Inaugural-Dissertation

zur Erlangung der Doktorwürde
der
Naturwissenschaftlich-Mathematischen Gesamtfakultät
der
Ruprecht-Karls-Universität
Heidelberg

vorgelegt von
Diplom-Mathematiker Thomas Richter
aus Marl

Tag der mündlichen Prüfung: 13.07.2005

Parallel Multigrid Method for Adaptive Finite Elements with Application to 3D Flow Problems

05.05.2005

1. Gutachter: Prof. Dr. Rolf Rannacher

2. Gutachter: Prof. Dr. Peter Bastian

Abstract

Aim of this work is the examination of numerical methods for the solution of large systems of PDE's. The equations under consideration arise from chemically reacting flows.

A focal point is the analysis of a finite element discretization with stabilized finite elements of degree two. Aspects of error estimation, solution techniques and mesh adaptivity are discussed with regard to the Navier-Stokes equations. Using a well established Navier-Stokes benchmark flow the discussed methods are verified.

To cope with the huge systems arising from reactive flow problems a parallel multigrid method on locally refined meshes is presented.

Finally, we will perform a simulation of a methane burner in a complex three dimensional geometry. We will use a detailed reaction mechanism with 39 chemical species.

Zusammenfassung

Gegenstand dieser Arbeit ist die Analyse von numerischen Verfahren zur Lösung von großen Systemen partieller Differenzialgleichungen. Die betrachteten Gleichungen treten z.B. bei der Simulation von reaktiven Strömungen auf.

Ein Schwerpunkt ist die Untersuchung einer stabilisierten Finite Elemente Diskretisierung mit quadratischen Ansatzräumen. Anhand der Navier-Stokes Gleichungen werden Aspekte wie das Lösen der Gleichungssysteme, Fehlerschätzung und Gitteradaption behandelt. Die vorgestellten Verfahren werden an einem etablierten Navier-Stokes Benchmark verifiziert.

Bei der Simulation von reaktiven Strömungen vergrößert sich die Anzahl der Lösungskomponenten um die Anzahl an chemischen Substanzen. Die implizit gekoppelte Lösung der entsprechenden Gleichungen stellt hinsichtlich Rechen- und Zeitaufwand eine enorme Anforderung an die Computer. Um eine Lösung überhaupt zu ermöglichen wird eine paralleles Mehrgitterverfahren auf adaptiven Gittern vorgestellt.

Schließlich werden Simulationsrechnungen einer Methanflamme in einem Brenner mit komplexer, dreidimensionaler Geometrie präsentiert. Die chemischen Reaktionen werden mit einem detailierten Reaktionsmechanismus unter Berücksichtigung von 39 chemischen Substanzen modelliert.

Contents

1. Introduction

1.1. Motivation

This work is devoted to the reliable solution of complex problems described by partial differential equations. The problems under consideration originate from various processes of nature specifically from flows and chemical reactions. Our focus here is the combination of both. Thinking of chemical reactors or flames in burners, the reacting species – whether fluid, gaseous, or a mixture of both – flow in some technical devices. In addition to basic 'cold flow' the reaction effects the flow by density variations of different species as well as temperature gradients aroused by the reactions. In order to accurately solve such combined problems, we have to treat the arising equations coupled simultaneously. Even with the use of modern parallel computers, the sheer size of the resulting system overstresses available capacity in terms of memory usage and computational time if we consider three dimensional problems involving large reaction systems.

Hence, the usual approach for the numerical treatment of reactive flow problems is made up of a two dimensional reduction of the geometry and a decoupling of the equations. This decoupling is either done by splitting the equations into a flow and a reaction part and iterating between the two or by some splitting scheme applied within the solution process.

However, if we like to apply adaptivity with reliable error control or if the problem is subject to some optimization, we need a coupled handling of the equations. Braack [Bra98] proposed an adaptive finite element scheme for two dimensional chemically reacting flows. The problem is treated fully coupled as a set of nonlinear equations. Error control and mesh adaption is applied following the framework of the dual weighted residual method by Becker & Rannacher [BR96], [BR01].

This work is dedicated to the extension of the already extensively analyzed adaptive finite element method to the three dimensional case. Beside enhancing the finite element discretization, a crucial point is the parallelization of the solvers. Modern parallel computers work with message passing protocols, in which the data is distributed to separate machines by passing data packages through a network. Consequently this communication is decidedly slow in comparison to local memory access. Communication between different machines has to be limited to a minimum.

The parallelization of finite element methods is not new. Mainly two approaches are used: one could split the computational domain into several parts and distribute local problems to different processors, see e.g. Quarteroni & Valli [QV99] for these "domain decomposition" methods. The second approach is the parallelization of the linear solver while the finite element method itself is kept unchanged. Details on the parallelization of an adaptive

multigrid solver are given e.g. in Bastian [Bas96]. This work is embedded into the finite volume context, through the main ingredients directly carry over to finite elements. The focus of the parallelization process in this work is not to obtain a highly efficient parallel algorithm but to open up the capacity of modern parallel computers for the existing numerical methods. The problems under considerations are highly coupled, a natural predetermined breaking point is not given. Furthermore the usage of adaptive mesh refinement complicates the parallelization from a technical viewpoint.

The remaining part of this chapter introduces two basic examples which will be used throughout this work: first a Navier-Stokes benchmark problem which is already well examined and will be used for verifying the developed methods, and second a configuration describing a methane burner, sufficiently challenging to exhaust modern techniques.

The second chapter gives an introduction to the finite element methods and explains the basic notations used in this work.

The used finite element discretization is discussed in the third chapter. We treat three dimensional Navier-Stokes flows since they already contain essential properties of the later on considered chemically reacting flows. Since the Galerkin discretization of the Navier Stokes equations yields various instabilities (e.g. Girault & Raviart [GR86]), efficient stabilization techniques form a principal part of this chapter. Always keeping reactive flows in mind, we address adaptive mesh refinement and error control in detail. Using well adapted meshes, we can significantly reduce the problem dimension without giving up accuracy. The presented finite element discretization will finally be validated using a three dimensional flow benchmark put forward by Schäfer & Turek [ST96].

In the fourth chapter, the parallelization of the given finite element solver is described. We will start with a basic summary of parallelization techniques and methods necessary for analyzing parallel algorithms. Considering parallel algorithms one is interested in the efficiency of the algorithms depending on the problem size and on the number of processors used, i.e. the scalability of the algorithm. Using the framework of iso-efficiency analysis presented by Grama et al. [GGKK03] we will see that for studying the efficiency of parallel algorithms we have to connect the problem size to the number of CPU. Roughly speaking, we will call an algorithm scalable if its efficiency remains constant while connecting the problem size to the number of processors by an algebraic coherency.

Further we will present the numerical methods used for solving the problems in detail and describe their parallelization. The focal point is set on the multigrid solver on adaptively refined meshes. Our primal interest is the preserving of the very robust coupled solution techniques in the parallel setting.

Finally in the sixth chapter we apply the developed methods to chemically reacting flows. As a model problem – which should not state an oversimplification – we use a methane burner which is an image of a real life configuration used to heat water. The presented calculations are to be understood as a feasibility study for three dimensional simulations of reacting flows with robust and reliable methods. The burner will be described in the next sections of this introduction.

The computations presented in this thesis are done with the finite element toolbox *Gascoigne* [BB$^+$]. In the context of this work, *Gascoigne* was extended for the use of parallel computers. Further various modifications where applied to many parts of the toolbox.

1.2. 3D Navier Stokes Benchmark Configuration

Under the Priority Research Program 'Flow Simulation on High Performance Computers' a set of benchmark problems has been defined by Schäfer & Turek [ST96]. The task is to calculate certain coefficients of a three dimensional flow around an obstacle. Two different obstacles – a cylinder with circular and with square cross-section – are considered. In both cases the inflow velocity is specified and yields a Reynolds number $Re = 20$ leading to a steady flow. The requested quantities are the drag coefficient, the lift coefficient and the pressure difference in two points near the obstacle.

$$\begin{aligned} c_{\text{drag}} &= C \int_{\Gamma_{\text{int}}} \left(\nu \frac{\partial v_t}{\partial \vec{n}} n_y - p n_x \right) \text{dS}, \quad C = \frac{2}{DH\bar{v}^2} = \frac{500}{0.41}, \\ c_{\text{lift}} &= C \int_{\Gamma_{\text{int}}} \left(\nu \frac{\partial v_t}{\partial \vec{n}} n_x - p n_y \right) \text{dS}, \\ \Delta p &= p(x_2) - p(x_1), \quad x_1 = (0.45, 0.2, 0.205),\ x_2 = (0.55, 0.2, 0.205). \end{aligned}$$

In summary, the searched quantities are three real numbers $\Delta p, c_{\text{drag}}, c_{\text{lift}}$ for each of the two configurations. The configuration of the benchmark problem is drawn in Figure 1.1.

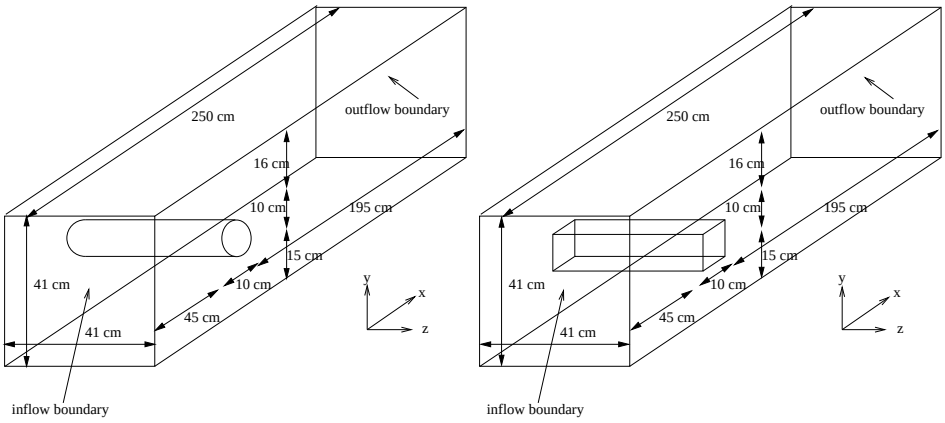

Figure 1.1.: Configurations of the benchmark problems. The obstacle is a cylinder with circular cross-section (Configuration 1, left) and square cross-section (Configurations 2, right).

Velocity v and pressure p of the flow are searched in the computational domain Ω. The

governing equations are the stationary Navier-Stokes equations:

$$\begin{aligned} -\nu\Delta v + v\cdot\nabla v + \nabla p &= f \\ \operatorname{div} v &= 0 \end{aligned} \quad \text{in } \Omega. \tag{1.1}$$

Considering the configuration displayed in Figure 1.1 the boundary of Ω is split into four parts:

$$\partial\Omega = \Gamma_{\text{in}} \cup \Gamma_{\text{wall}} \cup \Gamma_{\text{out}} \cup \Gamma_{\text{int}}.$$

A parabolic inflow is enforced as a Dirichlet boundary condition for the velocity on Γ_{in}, a no-slip condition – i.e. homogeneous Dirichlet condition for the velocity – on the interior obstacle and the wall $\Gamma_{\text{int}} \cup \Gamma_{\text{wall}}$ and a do-nothing outflow condition on Γ_{out}:

$$\begin{aligned} v &= g \quad \text{on } \Gamma_{\text{in}}, \\ v &= 0 \quad \text{on } \Gamma_{\text{wall}} \cup \Gamma_{\text{int}}, \\ \frac{\partial v}{\partial n} + pn &= 0 \quad \text{on } \Gamma_{\text{out}}. \end{aligned}$$

The outflow condition is called "'do-nothing"' condition, since arises from the variational formulation without introducing artificial terms. It allows various model flows without introducing boundary errors. Further, it fixes the absolute value of the pressure which is initially only given up to a constant (see equation 1.1). A good introduction to the derivation and the theory of Navier-Stokes equations is given by Galdi [Gal94a, Gal94b], details on the outflow condition are given in Heywood, Rannacher & Turek [HRT96].

1.3. 3D Methane Burner

The simulation of technical processes in realistic settings requires a huge computational effort. As an example we consider a usual methane burner as displayed in Figure 1.2. From below the fuel – methane – is introduced into the burner. In some mixing ducts the fuel is mixed with air to yield a nearly stoichiometric composition. This fuel-air mixture flows through lamellas to the surface of the burner. Settled atop these lamellas the fuel is burned. The lamellas all have a fixed distance of each other but are of different height. After a sequence of three different heights, the configuration is repeated (long - middle - short - middle - ...). To prevent the flame from entering into the burner cooling pipes are fitted into the lamellas. To allow a simulation of this specific flame we have to reduce the geometry to be presentable in a computer. First we assume the surface of the burner to be of infinite size with some periodicity and single out a small cuboid representing just the three different lamella sizes and a small portion of the cooling pipe. On the lateral boundaries of this imaginary cuboid we assume a symmetric continuation. Furthermore we consider a laminar inflow of premixed fuel and do not simulate the mixing process. On the right hand side of Figure 1.2 the computational domain is shown. Although these severe simplifications lead to a rather simple geometry, we need roughly 1 000 000 degrees of freedom to simulate a "cold flow" – i.e. just the simulation of the pressure and the velocity – within this domain. If we take into account the additional degrees of freedom representing the chemical species a corresponding simulation would overstress modern computers just in terms of memory usage.

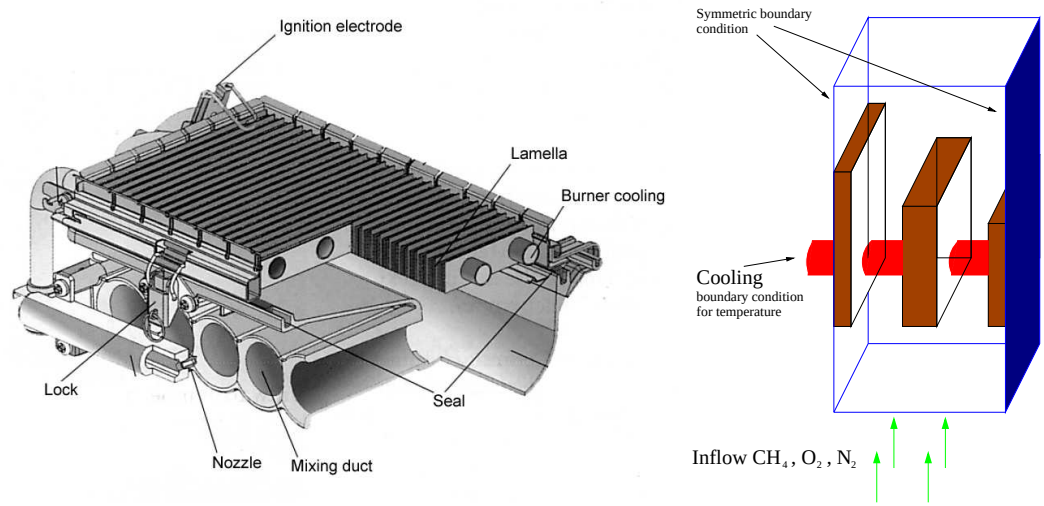

Figure 1.2.: Methane burner and reduced computational model of the geometry.

Relief is produced by the use of parallel computers. To cope with the size of the problem, we split it into parts and distribute them to several separated computers. The common way of parallelizing finite element simulations is done by domain decomposition methods, see Quarteroni & Valli [QV99]: the finite element triangulation is partitioned into parts of preferably the same size which are distributed to the different machines. In each subdomain the problem is solved, while additional conditions on the interface assure the global correctness of the solution. This decoupled way of solving the problem is not adequate for the simulation of chemically reacting flows, where the coupling and transport of information is essential to the character of the solution.

Hence we aim at transferring well established globally coupled algorithms such as a Newton and multigrid solver to parallel computers. These global methods are inherently ill-suited for parallelization since information in one point of the computational domain effects the whole solution. Nevertheless we try to hold up as much of the original methods as possible. In the next chapter the parallel solver is described in detail. We will see, that only the multigrid smoother has to be modified.

We denote the velocity by v, the pressure by p, the temperature by T, the n_s species mass fractions by w_k, $k = 1, \ldots, n_s$, and the density by ρ. The basic equations for reactive viscous flows express the conservation of total mass, momentum, energy, and species mass in the

form

$$\begin{aligned}
\partial_t \rho + \operatorname{div}(\rho v) &= 0\,, && (1.2)\\
\rho \partial_t v + \rho (v \cdot \nabla) v - \operatorname{div} \pi + \nabla p &= \rho g\,, && (1.3)\\
\rho c_p \partial_t T + (\rho c_p v + \alpha) \cdot \nabla T + \operatorname{div}(-\lambda \nabla T + \mathcal{Q}_{Duf}) & \\
+ \pi : \nabla v - v \cdot \nabla p - \partial_t p &= -\sum_{k=1}^{n_s} h_k m_k \omega_k\,, && (1.4)\\
\rho \partial_t w_k + \rho v \cdot \nabla w_k + \operatorname{div} \mathcal{F}_k &= m_k \dot{\omega}_k, && (1.5)
\end{aligned}$$

for $k = 1 \ldots n_s$, where c_p is the heat capacity of the mixture at constant pressure, λ the thermal conductivity, and for each species k, m_k is its molar weight, h_k its specific enthalpy, $\dot{\omega}_k$ its molar production rate, and $\mathcal{F}_k$ its diffusion flux.

In the course of solving reactive flow equations, several problems occur. First of all, the sheer complexity of the coupled systems is beyond the means of usual workstations. A rough estimation of the system matrix's expected size illustrates this issue: assume, a 3D simulation is performed on N grid points. The simulation includes 5 components for the flow problem (pressure, three velocity parts and the temperature) and further 20 chemical species. This is to say, we have to comprise $25N$ degrees of freedom. Thinking of second order finite elements, every grid point couples with all points at most 2 nodes away, i.e., on globally refined meshes each matrix row has $5^3 = 125$ couplings. In every node of the mesh nearly all components are coupled with each other, i.e., one matrix entry consists of $25^2 = 625$ entries. If we combine these estimates, the system matrix for a mesh containing N nodes consists of about $125 \cdot 625N$ entries. If we finally assume a triangulation of a 3D domain with 100.000 mesh points, and utilize single-precision arithmetic, the system matrix would require about 30 gigabytes of memory. This estimation is far from being a worst case scenario, since the assumed number of mesh points has to be regarded as an "entry-point" for technical settings and the given memory estimation only involves the system matrix. Using parallel computers and special storage techniques for the matrices we can reduce this aspect, but the capacity of available computer systems will always limit the problems to be considered with fully coupled solution strategies. The usage of numerical methods which no without assembling matrices is not a cure-all for this problem, since the computational effort is also aligned to the (then theoretical) matrix complexity and will then exhibit the bottleneck as matrix free methods will downgrade the efficiency with regard to the computational cost. Parallelization concepts and advanced storage techniques are described in later sections.

2. Basic notations and finite element spaces

In this section we give a short introduction to the finite element method and the related notations. We are interested in the solution u of the problem

$$\begin{aligned} \mathcal{A}u &= f \quad \text{in } \Omega, \\ \mathcal{B}u &= g \quad \text{on } \partial\Omega, \end{aligned}$$

in some domain $\Omega \subset \mathbb{R}^d$ with a given right hand side $f : \Omega \to \mathbb{R}^n$ and a second order (nonlinear) differential operator $\mathcal{A}$ and some boundary operator $\mathcal{B}$ with the usual boundary conditions of Dirichlet type:

$$u = u_0 \text{ on } \partial\Omega_D,$$

of Neumann type

$$\frac{\partial u}{\partial n} = g \text{ on } \partial\Omega_n,$$

or of Robin type:

$$u + \frac{\partial u}{\partial n} = w \text{ on } \partial\Omega_r,$$

with

$$\partial\Omega = \partial\Omega_d \cup \partial\Omega_n \cup \partial\Omega_r.$$

For example the inflow condition as well as the no-slip condition on the walls and the obstacle of the Navier-Stokes benchmark in section 1.2 are of Dirichlet type, while the outflow condition, the "do-nothing" condition is of Robin type.

2.1. Variational Formulation

By the usual route, problem 2.1 is transformed into the weak formulation (see e.g. Grossmann & Roos [RR94]) with a (semi-)linear form $a(\cdot)(\cdot)$. The solution u in the Hilbert space V is searched as

$$a(u)(\varphi) = f(\varphi), \quad \forall \varphi \in V.$$

The semilinear form is defined on the product space $V \times V$.

Remark 2.1. *For the treatment of Dirichlet boundary conditions on some part of the boundary $\Gamma_0 \subset \partial\Omega$ the solution u is searched in an affine space $u \in \hat{u} + V$ with $\hat{u} \in \hat{V}$ and $u = 0$ on Γ_0 for all $u \in V$, where $\hat{u} \in \hat{V}$ describes a prolongation of the Dirichlet boundary conditions into the domain. For simplicity of notation we assume $\hat{V} = V$ and $\hat{u} = 0$.*

Below, we introduce some notations from the theory of function spaces which will be used throughout this work. Detailed properties of these spaces can be found in most books on functional analysis, e.g. in Alt [Alt99], Riesz & Sz.-Nagy [RSN82] or Zeidler [Zei90].

For a domain $\Omega \subset \mathbb{R}^d$ we denote the Lebesgue space of square-integrable functions on Ω by $L^2(\Omega)$. It is a Hilbert space with scalar product and norm:

$$(v, w)_\Omega = \int_\Omega vw \,\mathrm{dx}, \qquad \|v\|_\Omega = (v, v)_\Omega^{\frac{1}{2}}$$

Analogous, $L^2(\partial\Omega)$ describes the space of square-integrable functions on the boundary of Ω equipped with the appropriate scalar product and norm. The Sobolev spaces $H^m(\Omega)$ consist of those functions $v \in L^2(\Omega)$ which possess (distributional) derivatives $\nabla^m v \in L^2(\Omega)^{d\times\cdots\times d}$ up to the order w. For these spaces we define the following half norms:

$$|v|_{H^m(\Omega)} := \|\nabla^m v\|_\Omega,$$

and the norm

$$\|v\|_{H^m(\Omega)} := \left(\sum_{k=0}^{m} |v|^2_{H^k(\Omega)}\right)^{\frac{1}{2}}.$$

Further there exists a continuous trace operator $\gamma : H^1(\Omega) \to L^2(\partial\Omega)$ with $v_{|\partial\Omega} := \gamma(v)$. It allows us to define the function space $H^1_0(\Omega)$ by:

$$H^1_0(\Omega) = \{v \in H^1(\Omega) \,|\, v_{|\partial\Omega} = 0\}.$$

For simplicity we neglect the subscript Ω of a norm or scalar product if the context is without any doubt.

2.2. Finite Element Triangulation

Considering finite elements the utilized function spaces are strongly bound to the *triangulation*. The domain $\Omega \subset \mathbb{R}^d$ with polygonal boundary $\partial\Omega$ is partitioned into open quadrilaterals (hexes in the three dimensional case) K – in the following called *cells* in two and three dimension – which constitute a non-overlapping covering. The case of non-polygonal boundaries is regarded later in this chapter. The triangulation is denoted by $\mathcal{T}_h = \{K\}$ with a mesh parameter h defined as the cell-wise constant function describing the diameter of the cell: $h|_K = h_K$. The intersections of the closure of two adjacent cells are called *faces* (regardless of the problems dimension). The intersections of two adjacent faces each with in three dimensions are called *edges*.

Following the literature a mesh is called regular if it fulfills the standard conditions for shape-regular finite element meshes (as proposed by Ciarlet [Cia78] or Braess [Bra03])

R1 $\bar{\Omega} = \bigcup_{K\in\mathcal{T}_h} \bar{K}$,

R2 $K_1 \bigcap K_2 = \emptyset, \quad \forall K_1, K_2 \in \mathcal{T}_h$ with $K_1 \neq K_2$,

R3 Any face of any cell $K_1 \in \mathcal{T}_h$ is either a subset of the boundary $\partial\Omega$, or a face of another cell $K_2 \in \mathcal{T}_h$.

Condition (R3) is weakened in two ways. To allow adaptive mesh refinement without using connection elements, *hanging nodes* are introduced: cells are allowed to have nodes which lie on midpoints of faces or edges of neighboring cells. At most one hanging node (see Figure 2.1) is permitted on each face.

Considering the non-polygonal boundary $\partial\Omega$ the requirements to a face being a subset of the boundary is alleviated by the matching of the vertices of the face and possibly some inner points of the face with the boundary. This is described in detail later.

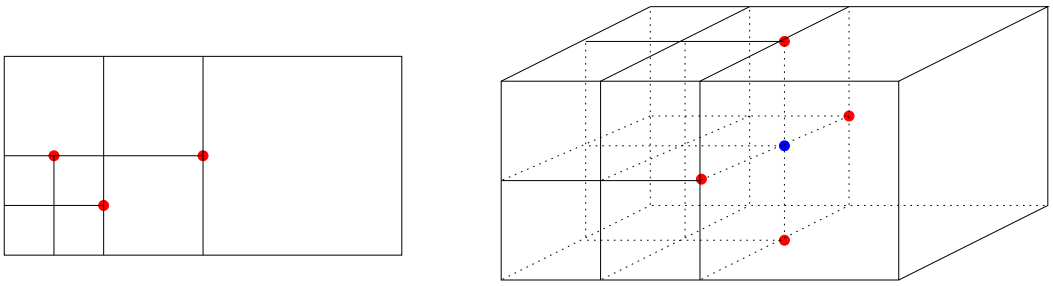

Figure 2.1.: A 2d and 3d mesh with hanging nodes. In three dimensions hanging nodes on faces and hanging nodes on edges are treated differently.

In addition we will sometimes require that the mesh is organized in a patch-wise manner. This means that the triangulation $\mathcal{T}_h$ results from global refinement of some mesh $\mathcal{T}_{2h}$ or even fourfold refinement of some mesh $\mathcal{T}_{4h}$. (Figure 2.2).

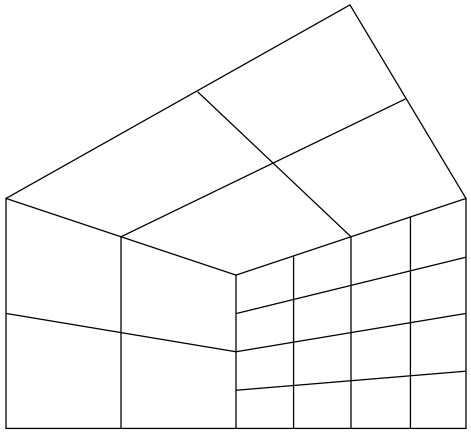

Figure 2.2.: A 2d mesh with hanging nodes and patch-structure with patchdepth one.

2.3. Finite Element Spaces

Following Ciarlet[Cia78], Brenner & Scott [BS94] or Johnson [Joh87] continuous finite element spaces $V_h \in V$ are constructed by

$$V_h = \{v \in V | v_{|K} \in Q(K), K \in \mathcal{T}_h\},$$

where $Q(K)$ denotes a suitable space of polynomial functions on the cell $K \in \mathcal{T}_h$. The polynomial spaces are defined on a *reference cell* $\widehat{Q}(\widehat{K})$, $K = (0,1)^d$ and mapped to the computational cell $T_K : \widehat{K} \to K$

$$\widehat{Q^p}(\widehat{K}) = \text{span}\left\{\prod_{i=1}^{dim} x_i^{\alpha_i}, \alpha_i \in \{0, \ldots, p\}\right\}$$

The simplest case of bilinear functions in two dimensions results in

$$\widehat{Q^1}(\widehat{K}) = \text{span}\,\{1, x_1, x_2, x_1 x_2\}$$

Using quadrilateral cells (respectively hexes in three dimensions) the mapping from the reference cell $\widehat{K}$ to the computational cell is not an affine one in the general case (See figure 2.3). In addition the faces are not even planes any more!

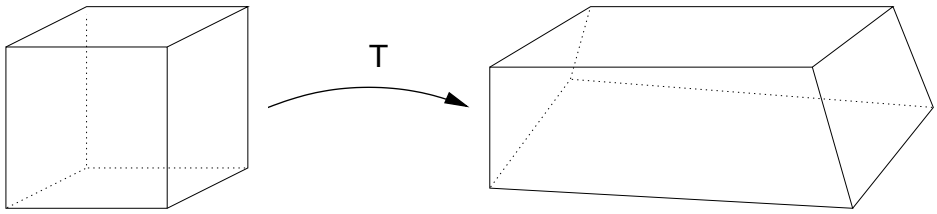

Figure 2.3.: Mapping T_K from the reference cell $\widehat{K}$ to a computational cell K.

To accomplish this kind of mappings one uses *isoparametric* finite elements. The mapping T_K itself lies in the space $\widehat{Q}(\widehat{K})$. Functions in the finite element space $Q(K)$ are therefore established as

$$Q(K) = \{T_K(\hat{v}) | \hat{v} \in \widehat{Q}(\widehat{K})\}.$$

As one necessary condition to the mapping we require its determinant to be bounded from below away from zero.

Regarding partial differential equations with more than one solution components, we use products of the presented function spaces. For example, for the Navier-Stokes equations in three dimensions with one pressure component and three velocity components, we usually denote the used function space by

$$V_h \times Q_h,$$

where V_h itself consists of a threefold product of an one dimensional space. All discretizations used within this work have the same order in all solution components, so called "equal order discretizations".

Considering isoparametric finite elements we get a clue for the treatment of non-polygonal boundaries. Regarding higher order finite elements (i.e. elements of second and higher order) we have degrees of freedoms placed on the midpoints of faces and edges. Using a biquadratic mapping from $\widehat{K}$ to K there is one additional degree of freedom for the mapping on the midpoint of the edges. In Figure 2.4, we display two possible distributions of the degrees of freedom. The possibility on the left-hand side in the figure may lead to the wrong order and deteriorate the accuracy of the finite element ansatz. We adjust the inner degrees of freedom in the biquadratic ansatz to the boundary (right-hand side of Figure 2.4) in order to obtain a boundary approximation of higher order.

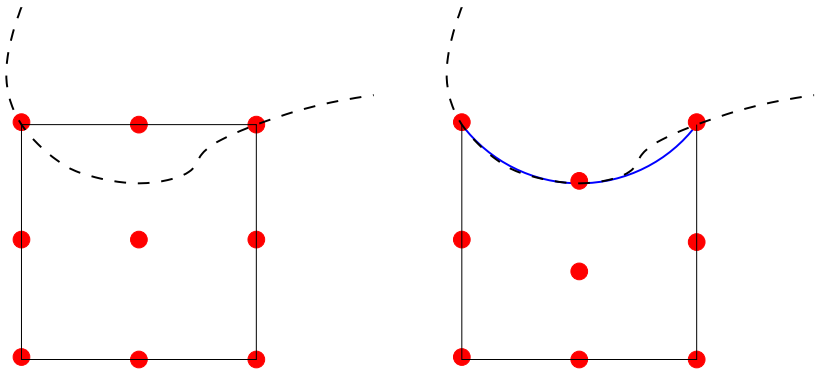

Figure 2.4.: Two possibilities for biquadratic elements on curved boundaries. Left: linear transformation, right: iso-parametric transformation (used in this work).

Using linear finite elements one may use a quadratic transformation onto the computational cell. With this approach we obtain a better accuracy of the method. In the second-order case one could even think of applying a third or fourth-order mapping. However, numerical results can not back up a gain of accuracy.

There are no degrees of freedom in hanging nodes. Instead these nodes are treated by a suitable interpolation of neighboring degrees of freedom. This interpolation has the same order as the finite element ansatz space. This implies continuity and therefore global conformity. For details see e.g. Carey & Oden [CO84]. For implementational details we refer to further sections.

In order to assure an approximation property of the finite element spaces additional conditions to the geometry of the cells, i.e. to the mapping T_K are required. A standard way of describing allowed cells is the *quasi-uniform* condition (see e.g. Grossmann & Roos [RR94]) demands:

$$\text{R4}\ \ \frac{h_K}{\rho_K} \leq C \quad \forall K \in \mathcal{T}_h,$$

where ρ_K denotes the diameter of the biggest ball inscribed in K. Using the lemmas of Lax-Milgram and Cea (see e.g. Alt [Alt99]) the approximation properties of finite element spaces can be characterized by estimates for interpolation errors. Throughout we use two types of interpolation operators $i_h : V \to V_h$: point-wise interpolation for continuous functions and

a generalized interpolation for functions in $H^1(\Omega)$, see Clement [Cle75] or Scott and Zhang [SZ91]. In the following proposition we collect the basic interpolation estimates (valid for the node-wise as well as the averaged interpolation operators of Clement or Scott & Zhang):

Lemma 2.2. *Let $\mathcal{T}_h$ be a quasi-uniform mesh and V_h a space of (isoparametric) finite elements of order p, then there exists a constant C_I depending on C_m and p such that there holds for $u \in H^{m+1}(\Omega)$ and $0 \leq m \leq p$:*

$$|u - i_h u|_{H^1(\Omega)} \leq C_I h^m |u|_{H^{m+1}(\Omega)}.$$

The proof is given e.g. in Braess [Bra03]. In fact, the statement of this lemma is also valid for more general meshes, neglecting the quasi-uniformity condition. Throughout the next chapter we will use strongly stretched (so called anisotropic) elements. Apel [Ape99] gave interpolation estimates on very general meshes. However, these estimates require the nonlinearity in the transformation to be small. In the following we give arguments why the nonlinear part of the transformation can be neglected under mesh refinement.

Lemma 2.3. *Let K be a cell transformed from the unit cell $\hat{K} = [0,1]^2$ by the bilinear transformation T_K:*

$$T_K(x,y) = T_{lin}(x,y) \circ T_{nl}(x,y), \quad T_{nl}(x,y) = \begin{pmatrix} x + \alpha xy \\ y + \beta xy \end{pmatrix} \tag{2.1}$$

The influence of the nonlinear part of the transformation is negligible under mesh refinement: $\alpha_r \leq \alpha \cdot \left(\frac{1}{2}\right)^r$, $\beta_r \leq \beta \cdot \left(\frac{1}{2}\right)^r$.

For simplicity we restrict this considerations to the two-dimensional case. The transformation to the computational cell can be split into the linear and the nonlinear part

$$T(x,y) = T_{lin}(x,y) \circ T_{nl}(x,y),$$

where the nonlinear part is given by

$$T_{nl}(x,y) = \begin{pmatrix} x + \alpha xy \\ y + \beta xy \end{pmatrix}. \tag{2.2}$$

The typical shape of a purely nonlinearly transformed element is given in Figure 2.5. We are interested in the behavior of the nonlinear transformation after refinement of the cells, i.e. in the nonlinearity of the children. Figure 2.6 illustrates the impact of the refinement on the nonlinearity. The path of the lower-left element in this figure will be discussed analytically:

The initial transformation to this element will be denoted by $T_0(x,y)$ of type (2.2). Only the upper-right corner of the cell is distorted from a regular square. The four corners are given by

$$x_1 = \begin{pmatrix} 0 \\ 0 \end{pmatrix}, x_2 = \begin{pmatrix} 1 \\ 0 \end{pmatrix}, x_3 = \begin{pmatrix} 1+\alpha \\ 1+\beta \end{pmatrix}, x_4 = \begin{pmatrix} 0 \\ 1 \end{pmatrix}.$$

The midpoint of this cell – and therefore the upper-right corner of the regarded child – is determined by

$$\frac{1}{4}\sum_{i=1}^{4} x_i = \frac{1}{2}\begin{pmatrix} 1+\frac{\alpha}{2} \\ 1+\frac{\beta}{2} \end{pmatrix},$$

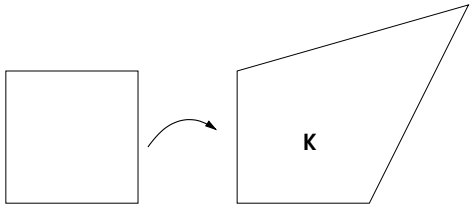

Figure 2.5.: Element K originated from a purely nonlinear transformation T_{nl}.

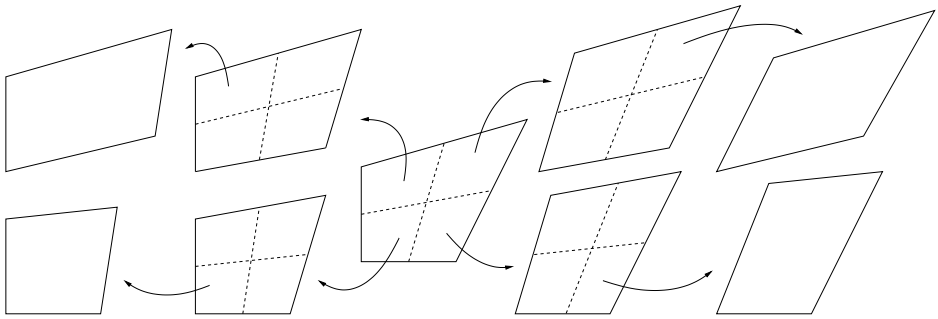

Figure 2.6.: Refinement of a nonlinearly transformed element K. Children are scaled to original size. Under multiple refinement they tend to regular parallelograms.

That is, the transformation $T_1(x, y)$ from the reference element to the lower-left child is given by

$$T_1(x, y) = \frac{1}{2} \begin{pmatrix} x + \frac{\alpha}{2}xy \\ y + \frac{\beta}{2}xy \end{pmatrix},$$

and generally, to the i-th child

$$T_i(x, y) = 2^{-i} \begin{pmatrix} x + 2^{-i} \cdot \alpha xy \\ y + 2^{-i} \cdot \beta xy \end{pmatrix}.$$

The influence of the nonlinearity therefore vanishes under mesh refinement. Similar calculations can be performed for the other three element types. Those will not tend to squares, but to parallelograms. During our further considerations we will treat all elements to be of affine shape.

3. FE discretization for 3D Navier-Stokes

In this chapter we study the finite element discretization of Navier-Stokes flows. As a model problem we will always have in mind the benchmark described in the introduction, see Figure 1.1 in Chapter 1.2. The finite element discretization presented in this chapter will be validated using this problem.

In the progress of this chapter we will derive the Galerkin formulation of the Navier-Stokes equations, followed by a detailed discussion of stabilization techniques with a focus on local projection methods.

3.1. Galerkin Formulation

To derive the Galerkin formulation of the Navier-Stokes equations (1.1), we multiply these equations by some function $\Phi := (\varphi, \xi) \in X := V \times Q$, $V = [H_0^1(\Omega)]^3$ and $Q = L^2(\Omega)$ or $Q = L^2(\Omega)/\mathbb{R}$ respectively if the Dirichlet boundary conditions are prescribed on the whole boundary of the domain. With integration over the domain Ω, the momentum equation is transformed to

$$\begin{aligned}(f, \varphi) =& (-\nu \Delta v + v \cdot \nabla v + \nabla p, \varphi) \\ =& (\nu \nabla v, \nabla \varphi) + (v \cdot \nabla v, \varphi) - (p, \operatorname{div} \varphi) - \int\limits_{\partial\Omega} \left(\nu \frac{\partial v}{\partial n} - pn \right) \varphi \, \mathrm{ds},\end{aligned} \tag{3.1}$$

for all $\varphi \in V$. Thus, on parts of the boundary where no Dirichlet condition is prescribed, the functions φ do not vanish and partial integration yields the "natural" outflow condition

$$\int\limits_{\partial\Omega_o} \left(\nu \frac{\partial v}{\partial n} - pn \right) \varphi \, \mathrm{ds} = 0, \quad \forall \varphi \in V.$$

The equation for the conservation of mass is kept unchanged:

$$(\operatorname{div} u, \xi) = 0, \quad \forall \xi \in Q. \tag{3.2}$$

We define the semi-linear form $a(\cdot)(\cdot)$ by

$$a(u)(\Phi) := (\nu \nabla v, \nabla \varphi)_\Omega + (v \cdot \nabla v, \varphi)_\Omega - (p, \operatorname{div} \varphi)_\Omega + (\operatorname{div} u, \xi)_\Omega,$$

and the functional $F(\cdot)$ by

$$F(\Phi) := (f, \varphi)_\Omega.$$

The weak formulation of the Navier-Stokes equations reads: find $u = (v, p) \in X$ with

$$a(u)(\Phi) = F(\Phi), \quad \forall \Phi \in X. \tag{3.3}$$

The standard Galerkin discretization consists in replacing the infinite dimensional Hilbert spaces V and Q by discrete spaces V_h and Q_h, i.e. find $u_h = (v_h, p_h) \in X_h := V_h \times Q_h$ with

$$a(u_h)(\Phi_h) = F(\Phi_h), \quad \forall \Phi_h \in X_h. \tag{3.4}$$

This approach does not lead to a stable discretization unless the pair of finite dimensional spaces V_h and Q_h fulfill the inf-sup condition (see for example Girault & Raviart [GR86]):

$$\inf_{p_h \in Q_h} \sup_{v_h \in V_h} \frac{(p_h, \operatorname{div} v_h)}{\|p_h\| \|\nabla v_h\|} \geq \gamma > 0.$$

Especially simple equal-order spaces as piecewise trilinear elements for the velocity as well as the pressure do not fulfill the inf-sup condition and the resulting discretization is therefore not stable. To solve the equations in the standard Galerkin formulation one has to utilize mixed interpolations, see Brezzi & Fortin [BF91] or the above mentioned book by Girault & Raviart [GR86].

