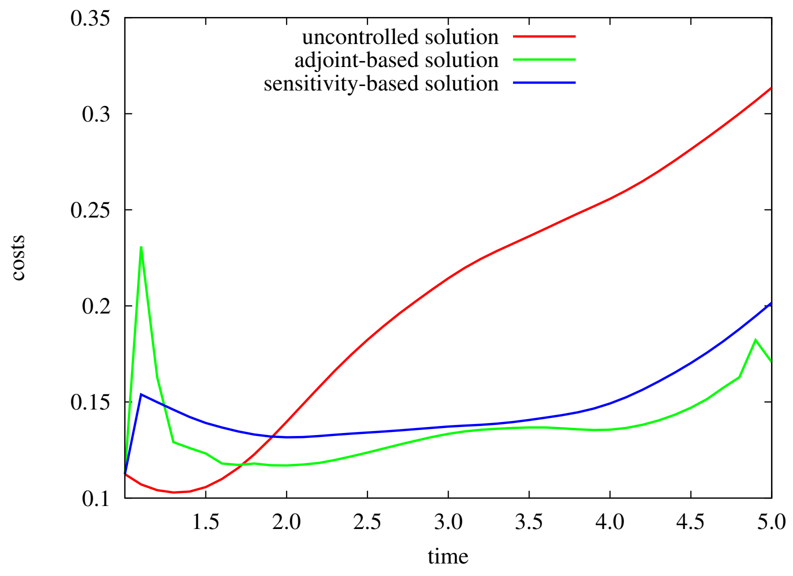


Figure 6.3: Evolution of cost functional $J_a(\mathbf{v}, \mathbf{c})$ resp. $J_s(\mathbf{v}, \alpha_k, \beta_k)$ for the three-dimensional backward facing step problem compared to uncontrolled case.

Only for $t > 4.0$ we have an increasing cost functional, which especially for the sensitivity-based approach can be explained by the fact that no special term at $t = T$ has been used within the cost functionals (6.1) and (6.4). For both controlled solutions, one furthermore observes a straight jump of costs in the beginning which is not present for the uncontrolled solution. This jump is related to the fact that the initial flow field $\mathbf{v}_0$ is not controlled and therefore a discontinuous boundary control appears within the first timestep yielding a noticeable change of the flow field. At a first glance, Figure 6.3 suggests that the gain which can be achieved by the adjoint-based approach compared to the sensitivity-based approach seems to be of minor importance since both result in a comparable evolution of the cost functional. Additionally, it remains to interpret the resulting boundary control by the adjoint-based method in terms of a physical feasibility.

To obtain an insight in the physical relevance of the adjoint-based solution, we plot in Figure 6.4 the flow field of the controlled, the uncontrolled and the desired case within the channel mid-height $z = 0.5$ at $t = 4.5$. For the uncontrolled Navier-Stokes flow, a re-attachment point of $x \approx 5$ can be found while this point is shifted to $x \approx 2.8$ for the adjoint-based optimisation. Also, the sensitivity-based solution shows a smaller recirculation area behind the step and the desired even flow field appears from $x \approx 3.5$ on. Although the adjoint-based optimisation approach allows much more degrees of freedom to control the flow field, the resulting solution is not significantly improved compared to the more physically motivated sensitivity-based approach. The vortex behind the step induced by the main flow stream is in both cases minimised by an additional inflow that is directed in negative y-direction. Figure 6.5 shows the boundary control at $t = 4.5$ for both optimisation approaches. On the one hand, one can clearly see the parabolic profile of x- and y-velocity components in the sensitivity-based case. The optimal design parameter α_k, β_k are given as $\alpha_0 = 1.58$, $\alpha_1 = -0.16$, $\beta_0 = 0.29$, $\beta_1 = 0.13$, i.e. the amplitude $A(t)$ slightly decreases from 1.58 in the beginning to 0.94 at $t = 5$ while the angle $\theta(t)$ increases from $17°$ to $47°$. On the other hand, the boundary control given by the adjoint-based optimisation shows a much more complex appearance. First the control of z-velocity component v_2 on Γ_c turns out to be of minor importance. The main characteristics of optimal control are given by a sharp inflow right below the step that reduces with decreasing y-direction – a profile that seems questionable to be realised within a concrete physical setting. At least the tendency of positive x-velocity indicating an injection and negative y-velocity to suppress the development of the recirculation area can be found in accordance to the sensitivity-based solution.

To summarise, it appears that both optimisation approaches result in a reduction of recirculation area behind the step. Although the adjoint-based solution performs slightly better in terms

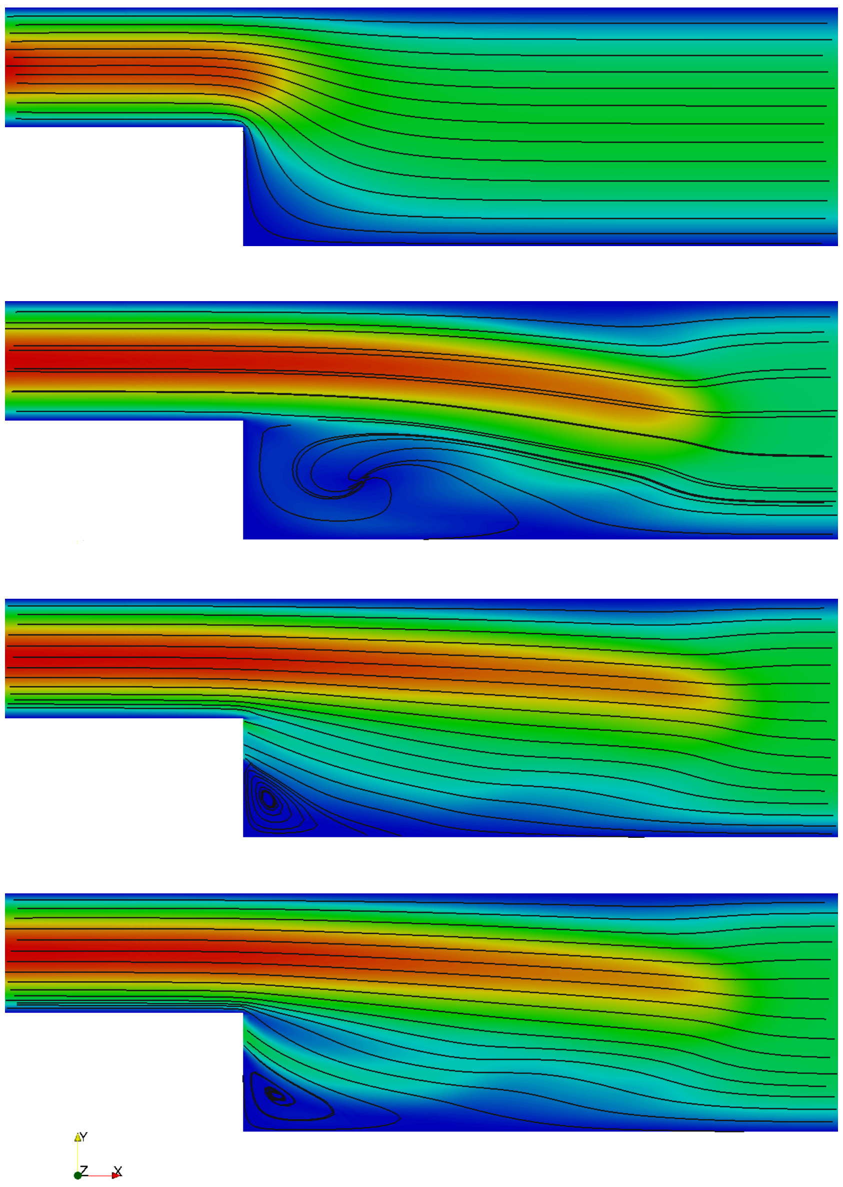


Figure 6.4: Desired flow field (top), uncontrolled Navier-Stokes flow (second), flow controlled by adjoint approach (third) and flow controlled by sensitivity approach (bottom) showing the absolute velocity and streamlines within cutplane $z = 0.5$ at time $t = 4.5$.

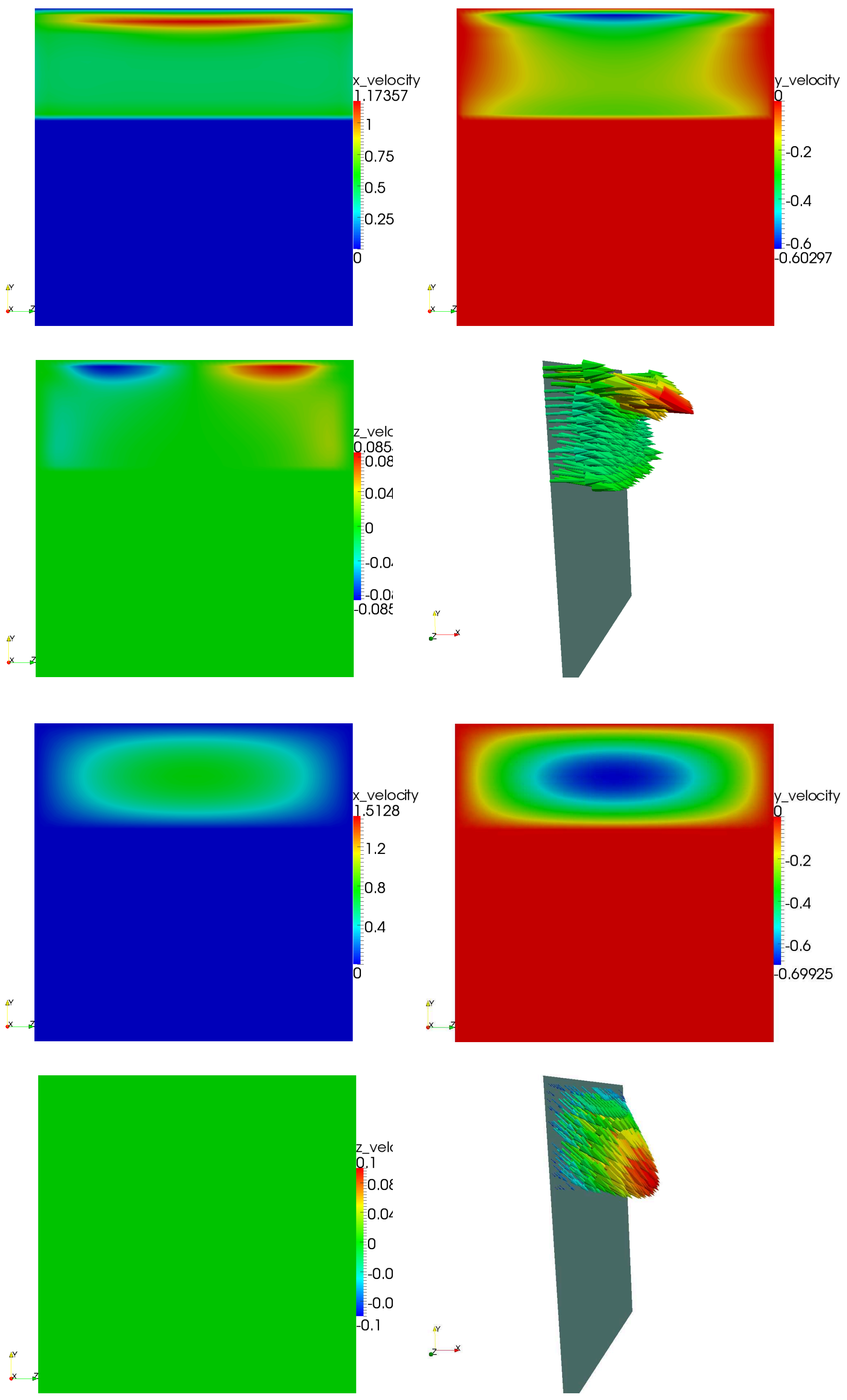


Figure 6.5: Control boundary values on Γ_c at time $t = 4.5$ given by adjoint-based optimisation (top) and sensitivity-based optimisation (bottom) – x-, y- and z-velocity components and resulting vector field are plotted.

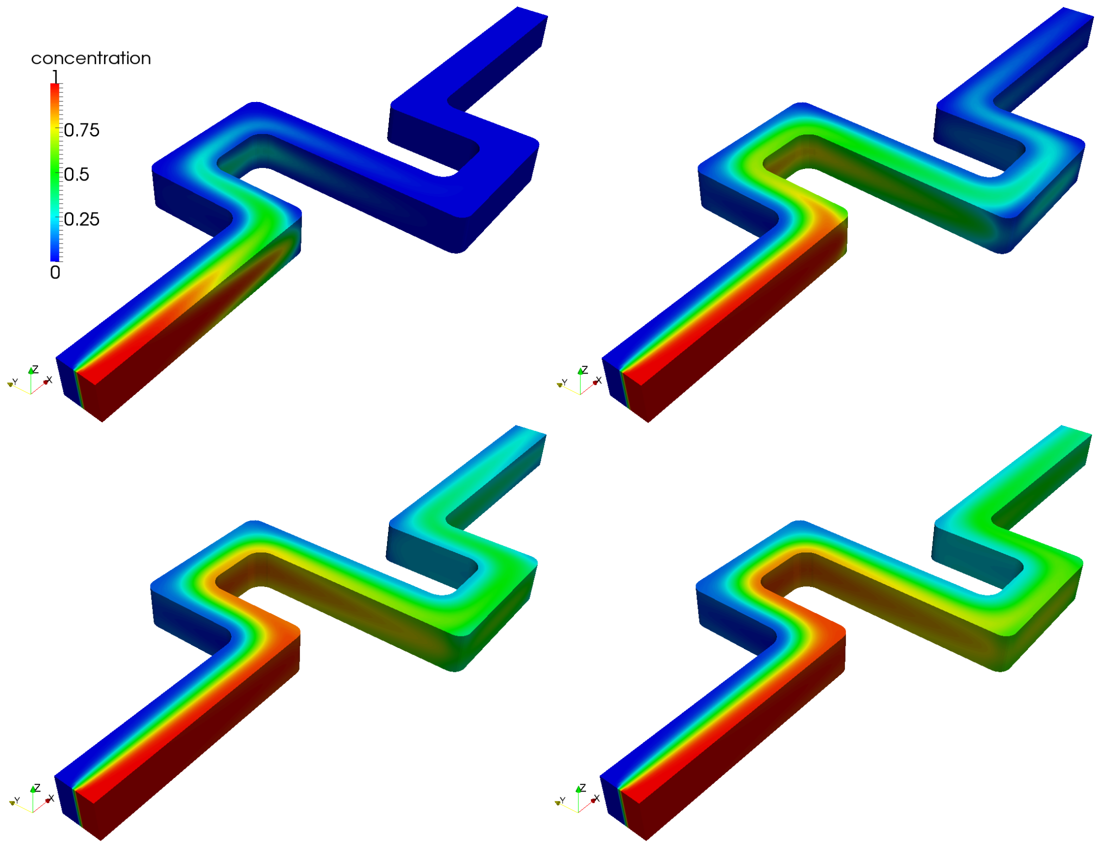


Figure 6.6: Simulation of concentration distribution within the channel during the inflow phase with a pure pressure-driven flow – top at $t = 10$ and $t = 20$, bottom at $t = 30$ and $t = 37$.

of cost functional evolution, the sensitivity-based solution seems physically more relevant. From the computational point of view both approaches have quite different requirements. While for the adjoint-based approach, we only need to solve one linear equation to get the gradient of the cost functional (see Chapter 5.1), the sensitivity-based approach in this example requires to solve 4 linear equations for the design parameters α_k, β_k. Since the optimisation algorithm in both cases nearly uses the same number of iterations (step-length searches and LBFGS updates), one roughly gets on the one hand a slowdown factor of four when the sensitivity-based optimisation is used. On the other hand, the requirements of code restructuring is much more involved for the adjoint-based approach than for the sensitivity-based approach. The very tight coupling of primal and adjoint equations, especially in the case of instationary problems, is already addressed in Chapter 5.2 where a solution for the backup of the primal equation is presented. In the case of sensitivity-based optimisation, this difficulty completely vanishes since the computation of all sensitivities can be done while the primal solver proceeds forward in time. Finally, the choice of suitable optimisation approach should rely on the available computational resources (both software and hardware) but even more on a physically meaningful formulation of the optimisation problem. The adjoint-based optimisation obviously results in a boundary control that is difficult to realise in a physical setting but illustrates the main idea how the control should be applied. Afterwards this information can be used to formulate the more physically motivated sensitivity-based optimisation having a good guess for the design parameter at hand.

