FACULTY
OF MATHEMATICS
AND PHYSICS
Charles University

DOCTORAL THESIS

Petr Lukáš

Numerical Solution of Convection-dominated Problems

Department of Numerical Mathematics

Supervisor of the doctoral thesis:	doc. Mgr. Petr Knobloch, Dr., DSc.
Study programme:	Mathematics
Study branch:	Computational mathematics

Prague 2021

I declare that I carried out this doctoral thesis independently, and only with the cited sources, literature and other professional sources.

I understand that my work relates to the rights and obligations under the Act No. 121/2000 Sb., the Copyright Act, as amended, in particular the fact that the Charles University has the right to conclude a license agreement on the use of this work as a school work pursuant to Section 60 subsection 1 of the Copyright Act.

In ..... date ..... signature of the author

This Thesis summarizes our research that took us already more than the last ten years. I would like to sincerely thank Petr Knobloch, without whom I would not have been able to complete the research, and without whom I would not have written it through all the scientific articles and this thesis. You are a wonderful supervisor, your insight and knowledge into the subject matter steered me through these years!

I am grateful for the support from my family. My mother encouraged me especially in earlier years of doctoral studies whereas my father is currently a great friend of mine, supporting me always and in all of my endeavors.

My colleagues at our Faculty also need to be mentioned here, they have supported me and provided me with much important feedback. It has been a great pleasure to be part of our SIAM Student chapter in Prague for ten years and to serve as an officer there for seven years, I am thankful for that opportunity.

I would like to express my sincere gratitude to Mrs. Urbankova, the representative of Karel Urbanek fund, who supported me during my stay in the USA at the University of Nevada, Reno in 2018.

Title: Numerical Solution of Convection-dominated Problems

Author: Petr Lukáš

Department: Department of Numerical Mathematics

Supervisor: doc. Mgr. Petr Knobloch, Dr., DSc., Department of Numerical Mathematics

Abstract: Numerical solution of the scalar convection–diffusion–reaction problems often possesses spurious oscillations which appear in the discrete solution when convection dominates diffusion and standard non-adaptive discretizations are used.

Numerical solution of convection-dominated problems requires special techniques to suppress these oscillations. Often stabilized methods are applied which involve free parameters. These parameters significantly influence the quality of the solution but their optimal choice is usually not known. We define them in an adaptive way by minimizing an error indicator characterizing the quality of the approximate solution. We consider new nonlinear limited-memory optimization methods.

A nontrivial requirement on the error indicator is that its minimization with respect to the stabilization parameters should suppress spurious oscillations without smearing layers. In this thesis novel error indicators are introduced and their suitability is considered on different benchmarks.

Keywords: FEM optimization SUPG SDFEM SOLD Error indicator

Contents

Introduction

In this thesis we consider numerical solution of the standard scalar convection–diffusion–reaction problem

$$-\varepsilon\,\Delta u + \mathbf{b}\cdot\nabla u + c\,u = f \ \text{ in } \Omega, \tag{1}$$

more details will be added later in Section 1.2. A very important aspect of any numerical solution of (1) are spurious oscillations that appear in the discrete solution when convection dominates diffusion and standard non-adaptive discretizations are used. Problem (1) is considered in many different areas of science, it describes physical phenomena where particles, energy, temperature, or other physical quantities are transferred inside a physical system due to diffusion and convection.

As an example, we can imagine that the quantity u is the concentration of a chemical. When concentration is low somewhere compared to the surrounding areas, the substance will tend to diffuse in from the surroundings, so the concentration will increase and conversely, if concentration is high compared to the surroundings, then the substance will diffuse out and the concentration will decrease. This is the effect of the diffusion term $-\varepsilon\,\Delta u$.

The second term, $\mathbf{b}\cdot\nabla u$, describes convection (or advection) of the quantity u due to the flow $\mathbf{b}$. The reaction term $c\,u$ describes the effect of the quantity u on itself, if c is positive then the quantity u will have a positive effect on its value, if c is negative it will have a negative one. The source term f describes the creation or destruction of the quantity u.

There are many complicated multiphysics simulations that are comprised of convection–diffusion–reaction subproblems. Equations of this form are important parts of many complex models but problem (1) is also important on its own.

To be able to numerically solve (1) we need to discretize it first. As discussed in John et al. [2018], an ideal discretization of a convection-dominated convection–diffusion–reaction equation should satisfy the following properties:

1. The numerical solution should possess sharp layers.
2. The numerical solution must not exhibit spurious oscillations.
3. There should be an efficient way for computing the numerical solution.

The first two properties are connected with the accuracy of the discretization and usually correlate with results from the finite element error analysis in sufficiently strong norms. Since the layer width is usually smaller than the mesh width, it follows from Property 1 that it is desirable that the layer of the numerical solution is not much wider than the mesh width.