3.2. Residual Based Stabilization Techniques

Particularly implementational aspects suggest the usage of equal order finite elements for all physical variables. Hughes et al. [HFM86] presented an alternative to the construction of spaces satisfying the inf-sup condition. Their idea is to modify the bilinear form (3.4) in order to get a stable discretization without introducing a noticeable additional error. This is usually done by adding mesh-dependent least squares terms to the Galerkin formulation:

$$a(u_h)(\psi_h) + s(u_h, \psi_h) = F(\psi_h) + F_s(\psi_h), \quad \forall \psi_h \in X_h.$$

The additional *Galerkin Least Squares* terms of Hughes read as follows:

$$\begin{aligned} s_{GLS}(u_h, \psi_h) &= \sum_{K \in \mathcal{T}_h} \delta_K(-\Delta v_h + v_h \nabla v_h + \nabla p_h, -\Delta \varphi_h + \varphi_h \nabla \varphi_h + \nabla \xi_h), \\ F_{GLS}(\psi_h) &= \sum_{K \in \mathcal{T}_h} \delta_K(f, -\Delta \varphi_h + \varphi_h \nabla \varphi_h + \nabla \xi_h), \end{aligned}$$

where $\delta_K \approx h_K^2$. The discrete equation to be solved for $u_h = (v_h, p_h) \in X_h$ is

$$a(u_h)(\psi_h) + s_{GLS}(u_h, \psi_h) = F(\psi_h) + F_{GLS}(\psi_h), \quad \forall \psi_h \in X_h.$$

This discretization is "fully consistent" in the sense that if u is the strong solution, the additional term $f_{GLS}(\psi) - s_{GLS}(u, \psi)$ vanishes for all ψ. A drawback of this stabilization method regarding Navier-Stokes equations is the introduction of boundary layers in the numerical solution which lead to a decrease of accuracy close to the boundary.

Perhaps more serious for complex problems is the numerical effort aligned to this stabilization technique. Within the stabilization terms, second derivatives are present. Using higher discretization order or even for linear elements on isoparametric meshes these derivatives do not vanish for the discrete functions. Their evaluation is very costly (mainly because second derivatives of the inverse transformation are necessary); further they are only needed for the stabilization. A negligence of these second derivatives would result in a loss of accuracy.

Regarding non stationary problems, the least squares term makes time stepping awkward: in order to conserve the consistency of the scheme, one has to consider space-time finite elements using discontinuous approximation in time. Further, mass lumping is impossible.

The algebraic structure of the stabilization term is quite nasty; artificial no-symmetric terms are introduced as well as artificial couplings between pressure and velocity. Taking the liberty of a preview to coming chapters, the effect of the least squares method on reactive flow equations must be discussed. With a large set of convection reaction diffusion equations for the chemical species added to the flow field, the least square terms introduce several artificial couplings between all species with the flow components. The algebraic structures generated by the stabilization terms are far more involved than the original Galerkin formulation itself. Special storage techniques based on mass lumping are not compatible with residual-based stabilization techniques. Details are given in Chapter 6.

3.3. Local Projection Stabilization

Becker & Braack [BB01] proposed a new stabilization technique based on the existence of an inf-sup stable subspace $\tilde{X}_h = V_h \times \tilde{Q}_h \subset X_h$ and a projection operator $i_{\tilde{Q}_h} : Q_h \to \tilde{Q}_h$ into this subspace. In the cited paper they proved the stability of the equal order $Q_r - Q_r$ Stokes elements. In this section we will cover the Stokes equations. The application of local projection methods to convective terms is given in Becker & Braack [BB04]

Lemma 3.1 (Local Projection Stabilization)**.** *Suppose that the inf-sup condition for the pair of finite element spaces $V_h \times \tilde{Q}_h$ is satisfied. Furthermore we require a continuous projection operator $i_{\tilde{Q}_h} : Q_h \to \tilde{Q}_h$*

$$\|i_{\tilde{Q}_h} p\| \leq c_1 \|p\|,$$

and a stabilization bilinear form $s(\cdot,\cdot)$ defined on $Q_h \times Q_h$ with the property

$$\|\pi p\|^2 \leq c_2 s(p,p), \quad \pi := id - i_{\tilde{Q}_h}, \tag{3.5}$$

with constants $c_1, c_2 > 0$. Then there holds $\forall p \in Q_h, \exists \varphi \in V_h$, such that

$$\|\nabla \varphi\| \leq \|p\| \text{ and } \gamma \|p\|^2 \leq (div\, \varphi, p) + c s(p,p), \tag{3.6}$$

with fixed constants $c > 0$ and $0 < \gamma$.

The proof is given in Becker & Braack [BB01].

As the underlying stable spaces we use the Taylor-Hood elements for higher order elements with

$$\pi := id - i_{Q_{r-1}},$$

or the $Q_r/$ iso Q_r element:

$$\pi := id - i_{2h}.$$

The iso Q_r element is the piecewise p-th degree element on the triangulation with mesh-size $2h$. This stabilization is established with help of the patch structure (see Chapter 2). With the defined projection operators we use as stabilization bilinear form:

$$s_{LPS}(p, \xi) = \sum_{K \in \mathcal{T}_K} \alpha_K (\nabla \pi p, \nabla \pi \xi), \tag{3.7}$$

with $\alpha_K \approx h_K^2$. The stable discretization for the Stokes equations reads

$$a(u_h)(\psi_h) + s_{LPS}(u_h, \psi_h) = F(\Phi_h), \quad \forall \Phi_h \in X_h. \tag{3.8}$$

The structure of the stabilization term is very easy; only (diagonal) couplings of the pressure with the corresponding test function are introduced. The artificial boundary layers as known from residual methods are no longer present, see Becker & Braack [BB01] for details. In addition, we do not need second derivatives for the evaluation of the stabilization form. The local projection scheme is not consistent in a way that the stabilization term applied on a strong solution u vanishes, but the additional error is of the same order as the discretization error.

3.4. Stokes Stabilization on Anisotropic Meshes

Considering anisotropic – i.e. stretched – finite elements the uniform choice of the stabilization parameter δ_K in (3.16) leads to a bad conditioning of the linear system as well as to an over-stabilization in certain directions. Figure 2.3 in Chapter 2 shows a typical stretched element appearing if we use bilinear transformations $T_K : \hat{K} \to K$ from the reference element to the computational cell. Throughout this section we cover the two dimensional case, the transfer to three dimensions is obvious. Further we only consider linear finite elements.

For a detailed analysis of anisotropic elements, an exact survey of the transformation mapping is crucial. For general bilinear transformations, the mapping can be split into a translation T_{trans}, the linear part T_{lin} and the nonlinear part T_{nl}:

$$T(x, y) = T_{trans}(x, y) \circ T_{nl}(x, y) \circ T_{lin}(x, y). \tag{3.9}$$

We will neglect the translation since it has no effect on the local approximation properties of the finite elements. Apel [Ape99] derived interpolation error estimates for general meshes originated from bilinear transformations comparable to Lemma 2.2. For $k \in \mathbb{N}, v \in$

$H^{k+1}(K)$, the following estimate was proven for cells K whose vertices are perturbed from parallelograms within some limit.

$$|v - I_h^{(k)} v|^2_{H^1(K)} \leq C \sum_{|\alpha|=k} h_K^{2\alpha} |D^\alpha v|^2_{H^1(K)} + C \sum_{r=[k/2]+1}^{k} h_2^{2(k-r)} \sum_{|\alpha|=2r-k-1} \sum_{|\beta|=k+1-r} h^{2\alpha} a^{2\beta} \|D^{\alpha+\beta} v\|^2_K, \tag{3.10}$$

where a indicates the degree of perturbance from the linear transformation. However, those estimates have two drawbacks or rather can be simplified for our purpose:

- Due to the nonlinearity, lower order terms are introduced for higher order ($k \geq 2$) elements. However, as discussed in Chapter 2, the nonlinearity can be neglected in the context of mesh refinement. More precisely: initially nonlinear cells tend to parallelograms under refinement. The factor a in (3.10) corresponds to the nonlinear perturbations α and β in (2.2) which asymptotically behave as the mesh size h_K.

- Beside the usual interior angle condition, Apel [Ape99] calls for a coordinate system condition: the angle ψ between the longest side of all quads and the x_1 axis should be bounded by the aspect ratio:

  $$|\sin \psi| \leq C h_2 / h_1,$$

  where h_1 is the longer side. This would rule out the possibility of anisotropic refinement around e.g. a circle, as perhaps required in the model problem (Figure 1.1). However, the coordinate system condition occurs due to the measurement of the derivatives according to the Cartesian basis. Considering anisotropic elements this is not natural. If we use (locally) adapted coordinate systems with directions η_i we can derive results without this restriction.

Using linear finite elements and under the assumption of linear transformation to the computational cells, the analysis interpolation estimates is far easier. We use an interpolation operator $i_h : V \to V_h$ of Clement-type adjusted for anisotropic meshes, see Becker [Bec95] . The averaging of the node-values is done along the long edges of the elements. Beyond the interior angle condition, Becker assumes the change of the aspect ratio h_2/h_1 between neighboring cells to be bounded by some constant. The element-wise aspect ratio however is allowed to be large. Becker showed the following stability estimates for the interpolation operator on rectangular stretched elements K with $h_x \geq h_y$:

$$\|\partial_x i_h v\|_K \leq c \|\nabla v\|_{\mathcal{N}(K)},$$
$$\|\partial_y i_h v\|_K \leq c \|\partial_y v\|_{\mathcal{N}(K)},$$

where $\mathcal{N}(K)$ is the patch of elements adjacent to K and the constant c is independent of the shape and size of K.

We derive local interpolation estimates which measure the error according to local coordinate systems. We restrict our considerations to cells obtained by affine transformations

of the reference element. Therefore, neglecting the translation, we can further split the transformation (3.9) into a rotational part, and a scaling and shear part:

$$T_{lin}(x,y) = T_{rot}(x,y) \circ T_{shear}(x,y)$$

which reads in matrix notation

$$T_{lin}(x,y) = \begin{pmatrix} \cos\varphi & -\sin\varphi \\ \sin\varphi & \cos\varphi \end{pmatrix} \begin{pmatrix} h_x & sh_y \\ 0 & h_y \end{pmatrix}. \tag{3.11}$$

The x-axis of the local coordinate system will be aligned with the longer side of the cell, the y axis with the orthogonal direction. I.e., we always have $h_x \geq h_y$. The rotation can be neglected in the analysis and the regarded transformation simplifies to

$$T(x,y) = \begin{pmatrix} h_x & sh_y \\ 0 & h_y \end{pmatrix}, \tag{3.12}$$

which describes all parallelograms whose longer side lie on the positive x-axis. Since the angle θ between the long and the short edge is estimated by

$$\cot\theta = s,$$

the interior angle condition can be rewritten as a condition to the shearing parameter s:

$$-s_* \leq s \leq s_*, \tag{3.13}$$

with s_* fixed and particularly independent of the mesh parameter h.

Using this notation, we can give interpolation estimates for the regarded kind of cells. All derivatives are expressed in the direction of the longest side and the according orthogonal one:

Lemma 3.2 (Anisotropic Interpolation Estimate)**.** *Assume that K is a parallelogram with its interior angles γ_i being bounded by $0 < \gamma_* \leq \gamma_i \leq \pi - \gamma_*$, $i = 1, \ldots, 4$, where the constant γ_* is independent of K. The change of the anisotropy $\kappa_K = h_{K,max}/h_{K,min}$ between adjacent cells K, K' with $K' \subset \mathcal{N}(K)$ is bounded. Further let η_1 be a unit vector aligned with the longer sides of K and η_2 the unit vector orthogonal on η_1. Then, for $v \in H^2(\mathcal{N}(K))$, $p \in H^1(\mathcal{N}(K))$ the following estimates holds for the interpolation operator i_h into linear elements:*

$$\begin{aligned}
\|v - i_h v\|_K^2 &\leq C\left(h_1^4 \|\partial_{\eta_1\eta_1} v\|_{\mathcal{N}(K)}^2 + h_1^2 h_2^2 \|\partial_{\eta_1\eta_2} v\|_{\mathcal{N}(K)}^2 + h_2^4 \|\partial_{\eta_2\eta_2} v\|_{\mathcal{N}(K)}^2\right) \\
\|\partial_{\eta_1}(v - i_h v)\|_K^2 &\leq C\left(h_1^2 \|\partial_{\eta_1\eta_1} v\|_{\mathcal{N}(K)}^2 + h_2^2 \|\partial_{\eta_1\eta_2} v\|_{\mathcal{N}(K)}^2\right) \\
\|\partial_{\eta_2}(v - i_h v)\|_K^2 &\leq C\left(h_1^2 \|\partial_{\eta_1\eta_1} v\|_{\mathcal{N}(K)}^2 + h_1^2 \|\partial_{\eta_1\eta_2} v\|_{\mathcal{N}(K)}^2 + h_2^2 \|\partial_{\eta_1\eta_2} v\|_{\mathcal{N}(K)}^2\right),
\end{aligned}$$

$$\begin{aligned}
\|p - i_h p\|_K^2 &\leq C\left(h_1^2 \|\partial_{\eta_1} p\|_{\mathcal{N}(K)}^2 + h_2^2 \|\partial_{\eta_2} p\|_{\mathcal{N}(K)}^2\right) \\
\|\partial_{\eta_1}(p - i_h p)\|_K^2 &\leq C\left(\|\partial_{\eta_1} p\|_{\mathcal{N}(K)}^2 + \kappa_K^{-2} \|\partial_{\eta_2} p\|_{\mathcal{N}(K)}^2\right) \\
\|\partial_{\eta_2}(p - i_h p)\|_K^2 &\leq C\left(\|\partial_{\eta_1} p\|_{\mathcal{N}(K)}^2 + \|\partial_{\eta_2} p\|_{\mathcal{N}(K)}^2\right)
\end{aligned}$$

with the constant C independent of K, h_1, the length of K in direction η_1 and $h_2 = |K|/h_1$.

PROOF: Since we will cell-wise align the coordinate system with the longer edge (w.l.o.g. the edge with length h_x, so that $h_y \leq h_x$), the rotation will be neglected and the transformation to be analyzed simplifies to (3.12).

The estimates are mainly a result of integral transformation back and forth to the reference element and usage of the stability of the interpolation operator proven by Becker [Bec95] and the Bramble Hilbert Lemma [Bra03] respectively. In the following we give the proof for one of these estimates:

$$
\begin{aligned}
\|\partial_y(v - i_h v)\|_K^2 &= h_x h_y \int_{\hat{K}} [\partial_y(\hat{v} - i_h\hat{v})(T^{-1}(x))]^2 \mathbf{d\hat{x}} \\
&= h_x h_y \int_{\hat{K}} \left[-\frac{s}{h_x}\partial_{\hat{x}}(\hat{v} - i_h\hat{v})(\hat{x}) + \frac{1}{h_y}\partial_{\hat{y}}(\hat{v} - i_h\hat{v})(\hat{x}) \right]^2 \mathbf{d\hat{x}} \\
&\leq C_1 C_{bhl} h_x h_y \int_{\hat{K}} \frac{s^2}{h_x^2}\left[\partial_{\hat{x}}\hat{\nabla}\hat{v}(\hat{x})\right]^2 + \frac{1}{h_y^2}\left[\partial_{\hat{y}}\hat{\nabla}\hat{v}(\hat{x})\right]^2 \mathbf{d\hat{x}} \\
&= C_1 C_{bhl} \int_{\mathcal{N}(K)} \frac{s^2}{h_x^2}\left[\partial_{\hat{x}}\hat{\nabla}v(T(\hat{x}))\right]^2 + \frac{1}{h_y^2}\left[\partial_{\hat{y}}\hat{\nabla}v(T(\hat{x}))\right]^2 \mathbf{dx} \\
&= C_1 C_{bhl} \int_{\mathcal{N}(K)} s^2 \left[(h_x + sh_y)v_{xx} + h_y v_{xy}\right]^2 + \\
&\qquad \left[(sh_x + s^2 h_y)v_{xx} + (h_x + 2sh_y)v_{xy} + h_y v_{yy}\right]^2 \\
&\leq C_1^2 s_*^4 C_{bhl} \left(h_x^2 \|v_{xx}\|_{\mathcal{N}(K)} + h_x^2 \|v_{xy}\|_{\mathcal{N}(K)}^2 + h_y^2 \|v_{yy}\|_{\mathcal{N}(K)}^2 \right)
\end{aligned}
$$

The constant C_{bhl} is given by the Bramble Hilbert Lemma, C_1 originates from $(\sum_{i=1}^N a_i)^2 \leq C_1(N)\sum_{i=1}^N a_i^2$ and s_* originates from (3.13). Combining the constants and after rotation of the coordinate system, the proof is finished.

□

To ensure a good behavior of the linear solver, the stabilization has to be aligned to the essential arrangement of the cells. We split the LPS stabilization term (3.8) into different

directions $\eta_i, i = 1, \ldots, d$ with $(\eta_i, \eta_j) = \delta_{ij}$:

$$\begin{aligned} s_{LPS}(p,\xi) =& \delta_0 \sum_{K\in\mathcal{T}_h} \left(\sum_{i=1}^{d} \sqrt{\delta_{K,i}} \left(\eta_i, \nabla\pi p\right) \eta_i, \sum_{i=1}^{d} \sqrt{\delta_{K,i}} \left(\eta_i, \nabla\pi\xi\right) \eta_i \right)_K \\ =& \delta_0 \sum_{K\in\mathcal{T}_h} \sum_{i,j=1}^{d} \sqrt{\delta_{K,i}\delta_{K,j}} \left(\partial_{\eta_i}\pi p \cdot \eta_i, \partial_{\eta_j}\pi\xi \cdot \eta_j\right)_K \\ =& \delta_0 \sum_{K\in\mathcal{T}_h} \sum_{i=1}^{d} \delta_{K,i} \left(\partial_{\eta_i}\pi p, \partial_{\eta_i}\pi\xi\right)_K . \end{aligned} \tag{3.14}$$

The specific choice of the stabilization parameters can be determined as result of an a-priori error estimate which follows later in this section. Another problem is the detection of the optimal directions η_i. If the transformation $T_K : \hat{K} \to K$ would just be a composition of translation, scaling and rotation, the image of the Cartesian unit vectors would notify the dominant directions. But considering the admitted elements (see Figure 2.3) the transformation also features a shear part and further is nonlinear.

Following the above discussion, we neglect the nonlinearity and replace the transformation by the linearization $T'(x, y)$ in the midpoint of the reference element

$$T'(x,y) := T\left(\frac{1}{2}, \frac{1}{2}\right) \begin{pmatrix} x \\ y \end{pmatrix} + x_0.$$

This linearization can be split into the basic mapping types: rotation, scaling and shearing as (3.11). The directions η_i are defines as follows: let

$$\tilde{\eta}_i = T'(e_i),$$

be the images of the Cartesian basis vectors and

$$\tilde{h}_i = |\tilde{e}_i|$$

their length. Further assume $h_i \geq h_{i+1}$. Due to the shearing, the vectors $\tilde{\eta}_i$ are not necessarily orthogonal. The final directions η_i are the orthogonalization of $\tilde{\eta}_i$ (while maintaining $\eta_1 = \tilde{\eta}_1$). Finally we set $h_i = |\eta_i|$ and normalize the directions.

Recapitulating the proof for the a-priori estimate given in Becker & Braack [BB01], we get a hint for the optimal choice of the stabilization parameters $\delta_{K,i}$. The following lemma is a modification of Theorem 3 in Becker & Braack [BB01] regarding the anisotropy:

Lemma 3.3. *Suppose the following regularity of the exact solution $v \in H^2(\Omega), p \in H^1(\Omega)$ of the Stokes equations. Then with $s(\cdot,\cdot)$ defined as in (3.14) and*

$$\delta_{K,i}\delta_{K,i} = \delta_0 h_i^2,$$

there holds for piecewise bilinear elements:

$$\|\nabla(v - v_h)\| + \|p - p_h\| \leq c \sum_i h_i (\|\partial_i p\| + \|\partial_i \nabla v\|).$$

PROOF: To simplify the proof, we introduce some notations:

$$\begin{aligned} \|p\|_\delta^2 &= s(p,p), \\ |||u|||^2 &= \|\nabla v\|^2 + \frac{\gamma^2}{4}\|p\|^2 + \|p\|_\delta^2. \end{aligned}$$

Following [BB01], the error $u - u_h$ is split into the interpolation error η and the projection error ξ:

$$u - u_h = \underbrace{(u - i_h u)}_{\eta} + \underbrace{(i_h u - u_h)}_{\xi}.$$

Using Lemma 3.2 we can directly treat the interpolation error η:

$$\begin{aligned} \|\nabla(v - i_h v)\|^2 + \|p - i_h p\|^2 &= \sum_{K\in\mathcal{T}} \left\{ \| \sum_i (\nabla(v - i_h v), \eta_i)\eta_i\|^2 + \|p - i_h p\|_K^2 \right\} \\ &\le \sum_{K\in\mathcal{T}} \left\{ \left(\sum_i \|\partial_{\eta_i}(v - i_h v)\|_K^2 \right) + \|p - i_h p\|_K^2 \right\} \\ &\le C \sum_{K\in\mathcal{T}} \Big\{ h_1^2 \|\partial_{\eta_1\eta_1} v\|_K^2 + (h_1^2 + h_2^2)\|\partial_{\eta_1\eta_2} v\|_K^2 + h_2^2 \|\partial_{\eta_2\eta_2} v\|_K^2 + \\ &\qquad h_1^2 \|\partial_{\eta_1} p\|_K^2 + h_2^2 \|\partial_{\eta_2} p\|_K^2 \Big\} \end{aligned}$$

From the stability condition (3.6)

$$\sup_{\psi\in V_h\times Q_h, |||\psi|||=1} \{a(u,\psi) + s(u,\psi)\} \ge \frac{1}{4}|||u|||,$$

we get

$$\frac{1}{4}|||\xi||| \le \sup_{|||\psi|||=1} \sup (a+s)(\xi,\psi).$$

This equals

$$\begin{aligned} (a+s)(\xi,\psi) &= -(a+s)(\eta,\psi) + (a+s)(u - u_h, \psi) \\ &= -a(\eta,\psi) - s(u - i_h u, \psi) + (a+s)(u - u_h, \psi) \\ &= -a(\eta,\psi) + a(u,\psi) - (a+s)(u_h,\psi) + s(i_h u, \psi) \\ &= -a(\eta,\psi) + s(i_h u, \psi). \end{aligned}$$

The Galerkin terms can be estimated by

$$\begin{aligned} a(\eta,\psi) &= (\nabla(v - i_h v), \nabla\varphi) - (p - i_h p, \operatorname{div}\varphi) + (\operatorname{div}(v - i_h v), \xi) \\ &\le \|\nabla(v - i_h v)\|\|\nabla\varphi\| + \|p - i_h p\|\|\nabla\varphi\| + \|\nabla(v - i_h v)\|\|\xi\| \\ &\le (\|\nabla(v - i_h v)\| + \|p - i_h p\|)(\|\nabla\varphi\| + \|\xi\|). \end{aligned}$$

Using the interpolation estimates we get the result. For the stabilization term we have

$$\begin{aligned} s(i_h u, \psi) &= \sum_{i,K\in\mathcal{T}} \delta_{K,i} \left(\partial_{\eta_i} \pi i_h p, \partial_{\eta_i} \pi \xi\right) \\ &\leq \|i_h p\|_\delta \|\xi\|_\delta \\ &\leq \left(\sum_{i,K\in\mathcal{T}} \delta_{K,i} \|\partial_{\eta_i}(\pi i_h p)\|_K^2 \right)^{\frac{1}{2}} |||\Psi||| \end{aligned}$$

Finally, using $\pi p = (id - i_{2h})p$ and Lemma 2.2, we have

$$\begin{aligned} \sum_i \delta_{K,i} \|\partial_{\eta_i} \pi i_h p\|_K^2 &\leq \sum_i \delta_{K,i} \|\partial_{\eta_i}(p - i_h p)\|_K^2 + \delta_{K,i} \|\partial_{\eta_i} \pi p\|_K^2 \\ &\leq C \sum_i \delta_{K,i} \|\partial_{\eta_i}(p - i_{2h} p)\|_K^2 \\ &\leq C \left((\delta_{K,1} + \delta_{K,2}) \|\partial_{\eta_1} p\|_K^2 + (\delta_{K,1} \kappa_K^{-2} + \delta_{K,2}) \|\partial_{\eta_2} p\|_K^2 \right) \end{aligned}$$

With $\delta_{K,i} = \delta_0 h_i^2$ and $\kappa_K = h_1/h_2$ we get

$$\sum_i \delta_{K,i} \|\partial_{\eta_i} \pi i_h p\|_K^2 \leq C \delta_0 \sum_i h_i^2 \|\partial_{\eta_i} p\|_K^2,$$

and finish the proof with the announced choice of the cell-wise stabilization parameters $\delta_{K,i}$:

$$\delta_{K,i} = \delta_0 h_{\eta_i}^2.$$

□

The application of the anisotropic stabilization is rather easy. If we reformulate (3.12) in the Cartesian basis, we get

$$\begin{aligned} s_{LPS}(p, \xi) &= \delta_0 \sum_{K\in\mathcal{T}_h} \sum_{i,k,l=1}^{d} \delta_{K,i} (\eta_i)_k \partial_k \pi p (\eta_i)_l \partial_l \pi \xi \\ &= \delta_0 \sum_{K\in\mathcal{T}_h} \sum_{i,k,l=1}^{d} \delta_{K,i} (\eta_i)_k (\eta_i)_l \partial_k \pi p \partial_l \pi \xi \\ &= \delta_0 \sum_{K\in\mathcal{T}_h} \left(\nabla \pi p, D_K \nabla \pi \xi\right)_K, \end{aligned}$$

with the matrix $D_K \in \mathbb{R}^{d\times d}$ given as

$$(D_K)_{k,l} = \sum_{i=1}^{d} \delta_{K,i} (\eta_i)_k (\eta_i)_l.$$

For the validation of this stabilization scheme we analyze a simple driven cavity problem. The configuration is given in Figure 3.1. We consider Navier-Stokes flow in a square driven

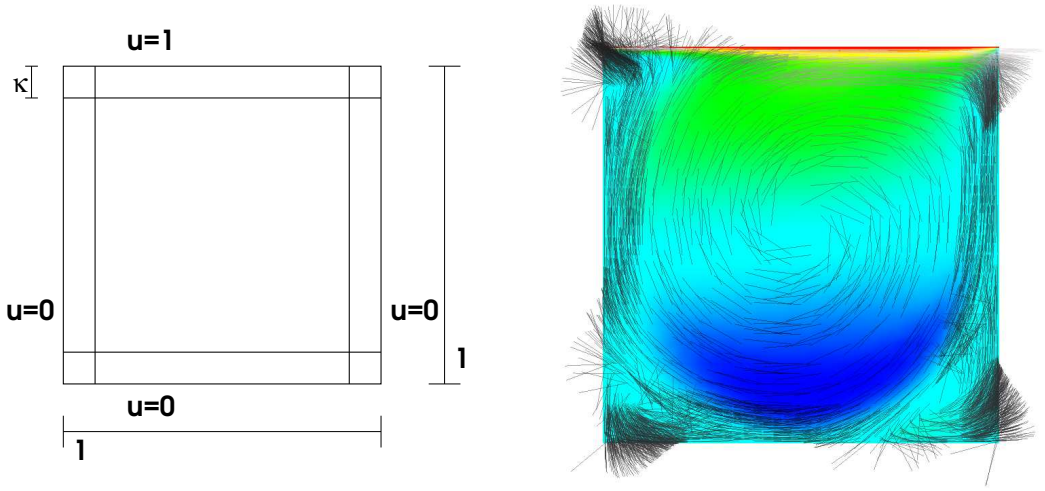

Figure 3.1.: Computational domain for driven cavity test. The anisotropy is controlled via adjusting the cells in the coarse mesh, i.e. by adjusting the size κ of the corner cells. The flow is driven by a prescribed velocity on the upper boundary.

by a prescribed velocity in horizontal direction on the upper boundary. On this part of the boundary the x-component of the velocity is set to $v_x = 1$. With the viscosity $\nu = 0.0005$ the problem yields the Reynolds number $Re = 2000$. As quantity of interest we measure the normal derivative of the horizontal velocity on the lower boundary:

$$j(u) = \int_{\Gamma_{low}} \frac{\partial u_x}{\partial n} \, \mathrm{ds}.$$

The anisotropy of the discretization is controlled by adjusting the 9 cells of the coarse mesh. In Table 3.1 we list the convergence rate of the linear solver and the discretization error for different anisotropies $1 : \kappa$ on a sequence of globally refined meshes. While we have to face problems with the isotropic stabilization scheme in the linear solver, we can solve all problems with a stable multigrid convergency using the modified LPS stabilization. Nevertheless, the convergence rates of the linear solver is not very good. This is aroused by the usage of a standard incomplete LU decomposition of the matrix as a smoother, which is well known to have problems with anisotropies. However, the analyzed aspect ratios in Table 3.1 cover the complete range used throughout this work. In the case of reactive flows – discussed in Chapter 6 – the correct application of the stabilization on anisotropic meshes decides about convergence or divergence of the solver. In this work, anisotropic meshes will be used to get simple coarse meshes of complex geometries. Thus, the aspect ratios keep bounded and smoothing can be performed with the standard ILU.

	std. LPS		anisotropic LPS	
$1:\kappa=1$	$j(e)$	ρ		
9 216	$5.14\cdot10^{-2}$	0.41		
36 864	$1.93\cdot10^{-2}$	0.40		
$1:\kappa=5$	$j(e)$	ρ	$j(e)$	ρ
9 216	$1.99\cdot10^{-2}$	0.67	$2.01\cdot10^{-2}$	0.55
36 864	$1.01\cdot10^{-2}$	0.75	$1.01\cdot10^{-2}$	0.54
$1:\kappa=10$	$j(e)$	ρ	$j(e)$	ρ
9 216	$6.06\cdot10^{-3}$	0.82	$5.67\cdot10^{-3}$	0.63
36 864	$1.86\cdot10^{-3}$	0.88	$1.88\cdot10^{-3}$	0.55

Table 3.1.: Discretization error and convergence rate ρ of the multigrid solver on a sequence of meshes. The ratio between long and short side in the meshes is $1:1$ in the upper table and $1:5$ in the middle and $1:10$ in the lower one. The values on the left side are obtained using isotropic LPS, for the values on the right side, the anisotropic modification of LPS was used.

3.5. Quadratic Adaptive Finite Elements

In this section we describe the finite element discretization of second order which we mainly use throughout this work. The solution is approximated in the space $u_h = (v_h, p_h) \in V_h \times Q_h$ with

$$\begin{aligned} V_h &= \{v \in [H^1(\Omega)]^d \,|\, v_{|K} \in Q^2(K)\}, \\ Q_h &= \{p \in L^2(\Omega)/\mathbb{R} \,|\, p_{|K} \in Q^2(K)\}. \end{aligned}$$

As mentioned in the previous section this function space does not fulfill the inf-sup condition, the resulting discretization is not stable. However this equal order ansatz with degrees of freedom in the nodes of the mesh simplifies the implementational effort.

3.5.1. Pressure Stabilization

Since this finite element space is not stable in the sense of the inf-sup condition we apply local projection stabilization with respect to the stable $Q_2 - Q_1$ Taylor-Hood element $\tilde{X}_h \subset X_h$ with

$$\tilde{Q}_h = \{p \in L^2(\Omega)/\mathbb{R} \,|\, p_{|K} \in Q^1(K)\}, \tag{3.15}$$

the continuous space of piecewise trilinear functions on the same triangulation $\mathcal{T}_h$. This is the well known stable Taylor-Hood element, see Cuvelier, Segal & Steenhoven [CSvS86]. The

LPS projection operator $i_{\tilde{Q}_h} : Q_h \to \tilde{Q}_h$ is the node-wise interpolation. The stabilization bilinear form $s_{LPS}(\cdot,\cdot)$ is chosen as:

$$s_{LPS}(p,\xi) = \sum_{K\in\mathcal{T}_h} \delta_K \left(\nabla\pi p, \nabla\pi\xi\right)_K . \tag{3.16}$$

We mainly favor this stabilized formulation over e.g. the Taylor-Hood element due to the easier handling of data structures using equal order spaces for all components of the solution. The drawback of an increased number of matrix couplings using the equal order element gets negligible for larger systems of equations as addressed in Chapter 6.

To apply the LPS stabilization, we have to check condition (3.5) in Lemma 3.1:

$$\|\pi p\|^2 \leq \sum_{K\in\mathcal{T}_h} \delta_K \left(\nabla\pi p, \nabla\pi p\right)_K .$$

This is easily seen by transformation of πp to the reference cell $\hat{K}$ and the fact that $\|\nabla(\cdot)\|_{\hat{K}}$ is a norm for all functions (πp) with $p \in V_h$. Using the equivalence of norms on finite dimensional spaces the estimate is proven with a constant $\delta_K \approx h_K^2$.

3.5.2. Convection Stabilization

Considering problems with dominant convection, i.e. large Reynolds numbers the convective term

$$(v_h \cdot \nabla v_h, \varphi_h)$$

imposes values to the secondary diagonals of the system matrix leading to numerical instabilities. This instabilities are corrected by the introduction of an additional stabilization term $s_{\text{conv}}(\cdot,\cdot)$ (and possibly additional right hand side terms $f_{\text{conv}}(\cdot)$). Well-known techniques for convection stabilization are the Upwind-Discretization of artificial diffusion, described i.e. in [CSvS86] or the streamline upwind Petrov-Galerkin method (SUPG) originally proposed by Brooks & Hughes [BH82]. All this methods impose a reduction of the convergence order, to $O(h)$ in the case of Upwind-Diffusion or artificial Diffusion and $O(h^{\frac{3}{2}})$ considering Streamline-Diffusion. All methods introduce additional diffusion to the problem. The following stabilization term was suggested by Becker & Braack [BB04]:

$$s_{conv}(u_h)(\psi) = \sum_{K\in\mathcal{T}_h} \delta_K \left((v_h \cdot \nabla)\pi v_h, (v_h \cdot \nabla)\pi\varphi\right)_K$$

with π as described for the pressure stabilization and the cell-wise constant δ_K depending on the local balance of convection and diffusion:

$$\delta_K = \delta_0 \frac{h_K^2}{6\nu + h_K\|v_h\|_K}.$$

This stabilization scheme can be regarded as a specification of the theory of Guermond [Gue99] with a special filter function. We add control over the fluctuations of the convective term with regard to a coarser mesh. For higher Reynolds numbers oscillation with exactly this frequency appear.

The usage of the Taylor-Hood element as the underlying stable subspace results in a loss of accuracy for highly convective problems. We have to use the $Q_2/$ iso Q_2 element as the stable space we project into. In a later section we will give details on the implementation and approximation properties of the local projection method based on the $Q_2/$ iso Q_2 element. However for the Stokes equations, there are no problems.

For both choices of the fluctuation operator π, the stabilized discrete problem reads: find $u_h \in X_h$ with

$$a(u_h)(\psi) + s_{\mathrm{LPS}}(p_h, \xi) + s_{\mathrm{conv}}(v_h)(\varphi) = F(\psi), \quad \forall \psi \in X_h.$$

For abbreviation both stabilization terms are combined in $s(\cdot,\cdot)$.

3.5.3. Implementational Aspects

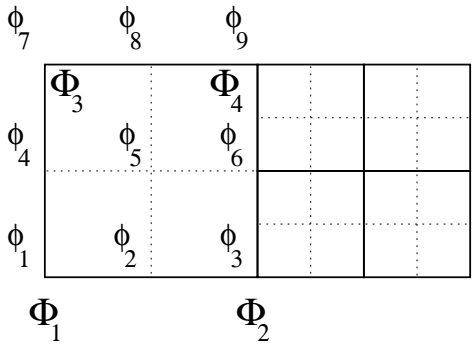

Figure 3.2.: Distribution of the basis functions in one Q_2 cell. The Q_1 basis functions $\Phi_1, \ldots, \Phi_4$ can be represented by using the Q_2 functions $\varphi_1, \ldots, \varphi_9$.

For simplicity we only present the two dimensional case and the projection into a space of lower degree. Considering higher order finite elements the degrees of freedom are settled in a nested sense (as shown in Figure 3.2 for biquadratic Q_2 finite elements). On the reference element we have a representation of the Q_1 basis functions by use of the quadratic functions. If we denote the 9 Q_2 ansatz functions on a cell K by $\varphi_1, \ldots, \varphi_9$ and the corresponding 4 Q_1 functions by $\Phi_1, \ldots, \Phi_4$, we get a algebraic coherency between the vectors $(\varphi)_i$ and $(\Phi)_i$ by the linear operator $I_h : V_h^{(2)} \to V_h^{(1)}$:

$$\begin{aligned} \Phi &= I_h \varphi \\ I_h &= \begin{pmatrix} 1 & \frac{1}{2} & 0 & \frac{1}{2} & \frac{1}{4} & 0 & 0 & 0 & 0 \\ 0 & \frac{1}{2} & 1 & 0 & \frac{1}{4} & \frac{1}{2} & 0 & 0 & 0 \\ 0 & 0 & 0 & \frac{1}{2} & \frac{1}{4} & 0 & 1 & \frac{1}{2} & 0 \\ 0 & 0 & 0 & 0 & \frac{1}{4} & \frac{1}{2} & 0 & \frac{1}{2} & 1 \end{pmatrix} \end{aligned}$$

Further if we represent the local solution u_K on a cell K by

$$u_K = (U, \varphi),$$

where U denotes the nodal vector of the solution in all degrees of freedom on the cell K and we further name the restriction of U to the 4 Q_1 degrees of freedom by $R_{Q_1} \in \mathbb{R}^{4\times 9}$, the projection operator $\pi := (id - i_{Q_1})$ is given by

$$\begin{aligned} \pi u_K &= u_K - i_{Q_1} u_K \\ &= (U, \varphi) - (R_{Q_1} U, I_h \varphi) \\ &= (U, \underbrace{(I - R_{Q_1}^T I_h)}_{\Pi} \varphi), \end{aligned}$$

with

$$\Pi = \begin{pmatrix} 0 & -1/2 & 0 & -1/2 & -1/4 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & -1/2 & 0 & 0 & -1/4 & -1/2 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & -1/2 & -1/4 & 0 & 0 & -1/2 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & -1/4 & -1/2 & 0 & -1/2 & 0 \end{pmatrix}.$$

Only ansatz functions which correspond to the Q_1 element have to be changed. The projection into a space of equal order on a coarse mesh can be handled in a simular way. Instead of really assembling this matrix – which is fixed for all elements in the triangulation – the modification is done e.g. by

$$\Pi\varphi_1 = -\frac{1}{2}\varphi_2 - \frac{1}{2}\varphi_4 - \frac{1}{4}\varphi_5.$$

Instead of representing linear test functions Φ_i with help of the quadratic test functions φ_i, we represent test functions in the coarse function space V_{2h} with the fine test functions. Another, yet comparable matrix Π describes this projection on a algebraic level. See [BR05c] for details.

Similar techniques are applied for the treatment of hanging nodes concerning higher order finite element spaces: if we regard two adjacent cells with different refinement levels (see Figure 2.1 in Chapter 2) the hanging nodes on the fine cell are replaced by interpolated values from the coarse cell. This interpolation is performed using a representation of the 'coarse' basis functions Φ_i with the 'fine' functions φ_i on a patch of fine cells.

$$\begin{pmatrix} \Phi_1 \\ \Phi_2 \\ \vdots \\ \Phi_c \end{pmatrix} = H \begin{pmatrix} \varphi_1 \\ \varphi_2 \\ \vdots \\ \varphi_f \end{pmatrix}$$

If we have to 'condense' hanging nodes in a solution vector $u_h = \sum u_i\varphi_i$, i.e. interpolate into the coarse ansatz space, we patch-wise multiply the vector of indices $(u_1, \cdots, u_N)$ with

α/δ	0.05	0.1	0.2	0.5	1.
0.05	$1.72e^{-3}$	$1.36e^{-3}$	$7.82e^{-4}$	$-3.41e^{-4}$	$-1.45e^{-3}$
0.10	$1.57e^{-3}$	$1.21e^{-3}$	$6.29e^{-4}$	$-4.85e^{-4}$	$-1.58e^{-3}$
0.20	$1.48e^{-3}$	$1.11e^{-3}$	$5.25e^{-4}$	$-5.85e^{-4}$	$-1.67e^{-3}$
0.50	$1.40e^{-3}$	$1.03e^{-3}$	$4.45e^{-4}$	$-6.65e^{-4}$	$-1.76e^{-3}$
1.00	$1.37e^{-3}$	$9.98e^{-4}$	$4.13e^{-4}$	$-6.98e^{-4}$	$-1.78e^{-3}$

α/δ	0.2	0.5	1.
0.2	$1.33e^{-3}$	$-1.44e^{-4}$	$-4.07e^{-3}$
0.5	$6.33e^{-4}$	$-8.71e^{-4}$	$-4.78e^{-3}$
1.	$-5.71e^{-4}$	$-1.88e^{-3}$	$-7.39e^{-3}$

Table 3.2.: Dependency of the accuracy on the choice of the stabilization parameters. Upper table: $Q_2 - Q_2$ LPS stabilized, lower table $Q_2 - Q_2$ PSPG/SUPG. The calculations are done on a structured mesh with 983 040 degrees of freedom.

the transformation matrix H. The same procedure is applied to the system matrix. First we cell-wise assemble the matrix using the fine space and afterwards we apply the matrix H from the left and from the right side. The matrix H has to assembled only once for a specific finite element space.