6.2 Simulation of Fluid Flow in a Twice-Folded Microchannel

Before we address the optimisation of the electrokinetic micromixer that was introduced in Chapter 1, we first compare our simulation results to the ones of Barz et al. in [10, 11]. We extend the pressure-driven laminar flow with mass transport as reported in [10] and with the electrically

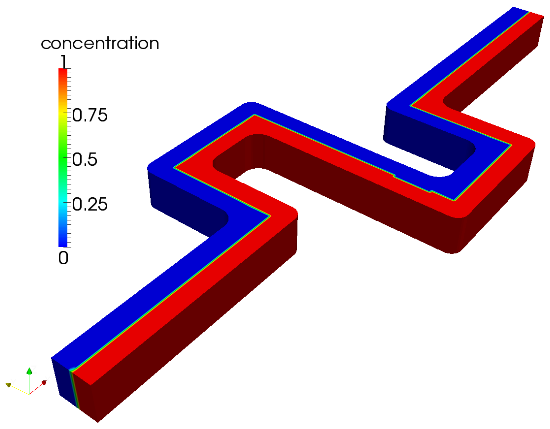


Figure 6.7: Idealised initial condition for the concentration distribution within the channel.

excited flow as in [11]. Furthermore, the geometry is slightly different. Barz et al. used sharp corners, simplifying the generation of the numerical mesh considerably, instead of the rounded corners resulting from the manufacturing process, as used here (see Appendix). Also a remark on the initial condition for the concentration has to be made beforehand. In a strict physical setting one should assume the channel to be filled in the beginning, i.e. at $t = 0$, with only one of the two liquids. A simulation of the ***inflow phase*** within the interval $t \in [0, 37]$ is shown in Figure 6.6 by some snapshots indicating the concentration front that propagates through the channel. While the pressure-driven flow field is fully developed within only a few timesteps, the propagation of the concentration of the species/dye takes longer. Using the supercomputer JUROPA hosted at the Forschungszentrum Jülich (cf. Chapter 4.3.4) with 512 processes, the simulation of this inflow phase takes about 12 hours (at a spatial discretisation of $2.8 \cdot 10^6$ degrees of freedom) and therewith reaches the batch job limit of maximal wallclocktime. As one can see in Figure 6.6 even at $t = 37$ the concentration is not fully developed within the channel. To overcome this physically correct but time consuming simulation, we use an idealised initial condition for the concentration as shown in Figure 6.7, namely an idealised distribution of both liquids without any mixture along the channel. This simplification allows a faster computation and is at least for the steady state solution ($t \to \infty$) of minor concern. But for the results in a certain window $t \in [0, t_{crit}]$ one has to be aware of the influence of the physically idealised initial condition.

The setting remains as introduced in Chapter 2, i.e. we solve the Navier-Stokes equations combined with the convection-diffusion equation (2.15) and an applied constant potential difference $\Delta\phi \neq 0$ if an electrical excitation is under investigation. The boundary conditions for the Navier-Stokes equations are given by a no slip condition $\mathbf{v} = 0$ on the channel walls Γ_0 whenever the potential difference vanishes – this corresponds to the setting in [10]. For a non-vanishing potential difference, i.e. if an electrical field is applied, the transition condition of velocity field between bulk solution and EDL on Γ_0 is implemented by the slip condition as presented in Chapter 2.2.3. Inflow and outflow conditions are implemented by the pressure drop formulation (see Chapter 2.2.3) such that the mean inflow velocity is $v_0 = 9.1 \cdot 10^{-4} \frac{m}{s}$ corresponding to a Reynolds number $Re = 0.1$ – this correlates to the setting in [11].

At the inflow section of the boundary, the concentration has given Dirichlet values $c = 1$ for $y < 0$ and $c = 0$ for $y \geq 0$ to mimic the upstream Y-junction of the microfluidic chip (see Figure 1.1). At the outflow section and at the channel walls resp. at the interface to EDL Γ_0 a vanishing flux is used, i.e. $\partial_n c = 0$. The diffusion coefficient of the concentration field is reported in [10] to be $D = 4.27 \cdot 10^{-10} \frac{m^2}{s}$ resulting in a Schmidt number of $Sc = 2340$. Furthermore, the potential difference across the computational domain is estimated in [11] to be $\Delta\varphi = 47V$ which correlates to an approximate electrical field of $\mathbf{E}_x \approx \pm 14.68 \frac{V}{mm}$. Here, we only use the situation of a negatively directed electrical field resulting in an electroosmotic flow directed against the pressure-driven flow.

In a first simulation, we use the stationary Navier-Stokes equations in combination with the convection-diffusion equation to evaluate the steady state of possible mixing. Figure 6.8 shows the concentration field at different height levels of the channel when no potential difference is applied.

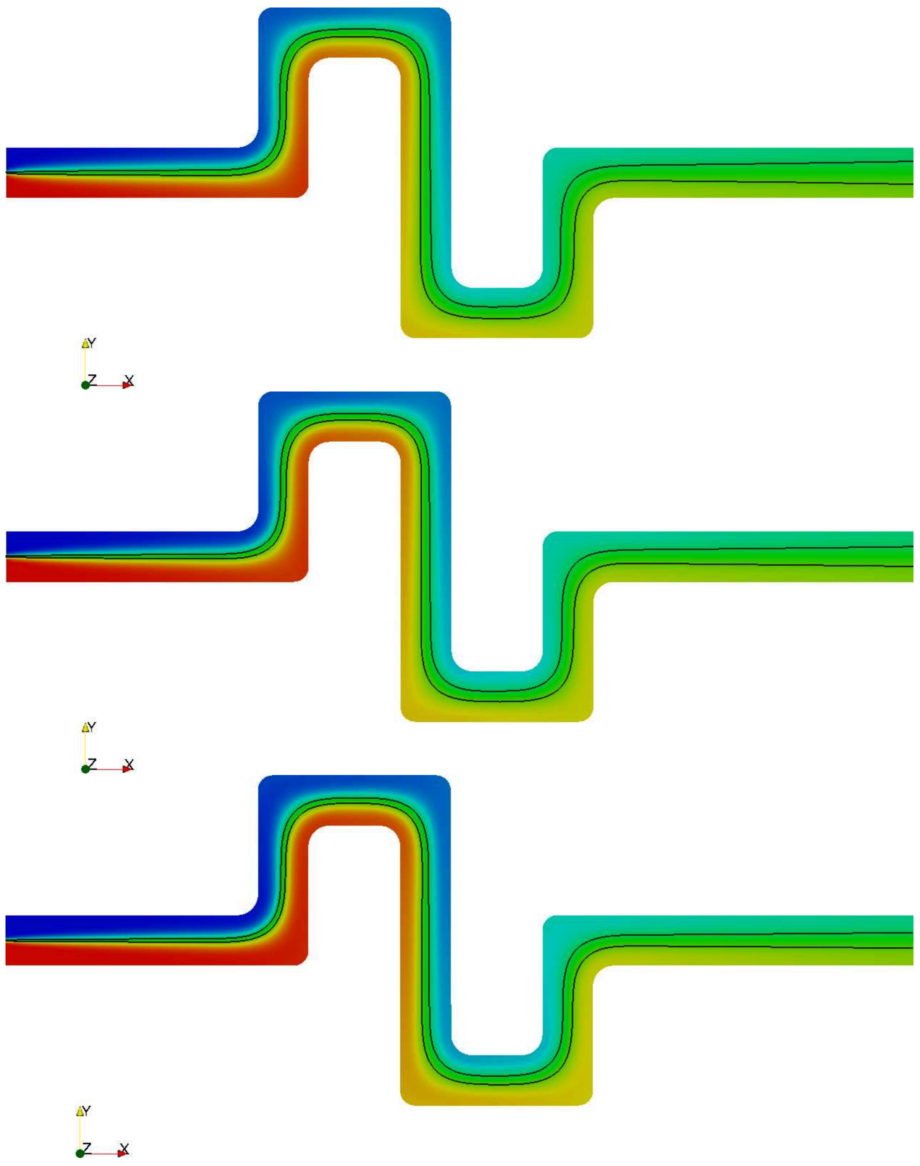


Figure 6.8: Concentration field at different height levels (top $z = -0.43$, middle $z = -0.32$, bottom $z = 0.00$) showing steady state solution of mixing for a three-dimensional Navier-Stokes flow at $Re = 0.1$ and $Sc = 2340$ without electrical field. Isolines indicate a concentration of 0.4 and 0.6.

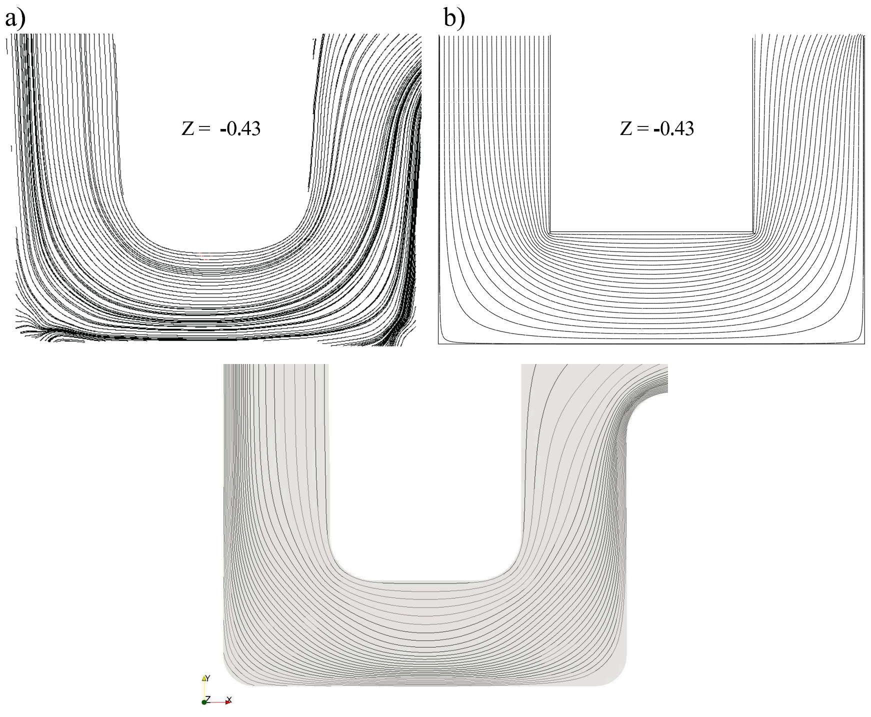


Figure 6.9: Streamlines of the electrically excited fluid flow at the height level $z = -0.43$, top: left measured and right simulated by Barz et al. [11], bottom: results by `HiFlow`^{2} using the parallel solver as presented in Chapter 4.

One can see that even for very low flow rates at $Re = 0.1$, i.e. a relative large residence time of the species enhancing mixing by diffusion, the outcome is not optimal. A slightly better mixing is given near the channel walls (upper two plots in Figure 6.8) compared to mid-height of the channel at $z = 0.0$ due to the much slower flow field within this region yielding a higher residence time. Typical Reynolds numbers in microfluidic applications are rather of the order of 1, though. Hence, the residence time is even shorter underlining the need to improve mixing by other means, e.g. due to the superimposition of an electroosmotic flow.

As a second step, we compare the resulting steady state flow fields (with non-vanishing constant potential difference $\Delta\phi$) within the lower U-bend of the microchannel, i.e. $x \geq 8$ in Figure 1 of the Appendix, to the measured and simulated results from Barz et al. [11]. Figures 6.9, 6.10 and 6.11 show the flow field at specific constant height levels z by means of streamlines. In each Figure the two top plots are kindly provided by Dominik P. J. Barz showing the a) measured and b) simulated results as they are given in [11]. The lower pictures are computed by `HiFlow`^{2} using the presented parallel solver of this thesis.

The flow fields in Figure 6.9 at a height level of $z = -0.43$ show a more or less even streamline configuration indicating a quasi steady flow. Due to the closeness to the bottom of the channel at $z = -0.5$ the flow within this region is dominated by the electroosmotic flow directed from the right to the left and there is no noticeable influence of the pressure-driven flow. Both simulated solutions are in very good agreement to the experimental results. At a slightly higher level of $z = -0.32$, as shown in Figure 6.10, the mutual influence of pressure-driven flow and opposite directed electroosmotic flow is considerably present. The flow topology appears to be very complex containing several saddle points (S) and vortices (V). While the simulation using sharp corners (top right in Figure 6.10) shows two vortices separated by a saddle point located at each inner bend, the modified geometry with rounded corners shows only one vortex point as it is also present in the experimental results. All other characteristics of the streamlines appear similar for the three illustrations. In the mid-height of the channel at $z = 0.0$, the pressure-driven flow appears as the main flow from the left to the right (Figure 6.11). Again a noticeable difference in the simulated results is given at the inner bends – usage of rounded corners clearly improves the agreement to the measured flow field. To summarise, the simulation of electroosmotic flow provides a very

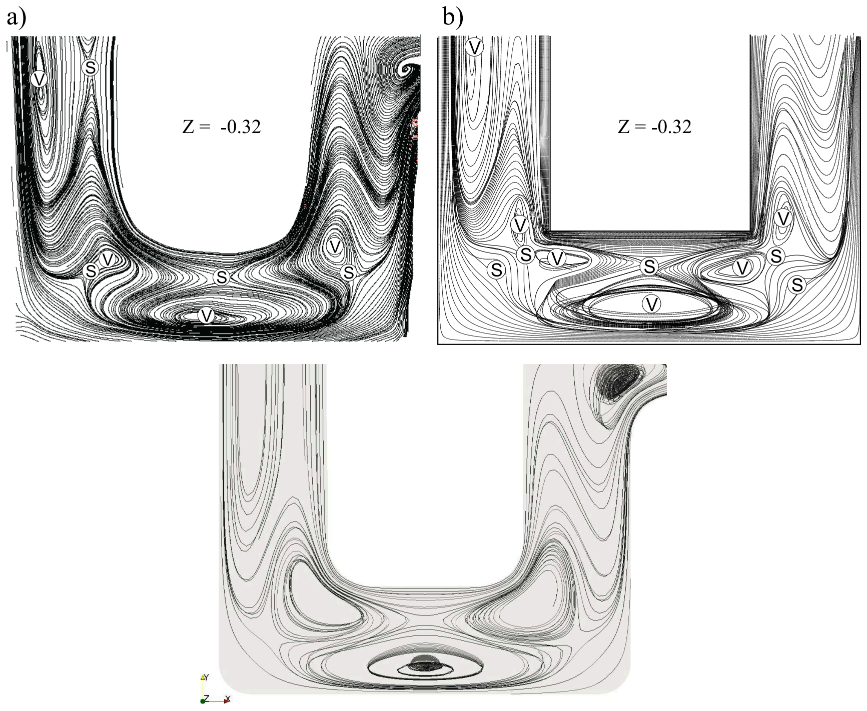


Figure 6.10: Streamlines of the electrically excited fluid flow at the height level $z = -0.32$, top: left measured and right simulated by Barz et al. [11], bottom: results by `HiFlow`^{2} using the parallel solver as presented in Chapter 4.