Property 2 is connected also with the physical fidelity of the numerical solution. Of course, spurious oscillations are erroneous, thus they diminish the accuracy. But even more important, they represent nonphysical situations, like negative concentrations.

Numerical solution of convection-dominated problems requires special techniques to suppress spurious oscillations in approximate solutions. Often stabilized methods are applied which involve user-chosen free parameters. These

parameters significantly influence the quality of the solution. The optimal values of these parameters are not known in general. Various stabilized methods have been proposed which often depend on the free parameters. Optimal choice of these parameters is usually not known. Theoretical bounds for the values of these parameters are derived in most cases. In this thesis we explore more different stabilized methods than just the streamline upwind/Petrov–Galerkin (SUPG) method which is the most widely used stabilized method. We explore other spurious oscillations at layers diminishing (SOLD) methods, those that add other terms to the SUPG method formulation.

There is a possibility to define the stabilization parameters in an adaptive way by minimizing an error indicator characterizing the quality of the approximate solution, this approach was introduced in John et al. [2011]. A nontrivial requirement on the error indicator is that its minimization with respect to the stabilization parameters should suppress spurious oscillations without smearing layers. In this thesis new error indicators are introduced and their suitability is tested on newly proposed benchmark problems for which previously proposed indicators do not provide satisfactory results.

The aim of the thesis is to present and to compare stabilization techniques based on an adaptive choice of parameters in the finite element method. These parameters are chosen so that they minimize a functional called error indicator, it is a nonlinear constraint optimization problem. The proposed methods are used to obtain better qualitative results than those of stabilized methods with a priori given parameters.

Since the number of variables of such an indicator is typically very large, there typically occur more local minima near the global minimum. We need a nonlinear optimization method to be able to not only minimize the error indicator but also to find a minimum that provides a physically meaningful solution. It is also important to perform these tasks in a robust way. In the thesis we study and further develop nonlinear optimization methods considered by the author in his diploma thesis, we use the standard Python–based optimization modules. A new limited-memory method based on rank-one updates of the hessian was developed and compared to the standard L-BFGS method.

One novelty of our approach consists in combining an error indicator with crosswind derivative control term and indicator with reduced residuals with nonlinear SOLD methods. Considered SOLD methods are adding an artificial diffusion either in all directions or only in the crosswind direction. We demonstrate that this approach can lead to more physically meaningful solutions than techniques considered before.

The indicator with reduced residuals is introduced as a new way for optimizing parameters in stabilized methods for the numerical solution of convection-dominated problems. Numerical results showed that the new indicator behaves better than indicators applied so far when, in regions away from layers, the exact solution is not constant in the crosswind direction. In contrast with the indicator adding the crosswind derivative control term, the new indicator is consistent with the approximated problem.

We show that the SOLD method with artificial diffusion in crosswind direction is a suitable candidate for optimization of parameters using the error indicator with reduced residuals and that the impact of the parameter optimization itself

is often crucial for this SOLD method.

The numerical experiments also comprise investigations of various initialization strategies for the SOLD parameter and revealed that initialization of the SOLD parameter by 0 leads to lower values of optimized free parameters and a less smeared solution.

We will show that using an optimization together with a stabilized method with a carefully chosen mesh can lead to a substantial increase of accuracy. For a setup on an isotropic mesh and layers aligned with the mesh, SUPG method with optimized free parameters can give us the rate of convergence of the SUPG method improved by the order of 1.

A SOLD method that adds diffusion isotropically is considered and it allows us to see where regions that need stabilization typically appear and how these regions behave in case of higher order FE spaces for both solution and the free parameters of this method.

An important question is how computationally expensive is our approach as the method is inherently nonlinear and we need multiple evaluations of a linear algebraic system corresponding to a discretization of (1) to obtain the result. It may seem as a huge overhead in comparison with linear stabilized methods but one should take into account that some very successful methods for solving (1) are also nonlinear since the artificial diffusion is added based on the computed solution.

As Godunov's theorem says (Godunov [1959]), linear numerical schemes for solving some linear partial differential equations, having the property of not generating new extrema (monotone scheme), can be at most first-order accurate. That is why nonlinear methods are important and used to get a higher-order method with a higher accuracy. In practice, the models we are solving are already nonlinear so the overhead, in general, may be relatively less important.