3.5.4. Computational Study

Since no computational experience with the stabilization techniques for higher order elements in three dimensions exists, we perform a numerical study to illustrate the behavior of the proposed scheme in comparison the residual type stabilization schemes PSPG/SUPG presented e.g. by Hughes et al. [HFM86], [BH82].

We are mainly interested in the robustness regarding the accuracy as well as the solver's behavior with respect to the choice of the stabilization parameters. Further we measure the numerical effort in terms of computational time. As model problem we use the already described Benchmark "laminar flow around a cylinder" and measure the error of the drag coefficient. For this specific configuration the influence of the convective terms is rather small, a loss of accuracy due to the stabilization term based on the Taylor-Hood element is not observed. Table 3.2 demonstrates the accuracy depending on different choices of the stabilization parameters for a fixed mesh with about 1 000 000 degrees of freedom. As one can see, the errors dependency on the pressure stabilization parameter α is very small. Depending on δ, i.e. the convection stabilization, the error in the drag value changes its sign. Needless to say, while passing zero, the error is reduced. Nevertheless, on "both sides" of zero, the absolute value of the error is rather the same. The residual bases stabilization technique shows a severe dependence of the accuracy on the correct choice of the parameters. For low values of α and δ, the linear systems could not be solved.

δ	0.05	0.1	0.2	0.5	1.
LPS	26	25	23	19	20
PSPG/SUPG	-	-	59	70	74

Table 3.3.: Dependency of the linear solving on the choice of the stabilization parameters. The pressure stabilization parameter is fixed to $\alpha = 0.1$, δ is changing. The calculations are done on a structured mesh with 983 040 degrees of freedom. Missing values in the lower line indicate severe solver problems with bad or even no convergency.

Table 3.3 gives the number of linear iterations necessary to solve the linear problems up to a given tolerance. While we keep a fixed value of the pressure stabilization parameter $\alpha = 0.1$, we vary the convection stabilization parameter δ. For a large range the number of linear steps necessary is almost constant regarding the local projection stabilization. As already mentioned, the residual method was not able to deliver results for all choices of α, furthermore, the number of steps necessary is much bigger and increasing with the stabilization parameter α.

In addition to different convergence behavior of the linear solver, we expect the local pressure stabilization to involve far less numerical effort. Since the LPS stabilized $Q_2 - Q_2$ results in the same matrix stencil as the PSPG/SUPG $Q_2 - Q_2$ element, all differences are connected to the integration of the residuals and the system matrix. In particular the evaluation of second derivatives on cells which differ from parallelograms is costly, since it requires evaluation of derivatives of the inverse transformation. However the correct evaluation of all derivatives is essential to obtain the optimal accuracy of the discretization. In Table 3.4 we sum up the times needed for residual and matrix assembling in three dimensions. Table 3.5 gathers the computational cost for the three-dimensional benchmark problem. The number of nonlinear as well as linear steps is the same for both stabilization techniques, i.e. the differences only occur because of the numerical integration. Further we give the discretization errors for LPS, PSPG/SUPG and for PSPG/SUPG with reduced second derivatives. For this last method we neglect all parts of the second derivatives belonging to nonlinearities in the transformation from the unit cell to the computational cell. In the following we denote by φ the basis functions on the computational cell K and by $\widehat{\varphi}$ the basis functions on the reference cell $\widehat{K}$, with $T : \widehat{K} \to K$. Instead of using the correct second derivatives given by

$$\nabla^2 \varphi = D^2 T^{-1} \widehat{\nabla} \hat{\varphi} + (DT^{-1})^2 \widehat{\nabla}^2 \hat{\varphi},$$

we assume DT^{-1} to be constant and the evaluation of the second derivatives simplifies to

$$\nabla^2 \varphi \approx (DT^{-1})^2 \widehat{\nabla}^2 \hat{\varphi}.$$

Although the cells in the triangulation are nearly parallelograms, a large lack in accuracy occurs, whereas the computational cost cannot be significantly decreased with this simplified approach.

	dof's per second		
dof's	LPS	PSPG/SUPG	PSPG/SUPG (*)
matrix	7 900	2 000	2 100
residual	23 000	18 000	21 000

Table 3.4.: Integration effort for the assembly of the system matrix and the residual. Values are given for a three dimensional problem as treated dof's per second since there is a linear dependency between the effort and the problem size. Pentium IV, 2800 MHz. (*) PSPG/SUPG with simplified second derivatives (assumption of linear transformation).

	LPS		PSPG/SUPG		PSPG/SUPG (*)	
dof's	error	time(s)	error	time(s)	error	time(s)
81,231	$2.41e^{-2}$	61	$1.99e^{-2}$	158	$4.81e^{-2}$	121
312 342	$5.13e^{-4}$	212	$5.51e^{-4}$	623	$9.41e^{-3}$	523

Table 3.5.: Comparison of numerical effort for LPS and PSPG/SUPG. $Q_2 - Q_2$ elements applied to the 3D benchmark flow around a cylinder. (*) PSPG/SUPG with simplified second derivatives (assumption of linear transformation).

3.5.5. LPS based on the $Q_2/$ iso Q_2 element

For problems with dominant convection, the projection into a space of one degree less will result in a loss of one order of accuracy. For the velocity we will only get the accuracy of the coarse space. Considering the pressure stabilization the projection into a space of lower polynomial degree does not reduce the order, since one degree less in the pressure accuracy is already optimal. Using the subspace

$$\tilde{V}_h = \{v \in [H^1(\Omega)]^{dim},\ v|_P \in [Q^2(P)]^{dim}, \forall P \in \mathcal{T}_{2h}\},$$

instead of (3.15) we get the correct order for the velocities. The implementation of the corresponding fluctuation operator $\pi := id - i_{2h}$ is performed analog to the discussion in Section 3.5.3, now a local matrix describes the representation of the basis functions $\Phi \in \tilde{V}_h = V_{2h}$ with help of the basis functions $\varphi \in V_h$. For a detailed discussion we refer to [BR05c].

In Table 3.6 we give a comparison of both fluctuation operators with respect to the approximation properties. As test-case we use the two dimensional simplification of the flow around a cylinder (see [ST96] or Chapter 1) at Reynolds numbers $Re = 20$ and $Re \approx 66$. The stabilization parameters α_0 and δ_0 are chosen as 0.2 for both cases. As indicated, for low Reynolds numbers the projection into the equal order space is not preferable in terms of accuracy. However, if the influence of the convection gets larger, accuracy is lost if we project into a subspace of lower degree. While we get a reduction factor of approximately 16 – indicating fourth order convergency of the drag evaluation – using the $Q_2/$ iso Q_2 element and for both projection types in the low Reynolds number case, the reduction rate

	$Re = 20$		$Re = 66$	
cells	$Q_2/Q1$	$Q_2/$ iso Q	$Q_2/Q1$	$Q_2/$ iso Q
640	$6.87 \cdot 10^{-3}$	$7.57 \cdot 10^{-3}$	$3.60 \cdot 10^{-2}$	$2.49 \cdot 10^{-2}$
2 560	$1.38 \cdot 10^{-3}$	$1.10 \cdot 10^{-3}$	$1.49 \cdot 10^{-2}$	$1.56 \cdot 10^{-2}$
10 240	$6.72 \cdot 10^{-5}$	$6.27 \cdot 10^{-5}$	$1.14 \cdot 10^{-3}$	$9.41 \cdot 10^{-4}$
40 960	$4.38 \cdot 10^{-6}$	$3.59 \cdot 10^{-6}$	$1.33 \cdot 10^{-4}$	$3.11 \cdot 10^{-5}$

Table 3.6.: Comparison between LPS with projection into space of lower degree and projection into equal order space on a coarser mesh. For two different Reynolds numbers 20 and 66 we list the relative errors in the drag evaluation on a sequence of globally refined meshes for the 2d Navier-Stokes benchmark.

for $Re = 66$ only yields a value of about 10 considering the projection into the lower order subspace.

The projection into a patched space of the same order requires a larger stencil in the system matrix. In three spacial dimensions using triquadratic finite elements, one patch includes 2 744 matrix couplings instead of 512 couplings necessary for the Galerkin part (or the stabilization based on the Taylor-Hood element). For this calculation one has to treat the different kind of unknowns separately: in every cell there are 6 unknowns on faces, each of it couples with 45 unknowns and is shared by 2 cells; this leads to $3 \cdot 45 = 135$ couplings per cell by unknowns on faces. Corresponding calculations are added for the unknowns on the edges and corners of each cell.

Thus, local projection stabilization with respect to the $Q_2/$ iso Q_2 element significantly increases the memory usage. Furthermore, patched meshes complicate the possibility of generating efficient locally refined meshes. This will be discussed in the next chapter. We can produce relief by applying the projection into a lower degree space as a preconditioner for the linear systems. This allows the usage of meshes without patches and reduces the memory demand. For the Newton residual the projection into the equal order space is utilized.

Conclusion

The local projection method for stabilizing the pressure coupling as well as convective terms seems to be a promising alternative to the standard residual methods PSPG/SUPG. The numerical examples confirm the advantages already mentioned: no additional couplings are introduced between different solution components, stabilization of pressure and of velocity are separated. Due to the easy structure an efficient implementation is possible which leads to a large reduction of the computational effort.

LPS is superior to the residual methods in terms of the robustness of the linear solver as well as regarding the robustness of the discretization error with respect to the choice of the stabilization parameters α_0 and δ_0.

In Chapter 6 where reactive flows will be considered, the separated structure of the local projection method will be of fundamental importance. Because no artificial couplings between different chemical species will be fed into the equations, we will be able to use special storage techniques which allow for a considerable reduction of the memory usage.

4. Error Estimation & Mesh Adaptation

The discussed adaptive finite element methods aim at efficiently calculating functional values of the solution, in our case the drag coefficient, the lift coefficient and the pressure drop at the obstacle. The basis of the adaptive approach is the dual-weighted-residual method (DWR) introduced in 1995 by Becker and Rannacher [BR96] and since then successfully applied to various problems such as parameter identification [BV04], chemically reactive flows [Bra98], optimization [Bec01], nonconforming finite elements and seceral others.

The new aspect in this work considering the error estimation and adaption process is the rigorous application to three dimensional problems with finite elements of degree two.

In this section we present our approach for a posteriori error control for output functionals and quadratic finite elements as well as our algorithm for mesh adaptation. The basis is the standard approach of dual weighted residuals by Becker & Rannacher [BR96, BR01], in which a dual solution is used for computing weights entering in the estimator. The focus of this section is on the approximation of the weights using second order finite elements and on an algorithm, to adapt the mesh. First we derive the error estimator and discuss implementational aspects. Afterwards we describe methods for mesh adaption.

4.1. Dual Weighted Residual Method

The aim is to get an a posteriori error estimate η for the discretization error measured in an output functional:

$$\eta \approx j(u) - j(u_h).$$

The error estimator is derived with a Lagrangian ansatz: minimize the error $j(u) - j(u_h)$, while the equation serves as a constraint: $a(u)(\psi) = f(\psi)$ for all $\psi \in X$. The Lagrange functional

$$\mathcal{L}(u, z) = j(u) - f(z) - a(u)(z)$$

inherits the Lagrangian multiplier $z \in X$ and yields the following essential conditions:

$$\mathcal{L}'_u(u, z)(\delta u) := j'(u)(\delta u) - a'_u(u)(\delta u, z) = 0, \quad \forall \delta u \in X, \tag{4.1}$$

$$\mathcal{L}'_z(u, z)(\delta z) := f(\delta z) - a(u)(\delta z) = 0, \quad \forall \delta z \in X. \tag{4.2}$$

Equation (4.2) equals the original equation to be solved, whereas we introduce an additional *dual problem* with equation (4.1). Both equations (4.1) and (4.2) have to be solved in the appropriate discretized space X_h:

$$\mathcal{L}'_u(u_h, z_h)(\delta u_h) := j'(u_h)(\delta u_h) - a'_u(u_h)(\delta u_h, z_h) = 0, \quad \forall \delta u_h \in X_h, \tag{4.3}$$

$$\mathcal{L}'_z(u_h, z_h)(\delta z_h) := f(\delta z_h) - a(u_h)(\delta z_h) = 0, \quad \forall \delta z_h \in X. \tag{4.4}$$

For solutions $(u, z) \in X \times X$ and $(u_h, z_h) \in X_h \times X_h$ we have the identity

$$j(u) - j(u_h) = \mathcal{L}(u, z) - \mathcal{L}(u_h, z_h).$$

To be precise, this identity even holds in the following way:

$$j(u) - j(u_h) = \mathcal{L}(u, \psi) - \mathcal{L}(u_h, \psi_h), \quad \forall \psi \in X \text{ and } \forall \psi_h \in X_h. \tag{4.5}$$

In the following we combine the *primal* and the *dual* solutions $x = (u, z)$ and $x_h = (u_h, z_h)$ and denote the error by $e^x = x - x_h$, the error identity (4.5) can be rewritten as

$$j(u) - j(u_h) = \int_0^1 \mathcal{L}'(x_h + se^x)(e^x)\,\mathrm{ds}$$

which we approximate with the trapezoidal rule

$$j(u) - j(u_h) = \frac{1}{2}\mathcal{L}'(x_h)(e^x) + \frac{1}{2}\mathcal{L}'(x)(e^x) + \mathcal{R}_h^{(3)}, \tag{4.6}$$

with the remainder term

$$\mathcal{R}_h^{(3)} = \frac{1}{2}\int_0^1 \mathcal{L}'''(x_h + se^x)(e^x, e^x, e^x)s(s-1)\,\mathrm{ds},$$

which is cubic in the error if $\mathcal{L}$ is three times differentiable. Due to Galerkin orthogonality the middle part of (4.6) $\frac{1}{2}\mathcal{L}'(x)(e^x)$ is zero for stationary points x, x_h. Denoting the residuals of the discretized primal equation (4.4) and the dual equation (4.3) by $\rho(\cdot)(\cdot)$ and $\rho^*(\cdot,\cdot)(\cdot)$ respectively,

$$\begin{aligned} \rho(u_h)(\varphi_h) &= f(\varphi_h) - a(u_h)(\varphi_h) \\ \rho^*(u_h, z_h)(\varphi_h) &= j'(u_h)(\varphi_h) - a'_u(u_h)(\varphi_h, z_h) \end{aligned} \tag{4.7}$$

we get from the error representation (4.5) using (4.3, 4.4) for all $y_h \in X_h$

$$\begin{aligned} j(u) - j(u_h) &= \frac{1}{2}\mathcal{L}'(x_h)(x - y_h) + \mathcal{R}_h^{(3)} \\ &= \frac{1}{2}\rho(u_h)(z - \varphi_h) + \frac{1}{2}\rho^*(u_h, z_h)(u - \psi_h) + \mathcal{R}_h^{(3)}. \end{aligned}$$

Especially for any interpolation operator $i_h : X \to X_h$ we get with

$$j(u) - j(u_h) = \frac{1}{2}\rho(u_h)(z - i_h z) + \frac{1}{2}\rho^*(u_h, z_h)(u - i_h u) + \mathcal{R}_h^{(3)}, \tag{4.8}$$

an error representation that only depends on the residuals of the two equations tested with some interpolation errors. A computable error estimator which differs from the error representation (4.8) only in higher order is achieved if we can approximate the interpolation errors of the continuous solutions $u \in X$ and $z \in X$:

$$j(u) - j(u_h) \approx \eta = \frac{1}{2}\rho(u_h)(\widehat{z - i_h z}) + \frac{1}{2}\rho^*(u_h, z_h)(\widehat{u - i_h u}). \tag{4.9}$$

If we can approximate $\widehat{x - i_h x}$ only with knowledge of the discrete solution x_h all the constituent parts of the error estimator η are known with the additional effort of computing the discrete dual solution (one auxiliary linear problem).

A well established way of approximating the interpolation error is the comparison of the discrete solutions u_h and z_h with their interpolation to a higher order space on a coarser mesh: $i_{2h}^* : V_h \to V_{2h}^*$.

$$u - i_h u \approx u_h - i_{2h}^* u_h.$$

If the triangulation possesses the patch structure denoted in Chapter 2 this interpolation is easily established. A detailed comparison for different kind of approximation techniques considering linear finite element spaces is e.g. found in Richter [Ric01]. However using for example second order finite elements the technique mentioned above would require the usage of a fourth order reconstruction for approximating the interpolation error. The associated numerical integration cost is very high. In addition this procedure requires a patch structure of the mesh. With increasing order of the finite element spaces, mesh refinement has to be "more locally" to give optimal complexity results. A very accurate refinement cannot be realized with a patch structured mesh. This aspect is discussed in more detail in the following section. It is not possible to use a simple equal order approximation of $z - i_h z$ since due to the Galerkin orthogonality the residuals $\rho(u_h)(\varphi_h)$ and $\rho^*(u_h, z_h)(\varphi_h)$ vanish for all $\varphi_h \in V_h$.

4.2. Error Estimation with Q_2 Elements

If we are mainly interested in quantities usable for mesh refinement we apply a simplified version of the error estimator. Details are found in Becker & Rannacher [BR01]:

Lemma 4.1. *For the Galerkin approximation of the Euler-Lagrange system (4.1, 4.2), we have the a posteriori error representation*

$$j(u) - j(u_h) = \rho(u_h)(z - \varphi_h) + \mathcal{R}_h^{(2)},$$

with the residual $\rho(u_h)(\cdot)$ defined in (4.7). The remainder term $\mathcal{R}_h^{(2)}$ is of second order in $(x - x_h)$ and vanishes if $a(\cdot)(\cdot)$ and $j(\cdot)$ are linear.

We have lost on order of magnitude in the remainder term, but the error representation is far easier, since we only need to evaluate the primal residual.

The traditional way of evaluating error estimators of residual type is to apply partial integration and use the strong operator form of the equation, see e.g. Becker & Rannacher [BR96]

for the dual weighted residuals methods or Verfürth [Ver96] for other error estimators:

$$\begin{aligned} j(u) - j(u_h) &= \rho(u_h)(z - i_h z) + \mathcal{R}_h^{(2)} \\ &\approx \rho(u_h)(z - i_h z) \\ &= f(z - i_h z) - a(u_h)(z - i_h z) \\ \Rightarrow |j(u) - j(u_h)| &\leq \left| \sum_{K \in \mathcal{T}_h} (f - \mathcal{A}u_h, z - i_h z)_K + (\mathcal{E}u_h, z - i_h z)_{\partial K} \right|, \\ &\leq c_{cs} \sum_{K \in \mathcal{T}_h} \left\{ \|f - \mathcal{A}u_h\|_K + \frac{1}{2} h_K^{-\frac{1}{2}} \|[\mathcal{E}u_h]\|_{\partial K} \right\} \omega_K, \\ \omega_K &= \max\{\|z - i_h z\|_K, h_K^{\frac{1}{2}} \|z - i_h z\|_{\partial K}\}. \end{aligned}$$

where $\mathcal{A}$ is the strong operator and $\mathcal{E}$ are edge remainder terms appearing since the derivatives of u_h are not steady across the edges. The constant c_{cs} is caused by the Cauchy-Schwarz inequality. Since this constant is unknown we cannot expect $|j(u) - j(u_h)|/\eta \to 1$ for decreasing h. Therefore we don't get a real error estimator but cell-wise error indicators:

$$\eta_K = \left\{ \|f - \mathcal{A}u_h\|_K + \frac{1}{2} h_K^{-\frac{1}{2}} \|[\mathcal{E}u_h]\|_{\partial K} \right\} \max\{\|z - i_h z\|_K, h_K^{\frac{1}{2}} \|z - i_h z\|_{\partial K}\}. \quad (4.10)$$

However as a benefit of this simplified method we can apply approximations for $\|z - i_h z\|$ which lie in the finite element space V_h. In addition, practical experience justifies the possibility of separating error estimation and obtaining error indicators for mesh adaption.

In the following we compare different approximation techniques for the interpolation error. All of them rely on super-approximation properties where theoretical results are missing for general type of equations and locally refined meshes. We separate two approaches for approximating the interpolation error. First we consider possibilities with $\widehat{u - i_h u} \notin V_h$, i.e. approximations that are suitable for application of the error identity (4.9). Second we introduce approximations of the norm of the interpolation error $\|u - i_h u\|_K$.

One possibility for estimating the interpolation error would be to compute the discrete function u_h in a higher order finite element space $u_{2h}^* \in X_{2h}^*$ on a coarse mesh and approximate the interpolation error by

$$u - i_h u \approx u_{2h}^* - i_h^* u_{2h}^*, \quad i_h^* : X_{2h}^* \to X_h. \quad (A1)$$

This approximation should deliver very good results but requires an even higher effort for the error estimator than for the solution of the problem and will therefor not be considered. As in the first order case a higher order reconstruction of the discrete solution is possible:

$$u - i_h u \approx i_{2h}^* u_h - u_h, \quad i_{2h}^* : X_h \to X_{2h}^*. \quad (A2)$$

This does not require the solution in the higher order space but it requires the numerical integration of higher order functions as well as the wasteful patch structure. The numerical integration effort is of order $O(p^d)$ if p is the order of the ansatz space and d the spacial

dimension, i.e. integration of a higher order function is eight times more expansive. Another approach to approximate the interpolation error is to compare the discrete solution u_h not with a better but with a slightly less accurate one:

$$\|u - i_h u\| \approx \|u_h - i_{2h} u_h\|, \quad i_{2h} : X_h \to X_{2h}. \tag{A3}$$

This equal-order interpolation to the patch is also element of the original ansatz space V_h, therefore this approximation is only feasible considering the simplified estimator (4.10). The interpolation to the coarser space X_{2h} could also be substituted with the calculated solution on this coarser mesh. This solution is in general available since we use a geometric multigrid solver for the solution of our problems. But again, this procedure requires the patched mesh.

A completely different approach to the approximation would be the use of an interpolation error estimate (for piecewise polynomials of order p) $\|u - i_h u\|_K \leq c_I h_K^{p+1} \|\nabla^{p+1} u\|_K$ and try to guess the high derivatives $\nabla^{p+1} u$. A major drawback of this method is the introduction of the unknown interpolation constant c_I. Further this interpolation estimate is only valid if the function u contains enough regularity properties. Considering problems with entering edges, the solution of the Navier-Stokes equation is not even in $H^2(\Omega)$. But it is possible to gain knowledge of $\nabla^{p+1} u$ by a reconstruction process without utilizing a higher order approximation or requiring the patch structure of the mesh.

Using finite elements of degree p we get an approximation for the p-th derivatives:

$$\nabla^p u_{|K} \approx \nabla^p u_{h|K} =: q_K.$$

This discontinuous function q_K is element of some space $V_h^{*,disc}$ which contains at least the cell-wise constant functions $P_h^{0,disc} \subset V_h^{*,disc}$. The function q_K is a tensor of dimension d^p. We now apply a projection of q_K into our original finite element space $[V_h]^{d^p}$:

$$g_h \in [V_h]^{d \times d \times \cdots \times d} : \quad (g_h - q_K, \varphi) = 0, \quad \forall \varphi \in V_h.$$

The gradient of the tensor g is an approximation for the searched $\nabla^{p+1} u$:

$$\|u - i_h u\| \approx h_K^{p+1} \|\nabla g_h\|. \tag{A4}$$

The projection can be replaced by a node-wise averaging of the unsteady values of $\nabla^p u_h$:

$$(g_h)_i := \sum_{K \in \mathcal{N}(i)} q_K(x_i).$$

This algorithm can be arranged without the patch structure and without higher order integration. On the other hand, the dimension of the tensor of the p-th derivatives is d^p which results in a comparable integration effort, which is even dominant for really high order spaces $p >> d$.

A simular approach with a local balancing of the gradient is used in the error estimator of Zienkiewicz & Zhu [ZZ87].

All theoretical results concerning the different approximation techniques are all based on super-approximation properties which asymptotically require meshes with patch-size of order $O(1)$. This conflicts with the necessity of accurate local mesh refinement. From there we

compare the presented methods in a computational study with respect to the approximation of the interpolation error, the numerical cost and particularly the usability for error estimation and mesh refinement. We won't consider the first possibility (A1) utilizing a higher order solution since the aligned numerical effort is not expected to be worthwhile.

As a first test we measure the approximation qualities of interpolation error by methods (A3) and (A4). We approximate a function with singularities in the derivatives

$$u(x, y, z) = \sqrt{x + y + z}, \tag{4.11}$$

of the unit box $\Omega = [0, 1]^3$. This function is not in $H^2(\Omega)$ but in some $H^{2-\alpha}(\Omega)$ with α small. On a sequence of meshes we compare the interpolation error $\|u - i_h u\|_\Omega$ with its coarse mesh approximation (A3) $\|u_h - i_{2h} u_h\|_\Omega$ and the interpolation estimate (A4) $h\|\nabla_h^3 u_h\|_\Omega$. In Table 4.1 we compare the absolute values of the (approximated) interpolation error as well as the cell-wise approximation quality $q^{(\mathrm{A})} := \|u - i_h u\|_K / \widehat{\|u - i_h u\|}_K^{(\mathrm{A})}$.

		(A3)		(A4), $c_I = 10^{-2}$	
# cells	$\|u - i_h u\|_\Omega$	$\|u_h - i_{2h} u_h\|$	$q^{(\mathrm{A3})}$	$c_I h^3 \|\nabla_h^3 u_h\|$	$q^{(\mathrm{A4})}$
8	$3.56 \cdot 10^{-3}$	$4.13 \cdot 10^{-3}$	$q \in [0.86, 0.86]$	$3.59 \cdot 10^{-3}$	$q \in [0.91, 0.91]$
64	$8.92 \cdot 10^{-4}$	$11.9 \cdot 10^{-4}$	$q \in [0.05, 0.81]$	$8.42 \cdot 10^{-4}$	$q \in [0.14, 1.08]$
512	$2.23 \cdot 10^{-4}$	$3.01 \cdot 10^{-4}$	$q \in [0.02, 0.80]$	$2.10 \cdot 10^{-4}$	$q \in [0.05, 1.08]$
4096	$5.58 \cdot 10^{-5}$	$7.73 \cdot 10^{-5}$	$q \in [0.01, 0.80]$	$5.25 \cdot 10^{-5}$	$q \in [0.02, 1.30]$

Table 4.1.: Approximation of the interpolation error with methods (A3) and (A4). We compare the overall approximation of the interpolation error as well as the range of the cell-wise approximation property.

As expected the approximation of the interpolation error via the coarse mesh interpolation (method A3) is quite good. Using the a priori unknown interpolation constant $c_I = 10^{-2}$ the overall approximation with method (A4) is even better. The good approximation property of method (A4) is perhaps misleading since this result depends on the right choice of the interpolation constant c_I. If we e.g. consider the smooth function

$$u(x, y, z) = \sin(\pi x) \sin(2\pi y) \sin(\pi z), \tag{4.12}$$

we get a simular result for method (A3), but we have to choose the interpolation constant as $2c_I \approx 10^{-2}$ to get a comparable result. The cell-wise values for the coarse mesh approximation are very accurate, we get $q \in [0.38, 0.45]$ for a mesh with 4096 cells using the smooth function.

Since we cannot detect a crucial benefit of one method we will further only consider method (A4) with the major advantage of not requiring the patch structure. Next we analyze the usability of the different approximation techniques for error estimation and for mesh adaption. In a first test we consider a simple model problem and test the ability of the methods for error estimation. Finally we apply the methods for mesh adaption on a three dimensional Navier-Stokes Flow.

Now we approximate the smooth function (4.12) as the solution of the Laplace equation

$$-\Delta u = f,$$

with f chosen appropriate. On a sequence of meshes we compare the numerical effort and the effectivity $\eta/|j(u)-j(u_h)|$ of two error estimators using (A2) and (A4) for approximating the weights. We do not regard method (A1) since the aligned numerical effort is to high, further we neglect method (A3) since is needs the patch structure without producing superior results compared to (A4). In Table 4.2 we list for a sequence of meshes the error, and the effectivity of the error estimator. The interpolation constant for method (A4) is set to $c_I = 10^{-4}$. The comparison is done for methods (A2) $(u_h - i^*_{2h}u_h)$ and (A4) $(h_K^3\|\nabla_h^3 u_h\|_K)$:

		(A2)	(A4)
cells	$\lvert j(u) - j(u_h)\rvert$	eff	eff
8	$1.10 \cdot 10^{-1}$	3.01	1.72
64	$2.01 \cdot 10^{-2}$	12.22	1.28
512	$1.05 \cdot 10^{-3}$	0.95	1.33
4.096	$6.16 \cdot 10^{-5}$	1.24	0.95
32.768	$3.79 \cdot 10^{-6}$	1.10	0.54

Table 4.2.: Effectivities for the error estimator with approximation methods (A2) and (A4). The solution is smooth: $u \in C^\infty(\Omega)$.

The results in Table 4.2 first of all indicate the lack of accuracy due to the Cauchy-Schwarz inequality. Although the approximation quality of the interpolation error with method (A4) is promising the method is not useful for error estimation. The effectivity for method (A2) is o.k.

Finally we compare the usability of both methods for mesh adaption applied to a three dimensional Navier-Stokes flow. We consider a flow in long direction through a channel of size $15 \times 2 \times 7$ with an obstacle of size $1 \times 1 \times 1$ fixed on the bottom of the channel. The quantity of interest is the drag coefficient of the obstacle. Due to the entering edges this problem involves singularities in the derivatives of the solution.

In Table 4.3 the achieved error, the effectivity and the numerical cost of the error estimator is listed for the application of methods (A2) and (A4). At first glance both methods deliver good results for adaptive mesh refinement. The effectivity of the estimator considering (A4) is subject to the discussed restrictions. Regarding this problem the trouble with patched meshes gets obvious. The calculation using method (A2) starts on a mesh with 1672 cells which equals 63540 degrees of freedom. After only two steps of local refinement the memory requirements of the system matrix exceeds a small workstation (half a gigabyte). Method (A4) is suitable for producing more economical meshes.

Conclusion

Using local recovery of derivative information, high qualitative mesh adaption is possible. However to ensure error estimation with good effectivities, the usage of Cauchy-Schwarz inequality has to be prevented. As discussed, evaluation methods with higher accuracy

(A2)

cells	$\lvert j(u) - j(u_h)\rvert$	eff	cost
1.672	$6.19 \cdot 10^{-2}$	0.03	0.11
5.144	$3.90 \cdot 10^{-3}$	2.81	0.14
17.464	$2.12 \cdot 10^{-3}$	1.31	0.15

(A4)

cells	$\lvert j(u) - j(u_h)\rvert$	eff	cost
209	$6.94 \cdot 10^{-1}$	0.27	0.08
832	$6.15 \cdot 10^{-2}$	0.84	0.09
2.316	$3.61 \cdot 10^{-3}$	5.80	0.12
6.306	$1.90 \cdot 10^{-3}$	6.51	0.13

Table 4.3.: Effectivity and numerical cost for the error estimator with approximation methods (A2) and (A4) for a Navier-Stokes test on adaptively refined meshes.

are linked with patched meshes which lead to wasteful refinements. But also the analysis of methods without the need of patch-structured meshes reveals a basic limit of adaptive methods based on total bisection of meshes, i.e. refining one cell into 8 small cells. The required amount of memory swells so rapidly that even large parallel computers are overstressed without having resolved singularities, boundary layers or regions of turbulence. For the solution of three dimensional problems with high Reynolds numbers anisotropic refinement seems to be necessary.

Though, the presented method for evaluating the error indicators leads to an efficient way of generating meshes adapted to the problem.

4.3. Mesh Adaption

In the previous section we have derived an a-posteriori error estimator η which has a cell-wise representation:

$$\eta = \sum_{K \in \mathcal{T}_h} \eta_K .$$

We have to choose a subset of cells $S \subset \mathcal{T}_h$ for refinement. Several standard approaches exist for choosing this subset. Throughout this work we use a scheme for adaption which differs from most other methods and was presented in [Ric01]. The cells $K_i \in \mathcal{T}_h$ are ordered with respect to the error indicators:

$$\eta_{K_i} \geq \eta_{K_{i+1}} .$$

The subset of cells to refine is always chosen as coherent queue $S_r = \{K_1, \ldots, K_r\}$. We assume a local convergency of the error indicators after refinement of one cell K as

$$\sum_{K' \in K} \eta_{K'} = \left(\frac{1}{2}\right)^{\alpha} \eta_K . \tag{4.13}$$

If we consider global refinement $\alpha = 2$ stands for quadratic convergence $\eta \leq ch^2$. This cell-wise convergence order α usually is not constant in the whole domain. Near singularities α should be chosen smaller. In [Bra98] it is shown, that under some regularity assumptions on the product of the primal and dual solution if we use optimal mesh refinement and if N denotes the number of nodes of the mesh, p the order of the finite element space and d the dimension, the error behaves as

$$E \approx CN^{-\frac{2p}{d}}.$$

The number r of cells to be refined is calculated as the minimal argument of

$$C(r) = E(r) \cdot N(r)^{\frac{2p}{d}}, \tag{4.14}$$

where $E(r)$ is as prediction of the error on the new mesh after refinement of r cells. This value is approximated using (4.13)

$$E(r) = \sum_{i=1}^{\#\text{cells}} \eta_{K_i} - \sum_{i=1}^{r} \left(1 - \left(\frac{1}{2}\right)^{\alpha}\right) \eta_{K_i}.$$

The value $N(r)$ is a prediction of the number of nodes on the new mesh:

$$N(r) = N + r(2^d - 1).$$

The minimum of (4.14) is determined by testing $E(r)$ with $r = 1, \cdots, N$.

For regular functionals it is known (see [Bra98]), that after an balancing of the cell-wise error indicators, global refinement is optimal. Contrary to this adaptation scheme, the standard approaches for mesh adaptation (fixed fraction, fixed number, ...) do not meet this request:

Remark 4.2. *If the error estimators are equilibrated $\eta_K = \eta_{K'}$ the described adaption process results in global refinement.*

If we fixate $\eta_K = \eta/N$ the resulting function for $E(r)$ has minimal values for $r = 0$ and $r = \#\text{cells}$ leading to global mesh refinement $r = N$, $S_r = \mathcal{T}_h$.

4.4. Numerical Results

In this section the described finite element method is applied to the 3D Navier-Stokes benchmark problem already given in Chapter 1, Figure 1.1. The results are taken from Braack & Richter [BR05a].

Although this problem has been formulated a couple of years ago accurate reference solutions are only determined for the regular geometry by John [Joh02]. A reference solution for the square cross-section is still missing. The entering edges bring about singularities in the derivatives of the solution which weaken the benefit of higher order finite elements. The use of adaptive mesh refinement combined with higher order finite elements is a promising way to handle this problem. Nevertheless the dimension of the discrete three-dimensional

#cells	#dof	Δp	c_{drag}	c_{lift}
9 360	98 128	0.1482	5.8431	6.14e-3
75 776	771 392	0.1605	5.9731	5.95e-3
606 208	6 116 608	0.1672	6.1043	7.92e-3

Table 4.4.: Values obtained by Schreiber [Sch96] with the Q_1^{rot} element for the circular cross-section.

#cells	#dof	Δp	c_{drag}	c_{lift}
9 440	97 736	0.1590	7.3069	3.48e-2
75 520	768 544	0.1683	7.5622	5.03e-2
604 160	6 094 976	0.1729	7.6138	6.00e-2

Table 4.5.: Values obtained by Schreiber [Sch96] with the Q_1^{rot} element for the square cross-section.

problems swiftly gets very large, therefore we further have to utilize parallel computers to obtain reliable reference solutions.

The quantities of interest in this benchmark problems where the drag value of the obstacle, its lift value and the pressure drop in two points adjacent to the obstacle.

In 1996, Schreiber [Sch96] – whose results are also published in Schäfer & Turek [ST96] – did an extensive study of the benchmark using the Rannacher-Turek element [RT92] with rotated trilinear velocities and piecewise constant pressure. In Tables 4.4 and 4.5 results for the drag value the lift and the pressure difference of both configurations are given. Although about 6 million unknowns were used, at most one digit can be assured, regardless of the specific configuration and the considered functional. At the time of this study, the values obtained by Schreiber can be regarded as the summit in terms of accuracy and effort (measured in number of unknowns). Nevertheless, the three dimensional benchmark was not considered to be managed.

Almost all participants of the study published by Schäfer & Turek utilized finite elements of degree one or finite volume discretizations. In the following we will apply the stabilized $Q_2 - Q_2$ element on the benchmark.

One achieves a higher order of accuracy, if the drag (as well as the lift) is not directly evaluated as the boundary integral

$$c_D = C \int_S \left(\nu \frac{\partial v_t}{\partial n} n_y - p n_x \right) \, \mathrm{ds},$$

instead transformed to an integral over the complete domain. With the tangential direction

$t = (n_y, -n_x, 0)$ on the obstacle, we get

$$\begin{aligned}\frac{\partial v_t}{\partial n} n_y =& \frac{\partial v_x}{\partial n} n_y^2 - \frac{\partial v_y}{\partial n} x_n n_y = \frac{\partial v_x}{\partial n} - n_x \left(\frac{\partial v_x}{\partial n} n_x - \frac{\partial v_y}{\partial n} n_y \right) \\ =& \frac{\partial v_x}{\partial n} - n_x \mathrm{div}\, v = \frac{\partial v_x}{\partial n}.\end{aligned}$$

Now if $u = (p, v)$ is the strong solution of the problem and with some test function $\hat{\Phi} = (\hat{\varphi}_0, \hat{\varphi}_1, \hat{\varphi}_2, \hat{\varphi}_3) \in [H^1(\Omega)]^4$, $\hat{\varphi}_i = 0$ for $i \neq 1$ we get

$$\begin{aligned}a(u)(\Phi) =& (\nu \nabla v_x, \nabla \hat{\varphi}_1) + (v \cdot \nabla v_x, \hat{\varphi}_1) - (p, \partial_x \hat{\varphi}_1) \\ =& (-\nu \Delta v_x + v \cdot \nabla v_x + \partial_x p, \hat{\varphi}) + \int_{\partial\Omega} \left(\nu \frac{\partial v_x}{\partial n} - p n_x \right) \hat{\varphi}_1 \, \mathrm{ds} \\ =& \int_{\partial\Omega} \left(\nu \frac{\partial v_x}{\partial n} - p n_x \right) \hat{\varphi}_1 \, \mathrm{ds}.\end{aligned}$$

If we take as test function for the horizontal velocity component $\hat{\varphi}_1|_{\partial\Omega/S} = 0$ and $\hat{\varphi}_1|_S = 1$ we obtain the drag coefficient as

$$a(u)(\hat{\varphi}) = c_{drag}.$$

This property can be used numerically by taking $\hat{\varphi}_1$ as a quadratic test function with Dirichlet values on $\partial\Omega$ as mentioned above:

$$a(u_h)(\hat{\varphi}_h) = c_{drag,h}.$$

For the obstacle with the circular cross-section it turns out that – using quadratic finite elements on uniform meshes – the error in the drag is evaluated as

$$c_{drag} - c_{drag,h} = O(h^4).$$

Details are given in Braack & Richter [BR05a].

For the obstacle with the circular cross-section (configuration 1), reference solutions are published in John [Joh02]. In that work, several discretizations are compared with respect to accuracy. Following the conclusion of the author, the most accurate finite element discretization for this problem under their consideration consists of piecewise tri-quadratic elements for the velocities (Q_2) and discontinuous piecewise linear pressure (P_1^{disc}). Their computation has been performed on structured meshes by the use of a parallel computer. The obtained values on the finest meshes for $Q_2 - P_1^{\mathrm{disc}}$ and extrapolated values are recapitulated in Table 4.6. Their finest mesh contains about 56 million of degrees of freedom (dof). The underlined digits in Table 4.6 are those ones which can be considered as reliable based on the convergence history. For the pressure drop, only the first two leading digits are ensured: $\Delta p = 0.17\ldots$. The drag coefficient was the "easiest" quantity to be computed: the first four digits are reliable: $c_{\mathrm{drag}} = 6.185\ldots$. The conjecture due to extrapolation based on the finest meshes is a value of $c_{\mathrm{drag}} = 6.1853\ldots$. If this is true, on the finest mesh also the fifth digit is correct. For the lift coefficient c_{lift}, the first three leading digits are known, $c_{\mathrm{lift}} = 9.40\ldots \cdot 10^{-3}$.

#cells	dof	Δp	c_{drag}	c_{lift}
245 760	7 035 840	0.170403	6.185234	9.40479e-3
1 966 080	55 666 560	0.170779	6.185327	9.40122e-3
extrapolated			6.185329	9.40098e-3

Table 4.6.: Reference values published in [Joh02] for the circular cross-section.

#cells	dof	Δp	c_{drag}	c_{lift}
480	18 720	0.188771	6.250365	1.88463e-1
3 840	136 000	0.178702	6.172750	1.02332e-2
30 720	983 040	0.173744	6.184551	9.46862e-3
245 760	7 864 320	0.171999	6.185323	9.43526e-3
1 966 080	62 914 560	0.171342	6.185331	9.40136e-3

Table 4.7.: Values obtained in this work for the circular cross-section on structured meshes.