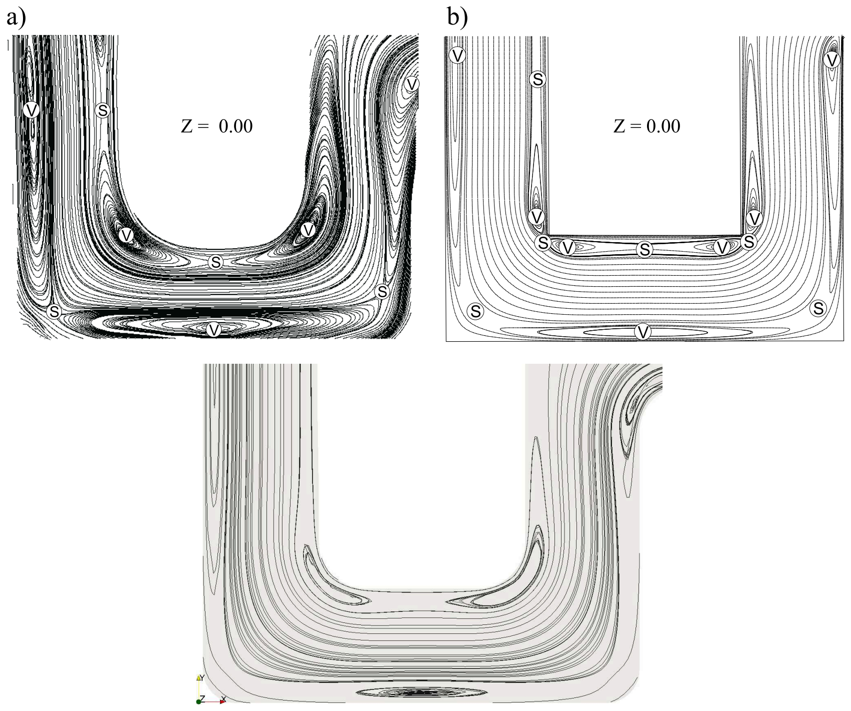


Figure 6.11: Streamlines of the electrically excited fluid flow at the height level $z = 0.00$, top: left measured and right simulated by Barz et al. [11], bottom: results by `HiFlow`^{2} using the parallel solver as presented in Chapter 4.

good agreement to the results presented in [11] and is even better due to the modification of the geometry. The presence of both, pressure-driven flow and electroosmotic flow, features a very complex flow topology that should be appropriate for the task of mixing by increasing the contact area of the two liquids. It remains to judge the quality of mixing and if necessary to improve the mixing by temporal alteration of the potential difference $\Delta\phi$.

We finally aim at solving the full three-dimensional system with a pressure-driven base flow and an additional electroosmotic flow in combination with the convection-diffusion equation for the species concentration to evaluate the influence on mixing quality according to Figure 6.8. It turns out that the simulation of the steady state, i.e. using a stationary solver, at Schmidt number $Sc = 2340$ leads to numerical instabilities due to the complicated flow field and limited mesh refinement capabilities. Even the attempt to solve the instationary system was not successful – especially within the top and bottom region of the channel, we encounter numerical instabilities due to the interference of pressure-driven and electroosmotic flow (see inflow area of the channel in Figure 6.12). A limitation to Schmidt number $Sc = 450$ and corresponding diffusion coefficient $D = 2 \cdot 10^{-9} m^2/s$ gives more stable results but the concentration also obviously shows a better mixing such that any optimisation is not worth discussing.

Based on the dimensionless parameter $Re = 0.1$ and $Sc = 2340$, we get the dimensionless diffusion coefficient $D = 1/(Re \cdot Sc) = 1/234$ such that a suitable mesh size can roughly be estimated by the (mesh) Peclet number. Requiring that $Pe = v_0 h/D < 1$ and assuming at least a mean velocity of 1, we have that $h < 1/234 \approx 0.004$ for the maximal size of a cell in the mesh. Realisation of such a fine mesh in 3D is beyond the actual computational capabilities accessible for this work. Hence, we concentrate in a first step on the 2D implementation of the optimisation for electrokinetic micromixer and have to face the very challenging problem in three dimensions in a future work. This work will be continued using modified algebraic solvers for very large HPC platforms (see also conclusion of Chapter 4) as well as discretisation schemes using stabilisation methods for the convection dominated flow and possibly adaptive refined finite element meshes.

6.3 Optimisation of an Electrokinetic Micromixer

As mentioned in the last section, the optimisation of a full three-dimensional electrokinetic micromixer is beyond the computational capabilities of available hardware. Therefore, we investigate the fundamental procedure for the two-dimensional case and address the question whether the results can be transfered to the three-dimensional case. Figure 6.13 shows the streamlines of the electrically excited flow fields in both two- and three-dimensional case as presented in the previous section for different Reynolds numbers. It is obvious that the two-dimensional approximation reveals a qualitatively good approximation for the channel mid-height. Hence, at least for the mid-height of the channel at $z = 0.0$, where the influence of the electrical double layer located at the top and bottom of the channel is not dominating, we can interpret the results of two-dimensional simulation/optimisation also for the three-dimensional case.

In addition to the previous section, we also use a Reynolds number of $Re = 1.0$ for the optimisation since such a flow regime is more realistic in terms of microfluidic applications. Whereas the flow field situation for $Re = 0.1$ and $Re = 1.0$ shows no significant difference (cf. Figure 6.13), the latter setting results in a lower residence time of a species within the channel such that a pure pressure-driven flow will not result in sufficient mixing (cf. Figure 6.14). Like in the comparison of the streamlines, we see that the resulting concentration at $Re = 0.1$ for the two-dimensional case (upper plot in Figure 6.14) is in a good agreement to the mid-height cutplane of the three-dimensional case (lower plot in Figure 6.8). An additional benefit of higher Reynolds numbers and shorter residence times is the better justification of the idealised initial condition of concentration given in Figure 6.7 which is similar to the steady state solution of pressure-driven flow in lower plot of Figure 6.14.

To judge the mixing of concentration, we need to define a measure that also will be used as cost functional for optimisation. Since a satisfying mixing is obviously obtained when the concentration

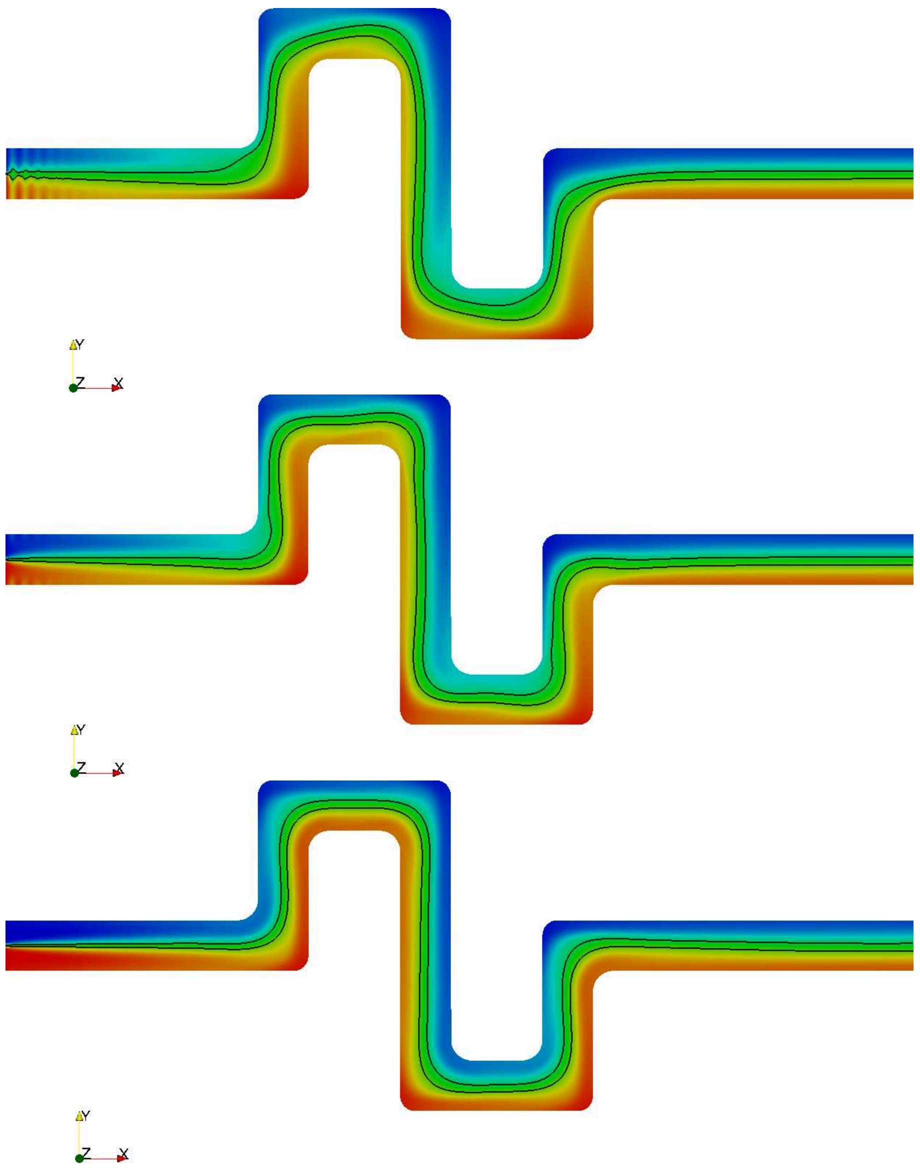


Figure 6.12: Concentration field at different height levels (top $z = -0.43$, middle $z = -0.32$, bottom $z = 0.00$) and at $t = 10.6$ for a three-dimensional Navier-Stokes flow at $Re = 0.1$ and $Sc = 2340$ within an electrically excited fluid flow. Isolines indicate a concentration of 0.4 and 0.6.

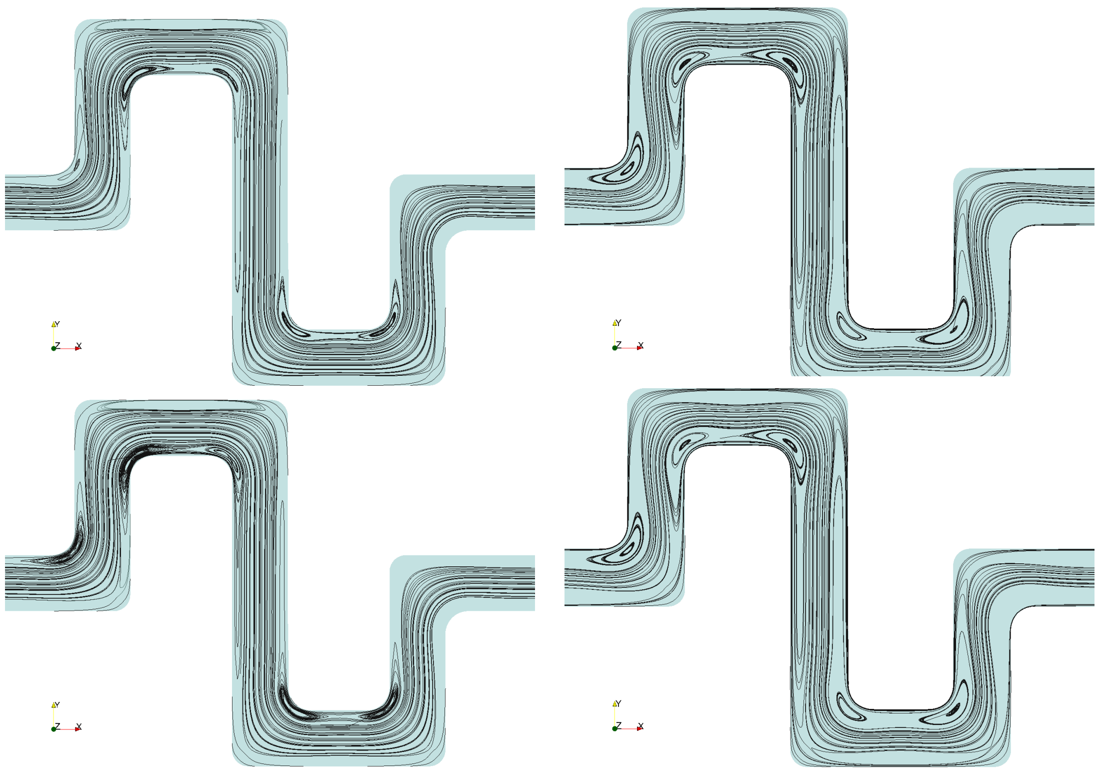


Figure 6.13: Left column: streamlines of the electrically excited fluid flow at the height level $z = 0.00$ using 3D simulation at Reynolds number $Re = 0.1$ (top) and $Re = 1.0$ (bottom). Right column: streamlines of the electrically excited fluid flow using 2D simulation at Reynolds number $Re = 0.1$ (top) and $Re = 1.0$ (bottom).