1. Fundamental definitions

1.1 Sobolev spaces

Let Ω be a bounded domain in $\mathbb{R}^d$, $d = 2, 3$. We say that $\partial\Omega$ is a Lipschitz boundary and Ω is called a Lipschitz domain, if for each point $s = (s_1, ..., s_d) \in \partial\Omega$ there exists a ball $B_r(s)$ of radius $r > 0$ centered at s and a Lipschitz function $\gamma : \mathbb{R}^{d-1} \to \mathbb{R}$ such that we have

$$\Omega \cap B_r(s) = \{x \in B_r(s); x_d > \gamma(x_1, ..., x_{d-1})\}$$

in a local coordinate system. For $1 \leq p < \infty$, let

$$\|u\|_{L^p(\Omega)} \equiv \left(\int_\Omega |u|^p dx\right)^{1/p},$$

for the special case of $p = \infty$, let

$$\|u\|_{L^\infty(\Omega)} \equiv \operatorname*{ess\,sup}_\Omega |u| = \inf\left\{C \in \mathbb{R}; \int_{\{\xi; |u(\xi)|>C\}} ds = 0\right\}.$$

We define the Lebesgue space $L^p(\Omega)$, $1 \leq p \leq \infty$, as

$$L^p(\Omega) \equiv \left\{u : \Omega \to \mathbb{R} \text{ measurable}; \|u\|_{L^p(\Omega)} < \infty\right\}.$$

Let $\alpha = (\alpha_1, ..., \alpha_d)$, $\alpha_i \in \mathbb{N}_0$, $1 \leq i \leq d$ be a multiindex and $|\alpha| = \sum_{i=1}^d \alpha_i$. By $C_c^\infty(\Omega)$ we denote the space of infinitely differentiable functions with compact support in Ω. For $\phi \in C_c^{|\alpha|}(\Omega)$ we define the "strong" multiindex derivative

$$D^\alpha \phi = \frac{\partial^{|\alpha|}\phi}{\partial x_1^{\alpha_1} \cdots \partial x_d^{\alpha_d}}.$$

We then define the space of locally integrable functions $L^1_{loc}(\Omega)$ as

$$L^1_{loc}(\Omega) \equiv \left\{u : \Omega \to \mathbb{R}; u \in L^1(K)\ \forall \text{ compact } K \subset \Omega\right\}.$$

We also have $L^p(\Omega) \subset L^1_{loc}(\Omega)$, $1 \leq p \leq \infty$. For $u \in L^1_{loc}(\Omega)$ we define the weak derivative $v = D^\alpha u$ such that

$$\int_\Omega v\phi\, dx = (-1)^{|\alpha|} \int_\Omega u D^\alpha \phi\, dx \quad \forall \phi \in C_c^\infty(\Omega).$$

Then according to Evans [1998], Section 5.2.1, the weak derivative, if it exists, is uniquely determined. We define the Sobolev space $W^{k,p}(\Omega)$, $k \in \mathbb{N}$, $1 \leq p \leq \infty$

$$W^{k,p}(\Omega) \equiv \{u \in L^p(\Omega), D^\alpha u \in L^p(\Omega) \forall \alpha; |\alpha| < k\}$$

with the norm

$$\|u\|_{k,p,\Omega} \equiv \sum_{|\alpha|\leq k} \|D^\alpha u\|_{L^p(\Omega)}.$$

For $p = 2$, $W^{k,2}(\Omega)$ is a Hilbert space that is usually denoted by

$$H^k(\Omega) \equiv W^{k,2}(\Omega)$$

with the norm $\|u\|_{k,\Omega} \equiv \|u\|_{k,2,\Omega}$. We define the seminorm $|u|_{k,\Omega}$ by

$$|u|_{k,p,\Omega} \equiv \sum_{|\alpha|=k} \|D^\alpha u\|_{L^p(\Omega)} \ .$$

According to this notation, we denote by $H^0(\Omega)$ the Hilbert space $L^2(\Omega)$ and its norm $\|u\|_{0,\Omega} \equiv \|u\|_{L^2(\Omega)}$.

If Ω is a domain with Lipschitz boundary we can define the space $H_0^1(\Omega)$ as functions from $H^1(\Omega)$ whose trace is zero on $\partial\Omega$.

1.2 Problem definition

We study the numerical solution of the scalar convection–diffusion–reaction equation

$$-\varepsilon\Delta u + \mathbf{b}\cdot\nabla u + cu = f \text{ in } \Omega, \quad u = u_b \text{ on } \Gamma^D \ , \ \ \varepsilon\frac{\partial u}{\partial \mathbf{n}} = g \text{ on } \Gamma^N, \tag{1.1}$$