For the square cross-section (configuration 2), the accuracy of simulations on structured meshes is only reliable up to the first two digits. For instance, the drag coefficient is believed to be $c_{\text{drag}} = 7.7\ldots$ for configuration 2.

In this work, we will apply the presented equal order tri-quadratic finite elements both for velocity and pressure. The stabilization takes advantage of the stable subspace $Q_2 - Q_1$. In order to validate that this discretization is at least as accurate as the established one $Q_2 - P_1^{\text{disc}}$ in [Joh02], we list our values for structured meshes in Table 4.7. The meshes used are exactly the same as those which were used to get the values in Table 4.6. The number of degrees freedom are moderately higher for the equal-order discretization compared to the one with discontinuous pressure. However, the accuracy is pretty much the same. For the pressure drop Δp, the first two digits are reliable. With respect to the drag coefficient, also the fifth digit can now be considered as reliable $c_{\text{drag}} = 6.1853\ldots$, which confirms the extrapolation in [Joh02].

This is a good starting point to analyze the effect of local mesh refinement in order to obtain similar values on meshes with less mesh points. Beyond this, we explore the far more difficult test case of the square cross-section (configuration 2). The solution can be found in a Sobolev space $H^{1+\alpha}(\Omega)$ with $0 < \alpha \ll 1$. Therefore, even for quadratic elements, the convergence order (on globally refined meshes) can be expected only to about $O(h)$. For completeness, in Table 4.8 we also gather the values obtained with stabilized $Q_2 - Q_2$ elements on structured meshes.

A confrontation of the best values obtained by Schreiber (Tables 4.4 and 4.5) with the $Q_2 - Q_2$ element yields a rather crushing defeat of lower order elements. Their results obtained with more than 6 000 000 unknowns are outdone on meshes with about 5 000 cells and 150 000 unknowns.

In the remainder of this section, we will enhance the accuracy by the use of a posteriori error estimation and analyze the described discretization – as well as the parallelization techniques – in more detail.

#cells	dof	Δp	c_{drag}	c_{lift}
78	3 696	0.183495	13.31491	-5.88042e-2
624	24 544	0.208050	8.04450	1.04015e-1
4 992	177 600	0.177370	7.97593	7.82310e-2
39 936	1 348 480	0.176165	7.78785	6.78254e-2
319 488	10 787 840	0.175759	7.75793	6.86428e-2
2 555 904	86 302 720	0.175677	7.76119	6.88130-2

Table 4.8.: Values obtained in this work for the square cross-section on structured meshes.

4.4.1. Adaptive Mesh Refinement

Using adaptive mesh refinement we have two goals in mind: first we aim at obtaining the values for the circular test case (see Table 4.7 with less degrees of freedom, second we want to determine reliable values for the test case concerning the square obstacle. In all cases we saddle at least one step of global refinement onto the adaptive refined meshes in order to validate the values achieved by adaptive refinement. Faulty local refinement could pretend a converging scheme but would result in a wrong limit. If the acquired values do not change significantly under global refinement we can trust the results.

In Tables 4.11, 4.12 and 4.13 the obtained values for the circular test are listed. In each table we separate the calculations done on a workstation from those on a parallel computer. A second parting rule indicates the switch from adaptive refinement to global refinement. In all cases the application of global refinements suggests the correctness of the adaptive scheme. Together with the number of cells, the number of degrees of freedom and the functional value we give the effectivity of the error estimator. This value η_{eff} is calculated as

$$\eta_{\text{eff}} = 10^{-4} \frac{|\eta(u_h, z_h)|}{|j(u) - j(u)|}.$$

In Table 4.9 we compare the necessary number of cells to reach a relative error of 1%, resp. 0.01%. As expected adaptive refinement has a large impact on the pressure drop. Both other test cases deliver a high convergence order on structured meshes. The profit of adaptive refinement is therefore limited. Considering functional values as quantities of interest it often occurs, that the error changes its leading sign. This can lead to very small errors which could be misconstrued as a reliable result of the algorithm. In this cases we did not consider these values for the comparison of adaptive and structured meshes.

	1%		0.01%	
	adaptive	global	adaptive	global
Δp	1 383	245 760	6 556	$>$ 1 966 080
c_{drag}	480	480	22 880	30 720
c_{lift}	21 536	30 720	149 104	1 966 080

Table 4.9.: Number of cells needed to reach a relative error of 1% respectively 0.01% considering the circular cross-section.

The square cross-section is somewhat more suited for adaptive refinement. Due to the entering edges structured meshes reveal a convergence order in the area of $O(h)$. Using adaptive refinement the singularities in the solution can be balanced, the convergence is expected to improve. In Tables 4.14, 4.15 and 4.16 we list the acquired values for all three functionals. Again we indicate the used computer architecture and the refinement strategy. Table 4.10 lists the needed number of cells to reach a relative error of 1% respectively 0.1%. For both error tolerances we get a large benefit from using adaptive refinement.

	1%		0.1%	
	adaptive	global	adaptive	global
Δp	2 255	4 992	17 795	319 418
c_{drag}	5 510	39 936	10 116	2 555 904
c_{lift}	5 776	39 936	131 930	>2 555 904

Table 4.10.: Number of cells needed to reach a relative error of 1% respectively 0.1% considering the square cross-section.

Conclusion

For many "real-life" cases an accuracy of 5% has to be considered as sufficient. Although the described benchmark problem has been published in 1996, a good reference solution was still missing for the square obstacle. Even the relative large bound of 5% was not reached for this simple geometry. With adaptive mesh refinement on quadratic finite elements, a reference value for all three functionals on both geometries could be defined with an error in the region of 0.01%.

This accuracy is far beyond the necessary one, but the benchmark configuration is very simple in comparison to "real-life" geometries, as for instance the methane burner described in Chapter 1.

The discretization presented in the previous chapter is well suited for Navier-Stokes flow, the accuracy is comparable to the $Q_2 - P_1^{\text{disc}}$ element analyzed by John [Joh02] and definitely superior to the Q_1^{rot} element analyzed by Schreiber [Sch96].

The DWR method has proven to be reliable for finite elements of degree two, even if the approximation of the weights is rather crude. The effectivities of the error estimator range between 0.25 and 5 for nearly all settings with a leading constant 10^{-4}.

#cells	dof	Δp	η_{eff}	machine
480	18 720	0.1892506787	2.41	
865	34 080	0.1798815121	2.06	
1 138	45 520	0.1746605518	1.64	
1 383	55 896	0.1729589594	1.32	
6 556	243 224	0.1712704796	0.56	
8 607	321 544	0.1710884110	0.29	
10 742	404 376	0.1710308434	0.13	serial
924 704	32 260 736	0.1710070986		parallel

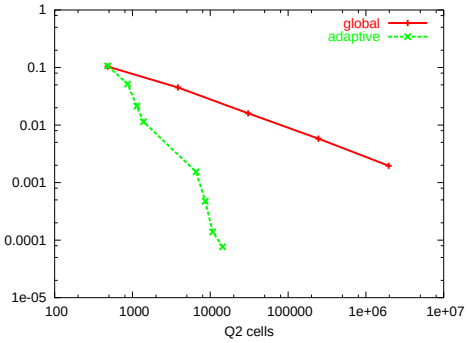

Table 4.11.: Values obtained for the circular cross-section on locally refined meshes with a posteriori error estimate of Δp.

#cells	dof	c_{drag}	η_{eff}	machine
480	18 720	6.2503650	1.32	
3 840	136 000	6.17275342	2.72	
22 880	784 384	6.18471848	0.28	serial
155 768	5 227 872	6.18533571		parallel
1 246 144	41 822 976	6.18533310		
extrapolated		6.18533293		

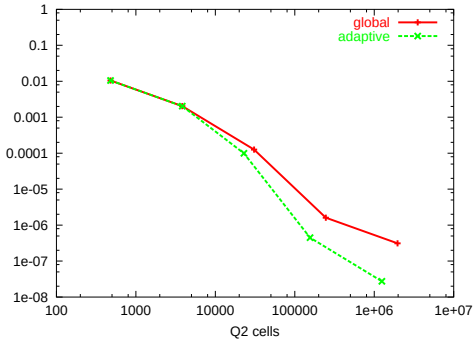

Table 4.12.: Values obtained for the circular cross-section on locally refined meshes with a posteriori error estimate of c_{drag}.

#cells	dof	c_{lift}	η_{eff}	machine
480	18 720	1.884632312e-1	0.05	
3 840	136 000	1.023315808e-2	0.23	
21 536	740 416	9.483555569e-3	0.27	serial
149 104	5 006 304	9.405158225e-3		parallel
1 192 832	40 050 432	9.401228291e-3		
extrapolated		9.40097e-3		

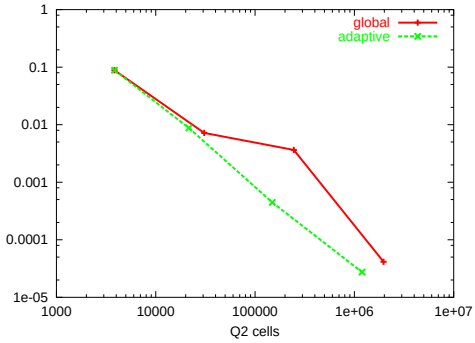

Table 4.13.: Values obtained for the circular cross-section on locally refined meshes with a posteriori error estimate of c_{lift}.

#cells	dof	Δp	η_{eff}	machine
78	3 696	0.183495025	0.21	
624	24 544	0.208049746	3.20	
2 248	83 952	0.177348465	0.49	
5 160	191 424	0.176449284	1.10	
14 008	511 952	0.175792355	0.76	
33 496	1 212 208	0.175713985	0.71	serial
83 000	2 969 936	0.175676527		parallel
664 000	23 759 488	0.175686487		

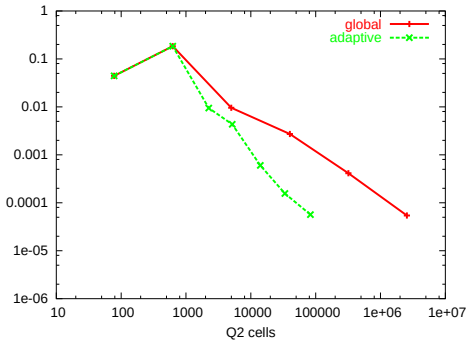

Table 4.14.: Values obtained in this work for the square cross-section on adaptively refined meshes for the pressure drop.

#cells	dof	c_{drag}	η_{eff}	machine
624	24 544	8.044496403	3.89	
2 472	91 120	7.974163476	13.87	
7 120	258 512	7.788092939	6.28	
16 808	615 408	7.759470239	7.53	
113 128	4 052 272	7.765846534		parallel
905 024	32 418 176	7.767272368		

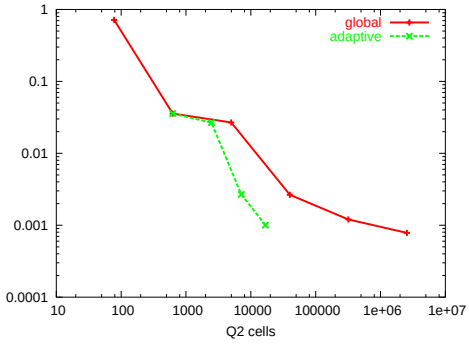

Table 4.15.: Values obtained in this work for the square cross-section on adaptively refined meshes for the drag coefficient c_{drag}.

#cells	dof	c_{lift}	η_{eff}	machine
78	3 696	$5.880420044e^{-2}$	2.59	
435	17 608	$1.040065281e^{-1}$	2.04	
1 611	61 616	$7.794480009e^{-2}$	3.64	
5 643	210 336	$6.750582790e^{-2}$	2.69	
8 506	327 008	$6.818400747e^{-2}$	3.29	
56 764	2 037 464	$6.873510495e^{-2}$		serial
454 112	16 299 712	$6.892755977e^{-2}$		parallel

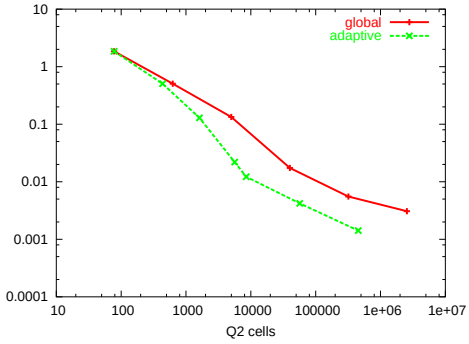

Table 4.16.: Values obtained in this work for the square cross-section on adaptively refined meshes for the lift coefficient c_{lift}.

5. Parallel Adaptive Finite Elements

Parallel methods for solving partial differential equations are well-established techniques. Domain decomposition or Schur Complement methods are inherently well suited for parallelization. Efficient multigrid methods have been ported to parallel computers. As early as 1978, Grosch [Gro78] analyzed multigrid methods with respect to parallelization. The promising approach to implement a multigrid method on a parallel computer is by domain decomposition and was advocated 1981 by Brandt [Bra81]. Each level of the triangulation is partitioned into subdomains and spread to the different processors. This approach is effective as long as the number of unknowns is larger than the number of processors. Since the demand will most certainly always exceed the capacity this "rule" should be satisfied. However the coarse meshes of the multigrid hierarchy will be of size $O(1)$, the number of processors might even outdo the number of unknowns. On coarse meshes, less processors are used (coarse mesh agglomeration), others are idle. As the biggest amount of work is done on the fine meshes, these idle processors will not gravely touch the effectivity.

To overcome the problem of idle processors, modifications of multigrid solvers, as the "concurrent multigrid algorithms" where examined for parallelization. Frederickson and McBryan [FM88] developed a parallel super-convergent multigrid method that creates useful work for idle processors. However, the question of parallel efficiency is not settled.

Considering adaptive mesh refinement, the problem of idle processors gets critical. The usual way to set up the multigrid hierarchy on locally refined meshes using triangulations of subdomains with equal refinement order on the finer meshes to ensure the optimal complexity with regard to the number of unknowns. Using this approach, the number of unknowns may be small compared to the number of processors on all levels of the mesh hierarchy. BPX, a variant of multigrid (see Bramble, Pasciak and Xu [BPX90]) is suitable for adaptive mesh refinement and was parallelized by Bastian [Bas93].

The difficulties of parallel multigrid on adaptively refined meshes are associated with the way of setting up the multigrid hierarchy. Becker & Braack [BB00] proposed a new way of assembling these meshes. On every level, the triangulation covers the whole domain. Coarser meshes are generated by "global coarsening". In the reference mentioned above, optimal complexity was demonstrated, unless strict refinement to one point is applied. For practical problems this method is well suited, and parallelization of standard multiplicative multigrid is possible.

Early implementations of parallel multigrid solvers only consider elliptic problems. Oswald [Osw01] implemented a parallel multigrid solver for the non-stationary Navier-Stokes equations based on Van Kan's projection scheme on structured meshes.

Since parallel adaptive multigrid methods require very complex data and program structures an implementation has to be applicable to a large variety of problems. For example a multigrid solver for the Navier-Stokes equation on anisotropic meshes requires very robust smoothers, e.g. special modification of the ILU decomposition, which are mostly not well suited for parallelization. A detailed examination of a parallel multigrid solver for adaptive meshes in two dimensions is given by Bastian [Bas96]. Bastian analyzed a parallel multigrid solver for scalar linear problems arising from finite volume discretizations in two dimensions. In Bastian [B$^+$99] a parallel multigrid solver for – amongst others – the Navier-Stokes equations in three dimension was analyzed.

The parallelization discussed in this work is based on the finite element toolbox Gascoigne [BB$^+$]. Gascoigne features an adaptive multilevel solver for a large range of problems in two and three dimensions, including the Laplace-(heat-)equation, Stokes, Navier-Stokes, compressible flows and reactive flows. The main components of the toolbox are identical for all problems, particularly there is only one multilevel solver, suitable for all these problems. For example the number of solution components or the choice of the specific equation or discretization does not influence the algorithmical treatment of the linear problems. In contrast to other approaches, the multigrid hierarchy is established by "global coarsening", thus the problem of idle processors is bypassed.

We try to preserve the sequential algorithm as far as possible. Therefore no special parallel techniques for the coarse mesh problem are applied. Actually, the only modification applied to the sequential algorithm is the domain decomposition approach used for the smoothing process – again the standard way of implementing parallel smoothers.

In the remainder of this chapter we give a short introduction to the basic analysis of parallel algorithms, afterwards we describe the algorithmical and implementational aspects of the solving techniques used in Gascoigne. During the description of the methods, we discuss their parallelization. Primary requirement for the parallelization process is the retainment of the original algorithms. Parallelization is always a trade-off between parallel efficiency and numerical quality of the underlying algorithms. Our focus is on the latter issue. Particularly, we will analyze the parallel smoother and prove its usability within a multigrid iteration. Finally we give several numerical examples demonstrating the capability and the parallel efficiency of the implemented methods.

5.1. Isoefficiency Analysis

In this section we gather some basic ideas about the examination of parallel algorithms. The starting point for a parallel algorithm should be the best sequential one; the first key value for the analysis of parallelization is the running time of this best sequential algorithm depending on the problem size N. Shortly we note some definitions for further usage following Grama et al. [GGKK03]

Definition 5.1 (Parallel Efficiency)**.**

sequential time *Time used in running the "best sequential algorithm" for problems with size N:*

$$T_S(N) = \text{"time of sequential algorithm with problem size } N\text{."}$$

parallel time *The time used in running the "parallel algorithm" on P CPU's for problem size N:*

$$T_P(N,P) = \text{"time of parallel algorithm on } P \text{ CPU's for problem size } N\text{."}$$

speedup *The quotient of sequential and parallel running time is called the speedup of the parallel algorithm:*

$$S(N,P) = \frac{T_S(N)}{T_P(N,P)}.$$

efficiency *The parallel efficiency of the algorithm is defined by the quotient of the speedup by the number of CPU's P:*

$$E(N,P) = \frac{T_S(N)}{P \cdot T_P(N,P)}.$$

With this definitions we can state a first basic fact about the parallelization of algorithms:

Remark 5.2. *For every parallel algorithm the speedup is subject to the following essential constraint*

$$S(N,P) \leq P.$$

Otherwise we would have

$$\begin{aligned} S(N,P) &= \frac{T_S(N)}{T_P(N,P)} > P \\ \Rightarrow T_S(N) &> P \cdot T_P(N,P). \end{aligned}$$

This would lead to a new "fastest" sequential algorithm by running the parallel one P times sequentially in contradiction to the assumption of $T_S(N)$ being optimal. The parallel efficiency is therefore bounded from above by 1.

Although we do not chase after optimal parallel efficiency, basic guidelines have to be considered to port an algorithm to a large scale parallel computer. Otherwise, communication and administration effort would finally dominate, the capacity of the parallel computer would not be usable. In the following section we briefly describe considerations on the scalability of parallel algorithms. From this considerations we derive directives for good implementations.

A simple model of parallel algorithms, Amdahl's Law [Amd67] states that if there is a pure sequential share s of work, the speedup is bounded independent of the number of CPU's. The remainder share of work is assumed to be totally parallel. Therefore taking the sequential time as

$$T_S(N) = sT_S(N) + (1-s)T_S(N),$$

we get for the parallel running time

$$T_P(N,P) = sT_S(N) + (1-s)\frac{T_S(N)}{P}.$$

Accordingly we have as speedup

$$S(N,P) = \frac{1}{s + (1-s)\frac{1}{P}}.$$

This rather depressing result states that there is a natural limit for speedup

$$S(N,P) \xrightarrow[P\to\infty]{} \frac{1}{s}$$

and the parallel efficiency is tending to zero with increasing P:

$$E(N,P) = \frac{1}{sP + (1-s)}.$$

In most numerical problems the sequential share of the algorithm is reduced with increasing problem size. But there are still limits to the efficiency and the number of CPU's has to be coupled to the problem size to get acceptable efficiencies. But nevertheless this well known computation points out the problem of a purely sequential share of work; in the worst case, even tasks like centralized file i/o dominate the whole running time, if they cannot be parallelized. The good numerical efficiency of multigrid solvers complicates the problem, since the actual computational work is given by the complexity $O(N)$, therefore in terms of order associated with the same effort as administrational tasks (file i/o).

The following definition introduces the idea of isoefficiency from Grama et al [GGKK03] and points up the dependency between problem size and number of CPU's.

Definition 5.3 (isoefficiency)**.**
An algorithm is called **isoefficient scalable** *if there is an isoefficiency function $P \mapsto N(P)$ and a constant $0 < E_{iso} \leq 1$, such that*

$$E(N(P),P) = E_{iso}.$$

For isoefficient scalable algorithms we can increase the number of CPU's according the problem size and preserve the efficiency of the parallel algorithm.

Remark 5.4. *Parallel algorithms with communication effort in the same order of magnitude as the sequential running time are not isoefficient scalable.*

Proof: the result follows directly from Amdahl's Law. The communication of order $T_S(N)$ establishes some pure sequential share of work $s > 0$. The efficiency decreases with the number of CPU's and does not depend on the problems size. Thus we cannot find a suitable isoefficiency function $N(P)$.

□

Since all the numerical algorithms have linear running time communication of size $O(N)$ has to be avoided. Therefore all data has to be distributed to the different CPU's. Since we use a server-client model we have to assure that no data of size $O(N)$ is gathered at the server. Gathering or dealing out this data to the different CPU's would require the critical amount of communication $O(N)$.

5.2. Parallel Finite Elements Discretization

Heart of the parallelization progress is the partitioning of the meshes described in Chapter 2. All the data is directly aligned to the nodes, edges or cells of the meshes, depending on the specific discretization. Further the computational work is connected to the data, therefore a good partitioning is the fundamental prerequisite for a parallel implementation. Our grid consists of cells (that are quads in two and hexes in three dimensions) and related nodes. We distribute the cells to P different subdomains and therefore while obtaining an unique partitioning of the cells, some nodes of the triangulation are given to several processes. Other approaches use a node-wise unique distribution; each node is assigned to one (and only one) subdomain. Usually some additional overlap of nodes is stored on each subdomain. For example all nodes which are at most 1 edge distant to the boundary of the subdomains are stored in both subdomains. Such an overlap allows to reduce the number of communication steps in every multigrid cycle during the calculation. Since we are interested in preserving the original algorithms and for simplicity of the implementation we do not use any overlap. With the focus on large systems of partial differential equations, as it is the case for reactive flows, considerations of parallel efficiency are less crucial. The number of matrix entries – and therefore the amount of local work – is of second order in the number of solution components, while the amount of data to be transfered across the interface is linear in the number of solution components. Thus, the local complexity of the regarded problems backs up the parallel efficiency

The nodes lying on the boundary between (at least) two subdomains are called the interface nodes. The subset of nodes in one mesh part p lying on the interface is denoted by $\mathcal{I}_p$. The number of interface nodes is called N_{int}^p. Communication will be aligned to the interface. Therefore it should be kept small to reduce the effort.

We denote the mapping of the N_{cells} cells to P CPU's by

$$m^c : \{1, \ldots, N_{cells}\} \to \{1, \ldots, P\}, \tag{5.1}$$

the (not unique) mapping of the N_{nodes} by

$$m^n : \{1, \ldots, N_{nodes}\} \to \mathcal{P}(\{1, \ldots, P\}), \tag{5.2}$$

and the interface on CPU p is a mapping from the N_{int}^p interface nodes to the set of adjacent CPU's which have the node in common:

$$m^i : \mathcal{I}_p \to \mathcal{P}(\{1, \ldots, P\}). \tag{5.3}$$

The cells of the triangulation $\mathcal{T}_h$ are divided to $p = 1, \ldots, P$ sub-triangulations $\mathcal{T}_h^p$:

$$\mathcal{T}_h^p = \{K \in \mathcal{T}_h : m^c(K) = p\}.$$

All nodes belonging to one cell $K \in \mathcal{T}_h^p$ also belong to the sub-triangulation. Therefore the intersection between two adjacent sub-triangulations is a set of nodes, the interface. The number of nodes in one subdomain is denoted by N^p.

On every subdomain $\mathcal{T}_h^p$ we define a local discretization V_h^p according to Chapter 2. On the new inner boundaries belonging to the interface nodes no boundary values at all are described instead, the problem is always considered on the union of the local discretizations:

$$V_h = \bigcup_{p=1}^{P} V_h^p.$$

This union yields the original boundary values. The parallelization of the algorithms is done on the algebraic level engaging with the solver.

The partitioning of the mesh has to fulfill various conditions to ensure optimal behavior of the distributed algorithms:

load balancing The data as well as the work have to be distributed uniformly. Since memory usage and computational effort linearly depend on the number of nodes in our algorithm we just have to ensure uniformly distributed data.

minimal communication Communication among different CPU's will be necessary for all nodes on the interface between two parts of the mesh. Consequently the size of the interfaces has to be minimized.

In general we can't achieve both goals at one time. The partitioning of the meshes is performed with the freely available program library *Metis* [Kar], a graph partitioning tool. The distribution process is discusses in detail within the next section.

5.2.1. Distributing the Data

In Finite-Element calculations the numerical data - this are mainly the system matrix, the solution vector and auxiliary vectors - is directly coupled to the mesh. We have chosen a cell-wise partitioning for matrices and a nodewise overlapping partitioning for the vectors. If we take some arbitrary node-vector $v_h \in V_h$ corresponding to the discretization on mesh $\mathcal{T}_h$ the values are stored on p CPU's. That is, on every subdomain there exists a vector $v_h^p \in V_h^p$ of size N_p. The values of v_h^p on the interface $\mathcal{I}_p$ of one CPU p are also stored on the other side(s) of the interface. Using (5.3) we get:

$$x_i \in \mathcal{I}_p \Rightarrow v_h^p(x_i) = v_h^q(x_i), \quad \forall q \in m^i(x_i). \tag{5.4}$$

Throughout the algorithms we have to assure this condition for vectors on the interface. Matrix entries are composed of cell-integrals over finite element functions. The distributed matrix A^p stores all entries belonging to degrees of freedom situated on cells $K \in \mathcal{T}_h^p$.

Values belonging to couplings of degrees of freedom on the interface therefore are spread to multiple subdomains, only if the coupling degrees of freedom have their support cells in both subdomains. In Figure 5.1 we describe a case where a matrix entry associated with interface nodes is kept in only one subdomain although both nodes are present in both subdomains. We stress this particular point since it will be of importance further on, considering the multigrid solver.

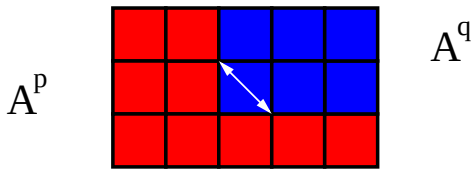

Figure 5.1.: A partitioning into two subdomains. The matrix entry belonging to the diagonal coupling of nodes is not stored in in A^p, although both degrees of freedom are situated in both subdomains.

For handling the distributed data we define some subdomain operators as restriction to the subdomain $\mathcal{R}_h^p : \mathcal{T}_h \to \mathcal{T}_h^p$ and a prolongation from the subdomain $\mathcal{P}_h^p : \mathcal{T}_h^p \to \mathcal{T}_h$. The restriction operator applied to a vector is defined as the nodewise restriction of v_h to v_p^h. If we use some local numbering $n^p : \{1, \dots, N_p\} \to \{1, \dots, N\}$ of the degrees of freedom $i = 1, \dots, N_p$ in $\mathcal{T}_h^p$ such that $x_i \in \mathcal{T}_h^p$ corresponds to $x_{n^p(i)} \in \mathcal{T}_h$ we get a matrix formulation of the restriction operator:

$$\mathcal{R}_h^p \in \mathbb{R}^{N_p \times N}, \quad r_{ij}^p = \begin{cases} 1 & n^p(i) = j, \\ 0 & n^p(i) \neq j \end{cases} . \tag{5.5}$$

The prolongation $\mathcal{P}_h^p$ is defined as the transposed of the restriction:

$$\mathcal{P}_h^p \in \mathbb{R}^{N \times N_p}, \quad \mathcal{P}_h^p = \left(\mathcal{R}_h^p\right)^T . \tag{5.6}$$

Using this definitions we can fix the conventions used for data distribution:

Definition 5.5 (Parallel Data Distribution)**.**
A vector $v_h \in V_h$ is correctly distributed to the p subdomains $v_h^p \in V_h^p$, if there holds

$$v_h^p = \mathcal{R}_h^p v_h$$

for all $p \in \{1, \dots P\}$. A matrix A_h is called correctly distributed, if we have

$$A_h = \sum_{p=1}^{P} \mathcal{P}_h^p A_h^p \mathcal{R}_h^p.$$

With this definition we can state some properties of the data distribution:

Remark 5.6.
If a vector v_h and a matrix A_h are distributed according to Definition 5.5 we get

1. $v_h^p = \mathcal{R}_h^p \mathcal{P}_h^p v_h^p$
2. $v_h \neq \sum_{p=1}^P \mathcal{P}_h^p \mathcal{R}_h^p v_h$
3. $A_h = \sum_{p=1}^P \mathcal{P}_h^p A_h^p \mathcal{R}_h^p$
4. $A_h^p \neq \mathcal{R}_h^p A_h \mathcal{P}_h^p$

PROOF: Point 1. is evident, since $\mathcal{R}_h^p \mathcal{P}_h^p$ is the identity on $\mathbb{R}^{N^p \times N^p}$. As degrees of freedom lying on the interface are restricted to all adjacent subdomains, these are considered multiple times in the summed prolongated vector. Therefore 2. follows. Point 3. is the definition and the matrix A_h^p is therefore not result of an node-wise restriction since the entries correspond to integrals over cells as discussed before and entries on the interface are spreaded over the different domains.

□

5.2.2. Implementation and Parallel Efficiency of the Matrix Vector Product

Due to the decomposition of matrices and vectors (definition 5.5) the matrix vector product can be calculated locally on every CPU. After performing the calculations we have to assure the parallel distribution condition of the resulting vector. Using the subdomain operators $\mathcal{R}_h^p$ and $\mathcal{P}_h^p$ the matrix vector product $b_h = A_h v_h$ reads:

$$\begin{aligned} b_h^p &= \mathcal{R}_h^p b_h = \mathcal{R}_h^p (A_h v_h) \\ &= \mathcal{R}_h^p \sum_{q=1}^P \mathcal{P}_h^q A_h^q \mathcal{R}_h^q v_h \\ &= \mathcal{R}_h^p \sum_{q=1}^P \mathcal{P}_h^q \left(A_h^q v_h^q \right). \end{aligned}$$

The evaluation of the matrix vector product can be split into two parts:

1. Local calculation of the matrix vector product $\tilde{b}_h^p = A_h^p v_h^p$.
2. Balancing of the values on the interface $b_h^p = \mathcal{R}_h^p \sum_{p=1}^P \mathcal{P}^q \tilde{b}_h^q$.

Step 1. of the matrix vector product can be performed in parallel on every CPU. Step 2. involves communication between adjacent CPU's along the interface. Step 2. is necessary to assure the conforming data distribution (Definition 5.5).

To analyze the parallel efficiency of the matrix vector product we assume the cost for one numerical operation as c_{num} and the average number of subdiagonals in the matrix as c_{diag}. The required time for the sequential matrix vector product $b_h = A_h v_h$ is given by

$$T_S(N) = c_{num} c_{diag} N. \tag{5.7}$$

If we assume the subdomains to be of almost equal size, the effort for Step 1. of the parallel matrix vector product is matched by

$$T_P^1(N,P) = \frac{1}{P} T_S(N) = c_{num} c_{diag} \frac{N}{P}. \tag{5.8}$$

Further we assume that the interface between two subdomains is of size $(N/P)^{(d-1)/d}$, i.e. the dimension of a line compared to a surface in two dimensions. The maximum number of subdomains in contact with one interface node is limited by c_{neigh} and the number of communication steps needed for exchanging data along the interfaces is bounded by $2c_{neigh}$. This is not an assumption but a result from Lemma 5.8 which will be given in Section 5.2.3. The cost for transferring data of size 1 is expressed by c_{comm}. Step 2. of the parallel matrix vector product involves communication of the interface nodes as well as local balancing of this nodes:

$$T_P^2(N,P) = (c_{num} + 2c_{neigh} c_{comm}) \cdot \left(\frac{N}{P}\right)^{\frac{(d-1)}{d}}. \tag{5.9}$$

The additional computational effort is neglected and included in the actual local matrix vector product. Combining (5.7), (5.8) and (5.9) we get the following results:

$$\begin{aligned} T_P(N,P) &= c_{num} c_{diag} \frac{N}{P} + 2c_{neigh} c_{comm} \left(\frac{N}{P}\right)^{\frac{(d-1)}{d}}, \\ S(N,P) &= \frac{1}{\frac{1}{P} + 2\frac{c_{neigh} c_{comm}}{c_{num} c_{diag}} N^{\frac{-1}{d}} P^{\frac{1-d}{d}}}, \\ E(N,P) &= \frac{1}{1 + 2\frac{c_{neigh} c_{comm}}{c_{num} c_{diag}} \left(\frac{P}{N}\right)^{\frac{1}{d}}}. \end{aligned} \tag{5.10}$$

We conclude that the matrix vector product is isoefficiently scalable with a linear isoefficiency function $N(P) \mapsto cN$.

Since we use a master-client concept for parallelization operations to be performed on the clients have to be activated by the master. Additional communication with effort $O(P)$ appears. If we include the startup of the matrix vector product, (5.10) yields:

$$\begin{aligned} T_P(N,P) &= c_{num} c_{diag} \frac{N}{P} + 2c_{neigh} c_{comm} \left(\frac{N}{P}\right)^{\frac{(d-1)}{d}} + c_{comm} P, \\ S(N,P) &= \frac{1}{\frac{1}{P} + 2\frac{c_{neigh} c_{comm}}{c_{num} c_{diag}} N^{\frac{-1}{d}} P^{\frac{1-d}{d}} + \frac{c_{comm}}{c_{num} c_{diag}} \frac{P}{N}}, \\ E(N,P) &= \frac{1}{1 + 2\frac{c_{neigh} c_{comm}}{c_{num} c_{diag}} \left(\frac{P}{N}\right)^{\frac{1}{d}} + \frac{c_{comm}}{c_{num} c_{diag}} \frac{P^2}{N}}. \end{aligned} \tag{5.11}$$

In the two dimensional case a closed formula for an isoefficiency function $N(P)$ can be given. Instead we try to analyze the three-dimensional case in common situations. The isoefficiency function will be suited somewhere between $N(P) = P$ and $N(P) = P^2$. On a usual computer with a performance of one gigaflop we set $c_{num} = 10^{-9}$, a high performance network yields a

transfer rate of about 1 gigabit a second, thinking of double precision representation of the numbers we set $c_{comm} = 1.5 \cdot 10^{-7}$. The number of subdiagonals using a Q_2 discretization is approximately $c_{diag} = 125$ and the maximum number of adjacent subdomains equals $c_{neigh} = 8$. Gathering this values we get as the parallel efficiency:

$$E(N,P) = \frac{1}{1 + 20\left(\frac{P}{N}\right)^{\frac{1}{3}} + \frac{P^2}{N}}. \tag{5.12}$$

If we expect at least 10 000 degrees of freedom for all subdomains, i.e. we use for (5.12) the coherency

$$N(P) = 10\,000P,$$

it follows that the efficiency will be above 0.5 if we use up to 1024 CPU's. Although this calculation is rather rough, the usability of the methods should be asserted.

Most algorithms used have a simular structure, including the initialization from the master $O(P)$, some local calculation $O(N/P)$ and finally communication on the interface of order $O((N/P)^{(d-1)/d})$. Altogether we will expect a parallel efficiency of at least 0.5 if we choose a reasonable proportion of P and N, as e.g. $N = 10\,000P$.

5.2.3. Distributed Communication

In this section we describe the distributed communication which occurs when values on the interface between different subdomains have to be exchanged. The layout of the subdomains and the neighborhood relations is a-priori unknown. Depending on the mesh layout, adaptive refinement and the number of subdomains, a large variety of partitions will occur. Every client p hast to communicate with all clients q neighboring to cells situated in $\mathcal{T}_h^p$. Without knowledge of the specific topology we have to assume that every client p has to communicate with all other clients. Efficient algorithms for exchanging data between all P processes involve a running time in order $O(P)$. Regarding e.g. the matrix vector product this would lead to a different isoefficiency function, approximately in the region $N = O(P^4)$ which is not usable for practical purpose.

In this section we will present an algorithm to create a communication sequence which allows the exchange of interface values in at most $2c_{neigh}$ steps, where c_{neigh} is the maximum number of adjacent subdomains in one interface node.

The parallel topology of the problem is transformed to a graph. Figure 5.2 displays a domain partitioned into 7 subdomains. The subdomains form the nodes of the graph, the edges describe the communication connections, i.e. the neighborhood relation. If values on the interface have to be exchanged, communication along all nodes of the graph is necessary. The algorithm aims at labeling the edges in such a manner that edges with equal label can be treated simultaneously, that is, all edges adjacent to one node must have different labels.

The number of nodes is denoted by P, the maximum number of edges meeting in one node is denoted by c_{neigh}. Each node p has a sorted list of $\#e_p$ edges connecting the nodes p and $e_p[i]$:

$$e_p[i] < e_p[i+1], \quad i = 1, \ldots, (\#e_p - 1).$$

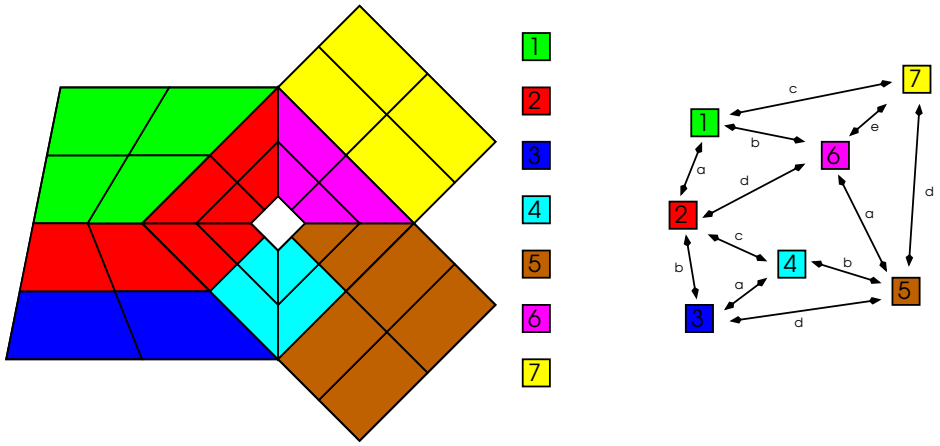

Figure 5.2.: Domain distributed to 7 subdomains and associated communication graph. Edges with the same letter can be treated simultaneously during communications.

The algorithm runs in parallel and stains the edges of the graph in the following way. See Bastian [Bas03] for details. On every CPU p:

Algoritm 5.7 (Communication Graph)**.**

```
for k = 1,...,#e_p:
  set q = e_p[k] neighboring node
  set c' = 0
  do
    set     c = minimal unused label ≥ c' on p
    send    c to node q
    receive c' = minimal unused label ≥ c from q
  while (c ≠ c')
  set label[q] = c
```

Lemma 5.8. *The maximum number of labels c_{maxlab}, handed out by this algorithm is*

$$c_{maxlab} \leq 2 \max_{p \in \{1,\ldots,P\}} \#e_p.$$

Proof: Otherwise two adjacent nodes would together hold more than $2c_{neigh}$ edges, which contradicts the assumption $\#e_p \leq c_{neigh}$.

□

Using this remark we can estimate the communication effort for exchanging values across the interface:

Remark 5.9. *The exchange of data of size $O(D)$ across the interface needs communication of order $O(c_{neigh}D)$.*

The running time of the algorithm is of order $O(c_{neigh}P)$. Once the communication graph is assembled it will be used frequently during the calculations. The edges in the example (see Figure 5.2) are labeled with 5 different labels. I.e. communication across all edges is performed in 5 steps.

As mentioned in Section 5.2.2 there are about $(N/P)^{(d-1)}d$ degrees of freedom on the interface in one subdomain. The overall communication cost is expected to of order

$$O\left(c_{neigh}\left(\frac{N}{P}\right)^{\frac{d-1}{d}}\right).$$

This communication is not evenly distributed over all edges, e.g. the data amount to be transfered between nodes 2 and 3 in Figure 5.2 is of order $O\left((N/P)^{(d-1)}d\right)$ but between nodes 1 and 6 only data of size $O(1)$ has to be transfered. Both edges are labeled with the same letter, the communication is performed in one step. Since the latter communication is fulfilled in a shorter time, the involved nodes have to wait for the end of the communication step. A small modification of the algorithm enhances the performance: we assemble two (or even more) communication graphs. First we only take into account communications of size $O\left((N/P)^{(d-1)}d\right)$. Since we neglect some couplings, we should get fewer communication steps. Afterwards we consider the exchange of values of order $O(1)$. In the three-dimensional case we could even insert a third communication graph responsible for the intersections of order $O\left((N/P)^{(d-2)}d\right)$ (that is: faces, edges, vertices). For example we consider domain $\Omega = [0,1]^3 \subset \mathbb{R}^3$ uniformly distributed into P^3 cuboids. Each subdomain (if we neglect the boundary) has 26 adjacent subdomains. The algorithm creates a communication graph with 26 steps, the effort of exchanging all values is of order $O\left(26(N/P)^{\frac{2}{3}}\right)$. If we split the type of edges into 6 edges of size $O\left((N/P)^{\frac{2}{3}}\right)$, 12 edges of size $O\left((N/P)^{\frac{1}{3}}\right)$ and 8 edges of size 1, the overall communication effort reduces to $O\left(6(N/P)^{\frac{2}{3}}\right)$ if we neglect the lower order terms and therefore is decreased to approximately a fourth.