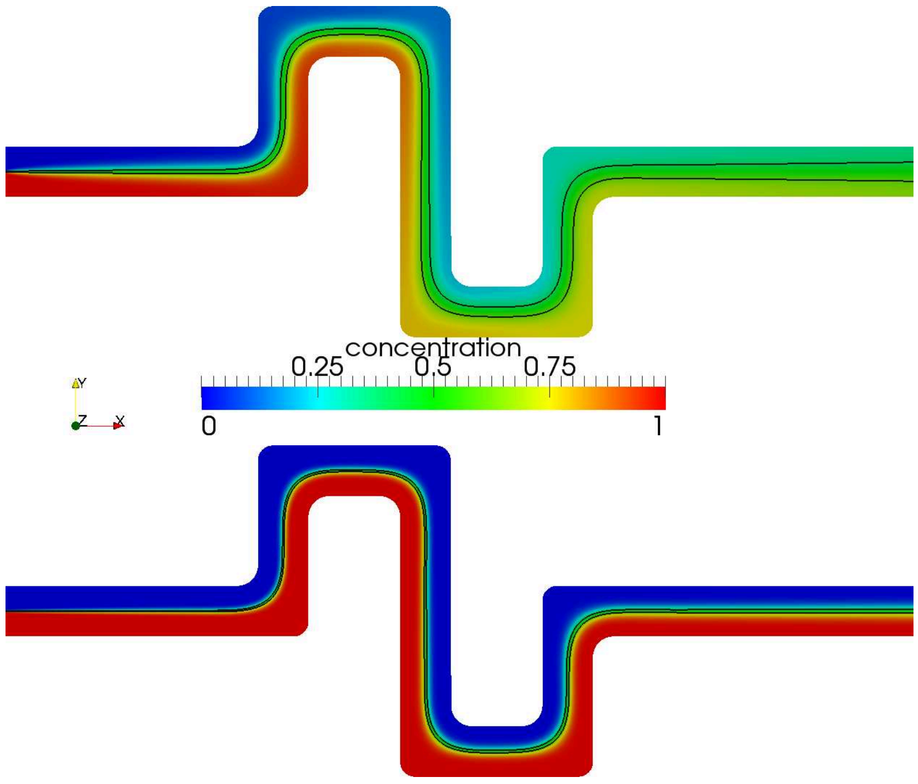


Figure 6.14: Steady state solution of the concentration field for a two-dimensional Navier-Stokes flow at $Re = 0.1$ (top), $Re = 1.0$ (bottom) and $Sc = 2340$ without electrical field. Isolines indicate a concentration of 0.4 and 0.6.

is ~ 0.5, a possible measures is given by the L^2 difference in Ω_s, i.e.

$$J_1(c) = \int\limits_{\Omega_s} |c - 0.5|^2 \, d\mathbf{x} \tag{6.5}$$

or by the fraction of unacceptable concentration

$$J_2(c) = \frac{1}{|\Omega_s|} \int\limits_{\Omega_s} \chi(c) \, d\mathbf{x}, \quad \chi(c) = \begin{cases} 0 & \text{if } c(\mathbf{x}) \in [0.5 - \epsilon, 0.5 + \epsilon] \\ 1 & \text{else} \end{cases} \tag{6.6}$$

where $\Omega_s \subset \Omega$ denotes the observation area of interest and ϵ determines an acceptable variance for which 0.1 is used in the following. Ω_s is chosen to be the outflow section of the channel, namely the straight part at $x \geq 11.65$ (cf. Appendix). For an ideal mixing both $J_1(c)$ and $J_2(c)$ should be zero.

The dimensionless system of partial differential equations describing the bulk flow within the electrokinetic micromixer is given by (2.49). Boundary conditions are already discussed in detail in Chapter 2.2. As in the simulation before, we use the pressure drop formulation for the velocity field, where the pressure difference between inflow and outflow boundary is adjusted such that the dimensionless mean velocity v_0 equals 1 to ensure the correct Reynolds number. In particular we use the following set of boundary conditions for the optimisation of the electrokinetic micromixer at $Re = 1.0$:

$$\begin{aligned} \mathbf{v}|_{\Gamma_0} &= -\Pi_2 \nabla \phi, & \partial_n \mathbf{v}|_{\Gamma_{in} \cup \Gamma_{out}} &= 0, & p|_{\Gamma_{in}} &= 304.2, \\ p|_{\Gamma_{out}} &= P_{out} = 0, & \phi|_{\Gamma_{in}} &= 0, & \phi|_{\Gamma_{out}} &= \phi_c(t, \alpha_k), \\ \partial_n \phi_{\Gamma_0} &= 0, & c|_{\Gamma_{in}} &= \begin{cases} 0 & \text{if } y \geq 0 \\ 1 & \text{if } y < 0 \end{cases}, & \partial_n c|_{\Gamma_0 \cup \Gamma_{out}} &= 0. \end{aligned} \tag{6.7}$$

The control of applied potential at Γ_{out} is given by an analytical function involving the design/optimisation parameter α_k to describe the temporal modifications.

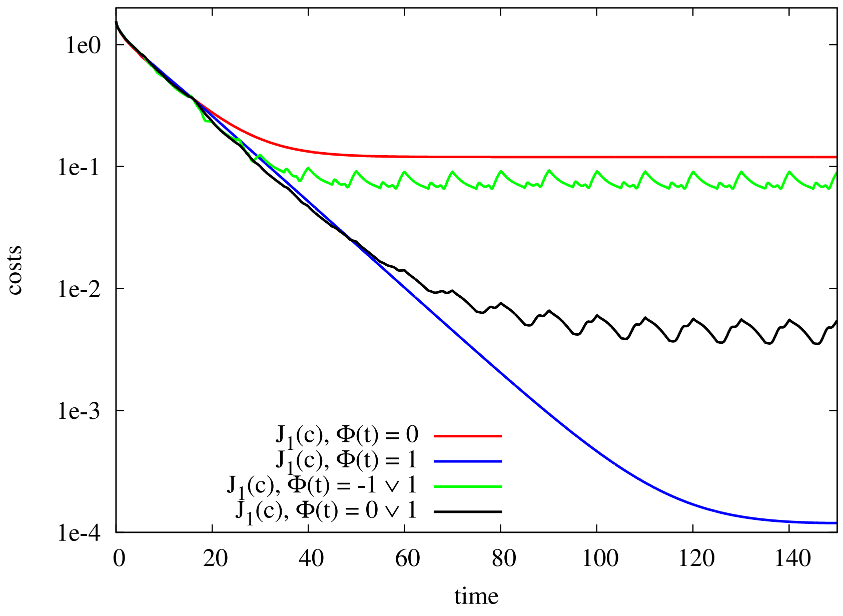
Before any definite function for the applied potential $\phi_c(t, \alpha_k)$ is set, one should observe the mixing for the pure pressure-driven flow as well as for the case of a constant and an alternating potential as reported in [11]. Hence, we first evaluate the evolution of the cost functionals (6.5), (6.6) using four fixed cases:

1. only pressure-driven flow, i.e. $\phi(t)|_{\Gamma_{out}} = 0$,
2. constant applied potential, i.e. $\phi(t)|_{\Gamma_{out}} = 1$,
3. rectangular alternating potential with period $t = 10$ and $\phi(t)|_{\Gamma_{out}} = -1 \vee 1$,
4. rectangular alternating potential with period $t = 10$ and $\phi(t)|_{\Gamma_{out}} = 0 \vee 1$.

The results are shown in Figures 6.15 and 6.16 for $Re = 0.1$ and $Re = 1.0$. A first observation can be made for the pure pressure-driven flow: after a certain time t_{diff} a steady state for the concentration field is given (cf. also Figure 6.14). The time t_{diff} can be interpreted as diffusion time of the underlying system. Depending on the Reynolds number and therefore on the residence time of a species within the microchannel, we have $t_{diff} \sim 40$ for $Re = 0.1$ and $t_{diff} \sim 25$ for $Re = 1.0$ which obviously results in an increasing value for the costfunctionals as the Reynolds number increases. For the cases of a non-vanishing applied potential one has to distinguish the constant potential and the alternating potential. The first one induces a steady flow field (cf. Figure 6.13) which for $t \to \infty$ gives a steady state of concentration field. Using a stationary solver, we determine the limits of cost functionals to be $J_1(c) = 1.3 \cdot 10^{-4}$, $J_2(c) = 0.0$ for $Re = 0.1$ and $J_1(c) = 0.11$, $J_2(c) = 0.82$ for $Re = 1.0$. Thus, for low Reynolds number a complete mixing can be achieved at least for $t \to \infty$ while for increasing Reynolds number the mixing gets worse due to a lower residence time of the species within the microchannel.

Both cases of an alternating applied potential result in a periodic solution. The limit of the costfunctionals can only be given by a mean value for the case of applied potential $\phi(t)|_{\Gamma_{out}} = -1 \vee 1$ while for the fourth case, i.e. $\phi(t)|_{\Gamma_{out}} = 0 \vee 1$, the costfunctional seems to tend to the same limit as for the constant applied potential. This behaviour can be explained by the fact that whenever $\phi(t)|_{\Gamma_{out}} < 0$ the electroosmotic flow is directed aligned to the pressure-driven flow. Therefore, the residence time of a species is even smaller than for the pure pressure-driven flow which degrades the mixing induced by the previously opposite directed electroosmotic flow. For the alternating potential that is non-negative one does not obtain this effect. Here, the phase of $\phi(t)|_{\Gamma_{out}} = 0$ only delays the mixing since within this phase the actual distribution of concentration field is transported like for the pure pressure-driven flow. Figure 6.17 shows a snapshot of the concentration field at $t = 96$ and at $Re = 1.0$ using a constant and an alternating applied potential. One can see the homogenous mixing in the constant case due to the steady flow field. The alternating potentials induce a much more complex flow field that is not necessarily well suited for a better mixing – at least within a long term observation.

Finally, a comparison of the two used cost functionals in Figures 6.15 and 6.16 indicates that both are equally suitable to evaluate the mixing of species concentration. While the L^2 difference used for $J_1(c)$ weights the variance of concentration field to desired value of 0.5 in a smooth manner, the cost functional $J_2(c)$ is more restrictive in this sense. Hence, if for any reason a mixing of species within the interval $c \in [0.4, 0.6]$ is crucial for the entire microfluidic setting on the chip (e.g. to start a chemical reaction), one should better use the cost functional $J_2(c)$. If the general possibility of mixing should be observed it is however also warrantable to use the cost functional $J_1(c)$. To summarise, three major results are given by the simulations: for $t \to \infty$



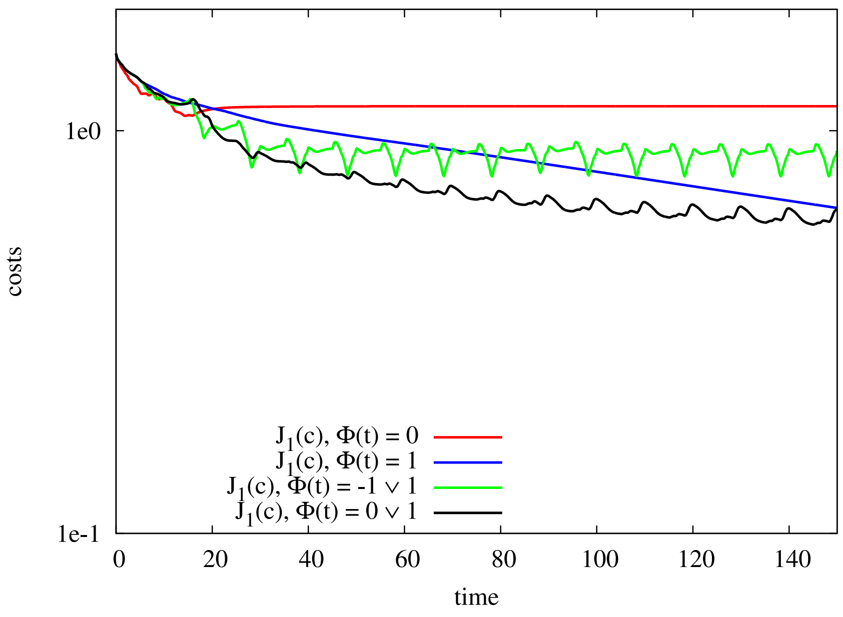
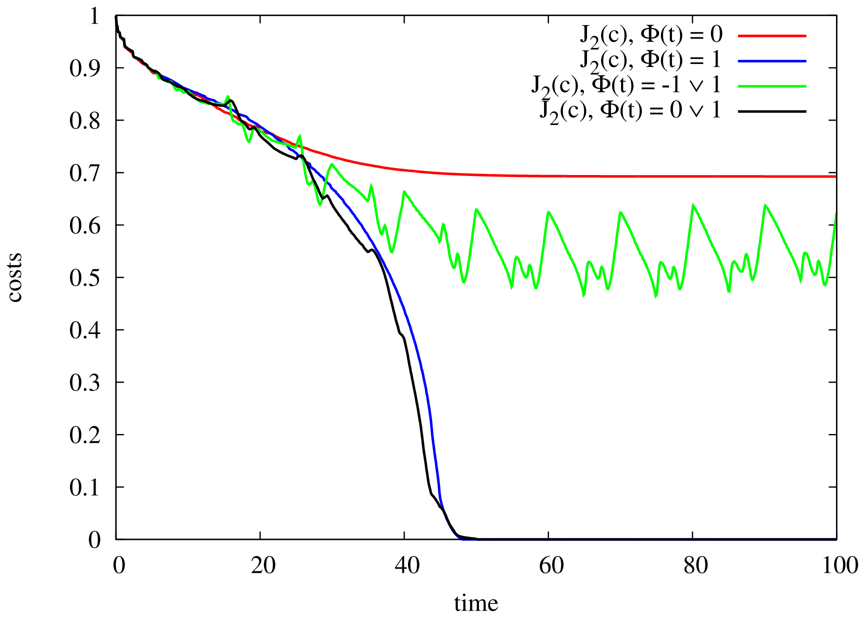


Figure 6.15: Evolution of cost functional $J_1(c)$ (6.5) with constant and alternating applied potential $\phi(t)|_{\Gamma_{out}}$ for $Re = 0.1$ (top) and $Re = 1.0$ (bottom).



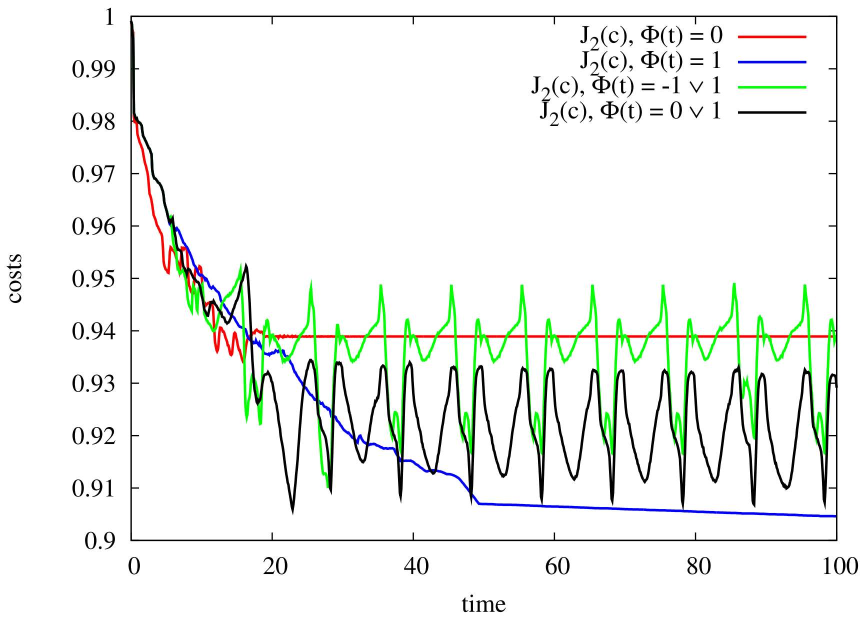


Figure 6.16: Evolution of the cost functional $J_2(c)$ (6.6) with constant and alternating applied potential $\phi(t)|_{\Gamma_{out}}$ for $Re = 0.1$ (top) and $Re = 1.0$ (bottom).