where $\Omega \subset \mathbb{R}^d$, $d = 2, 3$ is a bounded domain with a polygonal (or polyhedral in case $d = 3$) Lipschitz-continuous boundary $\partial\Omega$. Γ^D and Γ^N are relatively open subsets of $\partial\Omega$, $\Gamma^D \cap \Gamma^N = \emptyset$, and $\overline{\Gamma^D \cup \Gamma^N} = \partial\Omega$. Furthermore, $\varepsilon > 0$ is a constant diffusivity, $\mathbf{b} \in W^{1,\infty}(\Omega)^d$ is a convective field, $c \in L^\infty(\Omega)$ is a reaction coefficient, $f \in L^2(\Omega)$ is an outer source of u, $u_b \in H^{1/2}(\Gamma^D)$ is the Dirichlet boundary condition (less regular u_b can be considered as well), and $g \in L^2(\Gamma^N)$ is a function prescribing the Neumann boundary condition. We further assume that inflow boundary parts will only be subsets of Γ^D, more precisely $\{x \in \partial\Omega, (\mathbf{b}\cdot\mathbf{n})(x) < 0\} \subset \Gamma^D$, where $\mathbf{n}$ is the unit outward normal vector.

Let us introduce the usual assumption

$$c - \frac{1}{2}\mathrm{div}\mathbf{b} \geq 0, \tag{1.2}$$

which guarantees that (1.1) admits a unique weak solution for all $\varepsilon > 0$ (Roos et al. [2008]). In our examples we usually have $\mathrm{div}\mathbf{b} = 0$.

Numerical solution of (1.1) is still a challenge when convection is strongly dominant ($\varepsilon \ll |\mathbf{b}|$). The exact solution then typically possesses interior and boundary layers, where the derivatives of the solution are very large. These layers are often narrower than the mesh size and hence cannot be resolved properly. This leads to unwanted spurious (nonphysical) oscillations in the numerical solution. Multiple possible remedies to this problem are in scope of our work.

1.3 Weak formulation

Weak formulation of (1.1) is obtained by multiplying the partial differential equation by a test function and integrating over the domain Ω and applying integration by parts. The formulation of the problem is to find $u \in H^1(\Omega)$ such that $u - \tilde{u}_b \in \{v \in H^1(\Omega); v = 0 \text{ on } \Gamma^D\}$ and

$$a(u, v) = (f, v) + \langle g, v\rangle_{\Gamma^N} \quad \forall v \in \{v \in H^1(\Omega); v = 0 \text{ on } \Gamma^D\}, \tag{1.3}$$

where $(\cdot,\cdot)$ is the inner product in $L^2(\Omega)$ (in $L^2(\Omega)^d$, respectively), $\langle\cdot,\cdot\rangle_{\Gamma^N}$ is the standard scalar product in $L^2(\Gamma^N)$,

$$a(u,v) = \varepsilon(\nabla u, \nabla v) + (\mathbf{b}\cdot\nabla u, v) + (cu, v), \tag{1.4}$$

and $\tilde{u}_b \in H^1(\Omega)$ is an extension of u_b from $H^{1/2}(\Gamma^D)$ to the space $H^1(\Omega)$.

At the outflow parts of the boundary it is possible to consider the natural boundary condition. We will provide results for such situations in our numerical tests.

2. Discretization

To be able to compute the solution of (1.1) on a computer we need to move from a continuous problem to a discretized one. In the later parts of this chapter, we will treat problems which arise from the discretization by introducing methods which suppress spurious oscillations without extensively smearing layers originating from boundary conditions.

2.1 Galerkin discretization

Let $\{\mathcal{T}_h\}_h$ be a family of triangulations of $\overline{\Omega}$ parametrized by positive parameters h whose only accumulation point is zero. The triangulations $\mathcal{T}_h$ are assumed to consist of a finite number of open simplices T of Ω such that $\overline{\Omega} = \bigcup_{T\in\mathcal{T}_h} \overline{T}$ and the closures of any two different sets in $\mathcal{T}_h$ are either disjoint or possess either a common vertex, common edge or common face (if $d = 3$). This means the triangulation $\mathcal{T}_h$ complies with usual compatibility conditions and no hanging nodes are present. We assume that there exists a constant ν which satisfies

$$\frac{h_T}{\rho_T} \leq \nu \quad \forall T \in \cup\mathcal{T}_h, \tag{2.1}$$

where h_T is the diameter of the element, and ρ_T is the diameter of the largest inscribed ball in the element T.

For each h, we introduce a finite element space $W_h \subset H^1(\Omega)$ defined on $\mathcal{T}_h$ and approximating the space $H^1(\Omega)$ in the sense described in Ciarlet [1978]. The function $\tilde{u}_b$ is approximated by $\tilde{u}_{bh} \in W_h$. We define the space V_h as $V_h = W_h \cap \{v \in H^1(\Omega); v = 0 \text{ on } \Gamma^D\}$. We then say that $u_h \in W_h$ is a discrete solution of (1.1) if $u_h - \tilde{u}_{bh} \in V_h$ and

$$a(u_h, v_h) = (f, v_h) + \langle g, v_h\rangle_{\Gamma^N} \tag{2.2}$$

for all $v_h \in V_h$, where a is the bilinear form from (1.4).