5.2.4. Hanging Nodes and Multigrid

The methods introduced up to now are suitable for simple calculations on structured meshes. In this section we describe the necessary changes regarding adaptive mesh refinement and multigrid solvers. Especially the usage of multigrid solvers calls for various adaptations.

First we describe the modifications necessary due to the multigrid structure. A short introduction to multigrid solvers will be given in section 5.3, for a detailed analysis we refer to Hackbusch 1985 [Hac85] and Hackbusch 1993 [Hac93]. Shortly, multigrid methods are based on the recursive approximation of the problem with the usage of coarser subproblems. Starting with the discretized domain $\mathcal{T}_h$, we combine adjacent cells $K \in \mathcal{T}_h$ to new large cells and form a mesh $\mathcal{T}_h'$ consisting of these coarse cells. The actual problem to be solved on $\mathcal{T}_h$ is approximated on $\mathcal{T}_h'$. This coarse mesh approximation is recursively continued to

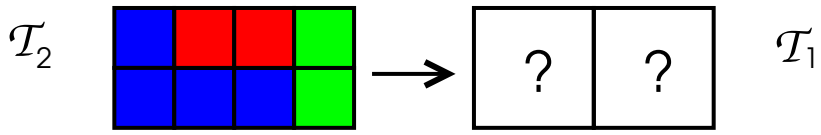

Figure 5.3.: Two meshes from a multigrid hierarchy. While the fine mesh (left) is distributed to three subdomains, the partitioning of the coarse mesh is ambiguous.

a final coarse mesh. We denote the coarsest mesh by $\mathcal{T}_1$ and the finest mesh $\mathcal{T}_h$ by $\mathcal{T}_L$. The practical creation of the mesh hierarchy is established in a reverse manner by adaptive mesh refinement. Starting with the coarse mesh $\mathcal{T}_1$ we successively solve the problem and choose cells for refinement. This mesh hierarchy is not necessary the multigrid hierarchy. The exact construction of the multigrid meshes is described in Becker & Braack [BB00]. The coarse mesh approximation requires vectors from one mesh level $\mathcal{T}_l$ to be transfered to the neighboring meshes in the hierarchy, i.e. $\mathcal{T}_{l+1}$ and $\mathcal{T}_{l-1}$. The mapping to the finer mesh $\mathcal{T}_{l+1}$ is called prolongation, the mapping to the coarse mesh $\mathcal{T}_{l-1}$ restriction. Thinking of distributed memory computers we would appreciate if the mesh transfer could be performed without communication between different processes. Therefore we have to distribute the whole mesh hierarchy in a nested way: a node $x_i \in \mathcal{T}_l^p$ in some subdomain p on mesh-level l should be situated in the same subdomain p on the adjacent mesh-levels, that is $x_i \in \mathcal{T}_{l+1}^p$ and $x_i \in \mathcal{T}_{l-1}^p$. The easiest way of establishing this kind of distribution would require a partitioning of the coarse mesh $\mathcal{T}_1$ which is kept during mesh refinement. If a cell $K \in \mathcal{T}_1^p$ is refined, the new (children cells) K' are arranged in the mesh $\mathcal{T}_2^p$. There are two reasons which oppose this simple strategy:

- If we use adaptive mesh refinement, thus we do not refine all cells $K \in \mathcal{T}_l$ and we do not a-priori know which cells we will refine, the partitioning of higher mesh levels degenerates. The number of cells in the different subdomains will show a large range. The efficiency of the parallel algorithms will lack, since the work is inhomogeneously distributed. ($\Rightarrow$ load balancing).

  An error estimator can be used to predict the necessary number of cells in each area of the domain. These predictions can be used to generate a good partitioning of coming meshes.

- Using multigrid methods it is desired to use coarse meshes as small as possible. The number of subdomains P to use may be bigger than the number of cells in the coarse mesh. But even if the number of cells is large enough we do not get a balanced partitioning unless we can guarantee an adequate quantity of cells per subdomain. This value may depend on the algorithm used for the partitioning of the mesh, using *Metis* [Kar] about 500 cells per subdomain is a rule of thumb regarding a three dimensional domain to create a partitioning with a difference of 5% between the largest and the smallest subdomain.

In particular the second point is crucial. We therefore have to choose finer meshes for partitioning. This will lead to a new problem illustrated in Figure 5.3. If we distribute the

fine mesh $\mathcal{T}_2$ we cannot transfer the partitioning to the coarse mesh $\mathcal{T}_1$ since cells $K^1, K^2 \in \mathcal{T}_2$ belonging to the same father cell $K \in \mathcal{T}_1$ are distributed to different subdomains. The problem even gets worse if the number of subdomains on mesh level $\mathcal{T}_2$ exceeds the number of cells in level $\mathcal{T}_1$. Since we cannot bypass this problem, we have to apply two adaptations while transferring a partitioning of a mesh $\mathcal{T}_l$ to the coarser mesh $\mathcal{T}_{l-1}$:

M1 If the children $K^i \in \mathcal{T}_l$ of a cell $K \in \mathcal{T}_{l-1}$ are situated in different subdomains, the cell K is attached to the subdomain where most children K^i are placed. If we reach a stand-off, the cell K is attached to the subdomain with the fewest number of cells up to now (of coarse we only consider subdomains in coherence with the children cells).

M2 If the number of cells in some subdomain the coarse mesh $\mathcal{T}_{l-1}^p$ is less a given limit, we combine this subdomain with the "next smallest" subdomain. Using this procedure we circumvent the problem of having more subdomains than cells. The pooling of subdomains is called *coarse mesh agglomeration*.

Relying on these adaptations we cannot assure the meshes to be distributed in a nested way. Therefore the mesh transfer is not totally local on every CPU. Since in multigrid context the lot of the work is done on the fine meshes and the coarse meshes are negligible we focus on the fine levels of the hierarchy. Since we retain a nested mesh structure if we carry forward the partitioning to finer meshes we just have to choose a mesh in the middle of the hierarchy to obtain the desired attributes on the fine meshes. The mesh which we actually distribute has to accomplish a trade-off between richness (to deliver a balanced partitioning) and coarseness (to assure local mesh transfer on fine mesh levels). For practical purposes we choose the coarsest mesh that holds at least a given number (approximately $100 \sim 500$) of cells per desired subdomain.

If we regard adaptive mesh refinement the partitioning of the multigrid hierarchy is more involved. We cannot choose an arbitrary mesh $\mathcal{T}_l$ in the multigrid hierarchy for partitioning, since we have to take the kind of adaptive refinement into account. If we split some mesh $\mathcal{T}_l$ into two subdomains of equal size, it might happen that all cells in subdomain $\mathcal{T}_l^1$ are chosen for refinement, but none in $\mathcal{T}_l^2$. The partitioning of the finer mesh $\mathcal{T}_{l+1}$ would be unbalanced. Thus the finest mesh has to be kept in mind during partitioning of some intermediate mesh. If we produce a balanced fine mesh, we abandon balanced coarse meshes, but since the main work (and memory usage) is connected with the fine mesh, this is no severe problem.

The toolbox *Metis* [Kar] used for partitioning actually provides general tools for graph partitioning and is not specially restricted to finite element meshes. A graph is partitioned in such way, that the number of nodes clustered in each subgraph is uniformly distributed and the number of edges cut for the partitioning is minimized. *Metis* allows the partitioning of weighted graphs. Thus to partition an intermediate mesh $\mathcal{T}_l$ we construct a graph where the nodes are cells $K \in \mathcal{T}_l$ with a weight equal to the number of descendants $K^i \in \mathcal{T}_L$ of K in the finest mesh. If we use this graph for partitioning, and the summed weights in each subgraph are balanced we get an even partitioning – in terms of cells – of the finest mesh.

In addition we can provide edge-weights, which indicate the number of cells separated in the finest mesh if we cut this specific edge. This minimizes the size of the interface on the finest mesh level $\mathcal{T}_L$.

Another problem arises concerning hanging nodes. During the calculation this nodes are replaced by an interpolation with neighboring nodes. If the neighboring nodes would be spreaded over different subdomains, communication would be required for resolving them. Therefore we have to gather all cells with hanging nodes prior to distributing them. The mesh actually chosen for partitioning will not be a mesh from the multigrid hierarchy and can be roughly described as "the coarsest mesh from the hierarchy which is still rich enough to allow Metis a good partitioning with some additional substitution of cells by coarser ones in order to gather hanging nodes in on subdomain."

In the following we describe all steps necessary for the distribution of the meshes. We require at least C_{sd} cells per subdomain and the meshes should be distributed to P domains:

Algoritm 5.10 (Mesh Partitioning)**.**

Given multigrid hierarchy $\mathcal{T}_1, \ldots, \mathcal{T}_L = \mathcal{T}_h$
1. set $\mathcal{T} = \mathcal{T}_l$ with $\mathcal{T}_l$ coarsest mesh with $\#\mathcal{T}_l/P > C_{sd}$
2. while some $K \in \mathcal{T}$ contains a hanging node
   a) set $K' = \texttt{father}(K)$
   b) remove all children K'^i of K in $\mathcal{T}$
   c) insert K' to $\mathcal{T}$
3. distribute mesh $\mathcal{T}$ to P subdomains $\mathcal{T}^p$
4. for all $K \in \mathcal{T}$
   a) insert all children K^i of K to the same subdomain as K
   b) mark all ancestors K^i of K for the same subdomain as K
5. insert all cells K marked in step 4.b) to the subdomain with most marks.
6. while subdomain $\mathcal{T}_l^p$ on level l exists with $\mathcal{T}_l^p < C_{sd}$
   a) combine subdomain $\mathcal{T}_l^p$ with smallest neighboring subdomain $\mathcal{T}_l^{p'}$

To illustrate the partitioning process, Figure 5.4 describes the complete progress of distributing a full multigrid hierarchy for an example configuration: In the toggling row of Figure 5.4 a multigrid hierarchy consisting of 4 mesh levels is given. This hierarchy is established by global coarsening of the finest mesh. All meshes should be partitioned into 2 subdomains. We require at least $C_{sd} = 5$ cells per subdomain for partitioning the resulting graph. The final meshes must have 2 cells per subdomain. In the following we describe all steps of the algorithm.

In the second row we start with Step 1. of Algorithm 5.10. We choose mesh $\mathcal{T}_3$ for starting since this mesh possesses $16 > 2 \cdot C_{sd}$ cells, mesh $\mathcal{T}_2$ with 7 cells does not fulfill this requirement. Next we combine all cells with hanging nodes (steps 2.a)-2.c)). The resulting mesh is not a mesh from the multigrid hierarchy but some intermediate one. The cellweights indicate the number of cells in the finest mesh. In Step 3. the mesh is partitioned under usage of the weighted graph as described in this section.

In the third row the partitioning is transfered to finer meshes (Step 4. and in Step 5. we choose the blue subdomain for the remaining cell. Finally in Step 6. we combine both subdomains on the coarsest mesh to fulfill the requirement of at least 2 cells per subdomain.

Finally we will give some clues on the parallel efficiency of the partitioning process. Since the partitioning is done on the master process and the partitioned meshes are handed out to the clients, we cannot expect isoefficient scalability. If the meshes are big in comparison to the problem size on every CPU, the partitioning is a severe bottleneck of the program. That implies that the implemented method is not an efficient parallel Poisson solver on adaptive refined meshes (which was not the aim of parallelizing the code). The guideline for a better mesh handling is obvious but requires a distributed mesh hierarchy. This is still subject to further work. The implementation of a distributed mesh handling for adaptively refined meshes is connected with large time exposure but will be necessary for the efficient treatment of non-stationary problems, where the question of load balancing gets fundamental.

As a trade-off we have the possibility of saddling steps of global mesh refinement upon a previously adaptive refined mesh hierarchy. These additional global mesh refinements are performed locally on every CPU and require communication only along the interfaces between different CPU's.

Conclusion

With the described partitioning of the meshes we have established the basics for a distribution of finite element discretizations. In the next section we give details on the numerical algorithms working on the data. Up to now, no change to the methodology was imposed due to the parallelization.

The partitioning and (statical) load balancing in well suited for the problems under consideration. These all have in common a very large system matrix compared to the number of mesh nodes. This leads to an essential share of local work to be done. Communication is not negligible, but after all of minor importance.

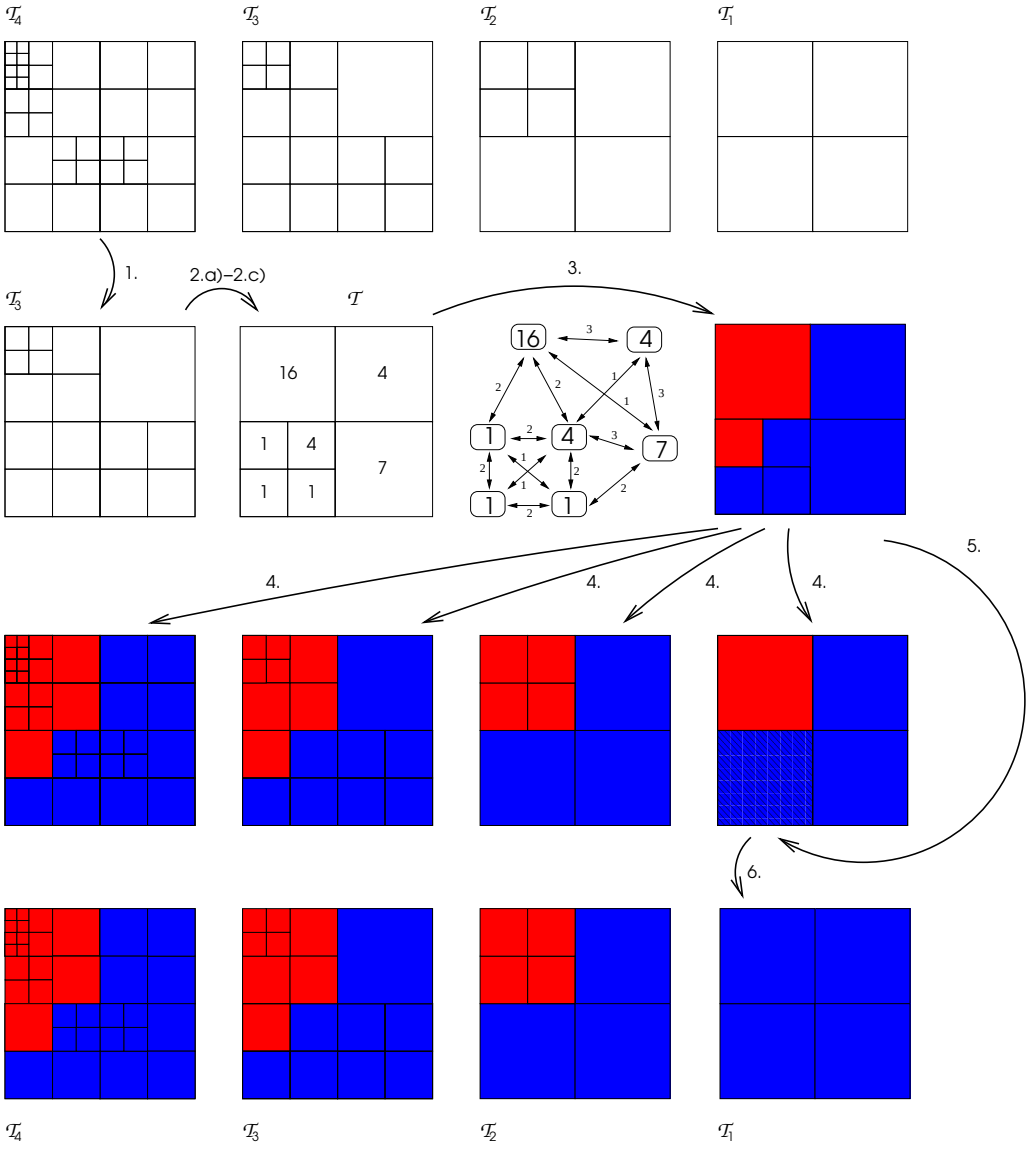

Figure 5.4.: Partitioning of a complete multigrid hierarchy.

5.3. Parallel Multigrid Solver

This section discusses the parallelization of the multigrid solver. A good introduction to multigrid solvers is given by Hackbusch 1985 [Hac85] and Hackbusch 1993 [Hac93]. Details on parallel implementations are found in Bastian [Bas96]. For multigrid on locally refined meshes we refer to Becker & Braack [BB00]. In the following we give a short introduction to multigrid methods. Concerning theoretical results we refer to the mentioned references.

The solution of linear systems $Ax = b$ using multigrid methods relies on the division of the defect $b - Ax$ into high an low frequencies. The high frequent errors are approximated (*smoothed*) with some steps of simple iterative methods as Jacobi, Gauss-Seidel, etc; the low frequent error parts are approximated on coarser meshes, *coarse mesh approximation*, thus with less numerical effort. The efficiency of multigrid methods is based on the observation, that usual linear iterations (as Jacobi, Gauss-Seidel, ...), though giving very bad solvers, rapidly reduce the high frequent error parts. The remaining low error frequencies are approximated on the coarse mesh. Since the dimension of the coarse mesh is smaller, the overall effort is reduced. If we approximate the coarse mesh problem again by splitting into high and low frequencies, the method is recursively iterated.

The mesh $\mathcal{T}_h = \mathcal{T}_L$ is the finest mesh of a multigrid hierarchy $\mathcal{T}_1, \ldots, \mathcal{T}_L$. These multigrid meshes are nested, i.e. the relationship $\mathcal{T}_l \subset \mathcal{T}_{l'}$ for $l' > l$ holds in a node-wise sense. On every mesh level l there exists a system matrix A_l and vectors $v_l \in V_l$ in the finite element space constructed on the specific mesh level. The *two-grid* iteration for the approximation of $A_l v_l = b_l$ on level l looks as follows:

Algoritm 5.11 (Two Grid Iteration)**.**

1. $v_l^{\mu_1} = \mathcal{S}_l^{\mu_1} v_1$	(pre-smooth)
2. $d_l = b_l - A_k v^{\mu_1}$	(defect)
3. $d_{l-1} = \mathcal{R}_l d_l$	(restriction)
4. $v_{l-1}^c = A_{l-1}^{-1}(d_{l-1})$	(coarse mesh approximation)
5. $v_l^c = v^{\mu_1} + \mathcal{P}_l v_{l-1}^c$	(prolongation)
6. $v_l^{\mu_2} = \mathcal{S}_l^{\mu_2} v_l^c$	(post-smooth)

Basically we can split the two-grid iteration into three parts: Steps 1. and 6. treat the smoothing of the high frequencies. The mesh transfer is performed in Steps 3. and 5., the approximation of the low frequencies is supplied in Step 4. If we bit-by-bit replace the coarse mesh approximation by the two-grid iteration we finally end up with the complete multigrid solver. Following the traditional analysis of multigrid solvers, the problem on the lowest level is solved exactly (coarse mesh problem). But in practice it is sufficient to utilize some steps of an iterative method for the coarse problem.

First we will analyze the mesh transfer operations. Since the partitioning of the meshes is mainly nested (see Section 5.2.4), the mesh transfer can be performed locally on every CPU. Only on coarser mesh levels some degrees of freedom must be fetched from neighboring CPU's. If we regard coarse mesh agglomeration, the mesh transfer between to different levels might require communication for nearly all degrees of freedom. Since coarse mesh

agglomeration is utilized only on coarse meshes of size $O(1)$ the aligned communication effort is no problem for the parallel efficiency. On intermediate meshes we have to expect communication of size of the interface $O\left((N/P)^{(d-1)/d}\right)$.

The evaluation of the defect in Step 2. of the multigrid algorithm employs a matrix vector product and therefore needs communication. This special issue is already discussed in Section 5.2.2.

The smoothing operation is the most delicate part of a multigrid solver. Usually few steps of iterative methods are used for smoothing. The parallelization of the smoother therefore strongly depends on the specific layout of the smoother. While methods like the Jacobi Iteration are inherently well suited for parallelization, others like the Gauss-Seidel Iteration have a sequential character. We mostly use a incomplete LU decomposition (ILU) of the system matrix as smoother. This method is well suited for a large range of problems as will be demonstrated below.

5.3.1. Parallel Multigrid Smoother

Since the ILU decomposition is a strictly global method, parallelization is not self-evident. In fact, the implemented parallel version numerically differs from the original sequential version. We neglect the subscript for the mesh level, as the smoothing process does not involve different levels:

Algoritm 5.12 (Sequential Multigrid Smoother)**.**

1. $d = b - A\ v$
2. $w = ILU(A)^{-1} d$
3. $v = v + w$

This consists of the evaluation of the defect $b_l - A_l v_l$ in step 1., the application of the ILU and finally the update in Step 3. In the parallel version we just replace the application of the ILU, i.e. Step 2. by a localized version:

2.a) $\tilde{w}^p = ILU(\widetilde{A^p})^{-1}\ d^p$
2.b) $w^p = \mathcal{R}^p \sum \mathcal{P}^p\ \tilde{w}^p$.

That is, we locally assemble an ILU decomposition of the matrices A^p (with slight modifications that are described later on) and afterwards balance the values on the interface to fulfill the data conventions (Definition 5.5). If we would solve the subproblems in Step 2.a) exactly, the proposed iteration would correspond to the standard Schwarz Iteration with minimal overlap. Schwarz [Sch69] suggested already 1869 an iterative method for solving differential equations using local problems on overlapping subdomains. Considering parallel computing, research on these domain decomposition methods was driven due to the natural parallel structure. An overview to domain decomposition is given by Quarteroni & Valli [QV99]. The major lack of the simple Schwarz iteration is the dependence of the convergence

order on the size of the overlap. In our approach the overlap is of size h, the local mesh size, and tends to zero. Therefore the error reduction $1 - O(h)$ tends to 1 under mesh refinement. Since the domain decomposition is applied as smoother and not as a self-contained solver, the approximation property is satisfactory in the multigrid context. Details are discussed later during this section.

If we regard a local part of the system matrix A_h on a subdomain $\mathcal{T}_h^p$ without contact to the border of the domain $\mathcal{T}_h$, this local matrix A^p corresponds to a homogeneous Neumann problem, thus the matrix is not regular and the existence of the ILU is not assured. We therefore apply a modification to the local system matrices before building the ILU:

$$\widetilde{A_h^p} = \mathcal{R}^p \left(\sum_{q=1}^{P} \mathcal{P}^q A_h^q \mathcal{R}^q \right) \mathcal{P}^p.$$

This modified matrix $\widetilde{A_h^p}$ is mainly the node-wise restriction of the global system matrix A_h to the subdomain $\mathcal{T}_h^p$. The additional note "mainly" is essential since we only consider matrix couplings across cells $K \in \mathcal{T}_h^p$. Figure 5.5 displays a situation, where matrix couplings are neglected in some subdomains.

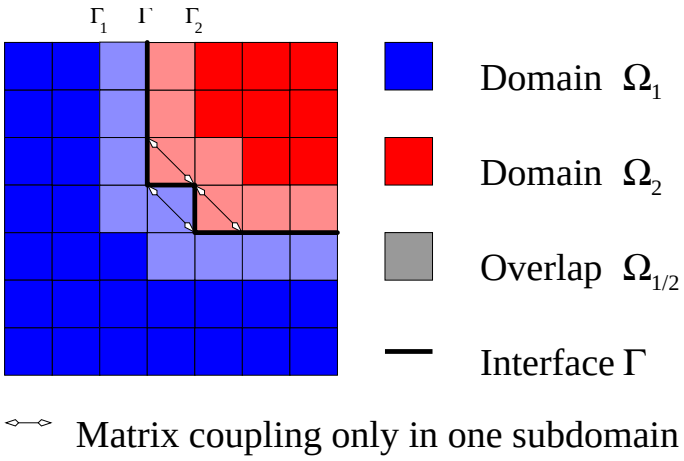

Figure 5.5.: Structure of the modified local matrices $\widetilde{A_h^p}$. Some matrix couplings are only considered in one subdomain, although both degrees of freedom lie on the interface.

Using this modified system matrix, the local problems can be interpreted as local Dirichlet problems, with homogeneous Dirichlet boundary values at the degrees of freedom one cell beyond the interface. If we would solve the local problems exactly, the smoother would equal the standard additive Schwarz Iteration with overlap of size h. Two further modifications are needed to derive the finally used smoother:

1. The subdomains are not solved exactly. Instead we apply an ILU decomposition of the modified local matrices $\widetilde{A_h^p}$. I.e. we use the original sequential smoother for approxi-

mating the local subproblems. The Schwarz iteration is a wrapper around the actual smoother.

2. The Schwarz iteration utilizes local Dirichlet problems with boundary values tapped from the neighboring domains. Since we deploy the Schwarz iteration in a multigrid context, where the solution of the defect correction problems tend to zero, we simplify the iteration by using homogeneous Dirichlet values.

The convergence of the smoother, i.e. the convergence of the high frequent error parts, is analyzed in the next section. Now we give the complete parallel smoothing algorithm and state some hints on the parallel efficiency of the smoother. Once on every mesh level $\mathcal{T}_l$ we have to construct the ILU decomposition. After gathering the matrix entries on the interface we employ the sequential algorithm on every CPU:

Algoritm 5.13 (Assembling the parallel ILU)**.**

1. $\widetilde{A_h^p} = \mathcal{R}^p \left(\sum \mathcal{P}^q A_h^q \ \mathcal{R}^q \right) \ \mathcal{P}^p$
2. $\mathtt{ILU}^p = ILU(\widetilde{A_h^p})$

Step 2. is performed locally on every CPU. Step 1. requires communication effort comparable to the matrix vector product. We expect a simular parallel efficiency. But since the ILU is assembled only once and frequently used, the parallel efficiency is of no significant importance. The application of the parallel smoother is a combination of the Schwarz iteration with the sequential version of the smoother (see Algorithm 5.12)

Algoritm 5.14 (Parallel Multigrid Smoother)**.**

1. $d^p = \mathcal{R}^p \sum \mathcal{P}^q \left(b^q - A_h^q v^q \right)$
2. $w^p = \mathcal{R}^p \sum \mathcal{P}^q \left(\mathtt{ILU}^p \ d^p \right)$
3. $v^p = v^p + w^p$

Step 1. in this algorithm is the already introduced parallel matrix vector product. In Step 2., we locally apply the ILU decomposition and further have to balance the values on the interface. The update is performed in Step 3. This step is local. Analyzing the parallel efficiency we get the same structure as the matrix vector product. Communication is required twice for balancing the interface. The computational work for applying the ILU decomposition is of the same magnitude as one matrix vector product. Therefore we can expect a simular parallel efficiency.

5.3.2. Convergence Analysis for the Schwarz Iteration

As a first approach we are looking at the "smoothing properties" of the Schwarz iteration on the following model problem:

$$\begin{aligned} -\Delta u &= f \text{ in } \Omega \in \mathbb{R}^2, \\ u &= 0 \text{ on } \partial\Omega. \end{aligned} \tag{5.13}$$

The domain $\mathcal{T}_h$ is chosen as shown in Figure 5.6 and split into two overlapping subdomains $\mathcal{T}_+$ and $\mathcal{T}_-$ with new internal boundaries Γ_+ for $\mathcal{T}_+$ and Γ_- for $\mathcal{T}_-$. Some discretization with mesh size h is applied and the overlap is of size $2h$. We observe the error on the line Γ throughout the Schwarz iteration. Even though the analysis is split into all frequencies which are presentable on the mesh, we always regard the analytical problem, i.e., the exact solutions of the Schwarz iteration.

From the standard analysis it is known, that the error reduction rate acts like $1 - O(h)$. Now we study the error with regard to its different frequencies. We will show a better reduction rate – independent of h – for high frequencies. The discussion given now is based on Hackbusch 1985 [Hac85]. The actual analysis of Schwarz iteration embedded into the multigrid solver is given in the subsequent sections.

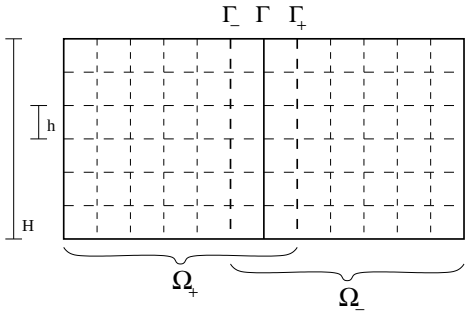

Figure 5.6.: Domain $\mathcal{T}_h$ split into two subdomains. Artificial boundaries are introduced with overlap of size h.

The largest error of the classical Schwarz iteration is expected near the interface of two adjacent subdomains. We therefore fix our interest on the line Γ in Figure 5.6.

Lemma 5.15 (Approximation properties of the Schwarz iteration)**.** *Consider the Laplace problem*

$$-\Delta u = f,$$

on the domain Ω given in Figure 5.6 partitioned into two subdomains with overlap of size $2h$. After one step of Schwarz iteration – with analytical solution of the subdomain problems – the error on the interface line Γ is reduced by some factor bounded away from 1 for all high frequencies which are not presentable on the coarser mesh of mesh size $2h$.

Proof: For some initial approximations u_-^0, u_+^0 satisfying $-\Delta u_i = f$ on both subdomains (but with $u_-^0 \neq u_+^0$ on the interface), we estimate the update v_i with Dirichlet values each grabbed from the other side, i.e. the problems:

$$\begin{aligned} -\Delta v_i &= 0 \text{ in } \mathcal{T}_i, \\ v_i &= 0 \text{ on } \partial\mathcal{T}_h, \\ v_- &= g_- = (u_+^0 - u_-^0) \text{ on } \Gamma_-, \\ v_+ &= g_+ = (u_-^0 - u_+^0) \text{ on } \Gamma_+. \end{aligned} \tag{5.14}$$

In the following we study the impact of the application of the left subdomain problem on $\mathcal{T}_-$ on the values on Γ. The boundary values on Γ_- are $g_- = u_+^0 - u_-^0$. If we assume N grid points along the line Γ_- the discrete boundary function g_- can be uniformly expressed by its frequencies

$$g_- = \sum_{\mu=1}^{N} a_\mu \sin\Big(\theta(\mu)y\Big), \qquad \theta(\mu) = \mu\frac{\pi}{H}, \quad N = \frac{H}{h}$$

in the grid points. The functions $\sin(\theta(\mu)y)$ tapped in the grid points form a basis of the discrete space. They can be interpreted as the different frequencies of g_-. Parts belonging to μ with $\mu > N/2$ are *high frequencies* since the are not visible on the according mesh with grid size $2h$. In our analysis we are interested in these high frequent error rates, we therefore solve equation (5.14) separately for each frequency. Considering $g_-^\mu = \sin(\theta(\mu)y)$ the analytical solution of the left subproblem reads:

$$v_-^\mu(x,y) = \frac{\sin\Big(\theta(\mu)y\Big)\sinh\Big(\theta(\mu)x\Big)}{\sinh\Big(\theta(\mu)(H+h)\Big)}.$$

Hence the update on the interface line Γ equals

$$v_\Gamma(y)^\mu = \lambda_\mu \sin\Big(\theta(\mu)y\Big), \quad \lambda_\mu = \frac{\sinh\Big(\theta(\mu)H\Big)}{\sinh\Big(\theta(\mu)(H+h)\Big)}. \tag{5.15}$$

Using the definition of the hyperbolic sine, we get

$$\lambda_\mu = \frac{e^{\mu\pi} - e^{-\mu\pi}}{e^{\mu\pi(1+\frac{h}{H})} - e^{-\mu\pi(1+\frac{h}{H})}}$$

If we are interested in the limit $h \to 0$, λ_μ has the following Taylor expansion

$$\lambda_\mu = 1 - \frac{e^{2\mu\pi}+1}{e^{2\mu\pi}-1}\mu\pi\frac{h}{H} + O\left(h^2\right).$$

As expected we get a relationship of λ_μ from the size of the overlap h. Since we are only interested in the high frequencies, worded by $\mu \geq N/2 = H/2h$, i.e. $\theta(\mu) \geq \pi/2h$ and considering that λ_μ is monotonically decreasing with regard to μ (for $m > 2$) we get from (5.15):

$$\begin{aligned}
\lambda_\mu &\leq \frac{\sinh\Big(\frac{\pi}{2h}H\Big)}{\sinh\Big(\frac{\pi}{2h}(H+h)\Big)} \\
&= \frac{e^{\frac{\pi H}{2h}} - e^{-\frac{\pi H}{2h}}}{e^{\frac{\pi(H+h)}{2h}} - e^{-\frac{\pi(H+h)}{2h}}} = \frac{e^{\pi N}-1}{e^{\pi(N+1)}-1}e^{\frac{\pi}{2}} \\
&= \frac{1-e^{-\pi N}}{e^{\pi}-e^{-\pi N}}e^{\frac{\pi}{2}} \to e^{-\frac{\pi}{2}} \approx 0.21,
\end{aligned}$$

if we neglect the higher order terms. Thus we get the desired result.

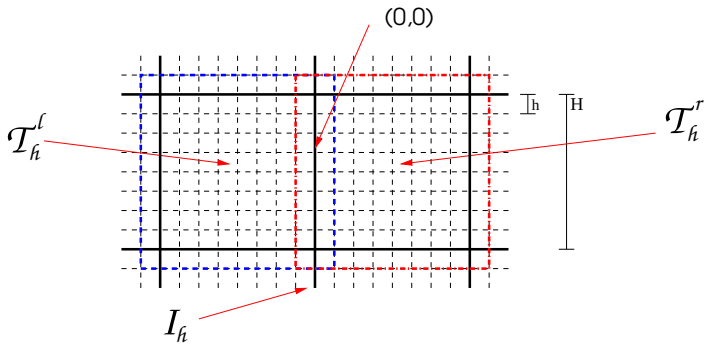

Figure 5.7.: Section of a globally refined mesh $\mathcal{T}_h$ with mesh-size h. Two subdomains $\mathcal{T}_h^l$ and $\mathcal{T}_h^r$ of size H are displayed.

□

Although the preceding analysis is not embedded into the multigrid context, we get an idea of the impact of Schwarz iteration on different error frequencies. In the following, we study the multigrid smoother as used in the implementation. Again we apply elements of Fourier analysis and are limited to the Laplace equation on a reference domain.

5.3.3. Convergence Analysis of the Multigrid Smoother

We analyze the smoothing problem for the Laplace equation on a structured mesh. Figure 5.7 displays two subdomains of a model partitioning. The domain is $\Omega = [0,1]^d$, with N grid points in every direction. The local mesh-size is $h = \frac{1}{N}$. The domain is partitioned into P^d subdomains of size $H = \frac{1}{P}$, a subdomain consists of $N_p = \frac{N}{P}$ grid points in every direction. The size of the subdomains is a natural multiple of the cell-size, i.e. $N_p \in \mathbb{N}$. For simplicity we only regard the two dimensional case. Again we neglect any discretization errors, instead we will map the discrete functions to a continuous Fourier basis:

Definition 5.16 (Fourier Components and Fourier Frequencies)**.** *For the Fourier frequencies $\boldsymbol{\theta} = (\theta_x, \theta_y)$ with $\theta_i = hi$, $i = 1, \ldots, N_p$ we define the Fourier components $\Phi(\boldsymbol{\theta}, \boldsymbol{x})$ as*

$$\Phi(\boldsymbol{\theta}, \boldsymbol{x}) = \sin\Big(\frac{\pi\theta_x}{H+2h}x\Big)\sin\Big(\frac{\pi\theta_y}{H+2h}y\Big).$$

Fourier frequencies $\boldsymbol{\theta} \in [0, \frac{1}{2}hN_p]^d$ are called low frequencies. *Other frequencies are called* high frequencies.

Discrete vectors v_h restricted to the subdomain $\mathcal{T}_h^p$ now have a unique representation using the grid functions:

$$v_h^p = \sum_{\boldsymbol{\theta}} v_{\boldsymbol{\theta}}^p \Phi(\boldsymbol{\theta}, \cdot),$$

coinciding in all grid points $x_i \in \mathcal{T}_h^p$. On the boundary of the extended domain $\widetilde{\mathcal{T}_h^p}$ all functions are continued with zero.

First we analyze the smoothing algorithm 5.14 under the assumption of exact subdomain solution:

Algoritm 5.17 (Multigrid smoother with exact subdomain solution)**.**

1. $d^p = \mathcal{R}^p \sum \mathcal{P}^q \left(b^q - A_h^q v^q\right)$
2. $w^p = \mathcal{R}^p \sum \mathcal{P}^q \left(\widetilde{A^p}\right)^{-1} d^p$
3. $v^p = v^p + w^p$

I.e. on every subdomain $\mathcal{T}^p$ we solve the local problem

$$\widetilde{A^p} w^p = d^p,$$

with homogeneous Dirichlet condition on the nodes closest to the boundary of the subdomain, i.e. on domains of size

$$\widetilde{\mathcal{T}_h^p} = [H + 2h]^d.$$

Lemma 5.18. *Algorithm 5.17 applied to the Laplace problem with homogeneous Dirichlet boundary condition*

$$\begin{aligned} -\Delta u &= f \text{ in } \mathcal{T}_h \\ u &= 0 \text{ on } \partial\mathcal{T}_h, \end{aligned}$$

where $\mathcal{T}_h$ distributed as described in this section is suitable as a multigrid smoother. High frequencies in the defect not visible on the coarse mesh are reduced at least by fixed rate bounded away from 1.

Within the subdomains the new defect will be zero since the local problems are solved exactly, the attention will be on an interface point between two subdomains. Without loss of generality this point is chosen as the origin $(0, 0)$. (See Figure 5.7). Prior to giving the proof we will introduce some more notations following Wienands [Wie01].

Remark 5.19. *For the Fourier components defined in Definition 5.16 there holds:*

$$\begin{aligned} \Phi(\boldsymbol{\theta}, (x-h, y)) + \Phi(\boldsymbol{\theta}, (x+h, y)) &= 2\Phi(\boldsymbol{\theta}, (x, y))\omega(\theta_x), \\ \Phi(\boldsymbol{\theta}, (x, y-h)) + \Phi(\boldsymbol{\theta}, (x, y+h)) &= 2\Phi(\boldsymbol{\theta}, (x, y))\omega(\theta_y). \end{aligned}$$

with

$$\omega(\theta) = \cos\Big(\frac{\pi\theta}{N_p}\Big).$$

With the basic properties of the trigonometrical functions this remark is easily proven. Next we introduce a stencil terminology for the discrete operator $\widetilde{A_h^p}$. For a fixed grid point $x_i \in \mathcal{T}_h^p$, we can describe the operator A_h^p on the space of the grid functions by

$$A_h^p v_h(x_i) = \sum_{\kappa \in J} l_\kappa v_h(x_i + \kappa h),$$

with stencil coefficients $l_\kappa \in \mathbb{R}$ and a certain finite subset $J \subset \mathbb{Z}^d$. This definition is unique for every discrete operator A acting on discrete functions. Using piece-wise bilinear elements for the Laplace equation the stencil in the two dimensional case looks like

$$[l_\kappa] = h^2 \frac{1}{3} \begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix}. \tag{5.16}$$

Combining the stencil notation for the model problem with the definition of the Fourier components the eigenvalues and eigenfunctions of the operator are given by the following remark:

Remark 5.20. *Under the assumption to the partitioning of the domain mentioned in this section and using piece-wise bilinear finite element functions for setting up the modified system matrix $\widetilde{A_h^p}$ on the subdomain $\mathcal{T}_h^p$ for the Laplace problem, we get the following correlation:*

$$\begin{aligned} \widetilde{A_h^p}\Phi(\boldsymbol{\theta}, \boldsymbol{x}) &= \lambda(\boldsymbol{\theta})\Phi(\boldsymbol{\theta}, \boldsymbol{x}) \\ \lambda(\boldsymbol{\theta}) &= \frac{h^2}{3}\left\{8 - 2\left[\omega(\theta_x) + \omega(\theta_y) + 2\omega(\theta_y)\omega(\theta_x)\right]\right\}. \end{aligned}$$

Proof: Using (5.16) we get

$$\widetilde{A_h^p}\Phi(\boldsymbol{\theta}, \boldsymbol{x}) = h^2 \frac{1}{3}\left\{8\Phi(\boldsymbol{\theta}, \boldsymbol{x}) - \Phi(\boldsymbol{\theta}, (x \pm h, y)) - \Phi(\boldsymbol{\theta}, (x, y \pm h)) - \Phi(\boldsymbol{\theta}, (x \pm h, y \pm h))\right\}.$$

With Remark 5.19 we get the desired result.

□

Now we can prove Lemma 5.18. We have to analyze the difference between the defect

$$d^p = \mathcal{R}^p \sum_q \mathcal{P}^p \left(b^q - A_h^q v^q\right) \tag{5.17}$$

before the application of the smoother with the consequent defect

$$d_1^p = \mathcal{R}^p \sum_q \mathcal{P}^p \left(b^q - A_h^q (v^q + w^q)\right) = d^p - \mathcal{R}^p \sum_q \mathcal{P}^p A_h^q w^q = d^p - \mathcal{R}^p A_h w \tag{5.18}$$

after smoothing, where w^q is the local update.