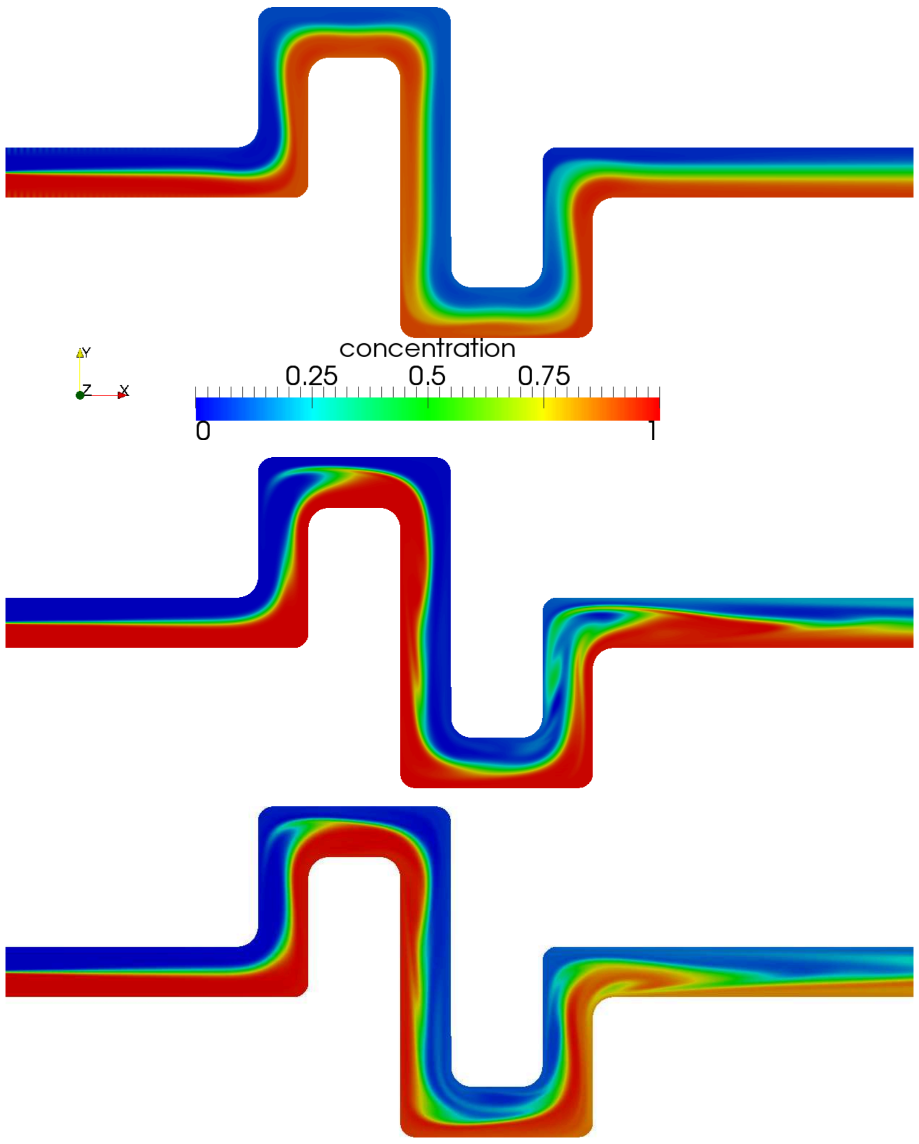


Figure 6.17: Concentration distribution for a two-dimensional electrically excited flow at $Re = 1.0$ and $Sc = 2340$. Solution at $t = 96$ for constant applied potential (top), alternating applied potential $\phi(t)|_{\Gamma_{out}} = -1 \vee 1$ (middle) and $\phi(t)|_{\Gamma_{out}} = 0 \vee 1$ (bottom).

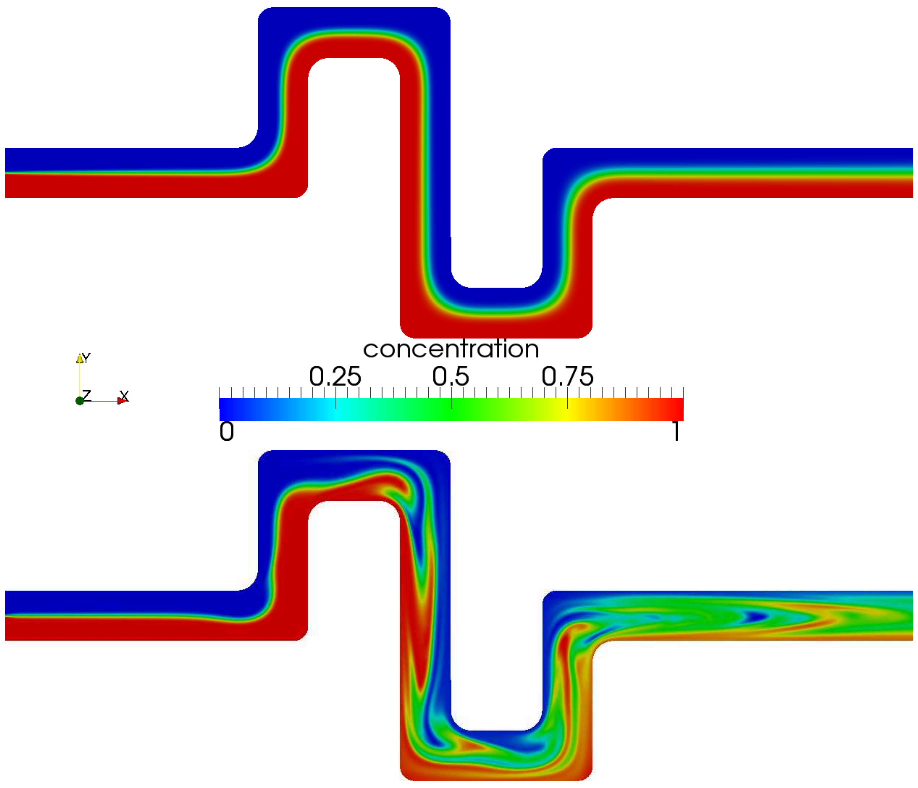


Figure 6.18: Concentration distribution for a two-dimensional flow at $Re = 1.0$ and $Sc = 2340$. Solution at $t = 30$ for the pure pressure-driven flow (top) and for the optimal electrically excited flow using $\phi_2(t, \omega, h)$ (bottom).

a constant applied potential gives the best mixing, for short time observations the alternating potential attains a more favourable mixing and finally the mixing at small temporal horizon is much more involved the bigger the Reynolds number is.

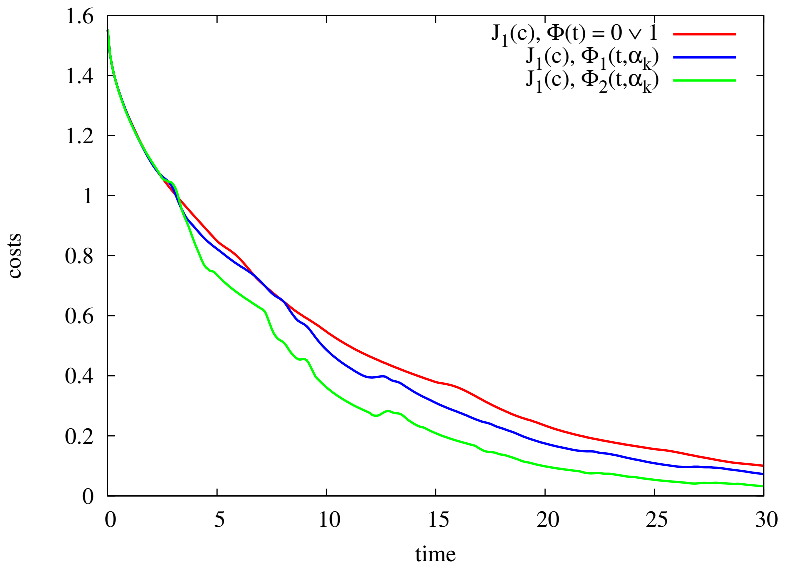
We conclusively aim at optimisation of the electroosmotic micromixer by means of minimising the cost functional $J_1(c, \alpha_k)$. Since a fast and homogeneous mixing of species is aimed for, we furthermore restrict the observed time interval to $t \in [0, 30]$. Within this interval, the overall decrease of the cost functional for all four simulated cases is comparable (see Figures 6.15 and 6.16) such that a demand for an optimisation is given. The optimisation problem of the electrokinetic micromixer involving the design parameters α_k is stated as

$$\min_{c,\alpha_k} J(c, \alpha_k) = \frac{1}{2} \int_0^T \int_{\Omega_s} |c - 0.5|^2 \, d\mathbf{x} \, dt + \frac{\lambda}{2} \sum_{k=0}^{n} |\alpha_k|^2 \tag{6.8}$$

such that the system (2.49) with boundary conditions (6.7) is fulfilled. We compare the optimal parameter settings to the results obtained by the alternating applied potential $\phi(t)|_{\Gamma_{out}} = 0 \vee 1$ with a period of $t = 10$ – see Figure 6.19. Motivated by the simulation results, we chose two different possibilities for the applied potential $\phi_c(t, \alpha_k)$ to check the influence of time dependent alteration. First a smooth applied potential described by the sinus function is used that also incorporates a constant term, namely

$$\phi_1(t, \alpha_k)|_{\Gamma_{out}} = \alpha_0 + \alpha_1 \sin(\alpha_2 t).$$

The optimal design parameter for $Re = 0.1$ are given by $\alpha_0 = 0.0366$, $\alpha_1 = 1.4518$, $\alpha_2 = 1.3256$ showing an alternating potential with a period of $t = 4.74$ and a nearly negligible constant offset. For the optimisation at $Re = 1.0$ the optimal parameter are comparable: the constant offset is even less ($\alpha_0 = 0.0184$) while the amplitude is slightly higher ($\alpha_1 = 1.4761$) at a similar frequency of $\alpha_2 = 1.3183$ respectively a period of $t = 4.76$. Compared to the simulation results (see Figure



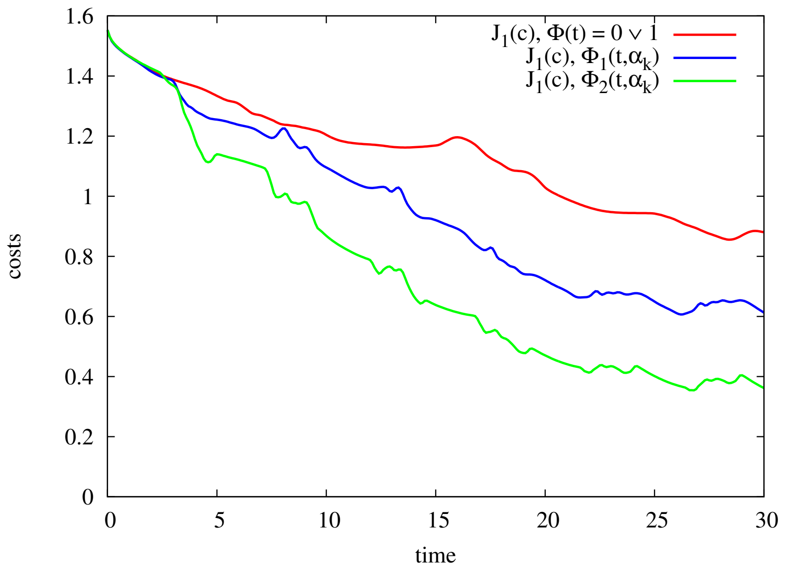


Figure 6.19: Evolution of the cost functional $J_1(c)$ for the alternating applied potential $\phi(t)|_{\Gamma_{out}} = 0 \vee 1$ and for the optimal controlled cases at $Re = 0.1$ (top) and $Re = 1.0$ (bottom).

6.19), the halved period results especially for $Re = 1.0$ in a visible benefit. With the frequency of the alternating potential that optimises the mixing ($\omega = 1.33$), we secondly chose a Fourier series

$$\phi(t, \alpha_k)|_{\Gamma_{out}} = \frac{\alpha_0}{2} + \sum_{k=1}^{n} \alpha_k \cos(k\omega t) + \sum_{k=n+1}^{2n} \alpha_k \sin((k-n)\omega t)$$

with $n = 2$ as control function to get a deeper insight on the weights of alternating and constant terms. However, this does not give any additional benefit, i.e. the optimisation algorithm does not result in a remarkable update due to very short step-length and small gradients showing that the frequency is optimal. Finally, the rectangular applied potential is optimised for the frequency $\alpha_0 := \omega$ and amplitude $\alpha_1 := h$ which can formally be written as

$$\phi_2(t, \omega, h)|_{\Gamma_{out}} = \frac{4h}{\pi} \sum_{k=1}^{\infty} \frac{\sin((2k-1)\omega t)}{2k-1}$$

The partial derivatives with respect to ω and h have to be evaluated for the sensitivity-based optimisation. For the frequency ω this gives a possibly non-convergent series

$$\frac{\partial \phi_2(t, \omega, h)}{\partial \omega} = \frac{4ht}{\pi} \sum_{k=1}^{\infty} \cos((2k-1)\omega t).$$

In order to still treat this ansatz, we restrict to the first ten terms of the series approximating the rectangular shape by a smooth function. Again for both Reynolds numbers we find very similar results. An optimal amplitude of $h \sim 1.49$ and a frequency of $\omega \sim 1.309$ are in nearly perfect accordance to the previous results of $\phi_1(t, \alpha_k)$. Nevertheless, the change from smooth function to clear switching of the applied potential results in a distinct gain in the evolution of cost functional (cf. Figure 6.19) and a noticeable better mixing compared to the pure pressure-driven flow within the microchannel (cf. Figure 6.19).

A last remark has to be made on the convergence of the optimisation algorithm. For one thing, smooth functions (as $\phi_1(t, \alpha_k)$ and also tested polynomial ansatz) give a steady reduction for the cost functional within $4 - 6$ LBFGS iterations while for the case $\phi_2(t, \alpha_k)$ only 1 or 2 iterations were successful. For another thing, optimisation at Reynolds number $Re = 0.1$ turns out to be somewhat more sensitive than at $Re = 1.0$, i.e. the line-search within the optimisation algorithm more often gets suitable step-lengths. Especially the optimisation of the frequency ω in this case is very difficult and a good initial guess is crucial. Such a behaviour illustrates the difficulty in choosing suitable design parameters combined with an analytical function to describe the control whenever the sensitivity-based optimisation is used. Therefore, an optimisation based on the adjoint equations will be an obvious future work, being aware of the fact that the results might not directly be physically meaningful as for the previous optimisation of vortex reduction behind a backward facing step, but should give a more precisely knowledge of the control capabilities for the electroosmotic micromixer.