In the following, we particularly let W_h be a conforming finite element space that consists of piecewise polynomials of degree k, i.e.,

$$W_h := \left\{v_h \in H^1(\Omega) : v_h|_T \in P_k(T) \quad \text{for all} \quad T \in \mathcal{T}_h\right\} \tag{2.3}$$

2.2 SUPG method

According to Roos et al. [2008], the Galerkin discretization is inappropriate if convection dominates diffusion, since then the discrete solution is globally polluted by spurious oscillations. An improvement can be achieved by adding stabilization terms to the Galerkin discretization. One of the most efficient procedures of this type is the streamline upwind/Petrov–Galerkin (SUPG) method developed by Brooks and Hughes [1982]. This method combines global stability properties with high accuracy in subdomains excluding boundary layers.

Since we are using piecewise polynomial approximations the second order derivatives are defined only element-wise in general ($\Delta u_h \notin L^2(\Omega)$). As an example of this, let us take a piecewise linear finite element space. There are jumps of

the first derivative across the element's sides, so the second order derivatives are not defined there in general. First, we define the residue $R_h(u)$:

$$R_h(u) = -\varepsilon\Delta_h u + \mathbf{b}\cdot\nabla u + cu - f, \tag{2.4}$$

where $\Delta_h u_h \in L^2(T)$ for each element T, so we calculate $\Delta_h u_h$ element by element.

In the formulation of the SUPG method we are adding a stabilization term $(R_h(u_h), \tau\mathbf{b}\cdot\nabla v_h)$ to the left-hand side of the Galerkin discretization. We will now seek $u_h \in W_h$ such that $u_h - \tilde{u}_{bh} \in V_h$ and

$$a(u_h, v_h) + (R_h(u_h), \tau\mathbf{b}\cdot\nabla v_h) = (f, v_h) + \langle g, v_h\rangle_{\Gamma^N} \quad \forall v_h \in V_h, \tag{2.5}$$

where τ is a nonnegative stabilization parameter. We assume that all admissible stabilization parameters form a set $Y_h \subset L^\infty(\Omega)$. When the parameter τ is chosen from a discrete finite element space we will denote it as τ_h. From now on we will use the discrete bilinear form a_h

$$a_h(u_h, v_h) = a(u_h, v_h) + (-\varepsilon\Delta_h u + \mathbf{b}\cdot\nabla u + cu, \tau\mathbf{b}\cdot\nabla v_h) \tag{2.6}$$

for all $u_h \in W_h$ and $v_h \in V_h$.

Since the residue of the exact solution is zero, the method is automatically consistent in the finite element sense - i.e., the solution of the original boundary value problem also satisfies the discrete variational system of equations (2.5).

Combining the consistency of the SUPG method with (1.3) yields the projection property

$$a_h(u - u_h, v_h) = 0 \quad \forall v_h \in V_h, \tag{2.7}$$

if the solution of (1.3) satisfies $u \in H^2(\Omega)$, this identity is known as Galerkin orthogonality.

2.2.1 Approximation and local inverse inequality

Let $u \in H^{k+1}(\Omega)$ with $k \geq 1$ so that $u \in C(\overline{\Omega})$. Then we can define an interpolant u^I in W_h and according to Grossmann et al. [2007] it enjoys an approximation property

$$|u - u^I|_{m,T} \leq C h_T^{k+1-m} |u|_{k+1,T} \quad \text{for } m = 0, 1, 2 \tag{2.8}$$

for all $T \in \mathcal{T}_h$, where the constant C does not depend on mesh size h_T and the polynomial degree k.

Using a scaling argument and the equivalence of norms in finite-dimensional spaces, we can prove the following local inverse inequality, see Ciarlet [1978], where μ_{inv} is independent of T and h:

$$||\Delta v_h||_{0,T} \leq \mu_{\text{inv}} h_T^{-1} |v_h|_{1,T} \quad \forall\, v_h \in V_h. \tag{2.9}$$

2.2.2 Stability and a priori error estimates

We will assume a slightly stronger condition than (1.2) in the following. Let us alter the condition from (1.2) by introducing an auxiliary constant σ_0:

$$c - \frac{1}{2}\text{div}\mathbf{b} \geq \sigma_0 > 0. \tag{2.10}$$

We will measure the stability and derive the error estimates in the following norm which is naturally related to the bilinear for a_h. We define τ_T as the value of the parameter τ_h on a given element $T \in \mathcal{T}_h$, for τ_h from the piecewise constant finite element space.