PROOF OF LEMMA 5.18: We write the defect (5.17) using the grid functions (definition 5.16):

$$d^p(\boldsymbol{x}) = \sum_{\boldsymbol{\theta}} d^p_{\boldsymbol{\theta}} \Phi(\boldsymbol{\theta}, \boldsymbol{x}). \tag{5.19}$$

Using Remark 5.20 the solution w^p of the local update problem $\widetilde{A^p_h} w^p = d^p$ is given by

$$w^p(\boldsymbol{x}) = \sum_{\boldsymbol{\theta}} \frac{d^p_{\boldsymbol{\theta}}}{\lambda(\boldsymbol{\theta})} \Phi(\boldsymbol{\theta}, \boldsymbol{x}). \tag{5.20}$$

The defect is analyzed in the grid-point $(0,0)$. The subdomains $\mathcal{T}^p_h$ and $\mathcal{T}^q_h$ meet in this point. The new part of the defect in (5.18) is given by:

$$\begin{aligned} A_h w &= \tfrac{h^2}{3}\left(8w(0,0) - \sum_{i\in\{-1,1\}} [w(i\cdot h,0) + w(0,i\cdot h) + w(-h,i\cdot h) + w(h,i\cdot h)]\right) \\ &= \tfrac{h^2}{3}\Bigg(\underbrace{4w^p(0,0) - \frac{1}{2}w^p(0,\pm h) - w^p(-h,\pm h) - w^p(-h,0)}_{\widetilde{d^p}} \\ &\qquad + \underbrace{4w^q(0,0) - \frac{1}{2}w^q(0,\pm h) - w^q(h,\pm h) - w^q(h,0)}_{\widetilde{d^q}}\Bigg). \end{aligned} \tag{5.21}$$

For using the Fourier components and frequencies we have to utilize different sets of Fourier components for both subdomains. Due to the simple layout of the domain (Figure 5.5) both Fourier components can be expressed with regard to a reference domain:

$$\begin{aligned} \Phi^p(\boldsymbol{\theta}, x, y) &= \Phi(\boldsymbol{\theta}, h - x, y), \\ \Phi^q(\boldsymbol{\theta}, x, y) &= \Phi(\boldsymbol{\theta}, h + x, y). \end{aligned} \tag{5.22}$$

We can continue (5.21) separately for both subdomains:

$$\widetilde{d^p} = \sum_{\boldsymbol{\theta}} w^p_{\boldsymbol{\theta}} \left(4\Phi(\boldsymbol{\theta}, h, 0) - \frac{1}{2}(\Phi(\boldsymbol{\theta}, h, \pm h) - \Phi(\boldsymbol{\theta}, 2h, \pm h) - \Phi(\boldsymbol{\theta}, 2h, 0)\right)$$

Using Remarks 5.19 and 5.20 we get

$$\begin{aligned} \widetilde{d^p} &= \sum_{\boldsymbol{\theta}} w^p_{\boldsymbol{\theta}} \Phi(\boldsymbol{\theta}, h, 0) \left(\underbrace{4 - \omega(\theta_y) - 2\omega(\theta_x) - 2\omega(\theta_x)\omega(\theta_y)}_{=\Psi(\theta)}\right) \\ &= \sum_{\boldsymbol{\theta}} \frac{d^p_{\boldsymbol{\theta}}}{\lambda(\theta)} \Phi(\boldsymbol{\theta}, h, 0)\Psi(\theta). \end{aligned}$$

With (5.22) we get the same result for the right subdomain. With Remark 5.20 the new part of the defect (5.21) yields

$$A_h w = \sum_{\boldsymbol{\theta}} (d^p_{\boldsymbol{\theta}} + d^q_{\boldsymbol{\theta}}) \Phi(\boldsymbol{\theta}, h, 0) \left(\frac{\Psi(\theta)}{2\Psi(\theta) + 2\omega(\theta_x)}\right),$$

and the new defect d_1 is given by

$$\begin{aligned} d_1(0,0) &= d(0,0) - (A_h w)(0,0) \\ &= \sum_{\boldsymbol{\theta}} (d^p_{\boldsymbol{\theta}} + d^q_{\boldsymbol{\theta}}) \Phi(\boldsymbol{\theta}, h, 0) \left(\frac{1}{2} - \frac{\Psi(\theta)}{2\Psi(\theta) + 2\omega(\theta_x)} \right), \\ &= \sum_{\boldsymbol{\theta}} \frac{1}{2} (d^p_{\boldsymbol{\theta}} + d^q_{\boldsymbol{\theta}}) \Phi(\boldsymbol{\theta}, h, 0) \left(\frac{\omega(\theta_x)}{\Psi(\theta) + \omega(\theta_x)} \right). \end{aligned}$$

The factor $\omega(\theta_x)/(\Psi(\theta) + \omega(\theta_x))$ is the reduction rate for the residual to a given Fourier component: Setting $x = \frac{\pi\theta_x}{N^p}$ and $y = \frac{\pi\theta_y}{N^p}$ we have to analyze the function

$$\frac{\omega(x)}{\Psi(x,y) + \omega(x)} = \frac{\cos(x)}{4 - \cos(x) - \cos(y) - 2\cos(x)\cos(y)} \tag{5.23}$$

for high frequencies $\mathcal{H}$ not visible on the coarse mesh $\mathcal{T}_{2h}$:

$$\mathcal{H} = \left\{ (x,y) \in [0,\pi]^2 : \; x \geq \frac{\pi}{2} \vee y \geq \frac{\pi}{2} \right\}$$

Basic analysis yields:

$$\begin{aligned} -\frac{1}{4} \leq \frac{\omega(x)}{\Psi(x,y) + \omega(x)} \leq \frac{1}{3}, &\qquad (x,y) \in \mathcal{H}. \\ \Rightarrow \quad \left| \frac{\omega(x)}{\Psi(x,y) + \omega(x)} \right| \leq \frac{1}{3}, &\qquad (x,y) \in \mathcal{H}. \end{aligned}$$

From this calculation we can conclude, that after one step of smoothing all components of the defect belonging to high frequencies are reduced by a factor of at least $\frac{1}{3}$ independent of the mesh-size and independent of the size of the subdomains. This completes the proof of Lemma 5.18.

□

Inexact Subdomain Smoothing

Up to now, we have analyzed the parallel smoothing Algorithm 5.14 under the assumption of exact subdomain solution. This assumption is necessary for the calculation of the update (5.20) in the proof of Lemma 5.18. There the solution was given by

$$w^p(\boldsymbol{x}) = \sum_{\boldsymbol{\theta}} \frac{d^p_{\boldsymbol{\theta}}}{\lambda(\boldsymbol{\theta})} \Phi(\boldsymbol{\theta}, \boldsymbol{x}).$$

Now let $\mathcal{S}$ be a smoothing operator for the approximation of $\widetilde{A^p}\widetilde{w^p} = d^p$. The ν-fold application of $\mathcal{S}$ is denoted by

$$w^{p,\nu} = \mathcal{S}^\nu(d^p),$$

with the component-wise representation:

$$w^{p,\nu}(\boldsymbol{x}) = \sum_{\boldsymbol{\theta}} w^{p,\nu}_{\boldsymbol{\theta}} \Phi(\boldsymbol{\theta}, \boldsymbol{x})$$

Lemma 5.21. *Let $\mathcal{S}$ be a smoothing operator with the following convergence properties for high frequencies $\boldsymbol{\theta} \in [0,\pi]^d/[0,\pi/2]^d$:*

$$|w_{\boldsymbol{\theta}}^{p,\nu} - w_{\boldsymbol{\theta}}^{p}| \leq c q_{\boldsymbol{\theta}}^{\nu}, \quad \nu \geq \nu_0,$$

with $q_{\boldsymbol{\theta}} \leq q < 1$. Then, the Schwarz iteration with ν-fold application of $\mathcal{S}$ on all subdomains is suitable as smoother for the Laplace problem for some

$$\nu \geq \nu_1 \geq \nu_0,$$

with ν_1 independent of the mesh size h

Proof: After ν-fold application of the smoother $\mathcal{S}$ we have

$$w_{\boldsymbol{\theta}}^{p}(1 - cq^{\nu}) \leq w_{\boldsymbol{\theta}}^{p,\nu} \leq w_{\boldsymbol{\theta}}^{p}(1 + cq^{\nu})$$

The update (5.20) is replaced by the worst possible

$$w^{p,\nu}(\boldsymbol{x}) = \sum_{\boldsymbol{\theta}} \frac{d_{\boldsymbol{\theta}}^{p}}{\lambda(\boldsymbol{\theta})}(1 + cq^{\nu})\Phi(\boldsymbol{\theta}, \boldsymbol{x}).$$

If we carry this modification through the proof of Lemma 5.18 we end up with a modified reduction factor (5.23):

$$\frac{\omega(\theta_x)}{\Psi(\boldsymbol{\theta}) + \omega(\theta_x)} - \frac{1}{2} cq^{\nu} \frac{\Psi(\boldsymbol{\theta})}{\Psi(\boldsymbol{\theta}) + \omega(\theta_x)}$$

Since the additional term is bounded for high frequencies, we can choose some $\nu_1 \geq \nu_0$ such that this term is small enough, more precisely such that

$$\left| \frac{1}{2} cq^{\nu} \frac{\Psi(\boldsymbol{\theta})}{\Psi(\boldsymbol{\theta}) + \omega(\theta_x)} \right| \leq q < \frac{2}{3}.$$

With this choice of ν_1 the proof is finished.

□

5.4. Implementational Aspects

Gascoigne [BB+] is a toolbox for solving partial differential equations with a focus on Navier Stokes, optimization and reactive flows. Although there is no stringent "physical" division, *Gascoigne* – the sequential version as well as the parallel version – mainly consists of three parts:

Mesh Agent The *Mesh Agent* takes care of the mesh geometry, hierarchically refined meshes and the multigrid hierarchy.

Master Process The *Master Process* controls all numerical algorithms, such as newton, multigrid, error estimator's on an algorithmical high level. The Master Process does not possess vectors or matrices.

Solver Process The *Solver Process* manages all the data (mainly matrices and vectors) and performs all direct work on the data; for instance, matrix vector multiplication, assembling of matrices.

The *Mesh Agent* is widely separated from the remaining program and is not used during solving equations. The border between the *Master Process* and the *Solver Process* is rather artificial and is mainly used for separating the data from the numerical algorithms. This segmentation of *Gascoigne* allows for keeping the numerical algorithms in the parallel version unchanged as far as possible.

These three ingredients also form the basic structure of the parallel version of *Gascoigne*. One master CPU holds the Mesh Agent and the *Master Process*, the Solver processes are distributed to the P slave CPU's. A new *Interface Class* replaces the *Solver Process* on the master CPU and forwards the tasks to the slaves. Figure 5.8 illustrates the structure of the implementation. In the following we gather the layout of the parallel program and state some differences to the sequential version:

- The *Master Process* is assigned to one dedicated CPU. Beside controlling the algorithms (this implies distributing the work - but never the data!) there is nearly no numerical work done on this CPU.
- Setting up on the mesh all data is distributed to various *Solver Processes* settled on different CPU's. Communication between the *Solver Processes* is done by these processes in parallel, the *Master Process* is not involved in transferring data. In the sequential program there is only one *Solver Process* running on the same CPU as the *Master Process*.
- In the parallel version, all calls from the *Master Process* to the *Solver Process* are captured by an *Interface Class*. This class controls all needed parallel structure: knowledge of the mesh distribution, layout of CPU's, ...

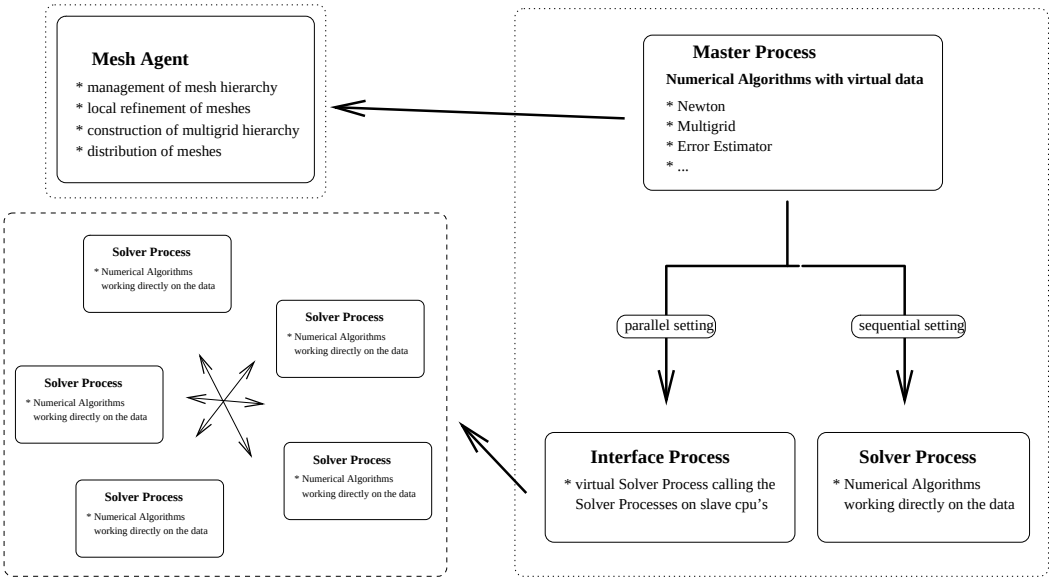

Figure 5.8.: Layout of the parallel structure. The numerical algorithm is controlled by the master CPU, the work is done by several slaves.

5.5. Numerical Study

In this section we perform some numerical studies on the efficiency of the implemented methods. All computations are done on the Linux cluster Helics [Hel] situated in Heidelberg.

- 256 nodes, 2× 1.6 GHz Athlon MP Processor, 2 GB memory
- High-speed Myrinet 2000 network 2 Gbits/sec.

In the following we document the results in three different studies. First, we will look at the multigrid smoother with regard to the smoothing property and the parallel efficiency. As a model problem we consider a two dimensional Laplace equation.

Second we consider a three dimensional Navier-Stokes benchmark flow (see Chapter 3) using global mesh refinement. Finally we connect this Navier-Stokes benchmark flow with error estimation and adaptive mesh refinement.

Quality of the Multigrid Smoother

This example illustrates the quality of the parallel multigrid smoother. Considering the Laplace equation in two dimensions

$$-\Delta u = f,$$

we estimate the convergence rate of the multigrid and the running time for the multigrid solver. First we consider two fixed meshes, one is globally refined and consists of 1 048 576

cells, the other one is locally refined with a random choice of the refined cells and consists of approximately 500 000 cells. Using this meshes we estimate the multigrid convergence rates. For pre- and post-smoothing we apply two smoothing steps. The coarse mesh problem is approximated by a fourfold application of the smoother. The results written in Table 5.1 indicate no negative dependency of the multigrid smoother on the number of subdomains. On the contrary the convergence rates even improve with more subdomains. This can be explained with the overlap between the subdomains. Although this overlap is minimal, degrees of freedom on the interface are smoothed twice in each step.

#CPU:	1	4	16	64	128	256
global refined mesh	0.025	0.024	0.022	0.022	0.018	0.015
local refined mesh	0.020	0.020	0.020	0.011	0.014	0.011

Table 5.1.: Multigrid convergence rates for different numbers of subdomains. The upper line belongs to a Laplace test with structured mesh. The lower line is computed on locally refined meshes.

cells	CPU's	time (sec)
4^8	1	21.16
4^9	4	20.45
4^{10}	16	23.69
4^{11}	64	25.65
4^{12}	256	38.10*

Table 5.2.: Running time for the Laplace Test. The problem size grows linear with the number of CPU's. (*) The poor value for the largest problem is caused by the network topology of *Helics*. Whilst 64 nodes are linked with a high-speed network, between this groups of computers the network speed is lower.

In Table 5.2 we list the time necessary for the solution of the Laplace problem, without considering the time for mesh adaption, file i/o or error estimation. The problem size is increased linearly with the number of CPU's. Ideal parallel efficiency $O(N/P)$ would result in a constant time for all problems.

3D Navier-Stokes Flow

We consider the flow through a channel presented in Chapter 1 and already analyzed in Chapter 4 with an obstacle as shown in Figure 1.1. The computation is performed on different numbers of subdomains. Each computation is performed on four succeeding meshes originated from global mesh refinement or until the memory is exhausted.

If we use the overall running time on one level as characteristic for the comparison, slight differences in the partitioning lead to large discrepancies in the resulting times. For example

if the partitioning of a mesh is not balanced enough to continue with the parallel global refinement strategy described at the end of Section 5.2.4 the refined mesh has to be newly partitioned. The question of when to apply which refinement strategy, i.e. the decision, if a mesh is assumed to be balanced depends on the choice of some tolerance parameter. A small exceeding of this parameter can lead to large increase in running time due to the costly mesh refinement using the administrational master process without delivering a superior partitioning. Nevertheless a pure comparison of the solving time would conceal the difficult (and up to now not parallelized) part in finite element computations. In Table 5.3 we therefore note the overall running time for this example on different numbers CPU's and different problem sizes. The results considering the sequential test case were obtained by using the original sequential version of Gascoigne. We require at least 500 cells per CPU. The problems belonging to coarse mesh levels do not use all available CPU's, therefore we get a stagnation of the times on small meshes. A direct comparison and estimation of parallel efficiencies is difficult since e.g. in the compute level belonging to 983 040 dof's the mesh was refined in parallel for small number of CPU's (up to 16) while we have to involve the master process for 32 and more CPU's.

#dof #CPU	1	4	8	16	32	64	128
18 720	57s	23s	13s	15s	16s	15s	16s
136 000	295s	103s	45s	35s	56s	60s	62s
983 040	–	863s	364s	207s	139s	127s	167s
7 864 320	–	–	–	–	812s	509s	312s

Table 5.3.: Overall running times for every level of refinement. Times include solution of the problem and mesh handling. If no values are given, the memory was exhausted.

#dof #CPU	1	4	8	16	32	64	128
18 720	51s	20s	10s	11s	10s	11s	10s
136 000	244s	90s	39s	29s	19s	19s	20s
983 040	–	754s	319s	187s	99s	69s	67s
7 864 320	–	–	–	–	720s	467s	291s

Table 5.4.: Running times for the actual solution of the problem without mesh handling. If no values are given, the memory was exhausted.

In Table 5.4 the running times only for the solving process is given. To get an idea of the parallel efficiency we list some index values describing the "time per calculated degree of freedom":

$$pe_i = 100\frac{time(sec)\cdot \#CPU}{\#dof's}. \tag{5.24}$$

The constant factor 100 is just for scaling reasons. If we would have optimal parallel complexity $O(N/P)$ (which we do not and cannot have!) the index pe_i should be constant. In Table 5.5 we list these indices.

#cells #CPU	1	4	8	16	32	64	128
18 720	0.27	0.43	0.43	0.94	1.71	3.76	6.84
136 000	0.18	0.26	0.23	0.34	0.45	0.89	1.88
983 040	-	0.31	0.26	0.30	0.32	0.45	0.87
7 864 320	-	-	-	-	0.29	0.38	0.51

Table 5.5.: Index for parallel efficiency ($time(sec) \cdot \#cpu/\#dof's$). Small values indicate good efficiency.

Considering this example we can conclude that the implemented methods have a parallel efficiency within reasonable limits. Although the meshing process does not support parallel computers, the method is well suited for the observed problems (considering the according problem size). Problems are expected if the work aligned with the meshing process is large in comparison to the effort of the actual solving process. This happens if we consider small scalar problems in two dimensions. But, parallelization was taken into account for large systems of equations, the Navier-Stokes equation is viewed just as a small test case.

Adaptive Refinement

This third example treats the 3D Navier Stokes benchmark with adaptive mesh refinement. Table 5.6 gives the overall running time per iteration including solving the problem, estimating the error and adaption of the meshes. Tables 5.7 and 5.8 list the running times and the efficiency index for the solution of the primal problem.

A short view on this last table indicates the quality of the parallelization. For the observed configuration the parallel efficiency does not degenerate. In Figure 5.9 we draw the parallel efficiencies

$$E(N,P) = \frac{T_S(N)}{P \cdot T_P(N,P)},$$

for this problem. Since large problems cannot be solved on a single processor, the running times for the sequential problem on large meshes are extrapolated with the assumption $pe_i = 0.53$ taken from Table 5.8.

Considering the rather small problem size and the adaptive mesh refinement the obtained parallel efficiencies are satisfactory. One has to take into account the additional master CPU which reduces the theoretically possible efficiency to

$$E(N,P) \leq \frac{P-1}{P}.$$

This explains the "low" efficiency of about $\frac{3}{4}$ for the case of 4 CPUs.

For the computations done for Chapter 4 the parallel version of *Gascoigne* was already used.

#dof #CPU	1	4	8	12	16
3 696	31	13	13	13	13
17 608	139	51	38	35	37
92 832	866	262	151	119	110
526 360	-	-	781	587	446
1 832 376	-	-	-	-	937

Table 5.6.: Parallel running time for the total iteration. Adaptively refined meshes.

#dof - #CPU	1	4	8	12	16
3 696	23	8	8	8	9
17 608	94	33	24	24	25
92 832	491	170	93	70	64
526 360	-	1009	477	323	259
1 832 376	-	-	-	-	872

Table 5.7.: Parallel running time for the solution of the primal problem. Adaptively refined meshes.

#dof - #CPU	1	4	8	12	16
3 696	0.62	0.87	1.73	2.60	3.90
17 608	0.53	0.75	1.09	1.64	2.27
92 832	0.53	0.73	0.80	0.90	1.10
526 360	-	0.77	0.72	0.74	0.79
1 832 376	-	-	-	-	0.76

Table 5.8.: Index for parallel efficiency. Adaptively refined meshes.

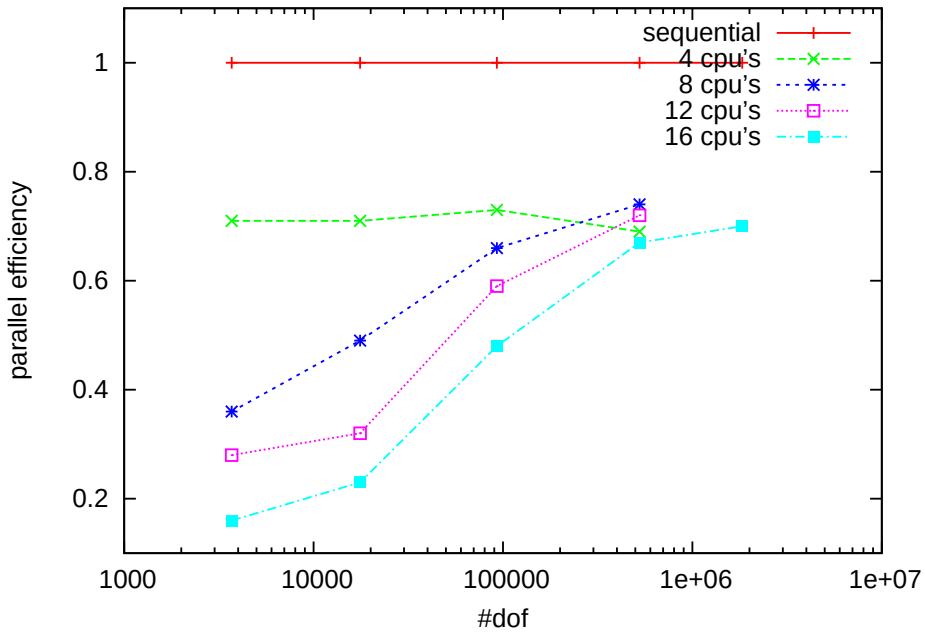

Figure 5.9.: Parallel efficiency for adaptively refined meshes $E(N,P) = \frac{T_S(N)}{T_P(N,P)}$ for different number of CPUs.

Conclusion

The numerical experiments in the last section have shown good parallel efficiencies for three dimensional Navier-Stokes flows on adaptively refined meshes. Without repeating the details, an efficiency of about 3/4 for reasonable proportions of problem size and number of CPU's can be observed.

The numerical methods are – with the exception of the ILU smoother – unchanged and still globally coupled. Therefore nearly no lack of the convergency rate is present when distributing the problems to different processors.

While the solution of the problems (primal and dual) as well as the error estimation is parallelized, the mesh handling is completely sequential. For very large problems, this will become the bottle-neck. However considering reactive flows, the situation is not that delicate, as we will see in the next chapter.

6. Reactive Flows

Gathering the numerical methods discussed in the previous chapters we now face complex problems involving chemical reactions. Details on the finite element solution of reactive flows are given in Braack [Bra98] or Becker, Braack & Rannacher [BBR99]. The system to be analyzed is the methane burner already introduced in the introduction, Chapter 1. The configuration describes a typical household burner used e.g. for water heating.

Considering this setting we have to deal with several problems: due to the complex geometry including some cooling devices we cannot apply a two dimensional reduction which still exhibits the main feature. The lamellae, which lead the fuel to the flame, impose edges entering into the computational domain and bring about singularities in the solution. Thus local mesh adaption will be crucial for a rigorous examination of the configuration.

We will use two different reaction systems for the simulation: the so-called C1 mechanism with 15 chemical species and 42 reversible reactions of Smooke, see [SMK89], and the more detailed C2 mechanism with 39 chemical species and 151 reversible reactions of Warnatz [WMD96]. This mechanism includes formation of several chemical species containing two carbon atoms. Both reactions systems are given in Appendix B.

The used reaction systems with 15 or 39 chemical species lead to a large, very stiff system matrix. This negatively influences the solution of the linear systems in two ways: advanced, stabilized methods are needed to gain convergence at all, while the size of the problem leads to such a large computational effort that parallel computers are necessary just to cope with the problem size; on the other hand we need globally coupled methods for treating the stiff systems. In Chapter 5 we described the parallelization process which virtually completely preserves the global algorithms. The presented multigrid solver is suitable for the equations under consideration.

In the progress of this chapter we will figure out the equations engaged, discuss the finite element discretization and the resulting linear systems. The focal point is set on the latter – the solution of the systems. Finally, we will present numerical results for the prescribed configuration, which will be introduced with more detail.

6.1. Equations

Reactive flow problems are described by a fully coupled system of equations which was already given in the introduction (1.2)-(1.5), namely the conservation of total mass

$$\partial_t \rho + \operatorname{div}(\rho v) = 0\,, \tag{6.1}$$

and of momentum

$$\rho\partial_t v + \rho(v\cdot\nabla)v - \operatorname{div}\pi + \nabla p = \rho g. \tag{6.2}$$

Further the equation describing the temperature

$$\rho c_p\partial_t T + (\rho c_p v + \alpha)\cdot\nabla T + \operatorname{div}(-\lambda\nabla T + \mathcal{Q}_{Duf}) + \pi : \nabla v - v\cdot\nabla p - \partial_t p = -\sum_{k=1}^{n_s} h_k m_k \omega_k\,, \tag{6.3}$$

and for each of the n_s species one diffusion-convection-reaction equation

$$\rho\partial_t w_k + \rho v\cdot\nabla w_k + \operatorname{div}\mathcal{F}_k = m_k\dot{\omega}_k, \tag{6.4}$$

describes the mass fraction of the species k. These conservation equations are completed by the ideal gas law

$$p = \frac{\rho RT}{\overline{m}}, \tag{6.5}$$

with the universal gas constant $R = 8.31451$, and mean molar weight $\overline{m}$ given by

$$\overline{m} = \left(\sum_{k=1}^{n_s}\frac{w_k}{m_k}\right)^{-1}.$$

If one sums up the species equations (6.4), it follows with (6.1)

$$\sum_{k=1}^{n_s} w_k = 1.$$

Therefore, we omit one equation in (1.5), say that of the last species, and use instead

$$w_{n_s} = 1 - \sum_{k=1}^{s} w_k\,.$$

The viscous tensor in (6.2) is given by

$$\pi = \mu\left\{\nabla v + (\nabla v)^T - \frac{2}{3}(\operatorname{div} v)I\right\},$$

where μ is the shear viscosity. The convective direction α

$$\alpha = \sum_{k=1}^{n_s} c_{p,k}\mathcal{F}_k \tag{6.6}$$

is associated with an enthalpy flux due to diffusion fluxes of species having different specific heat capacities $c_{p,k}$. In the governing equations, the Dufour effect given by the term

$$\operatorname{div}\mathcal{Q}_{Duf}$$

describes the heat flux due to concentration gradients. It is generally accepted to be of minor importance for the present flame configurations. Furthermore, heat release by pressure convection, volume viscosity effects, and viscous dissipation are neglected because of their small impact on laminar flames.

Details on the modeling of chemical reactions are given in Williams [Wil85] and Appendix A.

6.2. Simplified Model for Chemically Reacting Flows

For the aimed three dimensional simulations we simplify equations (6.1)-(6.5) by neglecting terms with low influence. In the temperature equation (6.3) we neglect the terms arising from the friction of the fluid in itself and all pressure effects on the temperature. As mentioned, the Dufour effect is not taken into account, further we neglect the term involving α which is small.

The chemical source terms are denoted by f_k and f_T for species and temperature, respectively:

$$\begin{aligned} f_T(T,w) &= -\sum_{k\in S} h_k m_k \omega_k, \\ f_k(T,w) &= m_k \dot{\omega}_k. \end{aligned}$$

The diffusion fluxes are strongly simplified by Fick's law (see Appendix A for details):

$$\mathcal{F}_k \approx -D_k \nabla y_k, \quad D_k := \rho D_k^*.$$

Gathering this simplifications, equations (6.1)-(6.5) are reduced to:

$$\partial_t \rho + \operatorname{div}(\rho v) = 0, \tag{6.7}$$

$$\rho \partial_t v + \rho (v\cdot\nabla) v - \operatorname{div}(\mu \nabla v) + \nabla p = \rho g, \tag{6.8}$$

$$\rho c_p \partial_t T + \rho c_p (v\cdot\nabla) T - \operatorname{div} \lambda \nabla T = f_T(T,w), \tag{6.9}$$

$$\rho \partial_t w_k + \rho (v\cdot\nabla) w_k - \operatorname{div} \rho D_k^* \nabla w_k = f_k(T,w),\ k = 1\ldots, n_s - 1, \tag{6.10}$$

$$p = \frac{\rho R T}{\overline{m}}. \tag{6.11}$$

Pressure Splitting for Flows at low Mach Number

The difference between compressible and incompressible flow is the density, which may vary in the first case and is constant in the incompressible case. The Mach number is defined as the ratio of the speed of sound c and the velocity of the flow v:

$$Ma = \frac{v}{c}.$$

Considering laminar flames, the maximal velocity of the flow is approximately $2m/s$ which leads to a low Mach number of about $Ma = 0.01$. The treatment of low Mach number flows with the classical Navier-Stokes equations including chemical reactions leads to numerical instabilities due to huge amplification of round-off errors. Therefore, we split the pressure into two parts, the thermodynamic pressure P_{th} which is constant in space and does not appear in the momentum equation (6.8) and the hydrodynamic p_{hyd} which is small and neglected in the gas law (6.11). See Braack [Bra98] for details:

$$p(x,t) = P_{th}(t) + p_{hyd}(x,t).$$

The thermodynamic pressure P_{th} is determined as the average pressure in Ω. The gas law is now an algebraic equation for the density instead of an equation for the total pressure. The derivatives of the density in the continuity equation (6.7) are expressed by the derivatives of the temperature T and the mean molecular mass $\overline{m}$:

$$\begin{aligned}\operatorname{div}(\rho v) &= \rho\left(\operatorname{div} v + \frac{1}{\rho} v\cdot\nabla\rho\right)\\ &= \rho\left(\operatorname{div} v + \frac{1}{\overline{m}}(v\cdot\nabla)\overline{m} - \frac{1}{T}(v\cdot\nabla)T + \frac{1}{p}(v\cdot\nabla)p\right),\end{aligned}$$

where the part regarding the pressure is neglected. The final system reads:

$$\operatorname{div} v + \frac{1}{\overline{m}}(v\cdot\nabla)\overline{m} - \frac{1}{T}(v\cdot\nabla)T = 0, \tag{6.12}$$

$$\rho\partial_t v + \rho(v\cdot\nabla)v - \operatorname{div}(\mu\nabla v) + \nabla p_{hyd} = \rho g, \tag{6.13}$$

$$\rho c_p\partial_t T + \rho c_p(v\cdot\nabla)T - \operatorname{div}\lambda\nabla T = f_T(T,w), \tag{6.14}$$

$$\rho\partial_t w_k + \rho(v\cdot\nabla)w_k + \operatorname{div}\mathcal{F}_k = f_k(T,w),\ k=1\ldots,n_s-1, \tag{6.15}$$

$$\rho = \frac{p\overline{m}}{RT}. \tag{6.16}$$

6.3. Finite Elements for Reactive Flows

The finite element discretization is based on the usual Galerkin formulation of (6.12)-(6.15) and the algebraic equation (6.16) for determining the density. First we will arrange the function spaces of the searched solutions. The boundary $\partial\Omega$ of the domain is split into some part where Dirichlet boundary conditions are applied Γ_d and the remaining part with boundary conditions of Neumann or Robin type. This splitting must not coincide for all components of the solution. We therefore denote the corresponding solution component with an additional index. As discussed in the introductory Chapter 2 the test space for the velocities v is defined by

$$V := \{v\in[H^1(\Omega)]^d :\ v|_{\Gamma_d^v} = 0 \text{ almost everywhere}\}.$$

The test spaces for the temperature and the chemical species are corresponding spaces R and R_k, with the necessary modifications for Dirichlet values:

$$\begin{aligned}R &:= = \{T\in H^1(\Omega) :\ T|_{\Gamma_d^T} = 0 \text{ almost everywhere}\}.\\ R_k &:= = \{w_k\in H^1(\Omega) :\ T|_{\Gamma_d^{w_k}} = 0 \text{ almost everywhere}\}.\end{aligned}$$

The test space for the pressure is given by $p\in Q := L^2(\Omega)$ if its absolute value is fixed by some Robin boundary condition and $Q := L^2(\Omega)/\mathbb{R}$ otherwise. For abbreviation we denote the product of the test spaces by

$$X := Q\times V\times R\times\{R_k\}.$$

Now we can give the weak formulation of equations (6.12)-(6.16) with help of a set of test functions $\Phi = (\xi,\varphi,\psi,\{\psi_k\})\in X$. Since we are looking for stationary solutions, we give the

stationary form of the equations. The equation for the conservation of mass is transformed to

$$a_1(u)(\xi) := (\operatorname{div} v, \xi) + (\frac{1}{\overline{m}}(v \cdot \nabla)\overline{m}, \xi) - (\frac{1}{T}(v \cdot \nabla)T, \xi), \tag{6.17}$$

the momentum conservation to

$$a_2(u)(\varphi) := (\rho(v \cdot \nabla)v, \varphi) + (\mu \nabla v, \nabla \varphi) - (p_{\text{hyd}}, \nabla \cdot \varphi) - (\rho g, \varphi), \tag{6.18}$$

with the implicitly given boundary term:

$$\int\limits_{\partial\Omega/\Gamma_d^v} \left(\mu \frac{\partial v}{\partial n} - pn\right) \varphi \, \mathrm{ds} = 0.$$

From the temperature equation (6.14) we derive

$$a_3(u)(\psi) := (c_p \rho(v \cdot \nabla)T, \psi) + (\lambda \nabla T, \nabla \psi) - (f_T(T, w), \psi), \tag{6.19}$$

with the additional boundary integral

$$\int\limits_{\partial\Omega/\Gamma_d^T} \lambda \frac{\partial T}{\partial n} \psi \, \mathrm{ds} = 0.$$

The $n_s - 1$ equations for the species are transformed to

$$a_4(u)(\psi_k) := \sum_{k=1}^{n_s-1} (\rho(v \cdot \nabla)w_k, \psi_k) + (\rho D_k \nabla w_k, \nabla \psi_k) - (f_k(T, w), \psi_k), \tag{6.20}$$

where again a boundary integral remains from integration by parts

$$\sum_{k=1}^{n_s-1} \int\limits_{\partial\Omega/\Gamma_d^{w_k}} \frac{\partial w_k}{\partial n} \psi_k \, \mathrm{ds} = 0.$$

These equations are enclosed by the algebraic equation for the pressure (6.16). With

$$a(u)(\Phi) := \sum_{j=1}^{4} a_j(u)(\Phi),$$

the complete system of equations in weak formulation is given by: find $u \in u_0 + X$, such that

$$a(u)(\Phi) = 0, \quad \forall \Phi \in X, \tag{6.21}$$

where u_0 is a continuation of the Dirichlet boundary values on Γ_d into the domain.

Simular to the discussion in Chapter 3, the actual finite element discretization is achieved by replacing the function space X by a finite dimensional one X_h consisting of piecewise polynomials. Again we have to face two kinds of instabilities connected with the discretization of (6.21). The first type of instabilities occurs, if the discrete function spaces do not fulfill

the "inf-sup" condition for the Stokes equations. As for the Navier-Stokes equations we will apply local projections stabilization (LPS) (see Becker, Braack, [BB01]) for this "pressure instabilities". The second kind of instabilities is aroused by convective terms. As described for the Navier-Stokes equations we will apply convection stabilization based on sub-grid modeling for the velocity, the temperature and the chemical species. For this purpose additional stabilization terms are added to the Galerkin formulation (6.21).

Actually, the choice of the finite element triangulation does not differ from the discretization of pure Navier-Stokes flows as described in the precious chapter. Neglecting the nonlinearity in the transformation T_K – which should be in reasonable limits – the shape of the elements is again only restricted by the minimum angle condition, see Section 3.4 for details.

6.3.1. Stabilization by Local Projections

The stabilization scheme discussed in Chapter 3 for the Navier-Stokes equations is transfered to reactive flow equations. For stabilizing the pressure, i.e. for achieving inf-sup stability we add the known bilinear form (3.8):

$$s_{LPS}(u, \Phi) = \sum_{K \in \mathcal{T}_h} \alpha_K \left(\nabla \pi p_{\text{hyd}}, \nabla \pi \xi \right)_K ,$$

with the cell-wise defined parameter $\alpha_K \approx h_K^2$ and the fluctuation operator $\pi := (id - i_{\tilde{Q}_h})$, with $i_{\tilde{Q}_h} : Q_h \to \tilde{Q}_h$ some projection into a inf-sup stable subspace. An analysis of this stabilization scheme is given in Chapter 3, further details are found in Becker & Braack [BB01, BB04]. On anisotropic meshes we use the modifications given in Section 3.4. In regions with dominant convection we utilize further stabilizations terms – again following the previously described scheme:

$$\begin{aligned} s_{CONV}(u)(\Phi) \quad = \quad & \sum_{K \in \mathcal{T}_h} \Big\{ \delta_K ((v \cdot \nabla)\pi v, (v \cdot \nabla)\pi \varphi)_K \\ & + \gamma_K ((v \cdot \nabla)\pi T, (v \cdot \nabla)\pi \psi)_K \\ & + \sum_k \gamma_{k,K} ((v \cdot \nabla)\pi w_k, (v \cdot \nabla)\pi \psi_k)_K \Big\}, \end{aligned}$$

where the cell-wise stabilizations parameters depend on the viscosity, the heat conduction and the chemical diffusion coefficients and are defined by

$$\begin{aligned} \alpha_K &= \alpha_0 f(\|\mu\|_{\infty,K}) \\ \delta_K &= \delta_0 f(\|\mu\|_{\infty,K}) \\ \gamma_K &= \gamma_0 f(\|\lambda\|_{\infty,K}) \\ \gamma_{i,K} &= \gamma_0 f(\|D_k\|_{\infty,K}) \\ \text{with } f(x) &= \left(\frac{6x}{h_K^2} + \frac{\|\rho v\|_{\infty,K}}{h_K} \right)^{-1}, \end{aligned}$$

with the modifications for anisotropic meshes as described in Section 3.4.

6.3.2. Solution Process

As discussed in the introduction, solving the presented equations is very difficult due to the size of the resulting problems. Another problem arises due to the quality of the equations. Beyond the pure complexity, the solution of the discrete problems is still difficult, among other things due to the stiffness of the source terms. In the following, the solution process and the utilized numerical methods are shortly described. Details are found in Braack [Bra98].

The outermost method for solving the nonlinear equations is a standard quasi-Newton solver. Due to the already mentioned properties of the equations, the solution of the linear systems within the Newton algorithm is more delicate. Shortly spoken we use a GMRES solver (see Saad [Saa03]) with a multigrid iteration as preconditioner. While the GMRES solver is standard, details on the multigrid solver with respect to adaptively refined meshes are given in Becker & Braack [BB00]; Chapter 5 of this work treats the parallelization aspects of the multigrid solver. Considering Navier-Stokes flows, we use a stabilized ILU decomposition of the matrix as smoother. In the parallel version of the multigrid solver, this smoothing is no longer a global operation (see Section 5.3). In reactive flow context, this may lead to problems. We therefore further stabilize the smoothing operation by enclosing the ILU into a BiCGStab solver, another Krylow space method (see Saad [Saa03] for details on BiCGStab). The entire smoothing process is replaced by this linear solver. The ILU now acts as a preconditioner for the BiCGStab iteration. A benefit is achieved due to some global – i.e. on the whole domain – orthogonalization and the line-search used within the linear solver. The BiCGStab iteration is not used as a self-contained solver for the smoothing problem, instead a fixed number of iterations is applied.