Chapter 7

Summary and Outlook

In this thesis, we investigate the simulation and optimisation of several fluid flow problems within microchannels with focus on the homogeneous and fast mixing of two liquids. This process is crucial to fulfill the lab-on-a-chip concept and serves as a prototype for several (bio)chemical mechanisms involving different species. For pure pressure-driven flows of an incompressible Newtonian fluid, we derive the Navier-Stokes equations describing the velocity and pressure field in the channel. A convection-diffusion equation is used to determine the distribution of species concentration. Since the optimisation of mixing is addressed by an induced secondary flow using the electrokinetic properties of a fluid in contact with an electrically charged surface, we need to further extend the governing equations by a model for the *electrical double layer (EDL)* near the channel walls. The outcome of the method of matched asymptotic expansions by Barz et al. [11] reveals a separation of the bulk flow from that within the EDL and enables to capture the influence of the EDL by suitable slip boundary conditions. Otherwise one would have to resolve the very thin boundary layer leading to impracticable fine meshes and very large discrete systems. This approach was already used within several works that do not resolve the EDL. Nevertheless, for a future work a direct comparison of the simulated flow field with the resolved EDL to experimental data should be conducted in order to validate the asymptotic approach.

Since the Laplace equation describing the potential within the fluid only couples to the Navier-Stokes equations by the slip boundary condition of velocity field and since the convection-diffusion equation can be treated totally self-contained once the velocity field is computed, we reduce the investigation on numerical solver and preconditioner to the discretised Navier-Stokes equations only. The presented finite element framework and the algebraic preconditioners however also cover the remaining partial differential equations (PDE) and the complete system. To obtain suitable preconditioners that mind the saddle-point structure of the Navier-Stokes equations, we incorporate the *Multilevel ILU preconditioner* provided by `ILU++` [112] to the finite element solver package `HiFlow`^{2} [20] and check its capabilities for the considered microflow problems. By appropriate preprocessing of the system matrix and especially by adjustment of the threshold used within the dropping rule of incomplete LU-decomposition, the `ILU++` preconditioner appears to be by far superior to any `ILU(p)` version. These convincing results for the sequential preconditioner need then to be transfered to a parallel framework to be able to solve three-dimensional problems in reasonable time. To this, we use a domain decomposition method based on a non-overlapping partition of the finite element mesh. The question arises which parallel data structure is well-suited to form a scalable extension to the sequential Multilevel ILU preconditioner. We implement and compare two possible parallel data structures, namely a *row block distribution* of data along the processes and a distribution motivated by the *Schur complement* formulation. Both data structures allow the usage of the sequential Multilevel ILU preconditioner locally on each process to get a parallel preconditioner for iterative Krylov subspace methods. While for the row block distribution a Block Jacobian approach simply skips all entries in the system matrix that couple several subdomains/processes, the Schur complement approach reduces the entire problem to a smaller one that is stated on the sceleton degrees of freedom between the subdomains. For the standard linear algebraic operations we find a similar scalability for both versions with a slight advantage for the purely local defined Schur complement data structure. The resulting parallel preconditioners allow for a variety of parameter to be set and show, on the one hand, that the Schur complement approach possesses some gainful properties to significantly reduce the number of global Krylov subspace iterations. On the other hand, we also see that there is a high sensitivity on chosen parameters and that the preconditioner is not unconditionally robust to arithmetic over- and/or underflows. In contrast to the linear algebraic operations, the absolute time to solve a linear system does not show a very good scalability. The essential drawbacks of both parallel preconditioners are obvious. For the Block Jacobian approach, too many couplings within the system matrix are skipped when the number of processes increases and therefore the preconditioner comes closer to the identity. For the Schur complement approach, an increasing number of processes leads to a bad ratio of interior to sceleton degrees of freedom such that the benefit of much smaller reduced problem vanishes. A future work within this direction should investigate the principal limits of scalability which are given by the fact that the system matrix is strongly coupled and cannot be reduced.

Once the parallel solver and preconditioner is established, the simulation of fluid flow within microchannels can be handled but the optimisation task still remains. The adjoint based optimisation of instationary problems owns some special features that have to be regarded within the software design. It is important to note that the adjoint problem is indeed always linear but has to be solved backward-in-time. Furthermore, if the primal problem or the cost functional is nonlinear each step of the adjoint solution is directly coupled to the corresponding step of the primal one. This requires special backup techniques of the primal solution which might be very costly on high-performance computing (HPC) systems due to increasing importance of data I/O. The proposed new approach which combines *parallel I/O* techniques with *checkpointing schemes* allows a considerable reduction of storage amount and the related time for storage access assuming an adequate trade off in the combination of both components. One should note that this approach does not take into account the case of an HPC platform with local storage resources. An extension of the derived scheme towards an hybrid method is an obvious next step and should allow to simultaneously tackle the issue of local and global parallel I/O in the context of PDE-constrained optimisation.

All parallel tools developed in this work are finally successfully tested for the example of a backward facing step flow using a more academic setting of tracking type optimisation governed by the instationary three-dimensional Navier-Stokes equations. We find that the sensitivity-based optimisation approach yields slightly worse results than the adjoint-based approach that has much more degrees of freedom to be controlled. Especially the physically more reasonable character of the sensitivity-based approach motivates of using this one for the *optimisation of electrically excited micromixer* as well. For the simulation of electrically excited fluid flow, we are able to expand former simulation results by Barz et al. [11]. Meshes closer to the real geometry which lead to discrete systems of nearly three million degrees of freedom per timestep requiring the usage of developed parallel solver/preconditioner with at least 512 processes on the supercomputer JUROPA. Nevertheless, an extension of the electrically excited fluid flow by a convection-diffusion equation to describe the mixing of species fails for the three-dimensional case due to numerical instabilities caused by a mesh that is still not fine enough. A possible reduction of the Schmidt number and therefore higher diffusion avoids this problem but also changes the physical situation. Alternatively, a further research might address adaptively refined meshes and stabilisation techniques for the convection dominated transport process. Also, the replacement of the convection-diffusion equation for the simulation of concentration field of the species might be a solution. If a numerical simulation of diffusion-free mass transport by means of particle tracking is in accordance with the physical problem, i.e. no diffusion of species can take place, the distribution of species can also simply be obtained by integration of the particle trajectory.

However, two-dimensional simulations can be done on the actual computational capabilities of available hardware showing that the velocity field of pure pressure-driven flow and electrically excited flow is in good agreement to the mid-height level of a three-dimensional simulation. Hence, the optimisation of the electrokinetic micromixer is carried out for the two-dimensional problem to get a first hint of the influence of the time-dependent applied potential on the quality of mixing. It turns out that the interpretation of mixing in terms of cost functionals is difficult and that the results differ significantly for long term simulations. Additionally, a careful choice of design parameters in combination with an analytical function for the applied potential is not straight forward and needs already some experience e.g. by previous simulations. For the case of fixed time interval in which an optimal mixing is aimed for, the best results are given by an alternating applied potential. In contrast to a constant applied potential inducing a steady electroosmotic flow that is directed oppositely to the pressure-driven flow, this approach induces a much more complex flow field resulting in a larger contact area of the species and a bigger residence time to increase the mixing. With respect to a future research, the extension to a fully three-dimensional setting used for the electrically excited micromixer would be desireful. The presented approaches show the fundamental capability for both simulation and optimisation of three-dimensional instationary problems. Nevertheless, a further improvement of the overall solution process with respect to the very high computational requirements is essential. This encompasses, on the one hand, a more elaborate discretisation by an adaptive mesh refinement and the stabilisation of the convection-diffusion equation and, on the other hand, a further revision of parallel solver and preconditioner in terms of scalability. Finally, it remains interesting to study whether an adjoint-based optimisation approach for the electrically excited micromixer is physically meaningful and yields new attractive results.

Appendix

Here, we show the parameter settings for both examples of this thesis as they are used in the numerical simulation and optimisation.

Electrokinetic Micromixer

The discretised geometry of the meander part of the microfluidic chip is given in Figure 1 showing a coarse mesh to indicate the used hexahedral cell type. Missings parts of the microfluidic chip are modelled by suitable inflow and outflow boundary conditions as presented in Chapter 2. The width of the square channel is $d_0 = 1.1 \cdot 10^{-4}\ m$ which is also used for scaling of the geometry. For the corners, we have to distinguish two types – in the direction of the fluid flow, we have concave corners possessing a radius of $r_0 = 4.5 \cdot 10^{-5}\ m$ while for the convex corners we have $r_1 = 3.2 \cdot 10^{-5}\ m$. Since the entire microchannel is made up of three glass layers (see Figure 1.1), the junction of top and bottom walls to side walls can certainly be modelled by idealised right angles. Any further parameters are taken from the experimental setting presented in [10, 11]:

- mean axial inflow velocity: $v_0 = 9.1 \cdot 10^{-4}\ \frac{m}{s}$
- fluid density: $\rho = 1000\ \frac{kg}{m^3}$
- dynamic viscosity: $\mu = 1 \cdot 10^{-3}\ \frac{kg}{ms}$
- diffusion coefficient of the species concentration: $D = 4.27 \cdot 10^{-10}\ \frac{m^2}{s}$
- Debye-length (approximate width of electrical double layer): $\ell_D = 9.7 \cdot 10^{-7}\ m$
- potential difference across simulated channel-segment: $\varphi_0 = 47\ V$
- charge density at microchannel wall: $q_\zeta = 5 \cdot 10^{-5}\ \frac{C}{m^2}$

Hence, we have for the dimensionless computations the characteristical quantities

$$Re = \frac{\rho v_0 d_0}{\mu} \approx 0.1,\ \ Sc = \frac{\mu}{\rho D} \approx 2340.$$

Furthermore, the electroosmotic effects within the microchannel are simulated by a slip condition from bulk flow to EDL involving the dimensionless parameter

$$\Pi_2 = \frac{\ell_D \varphi_0 q_\zeta}{v_0 d_0 \mu} \approx 22$$

as derived in [11].

Backward Facing Step Flow

In addition to the simulation and optimisation of an electically exited fluid flow, we also deal with the optimisation of a fluid flow described solely by the Navier-Stokes equations. The used backward facing step example is shown in Figure 2. For the sake of these academic tests, the realistic width of the channel d_0 is of minor interest. Assume the fluid is water at $\sim 20^\circ\ C$, i.e. ρ and μ as above, and think of the Reynolds number of values greater than 100 by higher mean inflow velocity v_0 or larger diameter of the channel d_0.

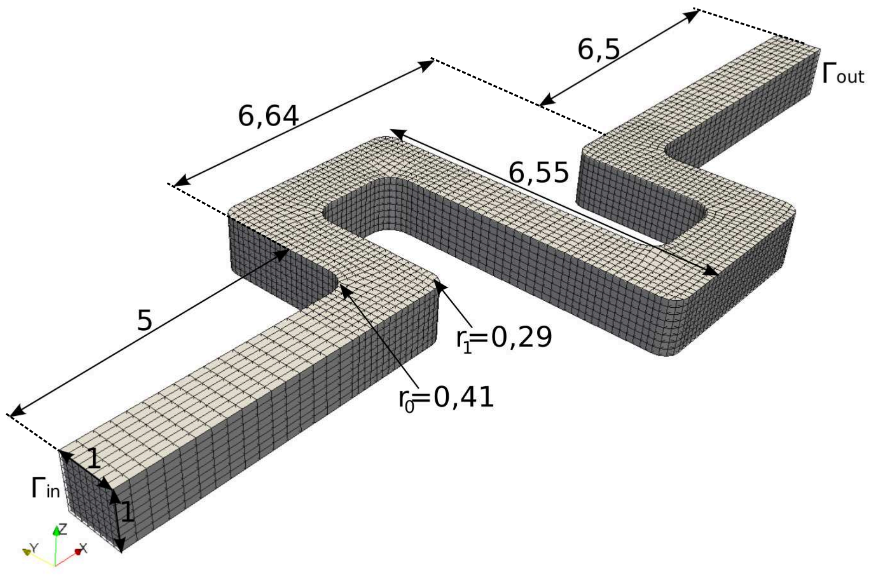


Figure 1: Scaled geometry of the meander part within the microfluidic chip with inflow boundary Γ_{in}, outflow boundary Γ_{out} and channel walls resp. transition layer to EDL Γ_0.

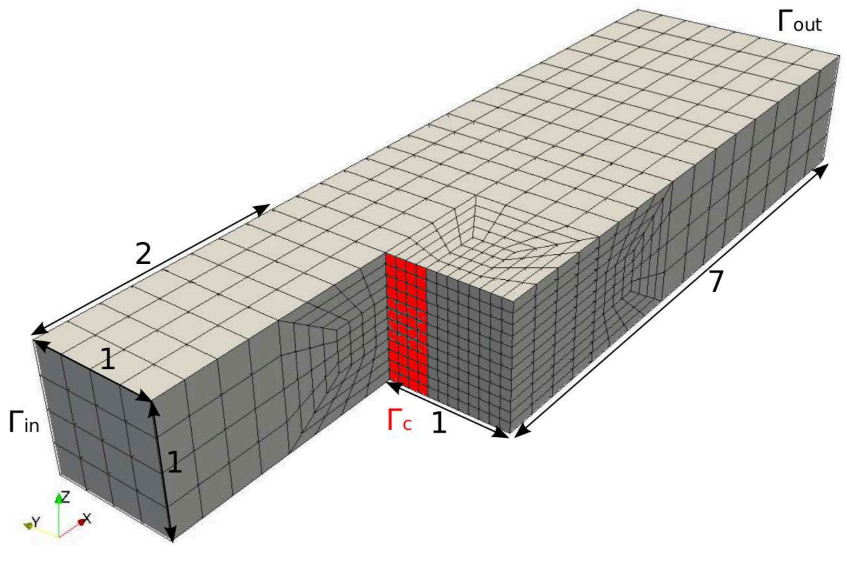


Figure 2: Scaled geometry of the backward facing step. Inflow and outflow boundary are given by Γ_{in} resp. Γ_{out}. The channel walls are Γ_0 except the control part Γ_c which is indicated by red colour.

Bibliography

[1] F. Abergel and R. Temam. On Some Control Problems in Fluid Mechanics. *Theoretical and Computational Fluid Dynamics*, 1:303–325, 1990.

[2] Robert A. Adams and John J. F. Fournier. *Sobolev spaces.* Pure and applied mathematics; 140. Academic Press, 2. repr. edition, 2006.

[3] V.I. Agoshkov. Poincaré-Steklov's operators and domain decompostion methods in finite dimensional spaces. In Roland Glowinski, Gene H. Golub, Gerard A. Meurant, and Jacques Periaux, editors, *First International Symposium on Domain Decomposition Methods for Partial Differential Equations*, pages 73–112. SIAM, 1988.

[4] P. R. Amestoy, I. S. Duff, J. Koster, and J.-Y. L'Excellent. A Fully Asynchronous Multifrontal Solver Using Distributed Dynamic Scheduling. *SIAM Journal on Matrix Analysis and Applications*, 23(1):15–41, 2001.

[5] Heng-Bin An, Ze-Yao Mo, and Xing-Ping Liu. A choice of forcing terms in inexact Newton method. *Journal of Computational and Applied Mathematics*, 200(1):47–60, 2007.

[6] Rutherford Aris. *Vectors, tensors and the basic equations of fluid mechanics.* Dover books on mathematics. Dover, New York, NY, 1. publ. edition, 1989.