$$|||v|||_{SD} := \left(\varepsilon |v|^2_{1,\Omega} + \sum_{T\in\mathcal{T}_h} \tau_T ||\mathbf{b}\cdot\nabla v||^2_{0,T} + \sigma_0 ||v||^2_{0,\Omega} + \frac{1}{2}\langle |\mathbf{b}\cdot\mathbf{n}|, v^2\rangle_{\Gamma^N} \right)^{1/2}. \tag{2.11}$$

Now, let us introduce two integration formulas which we will use later in the proof of Theorem 1:

$$\int_\Omega (\mathbf{b}\cdot\nabla v) v \, dx = \frac{1}{2}\int_\Omega \mathbf{b}\cdot\nabla(v^2)\, dx, \tag{2.12}$$

$$\int_\Omega \mathbf{b}\cdot\nabla(v^2)\, dx = \int_{\Gamma^N} v^2\, \mathbf{b}\cdot\mathbf{n}\, dS - \int_\Omega (\nabla\cdot\mathbf{b})\, v^2\, dx, \tag{2.13}$$

where (2.12) holds for all $v \in H^1(\Omega)$, whereas (2.13) holds only for all $v \in H^1(\Omega)$ with $v = 0$ on Γ^D.

Theorem 1 (Coercivity of a_h)**.** *Let the parameter τ_h of (2.5) satisfy the following inequality for all $T \in \mathcal{T}_h$:*

$$0 < \tau_T \le \frac{1}{2}\min\left\{ \frac{\sigma_0}{c_T^2}, \frac{h_T^2}{\varepsilon \mu_{\text{inv}}^2} \right\},$$

where $c_T = ||c||_{0,\infty,T}$. Then the discrete bilinear form is coercive, which means that it holds

$$a_h(v_h, v_h) \ge \frac{1}{2}|||v_h|||^2_{SD} \quad \forall v_h \in V_h.$$

Proof. For each $v_h \in V_h$ we obtain using the two integration formulas (2.12) and (2.13)

$$\begin{aligned} a_h(v_h, v_h) = &\int_\Omega \varepsilon |\nabla v_h|^2 dx + \int_\Omega c v_h^2 - \frac{1}{2}(\nabla\cdot\mathbf{b})\, v_h^2\, dx + \frac{1}{2}\int_{\Gamma^N} v_h^2\, \mathbf{b}\cdot\mathbf{n}\, dS \\ &+ \sum_{T\in\mathcal{T}_h} (-\varepsilon\Delta_h v_h + \mathbf{b}\cdot\nabla v_h + c v_h, \tau_h \mathbf{b}\cdot\nabla v_h)_T. \end{aligned}$$

From the last equation we get the initial estimate

$$\begin{aligned} a_h(v_h, v_h) \ge &\varepsilon |v_h|^2_{1,\Omega} + \sigma_0 ||v_h||^2_{0,\Omega} + \frac{1}{2}\langle |\mathbf{b}\cdot n|, v_h^2\rangle_{\Gamma^N} + \sum_{T\in\mathcal{T}_h} \|\mathbf{b}\cdot\nabla v_h\|^2_{0,T}\tau_T \\ &+ \sum_{T\in\mathcal{T}_h} \tau_T(-\varepsilon\Delta_h v_h + c v_h, \mathbf{b}\cdot\nabla v_h)_T. \end{aligned}$$

We see that the first part of the right-hand side of the last inequality is directly $|||v_h|||^2_{SD}$. Let us estimate the remaining term from above. The Young's inequality, local inverse inequality (2.9), and the hypothesis on τ_h give us

$$\begin{aligned} &\Big| \sum_{T\in\mathcal{T}_h} \tau_T(-\varepsilon\Delta_h v_h + c v_h, \mathbf{b}\cdot\nabla v_h)_T \Big| \\ &\quad \le \sum_{T\in\mathcal{T}_h} \varepsilon^2 \tau_T ||\Delta v_h||^2_{0,T} + \sum_{T\in\mathcal{T}_h} c_T^2 \tau_T ||v_h||^2_{0,T} + \frac{1}{2}\sum_{T\in\mathcal{T}_h} \tau_T ||\mathbf{b}\cdot\nabla v_h||^2_{0,T} \\ &\quad \le \frac{\varepsilon}{2}|v_h|^2_{1,\Omega} + \frac{\sigma_0}{2}||v_h||^2_{0,\Omega} + \frac{1}{2}\sum_{T\in\mathcal{T}_h} \tau_T ||\mathbf{b}\cdot\nabla v_h||^2_{0,T} \le \frac{1}{2}|||v_h|||^2_{SD}\,. \end{aligned}$$

This yields the desired coercivity result of the lemma.