6.3.3. Homotopy Methods

The regarded equations are highly nonlinear. Although we use stable algorithms throughout the entire solution process, the acquirement of solutions is very tough without a good initial guess. In the following we describe some strategies helpful for getting started:

- If the geometry allows for some – even crude – two dimensional restrictions, we use a prolongation of the corresponding 2d solution as starting value for the Newton solver. Thus, other techniques for reaching initial solutions are also applied to the 2d case.
- Simulations with the complex C2 mechanism are always initiated with the C1 mechanism. The corresponding solution of the C1 mechanism is expanded to the C2 mechanism by a local 0d time-stepping in every node of the mesh.
- Although being interested in stationary solutions, we use the implicit Euler as time stepping scheme for the solution process. The additional mass matrix stabilizes the system matrix. For really small time-steps, the system matrix can be regarded as a distortion of the identity. The employment of time stepping methods to get the desired solution may be very slow, but combined with time step control it marks out the most reliable strategy for gaining solutions. Time-step control is not aligned to some error analysis, but to the convergence rate of the nonlinear problem. The time step is chosen as large as possible, such that a solution of the problems is still possible.

- On coarse meshes the solution may not be adequately representable. The flame front, or layers of radicals may be too sharp to be described on large cells. By introducing additional diffusion to the system, a solution is found more easily. This is controlled by some homotopy parameters, i.e., by replacing all diffusion parameters in equations (6.12)-(6.16) with

$$\begin{aligned} \mu' &= H_\mu \mu \\ \lambda' &= H_\lambda \lambda \\ D'_k &= H_{D_k} D_k, \end{aligned}$$

  or even with $H_\mu = H_\lambda = H_{D_k} = H$. These parameters have to be chosen in reasonable limits, to simplify the solution process, but without leading to "false" branches. One could think of estimating the homotopy parameters with a-posteriori error estimators for model errors as given in Braack & Ern [BE03], but the introduction of the homotopy parameters is a must for getting a solution at all and not subject to error control.

All this strategies are applied interwoven and there is no stringent overall control of the parameters. While the time-step can be adjusted with the convergence rate of the Newton solver, the homotopy parameters may be reduced after each step of refinement. The automatic acquirement of initial solutions is still subject to research and up to now, user-interaction is necessary.

6.3.4. Advanced Linear Algebra

Braack [Bra98] developed an efficient method for storing the matrix entries. He considered two dimensional flow problems with linear finite elements. The method is based on the lumping of zero-th order terms in the matrices.

To describe the storage structure we have to give some details on the matrices. They are assembled as the Frechet derivatives of the semi-linear forms (6.17)-(6.20). For instance each of the species equations (6.20) is linearized in the direction $u^\Delta := (p^\Delta, v^\Delta, T^\Delta, \{w_k^\Delta\})$ by:

$$\begin{aligned} a'(u)(u^\Delta, \Phi) =& (\rho(v\cdot\nabla)w_k^\Delta, \psi_k) + (\rho D_k \nabla w_k^\Delta, \nabla\psi_k) - \Big(\frac{\rho}{T} D_k \nabla w_k T^\Delta, \nabla\psi_k\Big) \\ &+ \big(\rho(v^\Delta\cdot\nabla)w_k, \psi_k\big) - \Big(\frac{\rho}{T}(v\cdot\nabla)w_k T^\Delta, \psi_k\Big) - \sum_j \left(\frac{\partial f_k}{\partial w_j}(T,w) w_j^\Delta, \psi_k\right), \end{aligned} \tag{6.22}$$

where we have only taken out the derivatives of $\overline{m}$ whose influence is usually small. The derivatives of the other forms in (6.21) are derived in the same way. If we gather all solution components aligned in one node of the mesh, the matrix features a block structure with all couplings in this point gathered:

$$A_{ij} = \begin{bmatrix} B_{pp} & B_{pv} & B_{pT} & B_{pw} \\ B_{vp} & B_{vv} & B_{vT} & B_{vw} \\ B_{Tp} & B_{Tv} & B_{TT} & B_{Tw} \\ B_{wp} & B_{wv} & B_{wT} & B_{ww} \end{bmatrix}. \tag{6.23}$$

Each entry of A_{ij} is itself a matrix. For large reaction systems the block B_{ww} is dominant, a matrix of size:

$$B_{ww} \in \mathbb{R}^{n_s-1 \times n_s-1},$$

with all derivatives of the species equations. Amongst others in (6.22) B_{ww} includes for every species equations the sum

$$\cdots \sum_j \left(\frac{\partial f_k}{\partial w_j}(T,w) w_j^{\Delta}, \psi_k \right) \ldots .$$

Since the derivatives of the source terms $f_k(T,w)$ do not vanish, the matrix B_{ww} is a full matrix. However, only this source terms bring about couplings between different species. These couplings are of zero-th order. If we perform the numerical integration with the trapezoidal rule or the Simpson rule for quadratic finite elements respectively, the block B_{ww} is diagonal for all off diagonals A_{ij} with $i \neq j$. This very fact leads to the possibility of significantly reducing the storage effort. Again the system matrix A is composed in a block-wise manner. Now, we use different block structures for diagonal and off diagonal entries of A. While we store the full block (6.23) in the diagonal entries A_{ii}, we use a reduced structure (6.24) elsewhere. For the species couplings we utilize the matrix D_{ww} – now a diagonal one, the couplings between the species and the flow field are entirely left apart. Whilst the lumping of zero-th order terms does not change the properties of the solvers, neglecting some of the couplings reduces the Newton convergency.

The off diagonal blocks of the system matrix are given by

$$A_{ij} = \begin{bmatrix} B_{pp} & B_{pv} & B_{pT} & \\ B_{vp} & B_{vv} & B_{vT} & \\ B_{Tp} & B_{Tv} & B_{TT} & \\ & & & D_{ww} \end{bmatrix}, \quad \forall i \neq j. \tag{6.24}$$

If we compare the storage requirements for both types of matrix blocks, we end up with $(n_s+4)^2$ entries in the full block against $(24+n_s)$ entries in the reduced one. Considering linear finite elements with 27 matrix couplings in every row, the full matrix on a mesh with N grid points has

$$27(n_s+4)^2 N$$

entries. Using different matrix blocks for the off-diagonals, the storage usage reduces to

$$\left((n_s+4)^2 + 26(24+n_s)\right) N.$$

If we use the C2 reaction mechanism with $n_s = 39$ species, the memory needed to store a matrix is a factor 14.5 smaller than using the standard matrix. For the C1 mechanism, we save a factor of 7.5.

The derivatives given in (6.22) only account for the Galerkin formulation. One also has to consider the stabilization terms. At this point we recall the discussion on stabilization techniques from Section 3.1. The application of SUPG techniques as presented by Hughes

et al. [HFM86, BH82] to reactive flows would yield additional terms in the species equations. Among others, due to the source terms, the stabilization includes

$$s_{SUPG}(u)(\Phi) = \cdots \sum_{k=1}^{n_s-1} (f_k(T,w), (v\cdot\nabla)\psi_k)\ldots$$

This expression introduces additional couplings between all species among one another, since the source terms depend on all species. And unlike the couplings in the Galerkin term (6.22), these are first order couplings, where we cannot apply lumping. Thus there is no possibility of reducing the matrix B_{ww} to a diagonal matrix for off diagonals A_{ij}.

The usage of the local projection method adds the stabilization terms

$$s_{LPS}(u)(\Phi) = \cdots \sum_{k=1}^{n_s-1} \gamma_{i,K} \left((v\cdot\nabla)\pi w_k, (v\cdot\nabla)\pi\psi_k\right)_K \ldots$$

with only diagonal couplings of the species. No additional couplings between different species at all appear. Thus, solely the usage of local projection stabilization techniques allows for a reduction of the storage requirements by a factor of about 10 for the discussed setting.

Parallelization Aspects

Reactive flows do not introduce new difficulties to parallelization; in contrast, the chosen approach of parallelization is optimally suited for multicomponent problems, where the size of the matrix (and therefore the main computational effort) is utterly dominant. The bottleneck of the parallelization procedure presented in Chapter 5 is the maintenance and the distribution of the meshes. Considering a 3D problem with 39 chemical species, the solution exhibits 44 components. The number of solution components squared enters the matrix size. Thus, the system matrix for flow with 44 components is more than 120 times larger than the matrix for Navier-Stokes on the same mesh; therefore the share of effort used in every iteration for the handling of the meshes is 120 times smaller. The overall complexity estimate is crudely given by

$$T_P(N,P) = c_{mesh}N + 120c_{num}\frac{N}{P}.$$

Recalling the discussion regarding the parallel matrix vector product in Section 5.2.2, we expect a parallel efficiency of about 0.5 if we cluster about 10 000 nodes in every subdomain. Due to the large system matrices aligned with reactive flow problems, we get away with just 500 nodes per subdomain while maintaining the same parallel efficiency.

6.4. Numerical Study of a Methane Burner

In this section we use the proposed methods for a detailed analysis of a methane burner. Figure 1.2 illustrates the assembly of this burner. From beneath, fuel and air are fed into mixing ducts where a stoichiometric mixture of methane and air is established. This mixture

flows through a set of slots (with a uniform width of 2 mm each) between lamellae of different height (and a width of 1.5 mm each). Above this lamellae the actual flame is settled. The lamellae cool down the flow in order to prevent the flame from moving into the burner. Some cooling pipes are installed into the lamellae to prevent these from getting to hot. Since the overall size of the burner is quit large with respect to the local size of the slots, we single out some reference geometry and assume an uniform continuation to all directions. The right diagram in Figure 1.2 in the introduction shows this "unit"-configuration. On the lower part of the computational domain, we describe a Dirichlet inflow condition for mixed fuel and the oxidizer at room temperature. On the lamellae and the cooling pipe a no-slip condition for the velocity is prescribed. The temperature is enforced by a Dirichlet condition to values obtained by measurements, i.e., a profile from 372 to 490 Kelvin on the lamellae and 323 Kelvin on the cooling pipe. On the top, we have the usual "do-nothing" outflow condition together with homogeneous Neumann conditions for the temperature and the chemical species. All calculations are performed with both chemical reaction mechanisms involving 15 species and 84 elementary reactions and 39 species and 304 elementary reactions, respectively. These reaction systems are given in Appendix B.

A comparable burner, but with only two different lamellae sizes, was studied by Parmentier et al. [PBRW03]. The authors had the possibility to compare 1d and 2d simulations of the burner with experiments. Since no three dimensional simulations where possible, the cooling had to be neglected for the known reasons.

In this work, we won't carry out any parameter studies. Furthermore, no discussion from a chemical view point will follow the results. Instead, we regard the simulation as a feasibility test for complex reactive flow problems in three dimensions and we will give guidelines for the treatment and numerical solution of laminar combustion problems with detailed chemistry.

In a first approach, we further reduce the geometry and use a two dimensional simplification which will later be used as a starting guess for the simulations with the full geometry. (see Figure 6.1). Besides the cooling pipes, all the main ingredients of the burner are represented by this setting. Up to now, this was the usual approach since 3d simulations with detailed chemistry where not possible. The investigations in this chapter should give more hints on the reliability of 2d simplifications. In both settings, we are interested in some output functionals of the solution. For example we measure the mass fraction of formaldehyde CH_2O along a line stretched across the whole z-axis which could be experimentally measured with laser spectroscopic methods. Considering the two dimensional case, this functional simplifies to the evaluation of the mass fraction in a single point:

$$\begin{aligned} j_{2D}(u) &= w_{CH_2O}(x_0, y_0), \\ j_{3D}(u) &= \frac{1}{|J_{3D}|} \int_{J_{3D}} w_{CH_2O}(x_0, y_0, s)\,\mathrm{ds}. \end{aligned}$$

This functional is especially chosen to exhibit the three dimensional features of the problem. Along the evaluation line we expect a changing mass fraction profile in z-direction.

For both settings, in two and three dimensions, we will start the simulation with the smaller C1 mechanism. The corresponding solution will then be used as a starting guess for the

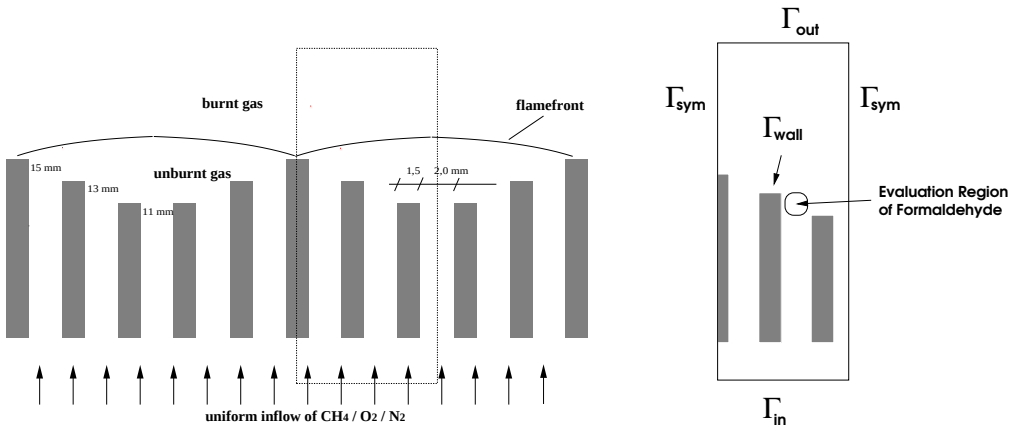

Figure 6.1.: 2D simplification of the methane burner.

more complex C2 mechanism.

6.4.1. 2D Simplification

A cut-out of the 2d-geometry is given in Figure 6.1. The lamellae are considered to be of infinite length in the direction of the z-axis. The actual computational domain is indicated by the box. We label the lower part of the boundary with Γ_{in}, the boundary of the lamellae with Γ_{wall}, outflow boundary with Γ_{out} and the remaining parts on the left and the right side with Γ_{sym}. The boundary conditions are given by

$$\begin{aligned} v = (0, 0.28),\ T = 288,\ w_{CH_4} = 0.0552,\ w_{O_2} = 0.22 \text{ on } \Gamma_{\text{in}} \\ v = 0,\ T = f(x),\ \frac{\partial w_k}{\partial n} = 0 \text{ on } \Gamma_{\text{wall}} \\ \frac{\partial v}{\partial n} + pn = 0,\ \frac{\partial w_k}{\partial n} = 0,\ \frac{\partial T}{\partial n} = 0 \text{ on } \Gamma_{\text{out}} \\ v^x = 0, \frac{\partial w_k}{\partial n} = 0,\ \frac{\partial T}{\partial n} = 0 \text{ on } \Gamma_{\text{sym}}, \end{aligned}$$

where $f(x)$ describes a linear temperature profile on the lamellae increasing from the bottom to the top, obtained by measurements and reaching from 372 to 402 Kelvin on the longest lamella, from 407 to 477 Kelvin on the middle and from 424 to 490 Kelvin on the short lamella.

As quantity of interest we look at the mass fraction of formaldehyde in one specific point (see Figure 6.1). The adaptivity of the meshes is controlled by this functional. We won‘t be able to give the usual plots on the convergency using the dual weighted error estimator since throughout the adaption process the homotopy parameters are adjusted and in addition there is no reference solution at all.

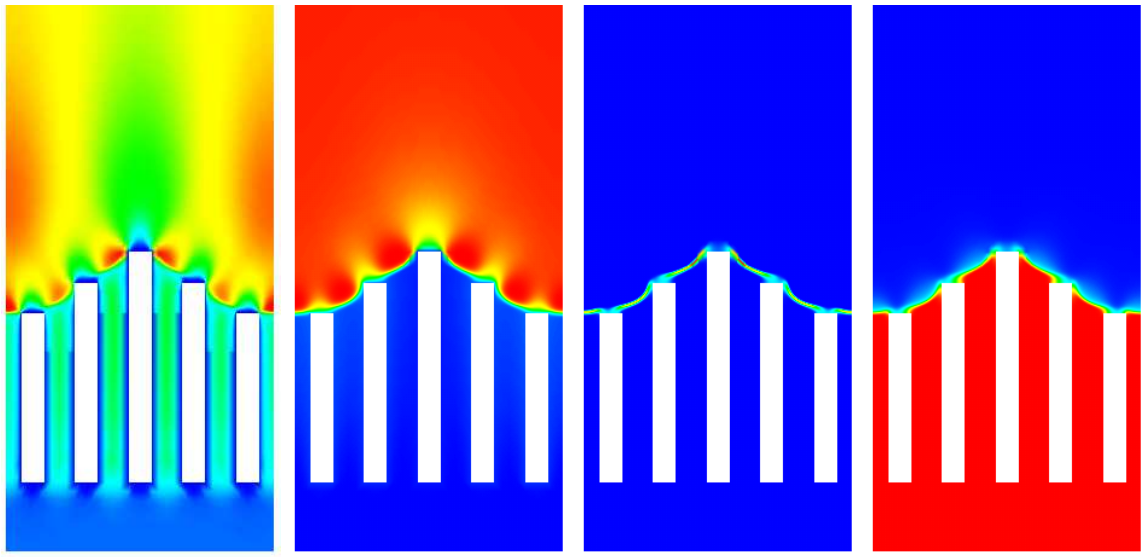

Figure 6.2.: Components of the 2D methane simulation obtained with the C1 mechanism B. From left to right: velocity in main flow direction, temperature, mass fraction of formaldehyde CH_2O and fuel CH_4.

Figure 6.2 gives plots of some components of the numerical solution (C1 mechanism). The specific evaluation point for the point functional is chosen to lie directly within this flame front – more precisely in the region of high formaldehyde mass fraction, somewhere between the two longest lamellae.

For completeness we give two components of the adjoint solution (C1 mechanism) in Figure 6.3. In both figures the evaluation point can spotted. The lower picture is the component belonging to the equation of formaldehyde. The upper picture shows the temperature component of the adjoint solution. This figure illustrates the bigger influence of the middle slot to the functional value.

Figure 6.4 shows two adaptively refined meshes. The cells are finest near the evaluation point and the edges of the lamellae.

With the solution computed up to now we start the simulation using the C2 mechanism. On a mesh of moderate size we get initial concentrations for the additional species (we have 24 new chemical species) by a zero dimensional time stepping in every node of the mesh. With very small time steps ($\Delta t = 10^{-7}$) we perform about 500 steps separately in every node. This approximation is usable as a starting guess for a time stepping of the coupled problem. Again we have to use some homotopy parameters $H > 1$ for the beginning.

A comparison between the two different reaction mechanisms is given in the next section together with the three dimensional results. Details on the solution of the problems and the computational effort will also be supplemented later on.

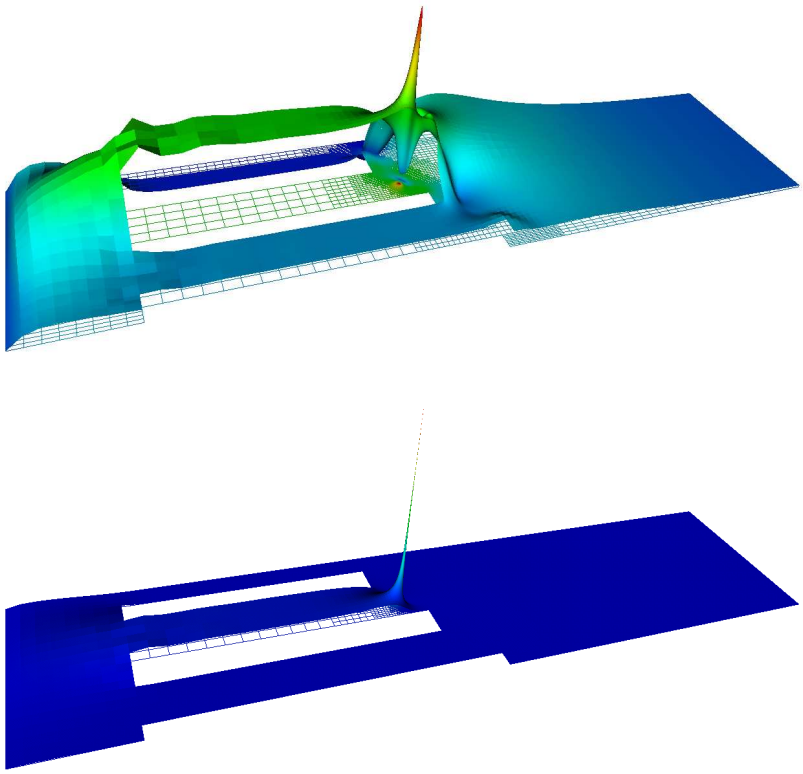

Figure 6.3.: The adjoint solution of the two dimensional methane burner. The upper figure shows the dual solution of the temperature equation, the lower one of the formaldehyde equation. The C1 mechanism (Appendix B) was used.

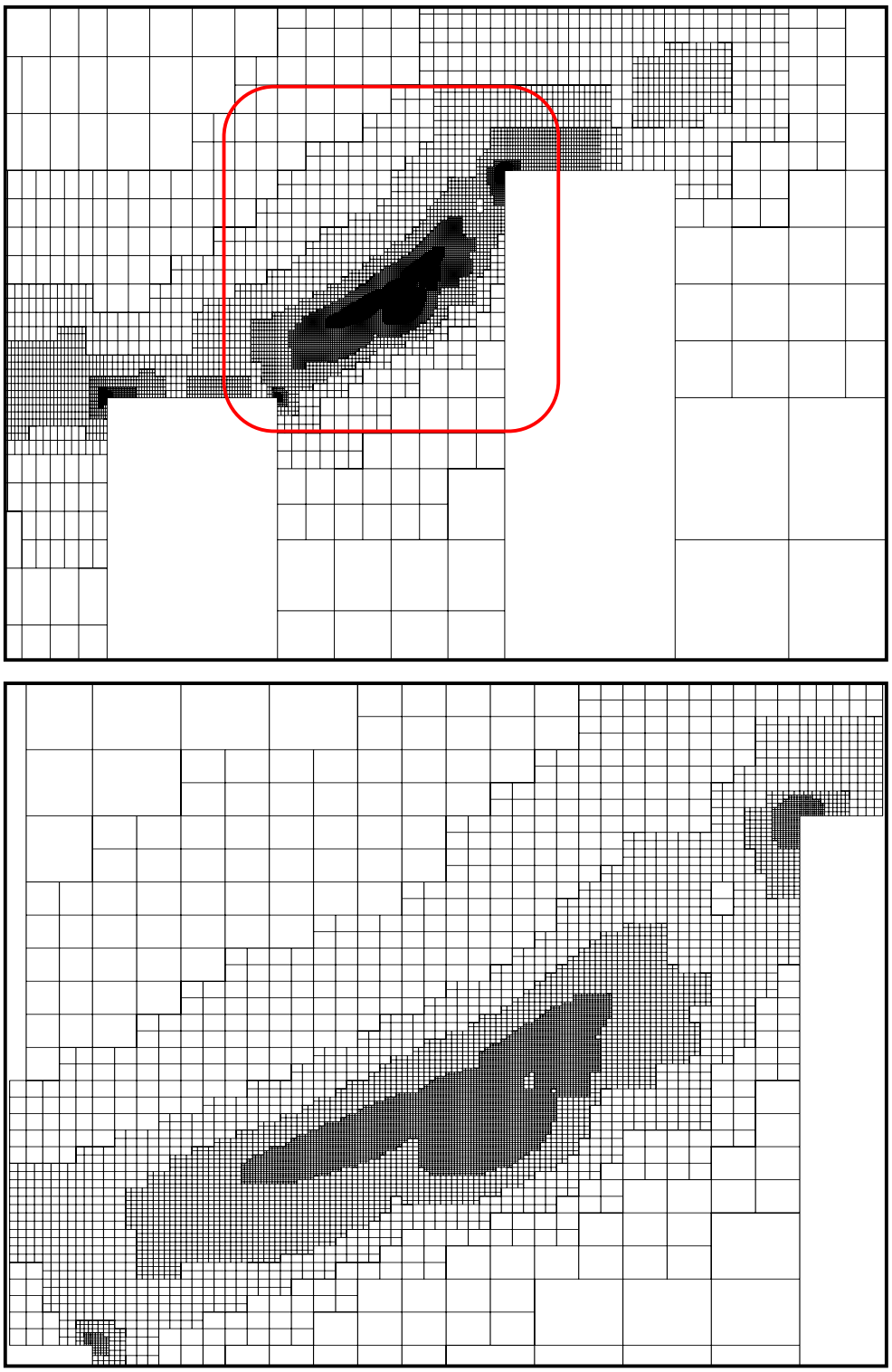

Figure 6.4.: Meshes from the 2D simulation. The left mesh is a section of the computational domain extended across the whole x-axis. The mesh on the right is a closeup from the area of the functional evaluation.

6.4.2. Numerical 3D Results

This section covers the full geometry of the burner. Particularly, we are interested in the influence of the cooling devices on the development of the flame-front. Again we have to face the whole variety of difficulties: the huge size of the problem, very stiff linear systems, nonlinearities and tiny but important details of the solution which are not representable on coarse meshes. As described in Section 6.3.2 the different solution approaches are all utilized hand in hand.

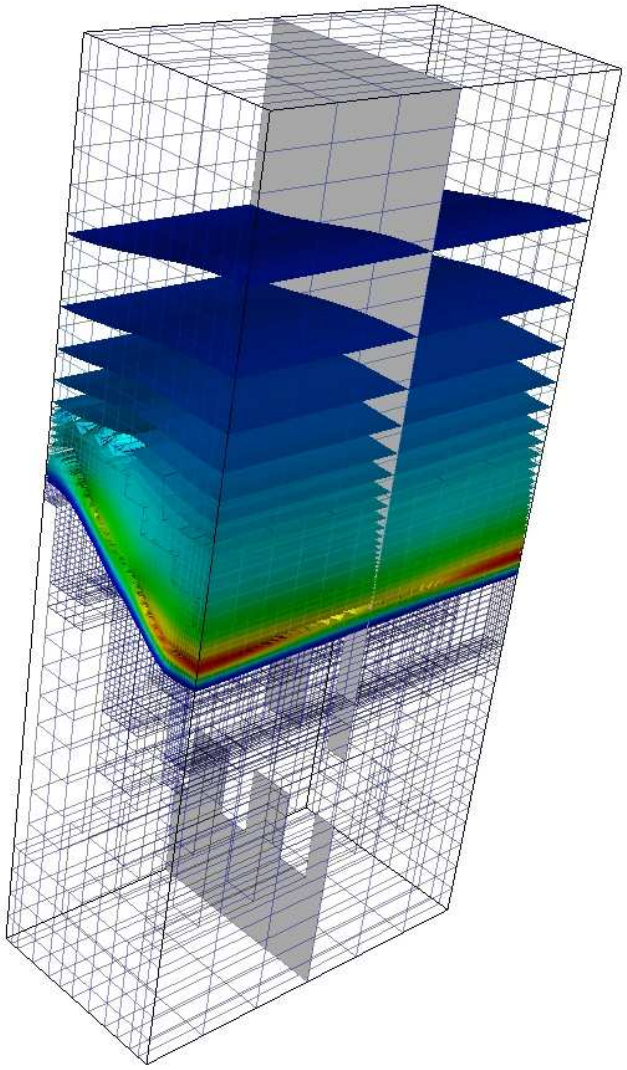

Figure 6.5.: Iso surfaces of the mass fraction of hydrogen H.

Prior to performing quantitative studies, we can discuss the main features of the solution with help of Figure 6.5. Here, iso surfaces of the hydrogen mass fraction are displayed. Settled atop the lamellae a very sharp flame front is established. Within this flame front the three dimensional influence is clearly visible. Atop the cooling pipes in the middle of the computational domain, the mass fraction of hydrogen is far less.

For a more detailed analysis of the burner we measure the mass fraction of some species and of the velocity field along a line through the computational domain. This could be experimentally measured with laser spectroscopic methods. As quantity of interest we identify the mass fraction of formaldehyde and of HO_2^- radicals as well as the temperature and the velocity field. The measurement line is situated within the flame front. In addition to exploring the values along the functional evaluation line through the domain, we can also compare the three dimensional simulation with the two dimensional simplification discussed in the previous section. Finally we compare the results for the two different reaction mechanisms.

The three dimensional computations using the C1 are initiated with the corresponding steady solution of the two dimensional simplification. The values are continued along the z-axis. No special modifications are applied in the area of the cooling pipes, only the correct boundary

cells	dof's	time (sec)	mem (MB)
15 872	364 572	1463	530
40 042	920 930	3423	1 383
51 415	1 160 386	4144	1 742

Table 6.1.: Initial time stepping for three dimensional solution using the C1 mechanism. Size of time steps $5 \cdot 10^{-5}$.

values are filled. With a mixture of time stepping, mesh refinement and adjustment of the homotopy parameters, the three dimensional continuation of the simplified solution is carried over to the solution of the full geometry. Since the meshes are relatively small, these first steps are feasible on a workstation. In Table 6.1 we gather some notes on these first simulations. The values are obtained on a single processor Opteron workstation with 1.6 GHz. This calculation would not have been possible on a single processor machine without having the special matrix structure described in Section 6.3.4. For instance the calculation on the mesh with 51 415 cells would have required more than 15 GB of memory. The problems on refined meshes are solved on the Linux cluster Helics [Hel].

A good visualization of the used finite element meshes is very difficult. Therefore, we refer to the two dimensional simplifications to get a guess of the adaptive scheme. The dual weights lead to mesh refinement in the region of the functional evaluation as well as all parts which are highly sensitive for this functional. This to say, we have a finer mesh between the middle lamellae where the functional's domain is settled. Next, the edges of the lamellae entering into the domain are resolved, again basically near the functional evaluation. The region around the cooling device does not need a fine mesh, since the boundary is approximated with a biquadratic transformation (see Section 2.3 on boundary approximations for details). Thus we do not have any singularities in this region and due to the small temperature gradients and no chemical reactions at all in this areas, the residual is rather small.

As for the two dimensional simplification we initiate the simulation with the C2 mechanism by zero dimensional time stepping on a mesh of moderate size. For the computation of the actual solution, we repeat the same procedure as for the C1 mechanism and the two dimensional case (see Section 6.3.2).

In Table 6.2 we list key values belonging to the solution process for different settings. The problem is too large and difficult to give detailed studies on the performance and efficiency of the solver. Instead we list values for time steps on different mesh levels throughout the calculation. All calculations belonging to the C1 mechanism where performed on a the Athlon XP cluster Helics [Hel]. A different cluster with Opteron nodes was used for the C2 mechanism. This second cluster consists of 20 nodes, each with 2 processors and 8 GB of memory. This cluster is slightly faster than Helics if the same number of processors is used. However, the nodes are connected by standard Gigabit instead of the high speed Myrinet network with much shorter latencies on Helics.

For judging the parallel efficiency, we use the parallel efficiency index defined in Section 5.5. Compared to the Navier-Stokes benchmark from Section 5.5 the values listed in Table 6.2 are

cells	dof's	time (sec)	#CPU	memory	pe_i	homo	mechanism
26 008	574 408	975	32	2100 MB	5.43	3.0	C1
65 880	1 415 842	293	80	4 500 MB	1.66	2.8	
185 020	3 785 509	1930	33	12 100 MB	1.68	2.5	
558 694	11 353 332	1350	105	41 600 MB	1.24	1.0	
15 872	825 084	1 772	3	1 132 MB	0.64	3.0	C2
29 648	1 505 946	1583	5	2 286 MB	0.53	8.0	
60 784	3 136 248	2248	10	5 379 MB	0.71	5.0	
197 816	9 863 082	6096	10	17 328 MB	0.61	1.5	
291 102	14 299 478	4681	18	25 858 MB	0.59	1.0	
291 102	14 299 478	2572	37	29 152 MB	0.66	1.0	

Table 6.2.: Calculations for the C1 and the C2 mechanism done on two Linux clusters. All simulations using the C1 mechanism are done on an AMD Athlon cluster, the C2 mechanism is computed on an Opteron cluster, slightly faster. On different meshes the number of cells, the number of degrees of freedom, the time necessary to solve one time step, the number of CPU's, the memory usage and the parallel efficiency index (see (5.24) in Section 5.5) is given. Further the used homotopy parameter is indicated.

in the same range, even though the numerical effort is linked to the number of matrix entries and not to the number of unknowns. Thus, the values indicate a higher parallel efficiency, which validates the discussion in Chapter 5: considering reactive flows the deal of local work is much larger compared to communication effort than for the Navier-Stokes equations.

As discussed in Section 6.3.2, mesh adaption, time stepping and reduction of the homotopy parameter are interwoven. With lower homotopy parameter, the Newton convergency is reduced, therefore the parallel efficiency index is slightly growing. The size of the time steps varies between 10^{-4} and 10^{-6}. Some smaller steps are necessary every time the mesh was changed.

Without the special matrix structure, the 3D computation with the detailed C2 mechanism on the finest mesh in Table 6.2 would have been beyond the means of even large parallel computers. Instead of 29 152 MB of memory, more than 500 000 MB would have been necessary. And also the time for computing the solution would be larger by a factor of more than 20.

The time values given in Table 6.2 are seconds necessary for one time step. With the available hardware, the overall running times started from scratch (neglecting the user interaction) are approximately given by: some hours the initial two dimensional solution with the C1 mechanism. The expansion of this solution to the C2 mechanism as well as the expansion to a three dimensional solution with the C1 mechanism requires a few days each. Starting with a three dimensional solution, the enlargement of the C1 mechanism to the detailed C2 calculation takes about seven days.

	C1 mechanism		C2 mechanism	
	2d	3d	2d	3d
velocity	3.39 m/s	2.61 m/s	3.51 m/s	3.31 m/s
temperature	2156 K	2039 K	2099 K	2036 K
CH_2O	$7.1e^{-3}$	$5.2e^{-3}$	$4.3e^{-3}$	$3.49e^{-3}$
HO_2	$2.3e^{-4}$	$6.2e^{-4}$	$4.9e^{-4}$	$3.85e^{-4}$

Table 6.3.: Maximal values for the velocity, the temperature as well as the mass fraction of formaldehyde and HO_2 obtained in the 2d and 3d simulation for both reaction mechanisms.

In Figure 6.6 we show the iso-surfaces for the velocity in the main flow direction, the temperature field and the mass fraction of formaldehyde and of CH_3CO for the C2 mechanism. The corresponding plots for the detailed C1 mechanism are not distinguishable by looking at the figures. The last species is not present in the C1 mechanism.

In Table 6.3 we gather the maximal values of the temperature, the velocity in main flow direction, the concentration of formaldehyde and of HO_2 for all four configurations. The lower value of the formaldehyde concentration in the case of the C2 mechanism was already observed in [BR05b] for a two dimensional study of a comparable flame with both mechanisms. However, as already mentioned, these results will not be discussed from a chemical point of view. The necessity of three dimensional simulations is evident and our results demonstrate the feasibility of such computations.

Finally, in Figure 6.7 plots of the behavior of the functional along the z-axis are given with comparisons to the two dimensional approximation of this functional. All of the four solution components exhibit a large influence in the z-direction of the domain. As expected, the velocity is lower above the cooling device, perhaps against intuition, the temperature is higher. But this can be explained by a shorter distance of the flame front to the lamellae due to the lower velocity and therefore a different cutting of the evaluation line through the flame. These plots belong to the C2 mechanism.

Some final remarks

The results of the simulations presented in this chapter call for some final discussion. First of all, the methane burner was taken as a feasibility test using a "real-life" configuration. Using the methods described in this work, a fully coupled simulation using detailed chemistry in a complex three dimensional domain is possible within "some few days". Without the usage of the robust finite element discretization on adaptively refined meshes described in Chapter 3, the parallel multigrid solver of Chapter 5 and the special matrix structures described in this chapter, a coupled solution of the regarded burner would not have been possible. But for all that, the solution of the problems is still challenging with huge memory effort and nonlinear equations, very difficult so solve.

The aim of this work was the development of numerical methods for the solution of large coupled systems of Pde's. This specific methane burner is only contemplated as a test-case of an application at the limit of possibility. From the chemical point of view there is always an interest in the simulation of combustion processes valid for a large range of species. Thus every extension of the reaction mechanism is of use for the prognosis of flame properties e.g. for the prediction of pollutants. However, the analysis of the detailed combustion process in a three dimensional burner is regarded as a feasibility test, thus, no discussion on the chemical aspect is given.

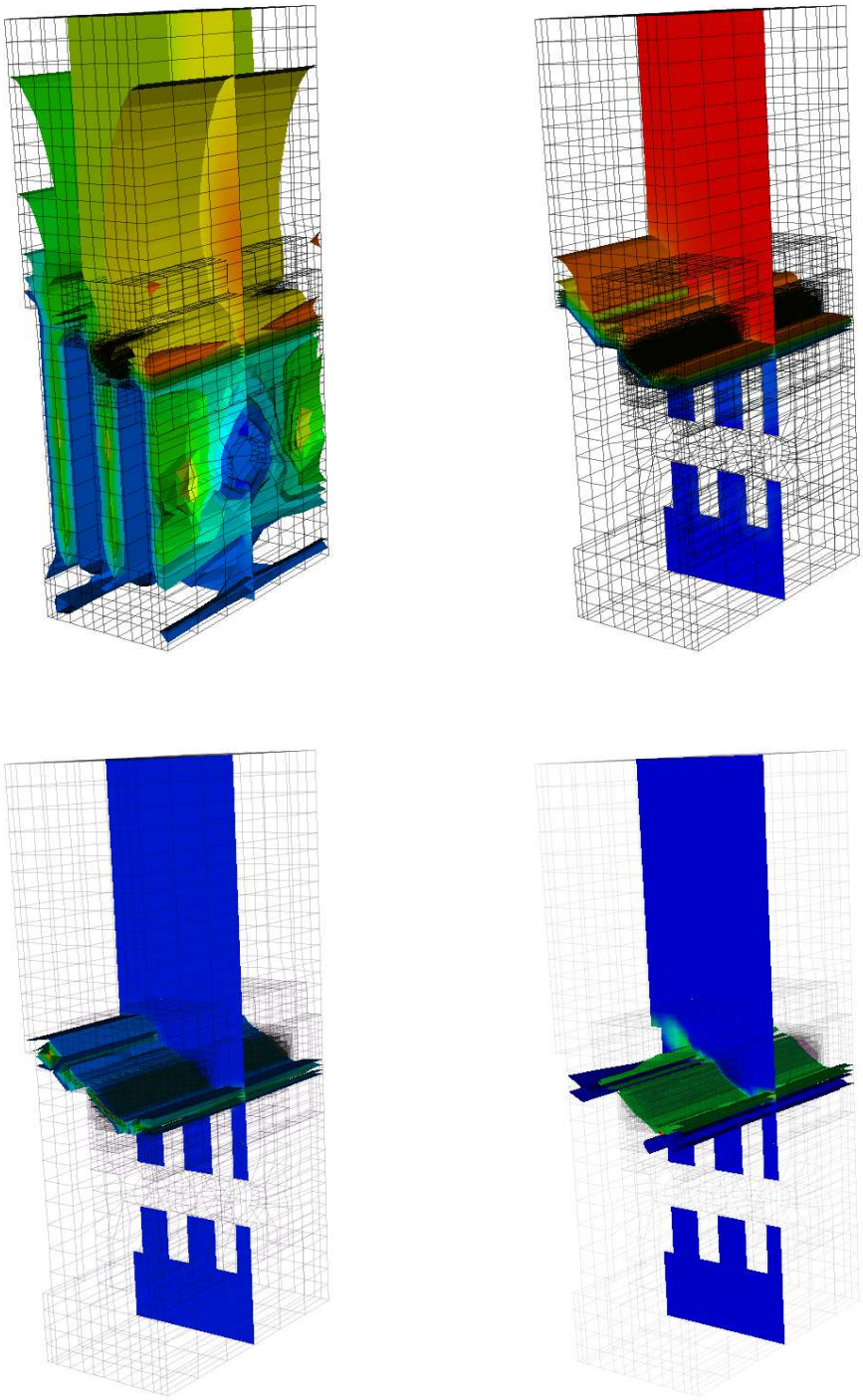

Figure 6.6.: Components of the 3D methane simulation with the C2 mechanism. From top left to the bottom right: velocity in main flow direction, temperature field, mass fraction of formaldehyde CH_2O and of CH_3CO radicals.