[7] Satish Balay, Kris Buschelman, Victor Eijkhout, William D. Gropp, Dinesh Kaushik, Matthew G. Knepley, Lois Curfman McInnes, Barry F. Smith, and Hong Zhang. PETSc users manual. Technical Report ANL-95/11 - Revision 3.0.0, Argonne National Laboratory, 2008.

[8] Satish Balay, Kris Buschelman, William D. Gropp, Dinesh Kaushik, Matthew G. Knepley, Lois Curfman McInnes, Barry F. Smith, and Hong Zhang. PETSc Web page, 2009. www.mcs.anl.gov/petsc.

[9] Satish Balay, William D. Gropp, Lois Curfman McInnes, and Barry F. Smith. Efficient Management of Parallelism in Object Oriented Numerical Software Libraries. In E. Arge, A. M. Bruaset, and H. P. Langtangen, editors, *Modern Software Tools in Scientific Computing*, pages 163–202. Birkhäuser Press, 1997.

[10] Dominik P. J. Barz, Hamid Farangis Zadeh, and Peter Ehrhard. Laminar Flow and Mass Transport in a Twice-Folded Microchannel. *American Institute of Chemical Engineers Journal*, 54(2):381–393, 2008.

[11] Dominik P. J. Barz, Hamid Farangis Zadeh, and Peter Ehrhard. Measurements and simulations of time-dependent flow fields within an electrokinetic micromixer. *under consideration for publication in Journal of Fluid Mechanics*, 2010.

[12] Dominik Peter Johannes Barz. *Ein Beitrag zu Modellierung und Simulation von elektrokinetischen Transportprozessen in mikrofluidischen Einheiten.* PhD thesis, Universität Karlsruhe (TH), 2005.

[13] George K. Batchelor. *An introduction to fluid dynamics.* Cambridge Univ. Pr., 1. repr. paperback edition, 1974.

[14] Andrew C. Bauer and Abani K. Patra. Performance of parallel preconditioners for adaptive hp FEM discretization of incompressible flows. *Communications in Numerical Methods in Engineering*, 18:305–313, 2002.

[15] Michelle Benzi, John C. Haws, and Miroslav Tůma. Preconditioning highly indefinite and nonsymmetric matrices. *SIAM Journal on Scientific Computing*, 22(4):1333–1353, 2000.

[16] M. Bercovier, M.J. Gander, R. Kornhuber, and O.B. Widlund, editors. *Domain decomposition methods in science and engineering XVIII.* Springer, 2009.

[17] Robert Byron Bird, Warren E. Stewart, and Edwin N. Lightfoot. *Transport phenomena.* Wiley, 2. edition, 2002.

[18] Stephan Blazy, Serguei Nazarov, and Maria Specovius-Neugebauer. Artifical boundary conditions of pressure type for viscous flows in a system of pipes. *Journal of Mathematical Fluid Mechanics*, 9(1):1–33, March 2007.

[19] Hendryk Bockelmann. Trust-Region und Checkpointing-Ansätze für die Optimale Kontrolle von Parabolischen Problemen. Master's thesis, Universität Karlsruhe (TH), 2006.

[20] Hendryk Bockelmann and Vincent Heuveline. HiFlow Web page, 2010. www.hiflow.de.

[21] Hendryk Bockelmann and Vincent Heuveline. Parallel I/O and Checkpointing Strategies for PDE-constrained Optimization. In *Proceedings of PARA 2008, 9th International Workshop on State-of-the-Art in Scientific and Parallel Computing*, 2010.

[22] Matthias Bollhöfer. A robust and efficient ILU that incorporates the growth of the inverse triangular factors. *SIAM Journal on Scientific Computing*, 25(1):86–103, 2003.

[23] Matthias Bollhöfer and Yousef Saad. Multilevel preconditioners constructed from inverse-based ILUs. *SIAM Journal on Scientific Computing*, 25(2):715–728, 2003.

[24] S. C. Brenner. The condition number of the Schur complement in domain decomposition. *Numerische Mathematik*, 83(2):187–203, 1999.

[25] M.O. Bristeau, R. Glowinski, and J. Périaux. Numerical methods for the Navier-Stokes equations. Applications to the simulation of compressible and incompressible viscous flow. *Computer Physics Reports*, 6:73–187, 1987.

[26] John C. Butcher. *Numerical methods for ordinary differential equations.* Wiley [u.a.], New York, 2003.

[27] Eduardo Casas. *Optimal Control of Viscous Flow*, chapter 4 - An Optimal Control Problem Governed by the Evolution Navier–Stokes Equations, pages 79–96. SIAM, 1998.

[28] Eduardo Casas and Fredi Tröltzsch. Second-order necessary and sufficient optimality conditions for optimization problems and applications to control theory. *SIAM Journal on Optimization*, 13(2):406–431, 2002.

[29] Tony F. Chan. Domain Decomposition Algorithms and Computational Fluid Dynamics. *International Journal of High Performance Computing Applications*, 2(4):72–83, 1988.

[30] Tony F. Chan and Tarek P. Mathew. Domain Decomposition Algorithms. In *Acta Numerica 1994*, pages 61–143. Cambridge University Press, 1994.

[31] I. Charpentier. Checkpointing schemes for adjoint codes: Application to the meteorological model Meso-NH. *SIAM Journal on Scientific Computation*, 22(6):2135–2151, 2001.

[32] C. M. Chen and V. Thomee. The Lumped Mass Finite Element Method for a Parabolic Problem. *Journal of the Australian Mathematical Society*, 26:329–354, 1985.

[33] Alexandre Joel Chorin. Numerical solution of the Navier-Stokes equations. *Mathematics of Computation*, 22:745–762, 1968.

[34] Philippe G. Ciarlet. *The finite element method for elliptic problems.* Classics in applied mathematics; 40. Society for Industrial and Applied Mathematics, Philadelphia, PA, unabridged republ. of the work first publ. by north-holland, 1978 edition, 2002.

[35] Peter Constantin and Ciprian Foias. *Navier-Stokes equations.* University of Chicago Press, 1989.

[36] M. Crouzeix and P. A. Raviart. Conforming and nonconforming finite element methods for solving the stationary Stokes equations. *RAIRO - Modélisation mathématique et analyse numérique*, 7:33–76, 1973.

[37] Cornelis Cuvelier, August Segal, and Anton A. van Steenhoven. *Finite element methods and Navier-Stokes equations.* Mathematics and its applications; 22. D. Reidel Publishing Company, Dordrecht [u.a.], 1986.

[38] Timothy A. Davis. Algorithm 832: UMFPACK V4.3 – an unsymmetric-pattern multifrontal method. *ACM Transactions on Mathematical Software*, 30:196–199, 2004.

[39] Ron S. Dembo, Stanley C. Eisenstat, and Trond Steihaug. Inexact Newton methods. *SIAM Journal on Numerical Analysis*, 19(2):400–408, 1982.

[40] Ron S. Dembo and Trond Steihaug. Truncated Newton algorithms for large scale unconstrained optimization. *Mathematical Programming*, 26(72):190–212, 1983.

[41] John E. Dennis and Robert B. Schnabel. *Numerical methods for unconstrained optimization and nonlinear equations.* Classics in applied mathematics; 16. Society for Industrial and Applied Mathematics, Philadelphia, 1996. Unabridged, corrected republ. of the work 1. printed by Prentice-Hall, Englewood Cliffs, NJ, 1983.

[42] Peter Deuflhard. *Newton methods for nonlinear problems: affine invariance and adaptive algorithms.* Springer series in computational mathematics; 35. Springer, Berlin [u.a.], corr. 2. edition, 2006.

[43] Maksymilian Dryja and Olof B. Widlund. An Additive Variant of the Schwarz Alternating Method for the Case of Many Subregions. Technical Report 339, also Ultracomputer Note 131, Department of Computer Science, Courant Institute, 1987.

[44] Stanley C. Eisenstat and Homer F. Walker. Choosing the Forcing Terms in an Inexact Newton Method. *SIAM Journal on Scientific Computation*, 17:16–32, 1994.

[45] Stanley C. Eisenstat and Homer F. Walker. Globally Convergent Inexact Newton Methods. *SIAM Journal on Optimization*, 4(2):393–422, 1994.

[46] Howard C. Elman, Youcef Saad, and Paul E. Saylor. A Hybrid Chebyshev Krylov Subspace Algorithm for Solving Nonsymmetric Systems of Linear Equations. *SIAM Journal on Scientific and Statistical Computing*, 7(3):840–855, 1986.

[47] Alexandre Ern and Jean-Luc Guermond. *Theory and practice of finite elements.* Applied mathematical sciences; 159. Springer, New York, 2004.

[48] Charbel Farhat and Francois-Xavier Roux. Implicit parallel processing in structural mechanics. *Computational Mechanics Advances*, 2(1):1–124, 1994.

[49] C. L. Fefferman. *The millenium price problems*, chapter Existence and smoothness of the Navier-Stokes equation, pages 57–67. Clay Math. Inst., 2006.

[50] Miloslav Feistauer. *Mathematical methods in fluid dynamics.* Longman Scientific and Technical [u.a.], 1. publ. edition, 1993.

[51] George J. Fix, Max D. Gunzburger, and Janet S. Peterson. On finite element approximations of problems having inhomogeneous essential boundary conditions. *Computers & Mathematics with Applications*, 9:687–700, 1983.

[52] Luca Formaggia, Marzio Sala, and Fausto Saleri. *Domain Decomposition Techniques*, volume 51 of *Lecture Notes in Computational Science and Engineering*, chapter 4, pages 135–163. Springer, 2006.

[53] Michel Fortin. An analysis of the convergence of mixed finite element methods. *RAIRO - Analyse numérique*, 11(4):341–354, 1977.

[54] MPI Forum. MPI: A Message-Passing Interface Standard. Technical report, Knoxville, TN, USA, 2008.

[55] Robert W. Fox, Philip J. Pritchard, and Alan T. McDonald. *Introduction to fluid mechanics.* Wiley, 2010.

[56] Valérie Frayssé, Luc Giraud, and Serge Gratton. A Set of Flexible GMRES Routines for Real and Complex Arithmetics on High-Performance Computers. *ACM Transactions on Mathematical Software*, 35(2):Article 13, July 2008 2008.

[57] Akihiro Fujii, Reiji Suda, and Akira Nishida. Parallel Matrix Distribution Library for Sparse Matrix Solvers. *International Conference on High Performance Computing and Grid in Asia Pacific Region*, 0:213–219, 2005.

[58] A. Fursikov, M. Gunzburger, L. S. Hou, and S. Manservisi. *Optimal Control Problems for the Navier-Stokes Equations*, pages 143–155. Lectures on Applied Mathematics. Springer, 2000.

[59] A. V. Fursikov. *Optimal Control of Distributed Systems: Theory and Applications.* American Mathematical Society, Boston, MA, USA, 2000.

[60] A. V. Fursikov, M. D. Gunzburger, and L. S. Hou. Boundary value problems and optimal boundary control for the Navier-Stokes system: the two-dimensional case. *SIAM Journal on Control and Optimization*, 36:852–894, 1998.

[61] Vivette Girault and Pierre-Arnaud Raviart. *Finite element approximation of the Navier-Stokes equations.* Lecture notes in mathematics; 749. Springer, Berlin [u.a.], 1979.

[62] Vivette Girault and Pierre-Arnaud Raviart. *Finite element methods for Navier-Stokes equations: theory and algorithms.* Springer series in computational mathematics, 5. Springer, Berlin, 1986.

[63] R. Glowinski. *Finite element methods for incompressible viscous flow*, pages 3–1176. Numerical methods for fluids (Part 3). Elsevier, 2003.

[64] R. Glowinski and J.F. Periaux. Numerical methods for nonlinear problem in fluid dynamics. In *INRIA Conference on Supercomputing: state-of-the-art*, pages 381–479, 1987.

[65] Anne Greenbaum. *Iterative methods for solving linear systems.* Frontiers in applied mathematics ; 17. SIAM, Philadelphia, 1997.

[66] Philip M. Gresho and Robert L. Sani. On pressure boundary conditions for the incompressible Navier-Stokes equations. *International Journal for Numerical Methods in Fluids*, 7:1111–1145, 1987.

[67] Philip M. Gresho and Robert L. Sani. *Incompressible Flow and the Finite Element Method*, volume 2: Isothermal Laminar Flow. John Wiley & Sons Ltd, 2000.

[68] Philip M. Gresho and Robert L. Sani. *Incompressible Flow and the Finite Element Method*, volume 1: Advection-Diffusion. John Wiley & Sons Ltd, 2000.

[69] Andreas Griewank and Andrea Walther. Algorithm 799: revolve: an implementation of checkpointing for the reverse or adjoint mode of computational differentiation. *ACM Transactions on Mathematical Software*, 26(1):19–45, 2000.

[70] William D. Gropp and David E. Keyes. Domain Decomposition Methods in Computational Fluid Dynamics. *International Journal for Numerical Methods in Fluids*, 14:147–165, 1992.

[71] Max Gunzburger. Adjoint Equation-Based Methods for Control Problems in Incompressible, Viscous Flows. *Flow, Turbulence and Combustion*, 65:249–272, 2000.

[72] Max D. Gunzburger. *Finite element methods for viscous incompressible flows: a guide to theory, practice, and algorithms.* Computer science and scientific computing. Academic Press, Boston [u.a.], 1989.

[73] Max D. Gunzburger. *Perspectives in flow control and optimization.* Society for Industrial and Applied Mathematics, 2003.

[74] Max D. Gunzburger and L. S. Hou. Treating inhomogeneous essential boundary conditions in finite element methods and the calculation of boundary stresses. *SIAM Journal on Numerical Analysis*, 29:390–424, 1992.

[75] Max D. Gunzburger, L. S. Hou, and T. Svobodny. Analysis and finite element approximation of optimal control problems for the stationary navier-stokes equations with dirichlet controls. *RAIRO - Modélisation mathématique et analyse numérique*, 6:711–748, 1991.

[76] Max D. Gunzburger, L. S. Hou, and Thomas P. Svobodny. Boundary velocity control of incompressible flow with an application to viscous drag reduction. *SIAM Journal on Control and Optimization*, 30(1):167–181, 1992.

[77] Max D. Gunzburger and S. Manservisi. The velocity tracking problem for Navier-Stokes flow with boundary control. *SIAM Journal on Control and Optimization*, 39(2):594–635, 2000.

[78] Max D. Gunzburger and Janet S. Peterson. On conforming finite element methods for the inhomogeneous stationary Navier-Stokes equations. *Numerische Mathematik*, 42:173–194, 1983.

[79] M.D. Gunzburger, L. Hou, and T.P. Svobodny. Analysis and Finite Element approximation of optimal control problems for the stationary Navier-Stokes equations with distributed and Neumann controls. *Mathematics of Computation*, 57(195):123–151, 1991.

[80] Wolfgang Hackbusch. *Iterative solution of large sparse systems of equations.* Applied Mathematical Sciences. Springer, 95 edition, 1994.