□

To get an error estimate we will first focus on the error between u_h and the interpolant u^I from (2.8). From now on we will need to consider $\tilde{u}_{bh}$ chosen in such a way so that $u^I - u_h \in V_h$. Theorem 1 and the Galerkin orthogonality (2.7) provide the following estimate for all $u \in H^2(\Omega)$

$$\frac{1}{2}|||u^I - u_h|||_{SD}^2 \leq a_h(u^I - u_h, u^I - u_h) = a_h(u^I - u, u^I - u_h) \ . \tag{2.14}$$

We will continue by estimating the right-hand side of (2.14) term by term. By invoking the interpolation properties from (2.8) we obtain for $u \in H^{k+1}(\Omega)$:

$$\varepsilon(\nabla(u^I - u), \nabla(u^I - u_h)) \leq \varepsilon^{\frac{1}{2}}|u^I - u|_{1,\Omega}|||u^I - u_h|||_{SD} \leq C\varepsilon^{\frac{1}{2}}h^k|u|_{k+1,\Omega}|||u^I - u_h|||_{SD},$$

$$\begin{aligned}
&(\mathbf{b}\cdot\nabla(u^I - u) + c(u^I - u), u^I - u_h) \\
&\quad = \Big((c - \nabla\cdot\mathbf{b})(u^I - u), u^I - u_h\Big) - \Big(u^I - u, \mathbf{b}\cdot\nabla(u^I - u_h)\Big) \\
&\qquad + \langle|\mathbf{b}\cdot\mathbf{n}|(u^I - u), u^I - u_h\rangle_{\Gamma^N} \\
&\quad \leq \left[C\left(\sum_{T\in\mathcal{T}_h}||u^I - u||_{0,T}^2\right)^{\frac{1}{2}} + \left(\sum_{T\in\mathcal{T}_h}\tau_T^{-1}||u^I - u||_{0,T}^2\right)^{\frac{1}{2}}\right]|||u^I - u_h|||_{SD} \\
&\quad \leq Ch^k\left[\sum_{T\in\mathcal{T}_h}h_T^2(1 + \tau_T^{-1})|u|_{k+1,T}^2\right]^{\frac{1}{2}}|||u^I - u_h|||_{SD} \ ,
\end{aligned}$$

$$\begin{aligned}
&\Big|\sum_{T\in\mathcal{T}_h}\tau_T\Big(-\varepsilon\Delta(u^I - u) + \mathbf{b}\cdot\nabla(u^I - u) + c(u^I - u), \mathbf{b}\cdot(u^I - u_h)\Big)_T\Big| \\
&\qquad \leq C\sum_{T\in\mathcal{T}_h}\tau_T^{\frac{1}{2}}(\varepsilon h_T^{k-1} + h_T^k + h_T^{k+1})|u|_{k+1,T}\,\tau_T^{\frac{1}{2}}||\mathbf{b}\cdot\nabla(u^I - u_h)||_{0,T} \\
&\qquad \leq C\left[\sum_{T\in\mathcal{T}_h}(\varepsilon + \tau_T)h_T^{2k}|u|_{k+1,T}^2\right]^{\frac{1}{2}}|||u^I - u_h|||_{SD} \ ,
\end{aligned}$$

where the inequality $\varepsilon\tau_T \leq Ch_T^2$ was used (from Theorem 1). Combining the upper results we get

$$|||u^I - u_h|||_{SD} \leq C\left[\sum_{T\in\mathcal{T}_h}(\varepsilon + \tau_T + \tau_T^{-1}h_T^2 + h_T^2)h_T^{2k}|u|_{k+1,T}^2\right]^{\frac{1}{2}} \ . \tag{2.15}$$

To extract the best possible convergence rate from (2.15) we need to balance the terms while respecting the definition of constraints on τ_T from Theorem 1. Let us set the parameter τ_T in the following manner:

$$\tau_T = \begin{cases} \tau_0 h_T/\|\mathbf{b}\|_{0,\infty,T} & \text{if } Pe_T > 1 \ \text{ (convection dominated case)}, \\ \tau_1 h_T^2/\varepsilon & \text{if } Pe_T \leq 1 \ \text{ (diffusion dominated case)}, \end{cases} \tag{2.16}$$

with appropriate positive constants τ_0 and τ_1, where the local mesh Péclet number Pe_T is defined as

$$Pe_T := \frac{\|\mathbf{b}\|_{0,\infty,T} h_T}{2\varepsilon} . \tag{2.17}$$

Theorem 2 (Global error estimate)**.** *Let the hypotheses of Theorem 1 be satisfied. Choose the parameter τ_T according to* (2.16)*. Then the solution u_h of the SDFEM satisfies the global error estimate*

$$|||u - u_h|||_{SD} \leq C(\varepsilon^{\frac{1}{2}} + h^{\frac{1}{2}})h^k |u|_{k+1,\Omega} .$$