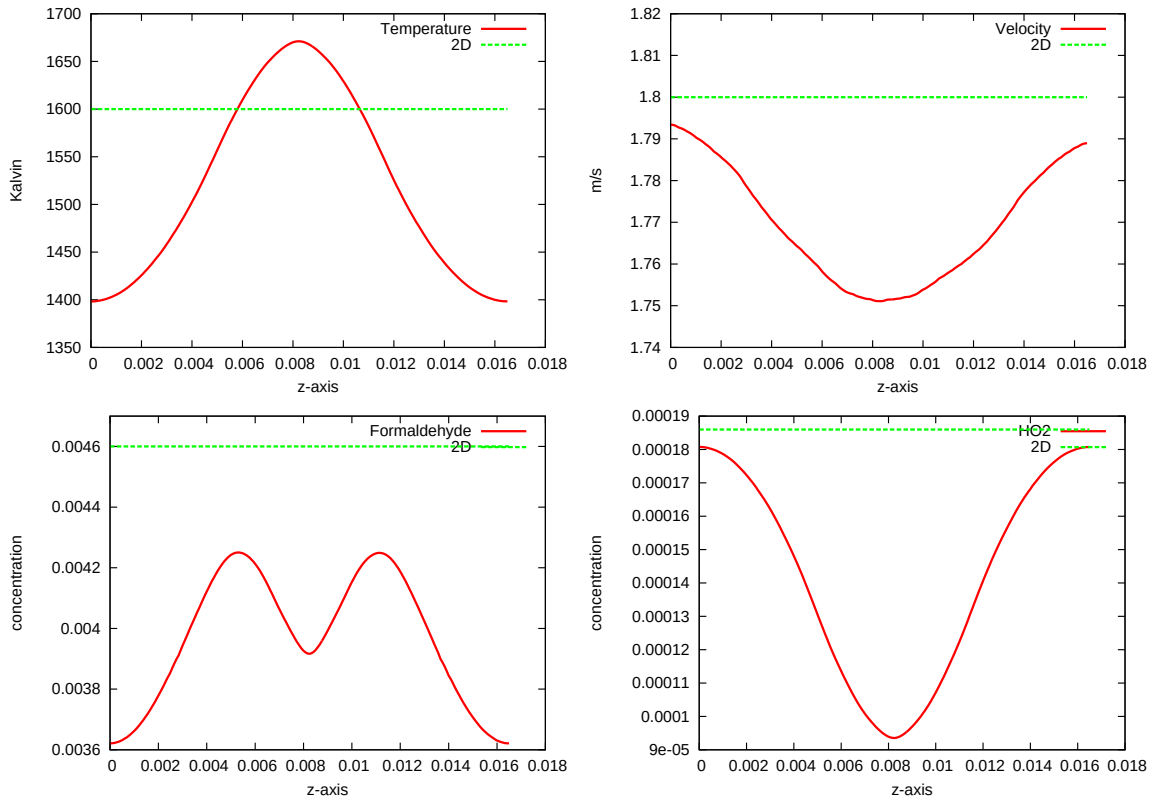

Figure 6.7.: Course of the temperature, the velocity in main flow direction and the mass fraction of formaldehyde and HO_2^- radicals along the z-axis on the functional evaluation point. The dashed line is the corresponding value obtained in the 2D simulation.

Acknowledgements

This work has been supported by the SFB 359 “Reaktive Strömung, Diffusion und Transport”. In subproject A2 numerical methods for the Navier-Stokes equations and chemical reactions where developed. The simulation of 3D flames with detailed chemistry was one goal of the SFB and finally reached within its last days. Further I was member of the Graduiertenkolleg “Complex Processes: Modeling, Simulation and Optimization”.

I would like to express my gratitude to Malte Braack for countless discussions and suggestions on my thesis as well as help with all the other tasks, especially for reading every bit of paper produced twice and thrice. My supervisor Rolf Rannacher always helped to work out the essential objectives of this work. Besides thanking for giving me the possibility to work on this topic I would like to thank for the encouragement and support to attend conferences as well as for helping me in gaining experience in research and teaching beyond my actual thesis.

Further my gratitude goes to Boris Vexler for always taking the time for discussion and to Josh Moore for canceling his own phd to get enough spare time for reading my manuscript.

This work would not have been possible without the software library Gascoigne, thus my thank goes to the whole team: Roland Becker, Malte Braack, Boris Vexler, Dominik Meidner, Michael Schmich and Tom Dunne.

Although this work has to be regarded as finished at this point, I would like to use the remaining space to express my gratitude to all those who helped to keep me from work.

I would like to thank Eva, Josh and Linnéa for all of it – relaxing, sharing adventures and discussions on hacking.

Further I extend my thanks to Friderike – hopefully in advance – for staying in Heidelberg and not only for sharing criminological affinities.

But I also thank those who left, not for it but tho: Diane, for a time in Heidelberg by far too short. Markus, although his move made life more healthy. Finally Sophie, for all the traveling and more.

I thank those who pulled parts of my life onto the streets, too many of them to mention all of, but surely amongst them Jenni, Matze, Manfred, Sven, Heinz and Hend.

Finally I thank all my colleagues and friends at the university, in particular Tom, Dirk and Anke for making the time at work more enjoyable and the latter also for distracting me from the cafeteria taste.

A. Modeling of Chemical Reactions

Elementary chemical reactions usually involve less than three reactants and three products, e.g. the oxidation of hydrogen atoms:

$$H + O_2 \rightarrow OH + O, \tag{A.1}$$

a reaction with two reactants and two product. Very often a third species is not affected by the reaction and only gives kinetic energy to initiate the reaction, e.g.. We write reactions of this type (A.1) as

$$\sum_{k\in S} \nu_{r,k}\chi_k \xrightarrow{k_r} \sum_{k\in S} \tilde{\nu}_{r,k}\chi_k, \quad r \in R = \{1,\ldots,m\},$$

where χ_k is the chemical species and $\nu_{r,k}, \tilde{\nu}_{r,k}$ the stoichiometric coefficients of the reaction r and k_r the reaction rate. For large reaction mechanisms, the matrices ν and $\tilde{\nu}$ describing all reactions are sparse. Typical sizes are from 9 species with 19 reactions for methane combustion up to ≥ 140 species with more than 2400 reactions for acethylen combustion.

Due to mass conservation there holds

$$\sum_{k\in S} m_k(\tilde{\nu}_{r,k} - \nu_{r,k}) = 0, \quad \forall r \in R. \tag{A.2}$$

The production rate $\dot{\omega}_k$ for species k in mole fractions is obtained by adding all reactions considered:

$$\dot{\omega}_k(T,w) = \sum_{r=1}^{n_r} \left\{ (\nu_{rk} - \tilde{\nu}_{rk}) k_r(T) \prod_{j=1}^{n_s} c_j^{\nu_{rj}}(w) \right\},$$

where c_j is the concentration of species j, given by $c_j = M_j^{-1}\rho w_j$. The chemical source terms for the species equations in mass fractions have the form

$$f_k(T,w) = M_k \cdot \dot{\omega}_k(T,w).$$

For (A.2) we conclude, that the sum over all n_s source terms vanishes,

$$\sum_{k=1}^{n_s} f_k = 0. \tag{A.3}$$

The dependence of the reaction rate on the temperature is given by an Arrhenius law

$$k_r(T) = A_r T^{\beta_r} \exp\left\{ \frac{-E_{ar}}{RT} \right\}.$$

The law itself is empirically validated, while the constants A_r, β_r and E_{ar} are determined by experiments. For the methane combustion analyzed in Chapter 6, these constants are given in Appendix B.

The source term f_T in the temperature equation has the form

$$f_T(T, w) = -\sum_{k=1}^{n_s} h_k(T) f_k(T, w),$$

where the enthalpy h_k of a species is given by

$$h_k(T) = h_{k,T^0} + \int_{T^0}^{T} c_{p,k}(T')dT',$$

with an enthalpy h_{i,T^0} for a reference temperature T^0. The partial heat capacity of species k is represented by $c_{p,i}$.

Building the sum over the reaction equations (6.15) and using (A.3) gives a trivial equation. The system degenerates and thus one species is dropped and replaced by

$$w_{n_s} = 1 - \sum_{k=1}^{n_s-1} w_k.$$

Backward Reaction

For elementary reactions $r \in R$ each reaction has its counterpart, the backward reaction $r^b \in R$, with reaction rate k_r^b. In order to avoid too many indices, from now on we focus on one specific reaction r, omit this index and introduce

$$\delta_j = \tilde{\nu}_{r,j} - \nu_{r,j}, \quad \forall j \in S.$$

The so-called equilibrium constant

$$k^e = \frac{k}{k^b} \tag{A.4}$$

depends on $\{\delta_j\}, \{m_j\}, T$ and p. Since δ_j depends on the reaction r, each reaction has its own equilibrium constant. These constants are taken from tables and the backward reaction in modeled via this constant and the forward reaction.

Transport Fluxes

The diffusion in the simplified species equation (6.15) consists only of the mass diffusion. While other parts are neglected for this work, the mass diffusion fluxes are modeled by Fick's law of diffusion [Fic55]:

$$\mathcal{F}_k^{\text{mass}} = -\rho D_k^* \frac{m_k}{\overline{m}} \nabla x_k,$$

with the species mole fractions x_k. The diffusion coefficients D_k^* are defined by

$$D_k^* = \frac{1 - y_k}{\sum_{l \neq k} \frac{x_l}{D_{kl}^{\text{bin}}}}.$$

B. Reaction Scheme for Methane Combustion

In Table B.1 the C_1-reaction mechanism for the methane combustion is given, see Smooke et al. [SMK89]. For each reaction we list the Arrhenius parameters of the forward reaction. Tables B.2 and B.3 give the details for the more complex C_2 mechanism.

Table B.1.: C1 Reaction mechanism used for the methane/air reaction including 15 species and 42 reversible reactions. Collision efficiencies: $M = 1$, $M'(H_2O, H_2) = (21, 3.3)$, $M''(H_2O, H_2) = (6, 3)$, $M'''(H_2O) = 20$.

Reaction	A_r	β_r	E_{ar}
$CH_4+M = CH_3+H+M$	6.30×10^{14}	0	435.14
$CH_4+O_2 = CH_3+HO_2$	7.90×10^{13}	0	234.30
$CH_4+H = CH_3+H_2$	2.20×10^{04}	3	36.61
$CH_2O+O = HCO+OH$	1.81×10^{13}	0	12.90
$HCO+OH = CO+H_2O$	5.00×10^{12}	0	0
$HCO+M = CO+H+M$	7.14×10^{14}	0	70.29
$HCO+H = CO+H_2$	4.00×10^{13}	0	0
$HCO+O = OH+CO$	1.00×10^{13}	0	0
$HCO+O_2 = CO+HO_2$	3.00×10^{12}	0	0
$CO+O+M = CO_2+M$	7.10×10^{13}	0	−19.00
$CO+O_2 = CO_2+O$	1.60×10^{13}	0	171.54
$CH_3+O_2 = CH_3O+O$	7.00×10^{12}	0	107.33
$CH_3O+H = CH_2O+H_2$	2.00×10^{13}	0	0
$CH_3O+O = CH_2O+OH$	1.00×10^{13}	0	0
$N+O_2$=$NO+O$	6.4×10^{09}	1	26.1
$H+O_2 = OH+O$	2.00×10^{14}	0	70.29
$O+H_2 = OH+H$	1.80×10^{10}	1	36.93
$H+HO_2 = H2+O_2$	2.50×10^{13}	0	2.93
$OH+HO_2 = H_2O+O_2$	5.00×10^{13}	0	4.18
$H+HO_2 = OH+OH$	2.50×10^{14}	0	7.95
$O+HO_2 = O_2+OH$	4.80×10^{13}	0	4.18
$CH_3+O = CH_2O+H$	6.80×10^{13}	0	0

Reaktion	A_r	β_r	E_{ar}
$CH_3O+O_2 = CH_2O+HO_2$	6.30×10^{10}	0	10.88
$CH_3+O_2 = CH_2O+OH$	5.20×10^{13}	0	144.66
$CH_3+OH = CH_2O+H_2$	7.50×10^{12}	0	0
$HO_2+CO = CO_2+OH$	5.80×10^{13}	0	95.96
$H_2+O_2 = OH+OH$	1.70×10^{13}	0	199.91
$OH+H_2 = H_2O+H$	1.17×10^{09}	1.3	15.17
$H+O_2+M' = HO_2+M'$	2.30×10^{18}	−0.8	0
$H+O_2+O_2 = HO_2+O_2$	6.70×10^{19}	−1.42	0
$H+O_2+N_2 = HO_2+N_2$	6.70×10^{19}	−1.42	0
$OH+OH = H_2O+O$	6.00×10^{08}	1.3	0
$H_2+M'' = H+H+M''$	6.99×10^{18}	−1	436.08
$O_2+M = O+O+M$	6.91×10^{18}	−1	496.41
$H+OH+M''' = H_2O+M'''$	2.10×10^{22}	−2	0
$CH_3O+M = CH_2O+H+M$	2.40×10^{13}	0	120.55
$CH_3O+OH = CH_2O+H_2O$	1.00×10^{13}	0	0
$CH_4+O = CH_3+OH$	1.60×10^{06}	2.36	30.96
$CH_4+OH = CH_3+4H_2O$	1.60×10^{06}	2.1	10.29
$CH_2O+OH = HCO+H_2O$	7.53×10^{12}	0	0.70
$CH_2O+H = HCO+H_2$	3.31×10^{14}	0	43.93
$CH_2O+M = HCO+H+M$	3.31×10^{16}	0	338.90
$CO+OH = CO_2+H$	1.51×10^{07}	1.3	−3.17
$N+NO$=N_2+O	3.27×10^{12}	0.3	0

Table B.2.: First part of the C2 mechanism of Warnatz, see [WMD96], for methane/air combustion includes 39 chemical species and 151 reversible reactions. Collision efficiencies: $M = 1$, $M'(O_2, H_2, H_2O, CO, CO_2, CH_4, N_2) = (0.4, 1, 6.5, 0.75, 1.5, 3, 0.4)$.

Reaction	A_r	β_r	E_{ar}
$O_2 + H = OH + O$	2.00e14	0	70.3
$H_2 + O = OH + H$	5.06e4	2.67	26.3
$H_2 + OH = H_2O + H$	1.00e8	1.60	13.8
$OH + OH = H_2O + O$	1.50e9	1.14	0.42
$H + H + M' = H_2 + M'$	1.80e18	−1.0	0
$O + O + M' = O_2 + M'$	2.90e17	−1.0	0
$H + OH + M' = H_2O + M'$	2.20e22	−2.0	0
$H + O_2 + M' = HO_2 + M'$	2.30e18	−0.8	0
$HO_2 + H = OH + OH$	1.50e14	0	4.2
$HO_2 + H = H_2 + O_2$	2.50e13	0	2.9
$HO_2 + H = H_2O + O$	3.00e13	0	7.2
$HO_2 + O = OH + O_2$	1.80e13	0	−1.7
$HO_2 + OH = H2O + O_2$	6.00e13	0	0
$HO_2 + HO_2 = H_2O_2 + O_2$	2.50e11	0	−5.2
$OH + OH + M' = H_2O_2 + M'$	3.25e22	−2.0	0
$H_2O_2 + H = H_2 + HO_2$	1.70e12	0	15.7
$H_2O_2 + H = H_2O + OH$	1.00e13	0	15.0
$H_2O_2 + O = OH + HO_2$	2.803e13	0	26.8
$H_2O_2 + OH = H_2O + HO_2$	5.40e12	0	4.2
$CO + OH = CO_2 + H$	6.00e06	1.5	−3.1
$CO + HO_2 = CO_2 + OH$	1.50e14	0	98.7
$CO + O + M' = CO_2 + M'$	7.10e13	0	−19.0
$CO + O_2 = CO_2 + O$	2.50e12	0	200.0
$CH + O = CO + H$	4.00e13	0	0
$CH + O_2 = CHO + O$	3.30e13	0	0
$CH + CO_2 = CHO + CO$	3.40e12	0	2.9
$CH + H_2O = 3CH_2 + OH$	5.70e12	0	−3.2
$CHO + M' = CO + H + M'$	7.10e14	0	70.3
$CHO + H = CO + H_2$	9.00e13	0	0
$CHO + O = CO + OH$	3.00e13	0	0
$CHO + O = CO_2 + H$	3.00e13	0	0
$CHO + OH = CO + H_2O$	1.00e14	0	0
$CHO + O_2 = CO + HO_2$	3.00e12	0	0
$CHO + CHO = CH_2O + CO$	3.00e13	0	0
$3CH_2 + H = CH + H_2$	6.00e12	0	−7.5
$3CH_2 + O \rightarrow CO + H + H$	8.40e12	0	0
$3CH_2 + 3CH_2 = C_2H_2 + H_2$	1.20e13	0	3.4
$3CH_2 + 3CH_2 = C_2H_2 + H + H$	1.10e14	0	3.4
$3CH_2 + CH_3 = C_2H_4 + H$	4.20e13	0	0
$3CH_2 + O_2 = CO + OH + H$	1.30e13	0	0
$3CH_2 + O_2 = CO + OH + H$	1.30e13	0	6.2
$3CH_2 + O_2 = CO_2 + H_2$	1.20e13	0	6.2
$1CH_2 + M' = 3CH_2 + M'$	1.20e13	0	0
$1CH_2 + O_2 = CO + OH + H$	3.10e13	0	0
$1CH_2 + H_2 = CH_3 + H$	7.20e13	0	0
$CH_2O + M' = CHO + H + M'$	5.00e16	0	320.0
$CH_2O + H = CHO + H_2$	2.30e10	1.05	13.7

Reaktion	A_r	β_r	E_{ar}
$CH_2O + O = CHO + OH$	4.15e11	0.57	11.6
$CH_2O + OH = CHO + H_2O$	3.40e09	1.20	−1.9
$CH_2O + HO_2 = CHO + H_2O_2$	3.00e12	0	54.7
$CH_2O + CH_3 = CHO + CH4$	1.00e11	0	25.5
$CH_2O + O_2 = CHO + HO_2$	6.00e13	0	170.7
$CH_3 + M' = 3CH_2 + H + M'$	1.00e16	0	379.0
$CH_3 + O = CH_2O + H$	8.43e13	0	0
$CH_3 + H = CH_4$	1.93e36	−7.0	38.0
$CH_3 + OH \rightarrow CH_3O + H$	2.26e14	0	64.8
$CH_3O + H \rightarrow CH_3 + OH$	4.75e16	−0.13	88.0
$CH_3 + O_2 \rightarrow CH_2O + OH$	3.30e11	0	37.4
$CH_3 + HO_2 = CH_3O + OH$	1.80e13	0	0
$CH_3 + HO_2 = CH_4 + O_2$	3.60e12	0	0
$CH_3 + CH_3 \rightarrow C_2H_4 + H_2$	1.00e16	0	134.0
$CH_3 + CH_3 = C_2H_6$	1.69e53	−12	81.24
$CH_3O + M' = CH_2O + H + M'$	5.00e13	0	105.0
$CH_3O + H = CH_2O + H_2$	1.80e13	0	0
$CH_3O + O_2 = CH_2O + HO_2$	4.00e10	0	8.9
$CH_2O + CH_3O \rightarrow CH_3OH + CHO$	6.00e11	0	13.8
$CH_3OH + CHO \rightarrow CH_2O + CH_3O$	6.50e9	0	57.2
$CH_3O + O = O_2 + CH_3$	1.10e13	0	0
$CH_3O + O = OH + CH_2O$	1.40e12	0	0
$CH_2OH + M' = CH_2O + H + M'$	5.00e13	0	105.0
$CH_2OH + H = CH_2O + H_2$	3.00e13	0	0
$CH_2OH + O_2 = CH_2O + HO_2$	1.00e13	0	30.0
$CH_3O_2 + M' \rightarrow CH_3 + O_2 + M'$	7.24e16	0	111.1
$CH_3 + O_2 + M' \rightarrow CH_3O_2 + M'$	1.41e16	0	−4.6
$CH_3O_2 + CH_2O \rightarrow CH_3O_2H + CHO$	1.30e11	0	37.7
$CH_3O_2H + CHO \rightarrow CH_3O_2 + CH2O$	2.50e10	0	42.3
$CH_3O_2 + CH_3 \rightarrow CH_3O + CH3O$	3.80e12	0	−5.0
$CH_3O + CH_3O \rightarrow CH_3O_2 + CH3$	2.00e10	0	0
$CH_3O_2 + HO_2 \rightarrow CH_3O_2H + O_2$	4.60e10	0	−10.9
$CH_3O_2H + O_2 \rightarrow CH_3O_2 + HO_2$	3.00e12	0	163.3
$CH_3O_2 + CH_3O_2 \rightarrow CH_2O + CH_3OH + O_2$	1.80e12	0	0
$CH_2O + CH_3OH + O_2 \rightarrow CH_3O_2 + CH_3O_2$	0	0	0
$CH_3O_2 + CH_3O_2 \rightarrow CH_3O + CH_3O + O_2$	3.70e12	0	9.2
$CH_3O + CH_3O + O_2 \rightarrow CH_3O_2 + CH_3O_2$	0	0	0
$CH_4 + H = H_2 + CH_3$	1.30e4	3.0	33.6
$CH_4 + O = OH + CH_3$	6.923e8	1.56	35.5
$CH_4 + OH = H_2O + CH_3$	1.60e7	1.83	11.6
$CH_4 + HO_2 = H_2O_2 + CH_3$	1.10e13	0	103.1
$CH_4 + CH = C_2H_4 + H$	3.00e13	0	−1.7
$CH_4 + 3CH_2 = CH_3 + CH_3$	1.30e13	0	39.9
$CH_3OH = CH_3 + OH$	9.51e29	−4.3	404.1
$CH_3OH + H = CH_2OH + H_2$	4.00e13	0	25.5
$CH_3OH + O = CH_2OH + OH$	1.00e13	0	19.6
$CH_3OH + OH = CH_2OH + H_2O$	1.00e13	0	7.1

Table B.3.: Second part of the C2 mechanism of Warnatz, see [WMD96], for methane/air combustion.

Reaktion	A_r	β_r	E_{ar}
$CH_3OH + HO_2 \rightarrow CH_2OH + H_2O_2$	6.20e12	0	81.1
$CH_2OH + H_2O_2 \rightarrow HO_2 + CH_3OH$	1.00e7	1.7	47.9
$CH_3OH + CH_3 = CH_4 + CH_2OH$	9.00e12	0	41.1
$CH_3O + CH_3OH \rightarrow CH_2OH + CH_3OH$	2.00e11	0	29.3
$CH_2OH + CH_3OH \rightarrow CH_3O + CH_3OH$	2.20e4	1.7	45.4
$CH_3OH + CH_2O \rightarrow CH_3O + CH_3O$	1.53e12	0	333.2
$CH_3O + CH_3O \rightarrow CH_3OH + CH_2O$	3.00e13	0	0
$CH_3O_2H = CH_3O + OH$	4.00e15	0	180.5
$OH + CH3O2H = H_2O + CH3O2$	2.60e12	0	0
$C2H + O = CO + CH$	1.00e13	0	0
$C2H + O_2 = HCCO + O$	3.00e12	0	0
$HCCO + H = 3CH_2 + CO$	1.50e14	0	0
$HCCO + O \rightarrow CO + CO + H$	9.60e13	0	0
$HCCO + 3CH2 = C2H3 + CO$	3.00e13	0	0
$C2H2 + M' = C2H + H + M'$	3.60e16	0	446.0
$C2H2 + O_2 = HCCO + OH$	2.00e8	1.5	126.0
$C2H2 + H = C2H + H2$	1.50e14	0	79.6
$C2H2 + O = 3CH2 + CO$	1.72e4	2.8	2.1
$C2H2 + O = HCCO + H$	1.72e4	2.8	2.1
$C2H2 + OH = H_2O + C2H$	6.00e13	0	54.2
$C2H2 + C2H = C4H2 + H$	3.00e13	0	0
$CH2CO + M' = 3CH2 + CO + M'$	1.00e16	0	248.0
$CH2CO + H = CH3 + CO$	3.60e13	0	14.1
$CH2CO + O = CHO + CHO$	2.30e12	0	5.7
$CH2CO + OH = CH2O + CHO$	1.00e13	0	0
$C_2H_3 = C_2H_2 + H$	4.73e40	−8.8	194.50
$C_2H_3 + OH = C_2H_2 + H_2O$	5.00e13	0	0
$C_2H_3 + H = C_2H_2 + H_2$	1.20e13	0	0
$C_2H_3 + O = C_2H_2 + OH$	1.00e13	0	0
$C_2H_3 + O = CH_3 + CO$	1.00e13	0	0
$C_2H_3 + O = CHO + 3CH_2$	1.00e13	0	0
$C_2H_3 + O_2 = C_2H_2 + HO_2$	5.40e12	0	0
$CH_3CO = CH_3 + CO$	2.32e26	−5.0	75.12
$CH_3CO + H = CH_2CO + H_2$	2.00e13	0	0
$CH_2CHO + H = CH_2CO + H_2$	2.00e13	0	0
$C_2H_4 + M' = C_2H_2 + H_2 + M'$	2.50e17	0	319.8
$C_2H_4 + M' = C_2H_3 + H + M'$	1.70e18	0	404.0
$C_2H_4 + H = C_2H_3 + H_2$	1.70e15	0	62.9
$C_2H_4 + O = CH_2CHO + H$	5.20e5	2.08	0
$C_2H_4 + O = CHO + CH_3$	1.21e6	2.08	0
$C_2H_4 + OH = C_2H_3 + H_2O$	6.50e13	0	24.9
$CH_3CHO + M' = CH_3 + CHO + M'$	7.00e15	0	342.8
$CH_3CHO + H = CH_3CO + H_2$	2.10e9	1.16	10.1
$CH_3CHO + H = CH_2CHO + H_2$	2.00e9	1.16	10.1
$CH_3CHO + O = CH_3CO + OH$	5.00e12	0	7.6
$CH_3CHO + O = CH_2CHO + OH$	8.00e11	0	7.6
$CH_3CHO + O_2 = CH_3CO + HO_2$	4.00e13	0	164.3

Reaktion	A_r	β_r	E_{ar}
$CH_3CHO + OH = CH_3CO + H_2O$	2.30e10	0.73	−4.7
$CH_3CHO + HO_2 = CH_3CO + H_2O_2$	3.00e12	0	50.0
$CH_3CHO + 3CH_2 = CH_3CO + CH_3$	2.50e12	0	15.9
$CH_3CHO + CH_3 = CH_3CO + CH_4$	$2.e-6$	5.64	10.3
$C_2H5 = C_2H_4 + H$	1.02e43	−9.1	224.15
$C_2H5 + H = CH_3 + CH_3$	3.00e13	0	0
$C_2H5 + O = CH_3CHO + H$	5.00e13	0	0
$C_2H5 + O = CH_2O + CH_3$	1.00e13	0	0
$C_2H5 + O_2 = C_2H_4 + HO_2$	1.10e10	0	−6.3
$C_2H5 + CH_3 = C_2H_4 + CH_4$	1.14e12	0	0
$C_2H5 + C_2H5 = C_2H_4 + C_2H6$	1.40e12	0	0
$C_2H5O = CH_3CHO + H$	2.51e14	0	97.0
$C_2H5O = CH_2O + CH_3$	1.00e15	0	90.4
$C_2H5O + O_2 = CH_3CHO + HO_2$	5.01e12	0	16.7
$C_2H5O + OH = CH_3CHO + H_2O$	1.32e12	0	0
$C_2H5O + H = CH_3CHO + H_2$	1.80e13	0	0
$CH_3CHOH = CH_3CHO + H$	1.00e14	0	105.0
$CH_3CHOH + H = CH_3CHO + H_2$	3.00e13	0	0
$CH_3CHOH + OH = CH_3CHO + H_2O$	1.51e13	0	0
$CH_3CHOH + O = CH_3CHO + OH$	1.20e14	0	0
$CH_3CHOH + O_2 = CH_3CHO + HO_2$	1.20e13	0	0
$CH_2CH_2OH = C_2H_4 + OH$	1.00e14	0	140.0
$CH_2CH_2OH + H = CH_3CHO + H_2$	5.00e13	0	0
$C_2H6 + H = C_2H5 + H_2$	1.40e9	1.5	31.1
$C_2H6 + O = C_2H5 + OH$	1.00e9	1.5	24.4
$C_2H6 + OH = C_2H5 + H_2O$	7.20e6	2.0	3.6
$C_2H6 + HO_2 = C_2H5 + H_2O_2$	1.70e13	0	85.9
$C_2H6 + O_2 = C_2H5 + HO_2$	6.00e13	0	217.0
$C_2H6 + 3CH_2 = C_2H5 + CH_3$	2.20e13	0	36.3
$C_2H6 + CH_3 = C_2H5 + CH_4$	$1.5e-7$	6.0	25.4
$C_2H5OH = CH_3 + CH_2OH$	2.51e16	0	353.0
$C_2H5OH + OH = CH_3CHOH + H_2O$	5.25e6	2.0	1.9
$C_2H5OH + OH = C_2H5O + H_2O$	1.15e6	2.0	3.8
$C_2H5OH + OH = CH_2CH_2OH + H_2O$	8.13e6	2.0	2.5
$C_2H5OH + O = CH_3CHOH + OH$	7.94e12	0	13.6
$C_2H5OH + O = C_2H5O + OH$	4.79e13	0	28.7
$C_2H5OH + O = CH_2CH_2OH + OH$	1.00e14	0	31.3
$C_2H5OH + H = CH_3CHOH + H_2$	4.40e12	0	19.1
$C_2H5OH + H = C_2H5 + H_2O$	5.90e11	0	14.4
$C_2H5OH + HO_2 = CH_3CHOH + H_2O_2$	6.30e12	0	81.1
$C_2H5OH + CH_3 = CH_3CHOH + CH_4$	2.04e11	0	36.4
$C_2H5OH + CH_3 = CH_2CH_2OH + CH_4$	2.04e11	0	36.4
$C_2H5OH + CH_3 = C_2H5O + CH_4$	7.49e10	0	39.3
$C_2H5OH + CH_3O = CH_3CHOH + CH_3OH$	2.00e11	0	29.3
$C_2H5OH + CH_2O = C_2H5O + CH_3O$	1.53e12	0	333.2
$C_2H5OH + C_2H5O = C_2H5OH + CH_3CHOH$	2.00e11	0	29.3

Bibliography

[Alt99] H.W. Alt. *Lineare Funktionalanalysis*. Springer, Berlin, 1999.

[Amd67] G. Amdahl. Validity of the single processor approach to achieving large-scale computer capabilities. In *AFIPS Conference Proceedings*, volume 30, pages 383–385, 1967.

[Ape99] T. Apel. *Anisotropic finite elements: Local estimates and applications*. Advances in Numerical Mathematics. Teubner, Stuttgart, 1999.

[B+99] P. Bastian et al. A parallel software-platform for solving problems of partial differential equations using unstructured grids and adaptive multigrid methods. High performance computing in science and engineering. Springer, 1999.

[Bas93] P. Bastian. Parallel adaptive multigrid methods. Technical Report 93–60, Interdisziplinäres Zentrum für Wissenschaftliches Rechnen, 1993.

[Bas96] P. Bastian. *Parallele adaptive Mehrgitterverfahren*. Teubner Skripten zur Numerik. B.G. Teubner, Stuttgart, 1996.

[Bas03] P. Bastian. Paralleles Rechnen I. Vorlesungsskriptum, 2003.

[BB+] R. Becker, M. Braack, et al. Gascoigne 3d - a finite element toolbox. http://gascoigne.uni-hd.de.

[BB00] R. Becker and M. Braack. Multigrid techniques for finite elements on locally refined meshes. *Numerical Linear Algebra with Applications (Special Issue)*, 7:363–379, 2000.

[BB01] R. Becker and M. Braack. A finite element pressure gradient stabilization for the stokes equations based on local projections. *Calcolo*, 38:173–199, 2001.

[BB04] R. Becker and M. Braack. A two-level stabilization scheme for the navier-stokes equations. In Feistauer et al., editors, *Enumath 2003*, pages 123–130, Prague, 2004. Springer.

[BBR99] R. Becker, M. Braack, and R. Rannacher. Numerical simulation of laminar flames at low mach number with adaptive finite elements. *Combustion Theory and Modelling*, 3(3):503–534, 1999.

[BE03] M. Braack and A. Ern. A posteriori control of modeling errors and discretization errors. *SIAM J. Multiscale Modeling and Simulation*, 1(2):221–238, 2003.

[Bec95] R. Becker. *An Adaptive Finite Element Method for the Incompressible Navier-Stokes Equations on Time-dependent Domains*. PhD thesis, Universität Heidelberg, 1995.

[Bec01] R. Becker. *Adaptive Finite Elements for Optimal Control Problems*. Habilitationsschrift, Institut für Angewandte Mathematik, Universität Heidelberg, 2001.

[BF91] F. Brezzi and M. Fortin. *Mixed and hybrid finite element methods*. Springer Series in Computational Mathematics. Springer, New York, 1991.

[BH82] A. Brook and T. Hughes. Streamline upwind/petrov-galerkin formulation for convection dominated flow with particular emphasis on the incompressible navier-stokes equations. *Comp. Meth. Appl. Mech. and Engng.*, 32:199–259, 1982.

[BPX90] J. H. Bramble, J. E. Pasciak, and J. Xu. Parallel multilevel preconditioners. *Math. Comp.*, 55:1–22, 1990.

[BR96] R. Becker and R. Rannacher. A feed-back approach to error control in finite element methods: Basic analysis and examples. *East-West J. Numer. Math.*, 4(4):237–264, 1996.

[BR01] R. Becker and R. Rannacher. An optimal control approach to a posteriori error estimation in finite element methods. In A. Iserles, editor, *Acta Numerica 2001*. Cambridge University Press, 2001.

[BR05a] M. Braack and T. Richter. Solutions of 3d navier-stokes benchmark problems with adaptive finite elements. *Computers & Fluids (submitted)*, 2005.

[BR05b] M. Braack and T. Richter. Solving multidimensional reactive flow problems with adaptive finite elements. In R. Rannacher et. al., editor, *Reactive Flows, Diffusion and Transport*. Springer, 2005. to appear.

[BR05c] M. Braack and T. Richter. Stabilized finite elements for 3d reactive flows. *International journal for numerical methods in fluids*, 2005. submitted.

[Bra81] A. Brandt. Multigrid solvers on parallel computers. In M. H. Schultz, editor, *Elliptic Problem Solvers*, pages 39–83. Academic Press, New York, 1981.

[Bra98] M. Braack. *An Adaptive Finite Element Method for Reactive Flow Problems*. Dissertation, Universität Heidelberg, 1998.

[Bra03] D. Braess. *Finite Elemente. Theorie, schnelle Löser und Anwendungen in der Elastizit"atstheorie*. Springer, Berlin, 2003.

[BS94] S. Brenner and R.L. Scott. *The mathematical theory of finite element methods*. Springer, Berlin Heidelberg New York, 1994.

[BV04] R. Becker and B. Vexler. A posteriori error estimation for finite element discretization of parameter identification problems. *Numerische Mathematik*, 96(3):435–459, 2004.

[Cia78] P.G. Ciarlet. *The Finite Element Method for Elliptic Problems*. North-Holland Publishing Company, Amsterdam, 1978.

[Cle75] Ph. Clement. Approximation by finite element functions using local regularization. *Revue Franc. Automat. Inform. Rech. Operat.*, 9(R-2):77–84, 1975.

[CO84] C.F. Carey and J.T. Oden. *Finite Elements, Computational Aspects, Vol III*. Prentice-Hall, New Jersey, 1984.

[CSvS86] C. Cuvelier, A. Segal, and A.A. van Steenhoven. *Finite Element Methods and Navier-Stokes Equations*. Mathematics and Its Applications. D. Reidel Publishing Company, Dordrecht, 1986.

[Fic55] A. Fick. Über Diffusion. *Annu. Phys.*, 94:59–86, 1855.

[FM88] P. O. Frederickson and O. A. McBryan. Parallel superconvergent multigrid. In S. F. McCormick, editor, *Multigrid Methods: Theory, Applications, and Supercomputing*, volume 110 of *Lecture Notes in Pure and Applied Mathematics*, pages 195–210. Marcel Dekker, New York, 1988.

[Gal94a] G.P. Galdi. *An Introduction to the Mathematical Theory of the Navier-Stokes Equations. Volume I, Linearised Steady Problems*, volume 38 of *Springer Tracts in Natural Philosophy*. Springer, New York, 1994.

[Gal94b] G.P. Galdi. *An Introduction to the Mathematical Theory of the Navier-Stokes Equations. Volume II, Nonlinear Steady Problems*, volume 38 of *Springer Tracts in Natural Philosophy*. Springer, New York, 1994.

[GGKK03] A. Grama, A. Grupta, G. Karypis, and V. Kumar. *Introduction to parallel Computing, second edition*. Pearson Education. Addison-Wesley, 2003.

[GR86] V. Girault and P.A. Raviart. *Finite Element Methods for Navier-Stokes Equations*. Springer, Berlin, 1986.

[Gro78] C.E. Grosch. Poisson solvers on large array computers. In B.L. Buzbee and J.F. Morrison, editors, *LANL Workshop on vector and parallel processors*, volume 198, 1978.

[Gue99] J.-L. Guermond. Stabilization of galerkin approximations of transport equations by subgrid modeling. *Modél, Math. Anal. Numér.*, 33(6):1293–1316, 1999.

[Hac85] W. Hackbusch. *Multi-Grid methods and applications*. Springer, 1985.

[Hac93] W. Hackbusch. *Iterative solution of large sparse systems of equations*, volume 95 of *Applied Math. Sciences*. Springer, 1993.

[Hel] Helics. Heidelberg linux cluster system. http://www.helics.de.

[HFM86] T.J.R. Hughes, L.P. Franca, and M. Malestra. A new finite element formulaion for computational fluid dynamics. v. circumventing the babuska-brezzi condition: a stable petrov-galerkin formulation of the stokes problemn accommodating equal-order interpolation. *Comp. Methods Appl. Mech. Engrg.*, 59:85–99, 1986.

[HRT96] J. Heywood, R. Rannacher, and S. Turek. Artificial boundaryies and flux and pressure conditions for the incompressible navier-stokes equations. *Int. J. Numer. Meth. Fluids*, 22:325–352, 1996.

[Joh87] C. Johnson. *Numerical Solution of partial differential equations by finite element method*. Cambridge University Press, Cambridge, 1987.

[Joh02] V. John. Higher order finite element methods and multigrid solvers in a benchmark problem for the 3d navier-stokes equations. *Int. J. Numer. Math. Fluids*, pages 775–798, 2002.

[Kar] G. Karypis. Metis - a family of multilevel partitioning algorithms. http://www-users.cs.umn.edu/ karypis/metis/index.html.

[Osw01] H. Oswald. *Parallele Lösung der instationären Navier-Stokes Gleichungen*. PhD thesis, Universtität Heidelberg, 2001.

[PBRW03] S. Parmentier, M. Braack, U. Riedel, and J. Warnatz. Modeling of combustion in a lamella burner. *Combust. Sci. and Tech.*, 175:185–206, 2003.

[QV99] A. Quarteroni and A. Valli. *Domain Decomposition Methods for Partial Differential Equations*. Clarendon Press, Oxford, 1999.

[Ric01] T. Richter. Funktionalorientierte Gitteroptimierung bei der Finite-Elemente-Approximation elliptischer Differentialgleichungen. Diploma thesis, Universität Heidelberg, 2001.

[RR94] Ch. G. Rossmann and H.-G. Roos. *Numerik partieller Differentialgleichungen*. B.G. Teubner, Stuttgart, 1994.

[RSN82] F. Riesz and B. Sz.-Nagy. *Vorlesungen über Functionalanalysis*. VEB Deutscher Verlag der Wissenschaften, 1982.

[RT92] R. Rannacher and S. Turek. A simple nonconforming quadrilateral stokes element. *Numer. Meth. Part. Diff. Equ.*, 8:97–111, 1992.

[Saa03] Y. Saad. *Iterative Methods for Sparse Linear Systems*. SIAM, Philadelphia, 2003.

[Sch69] H.A. Schwarz. Über einige Abbildungsaufgaben. *J. Reine Angew. Math.*, 70:105–120, 1869.

[Sch96] P. Schreiber. *Eine nichtkonforme Finite-Elemente-Methods zur Lösung der inkompressiblen 3-D Navier-Stokes Gleichungen*. PhD thesis, Universität Heidelberg, 1996.

[SMK89] M. D. Smooke, R. E. Mitchell, and D. E. Keyes. Numerical solution of two-dimensional axisymmetric laminar diffusion flames. *Comb. Sci. and Tech.*, 67:85–122, 1989.

[ST96] M. Schäfer and S. Turek. Benchmark computations of laminar flow around a cylinder. *Notes Numer. Fluid Mech.*, 42:547–566, 1996. Flow Simulations with High-Performance Computers II. DFG priority research programm results 1993-1995.

[SZ91] R. Scott and D. Zhang. Interpolation of non-smooth functions. *Math. Comp.*, 13:1311–1328, 1991.

[Ver96] R. Verfürth. *A Review of a Posteriori Error Estimation and Adaptive Mesh-Refinement Techniques*. Wiley and Teubner, 1996.

[Wie01] R. Wienands. Extended local fourier analysis for multigrid: Optimal smoothing, coarse grid correction, and preconditioning. Technical Report 20, GMD Research Series, St. Augustin, 2001.

[Wil85] F. A. Williams. *Combustion Theory*. Addison-Wesley Publishing Company, Redwood City, 1985.

[WMD96] J. Warnatz, U. Maas, and R.W. Dibble. *Combustion*. Springer, New York, 1996.

[Zei90] E. Zeidler. *Nonlinear functional analysis and its applications I-IV*. Springer, Berlin, 1985-1990.

[ZZ87] O.C. Zienkiewicz and J.Z. Zhu. A simple error estimator and adaptive procedure for practical engineering analysis. *Int. J. Numer. Meth. Engrg.*, 24:337–357, 1987.