[81] Ernst Hairer and Gerhard Wanner. *Solving ordinary differential equations*, volume 1: Nonstiff problems of *Springer series in computational mathematics; 8*. Springer, Berlin, 2. rev. edition, 2000.

[82] Vincent Heuveline and Andrea Walther. Online Checkpointing for Parallel Adjoint Computation in PDEs: Application to Goal-Oriented Adaptivity and Flow Control. In W. et al. Nagel, editor, *Proceedings of Euro-Par 2006*, volume 4128 of *LNCS*, pages 689–699. Springer, 2006.

[83] John G. Heywood and Rolf Rannacher. Finite element approximation of the nonstationary navier-stokes problem. part i: Regularity of solutions and second-order error estimates for spatial discretization. *SIAM Journal on Numerical Analysis*, 19:275–311, 1982.

[84] John G. Heywood, Rolf Rannacher, and Stefan Turek. Artifical boundaries and flux and pressure conditions for the incompressible Navier-Stokes equations. *International Journal for Numerical Methods in Fluids*, 22:325–352, 1996.

[85] Michael Hintermüller, Karl Kunisch, Yulian Spasov, and Stefan Volkwein. Dynamical systems-based optimal control of incompressible fluids. *International Journal for Numerical Methods in Fluids*, 46(4):345–359, 2004.

[86] M. Hinze and K. Kunisch. Second order methods for boundary control of the instationary Navier-Stokes system. *ZAMM Journal of Applied Mathematics and Mechanics*, 84:171–187, 2004.

[87] Michael Hinze. *Optimal and instantaneous control of the instationary Navier-Stokes equations.* Habilitation, Technische Universität Dresden, Juli 2002.

[88] Michael Hinze, Rene Pinnau, Micheal Ulbrich, and Stefan Ulbrich. Modelling and Optimization with Partial Differential Equations. Lecture Notes, Autumn School Hamburg, September 26–30, 2005.

[89] Michael Hinze and Andrea Walther. An optimal memory-reduced procedure for calculating adjoints of the instationary Navier-Stokes equations. *Optimal Control Applications and Methods*, 27(1):19–40, 2006.

[90] E. H. Hirschel, editor. *Benchmark computations of laminar flow around cylinder*, volume 52 of *Notes on Numerical Fluid Mechanics*. Vieweg, 1996. co. F. Durst, E. Krause, R. Rannacher.

[91] P. Jamet and P. A. Raviart. Numerical solution of the stationary Navier-Stokes equations by Finite Element Methods. *Computing Methods in Applied Sciences and Engineering*, 10:193–223, 1974.

[92] Volker John, Gunar Matthies, and Joachim Rang. A comparison of time-discretization/linearization approaches for the incompressible Navier-Stokes equations. *Computer Methods in Applied Mechanics and Engineering*, 195:5995–6010, 2006.

[93] Masays Kakuta, Fiona G. Bessoth, and Andreas Manz. Microfabricated Devices for Fluid Mixing and Their Application for Chemical Synthesis. *The Chemical Record*, 1:395–405, 2001.

[94] George Karniadakis, Ali Beskok, and Narayan Aluru. *Microflows and Nanoflows.* Springer, 2005.

[95] C. T. Kelley. *Iterative methods for linear and nonlinear equations.* Frontiers in applied mathematics; 16. SIAM, Philadelphia, 1995.

[96] C. T. Kelley. *Iterative methods for optimization.* Frontiers in applied mathematics; 18. Society for Industrial and Applied Mathematics, Philadelphia, Pa., 1999.

[97] P. Kloucek and F. S. Rys. Stability of the fractional step θ-scheme for the nonstationary Navier-Stokes equations. *SIAM Journal on Numerical Analysis*, 31:1312–1335, 1994.

[98] Mathias Krause. *Fluid Flow Simulation and Optimisation with Lattice Boltzmann Methods on High-Performance Computers.* PhD thesis, Universität Karlsruhe (TH), 2010.

[99] Jaroslav Kruis. *Domain decomposition methods for distributed computing.* Saxe-Coburg Publications, 2006.

[100] K. Kunisch and B. Vexler. Optimal vortex reduction for instationary flows based on translation invariant cost functionals. *SIAM Journal on Control and Optimization*, 46(4):1368–1397, 2007.

[101] Olga A. Ladyzenskaja. *The Mathematical theory of viscous incompressible flow.* Gordon & Breach, 2. rev. and enlarged. edition, 1969.

[102] Roland Laifer. Experiences with HP SFS/Lustre at SSCK, SGPFS 5 Stuttgart. Technical report, Universität Karlsruhe (TH), 2006. http://www.rz.uni-karlsruhe.de/dienste/lustretalks.

[103] Na Li, Yousef Saad, and Edmond Chow. Crout Versions of ILU for General Sparse Matrices. *SIAM Journal on Scientific Computation*, 25(2):716–728, 2003.

[104] Chia-Ch'iao Lin and Lee A. Segel. *Mathematics applied to deterministic problems in the natural sciences.* MacMillan, Collier Macmillan, 1974.

[105] Jacques Louis Lions. *Optimal control of systems governed by partial differential equations.* Springer, 1971.

[106] Pierre Louis Lions. On the Schwarz alternating method I. In Roland Glowinski, Gene H. Golub, Gerard A. Meurant, and Jacques Periaux, editors, *First International Symposium on Domain Decomposition Methods for Partial Differential Equations*, pages 1–42. SIAM, 1988.

[107] Pierre Louis Lions. On Schwarz alternating method II. In Tony F. Chan, Roland Glowinski, Jacques Periaux, and Olof B. Widlund, editors, *Domain Decomposition Methods*, pages 47–70. SIAM, 1989.

[108] Pierre Louis Lions. On the Schwarz alternating method III: A variant for nonoverlapping subdomains. In Tony F. Chan, Roland Glowinski, Jacques Periaux, and Olof B. Widlund, editors, *Third International Symposium on Domain Decomposition Methods for Partial Differential Equations*, pages 202–223. SIAM, 1990.

[109] S. H. Lui. On Schwarz Alternating Methods for the Incompressible Navier-Stokes Equations. *SIAM Journal on Scientific Computation*, 22(6):1974–1986, 2000.

[110] Tarek Poonithara Abraham Mathew. *Domain decomposition methods for the numerical solution of partial differential equations.* Number 61 in Lecture Notes in Computational Science and Engineering. Springer, 2008.

[111] H. Maurer and J. Zowe. First and second-order necessary and sufficient optimality conditions for infinite-dimensional programming problems. *Mathematical Programming*, 16(1):98–110, Dezember 1979.

[112] Jan Mayer. ILU++ software package for solving large linear systems of equations by iterative methods using state of the art preconditioners . www.iluplusplus.de.

[113] Jan Mayer. Alternative Weighted Dropping Strategies for ILUTP. *SIAM Journal on Scientific Computation*, 27(4):1424–1437, 2006.

[114] Jan Mayer. Symmetric Permutations for I-matrices to Delay and Avoid Small Pivots. *SIAM Journal on Scientific Computation*, 30(21):982–996, 2008.

[115] Dominik Meidner and Boris Vexler. Adaptive space-time finite element methods for parabolic optimization problems. *SIAM Journal on Control and Optimization*, 46(1):116–142, 2007.

[116] Ingo Meisel and Peter Ehrhard. Electrically-excited (electroosmotic) flows in microchannels for mixing applications. *European Journal of Mechanics B/Fluids*, 25:491–504, 2006.

[117] S. Müller-Urbaniak. *Eine Analyse des Zwischenschritt-θ-Verfahrens zur Lösung der instationären Navier-Stokes-Gleichungen.* PhD thesis, University of Heidelberg, 1994.

[118] Yi-Yong Nie and Vidar Thomee. A Lumped Mass Finite-Element Method with Quadrature for a Non-linear Parabolic Problem. *IMA Journal of Numerical Analysis*, 5:371–396, 1985.

[119] Jorge Nocedal and Stephen J. Wright. *Numerical optimization.* Springer, 2. edition, 2006.

[120] John Tinsley Oden and Junuthula N. Reddy. *An introduction to the mathematical theory of finite elements.* Pure and applied mathematicsA Wiley-Interscience publication. Wiley, New York [u.a.], 1976.

[121] James M. Ortega and Werner C. Rheinboldt. *Iterative solution of nonlinear equations in several variables.* Classics in applied mathematics; 30. Society for Industrial and Applied Mathematics, Philadelphia, Pa., 2000.

[122] F. Pellegrini. SCOTCH 5.1 user's guide. Technical report, LaBRI, 2008.

[123] F. Pellegrini. SCOTCH Web page, 2010. www.labri.fr/perso/pelegrin/scotch.

[124] Michael Pernice and Homer F. Walker. NITSOL: A Newton Iterative Solver for Nonlinear Systems. *SIAM Journal on Scientific Computation*, 19(1):302–318, 1998.

[125] Roger Peyret, editor. *Handbook of computational fluid mechanics.* Academic Press, 1996.

[126] Alfio Quarteroni and Alberto Valli. *Numerical approximation of partial differential equations.* Springer series in computational mathematics (23). Springer, Berlin, 1994.

[127] Alfio Quarteroni and Alberto Valli. *Domain decomposition methods for partial differential equations.* Clarendon Press, repr. edition, 2005.

[128] Thomas Rauber and Gudula Rünger. *Parallele Programmierung.* Springer, 2007.

[129] Thomas Rauber and Gudula Rünger. *Parallel Programming: for Multicore and Cluster Systems.* Springer, 2010.

[130] Junuthula N. Reddy. *Applied functional analysis and variational methods in engineering.* McGraw-Hill, 1986.

[131] Hans-Görg Roos, Martin Stynes, and Lutz Tobiska. *Robust Numerical Methods for Singularly Perturbed Differential Equations: Convection-Diffusion-Reaction and Flow Problems.* Springer-Verlag, 2008.

[132] Youcef Saad. A flexible inner-outer preconditioned GMRES algorithm. *SIAM Journal on Scientific Computation*, 14(2):461–469, March 1993 1993.

[133] Youcef Saad. *Iterative methods for sparse linear systems.* SIAM, Society for Industrial and Applied Mathematics, 2. edition, 2003.

[134] Youcef Saad and Martin H Schultz. GMRES: a generalized minimal residual algorithm for solving nonsymmetric linear systems. *SIAM Journal on Scientific and Statistical Computation*, 7(3):856–869, 1986.

[135] Yousef Saad. Multilevel ILU with reorderings for diagonal dominance. *SIAM Journal on Scientific Computation*, 27:1032–1057, 2006.

[136] Yousef Saad and Masha Sosonkina. Distributed Schur Complement techniques for general sparse linear systems. *SIAM Journal on Scientific Computation*, 21:1337–1356, 1997.

[137] H. A. Schwarz. *Gesammelte Mathematische Abhandlungen*, volume 2, pages 133–143. Springer, Berlin, 1890. First published in Vierteljahrsschrift der Naturforschenden Gesellschaft in Zürich, volume 15, 1870, pp. 272–286.

[138] Ridgway Scott. Interpolated boundary conditions in the Finite Element Method. *SIAM Journal on Numerical Analysis*, 12:404–427, 1975.

[139] Lee A. Segel and George H. Handelman. *Mathematics applied to continuum mechanics.* Society for Industrial and Applied Mathematics, reprint of the 1977 original edition, 2007.

[140] John N. Shadid, Ray S. Tuminaro, and Homer F. Walker. An inexact Newton method for fully coupled solution of the Navier-Stokes equations with heat and mass transport. *Journal of Computational Physics*, 137(1):155–185, 1997.

[141] Helmut E. Siekmann and Paul Uwe Thamsen. *Strömungslehre.* Springer-Lehrbuch. Springer, Berlin, 2008.

[142] Barry F. Smith. A parallel implementation of an iterative substructuring algorithm for problems in three dimensions. *SIAM Journal on Scientific Computation*, 14:406–423, 1993.

[143] Barry F. Smith, Petter E. Bjørstad, and William D. Gropp. *Domain decomposition, Parallel Multilevel Methods for Elliptic Partial Differential Equations.* Cambridge Univ. Press, 1. paperback edition, 2004.

[144] Hermann Sohr. *The Navier-Stokes equations: an elementary functional analytic approach.* Birkhäuser, 2001.

[145] S. S. Sritharan, editor. *Optimal Control of Viscous Flow.* SIAM, 1998.

[146] Rolf Stenberg. Analysis of Mixed Finite Element Methods for the Stokes problem: A Unified Approach. *Mathematics of Computation*, 42:9–23, 1984.

[147] Rolf Stenberg. On some three-dimensional Finite Elements for incompressible media. *Computer Methods in Applied Mechanics and Engineering*, 63:261–269, 1987.

[148] Gilbert Strang and George J. Fix. *An analysis of the finite element method.* Prentice-Hall series in automatic computation. Prentice-Hall, Englewood Cliffs, NJ, 1973.

[149] P. Stumm and A. Walther. Multistage approaches for optimal offline checkpointing. *Preprint SPP1253-15-03*, 2008.

[150] Sun Microsystems. Lustre documentation. Technical report. http://www.lustre.org, http://www.sun.com/software/products/lustre.

[151] C. Taylor and P. Hood. A numerical solution of the Navier-Stokes equations using the finite element technique. *Computers and Fluids*, 1:73–100, 1973.

[152] Roger Temam. *Navier-Stokes equations: theory and numerical analysis.* Studies in mathematics and its applications ; 2. North-Holland, Amsterdam [u.a.], rev. edition, 1979.

[153] Roger Temam. *Navier-Stokes equations and nonlinear functional analysis.* Regional conference series in applied mathematics ; 41. Society for Industrial and Applied Mathematics, Philadelphia, Pa., 1983.

[154] Roger Temam and Alain Miranville. *Mathematical modeling in continuum mechanics.* Cambridge University Press, 2001.

[155] Andrea Toselli and Olof Widlund. *Domain decomposition methods - algorithms and theory.* Springer, 2005.

[156] Fredi Tröltzsch. Optimality conditions for parabolic control problems and applications. *Teubner-Texte zur Mathematik*, 62:156–164, 1984.

[157] Fredi Tröltzsch. *Optimale Steuerung partieller Differentialgleichungen: Theorie, Verfahren und Anwendungen.* Vieweg+Teubner, 2. edition, 2009.

[158] R. Verfürth. Error estimates for a mixed finite element approximation of the Stokes equations. *RAIRO Anal. Numér.*, 18:175–182, 1984.

[159] Daniel Wachsmuth. *Optimal control of the unsteady Navier-Stokes equations.* PhD thesis, Technische Universität Berlin, 2006.

[160] Andrea Walther and Andreas Griewank. *Advantages of binomial checkpointing for memory-reduced adjoint calculations.*, pages 834–843. Numerical mathematics and advanced applications. Proceedings of ENUMATH 2003, the 5th European conference on numerical mathematics and advanced applications. Springer, 2004.

[161] Barbara Wohlmuth. *Discretization methods and iterative solvers based on domain decomposition.* Springer, 2001.

[162] W. L. Wood. *Practical time-stepping schemes.* Oxford applied mathematics and computing science series. Clarendon Pr., Oxford [u.a.], 1990.

[163] Kôsaku Yosida. *Functional Analysis.* Springer, 1995.