Proof. From (2.15) and (2.16) follows

$$|||u^I - u_h|||_{SD} \leq C(\varepsilon^{\frac{1}{2}} + h^{\frac{1}{2}})h^k |u|_{k+1,\Omega}.$$

By employing the interpolation properties (2.8) we get

$$|||u - u_I|||_{SD} \leq C(\varepsilon^{\frac{1}{2}} + h^{\frac{1}{2}})h^k |u|_{k+1,\Omega}.$$

Triangle inequality then completes the proof. □

If convection dominates diffusion we have $\varepsilon < \|\mathbf{b}\|_{0,\infty,T}\frac{h_T}{2}$ and $\tau_T = \tau_0 \frac{h_T}{\|\mathbf{b}\|_{0,\infty,T}}$. So on meshes satisfying $h < Ch_T$ one obtains the global estimate

$$||u - u_h||_{0,\Omega} + h^{\frac{1}{2}}\Big(\sum_{T\in\mathcal{T}_h} \|\mathbf{b}\cdot\nabla(u - u_h)\|_{0,T}^2\Big)^{\frac{1}{2}} \leq Ch^{k+\frac{1}{2}}|u|_{k+1,\Omega}.$$

We see from the approximation properties (2.8) that the L^2 error of the derivative in the streamline direction (second term) is optimal, but the bound on $\|u-u_h\|_{0,\Omega}$ is of the order $\frac{1}{2}$ less than optimal.

2.2.3 Choice of τ_h parameter in one dimension

Theorem 1 provides bounds for τ for which the SUPG method is stable and leads to a quasi-optimal convergence of the discrete solution u_h. Although the bounds for τ are known, the accuracy can be dramatically influenced by the choice of the parameter inside these bounds.

It has been shown in Brooks and Hughes [1982] that for a one-dimensional case of (1.1) with constant data (in one dimension the convection is a scalar so let us denote it by b instead of $\mathbf{b}$), Dirichlet boundary conditions, equidistant mesh, and P_1 finite elements the nodally exact solution is obtained if and only if τ is defined as

$$\tau = \frac{h}{2\,|b|}\xi_0(Pe), \quad \text{where } \xi_0(\alpha) = \coth\alpha - \frac{1}{\alpha},\ Pe = \frac{|b|\,h}{2\varepsilon}, \tag{2.18}$$

where h is the element length and Pe is the local Péclet number defined in (2.17) in one dimension.

2.2.4 Choice of τ_h parameter in more dimensions

To find a suitable generalization of (2.18) for multidimensional cases is not as easy as one might expect. Heuristic generalization suggests:

$$\tau|_T \equiv \tau_T = \frac{h_T}{2\|\mathbf{b}\|_T}\xi(Pe_T), \quad \text{where } Pe_T = \frac{\|\mathbf{b}\|_T h_T}{2\varepsilon}, \tag{2.19}$$

where T is an element of $\mathcal{T}_h$, h_T is a characteristic length of T, $\|\mathbf{b}\|_T$ is a suitable norm of $\mathbf{b}$, ξ is an upwind function, such that $\xi(\alpha)/\alpha$ is bounded for $\alpha \to 0+$.

How to choose the optimal norm $\|\cdot\|_T$, h_T and ξ is not known. The correspondence between one-dimensional and d-dimensional case is not at all straightforward and universally optimal choice of these parameters does not exist, for further discussions about the choice of the parameter in multidimensional cases see John and Knobloch [2007]. Knobloch [2009] also discussed the choice of the SUPG parameter at outflow boundary layers. In the following we will choose h_T as the element diameter in the direction of the convection vector $\mathbf{b}$ and $\|\mathbf{b}\|_T$ is the Euclidean norm of vector $\mathbf{b}$.

As can be seen from (2.5), adding the stabilization term to the equation has an effect of adding a diffusion in the streamline direction, see the second term on the right-hand side of (2.4). However, Brooks and Hughes [1982] showed on one-dimensional example, that the standard Galerkin discretization is in general underdiffuse. So it is not entirely correct to speak about just adding a diffusion, but rather about creating a better finite element scheme, particularly more accurate. Recalling that the residual for the exact solution is zero, we immediately obtain its consistency. Moreover, according to Roos et al. [2008], this method has good stability properties and an "extra accuracy" of half of the power of h in the streamline direction in comparison with the Galerkin method.

2.3 Galerkin Least Squares FEM

In the SUPG method (2.5) the standard Galerkin method is augmented by the addition of terms that are proportional to the residual of the original differential equation. Since the residual of the exact solution is zero, the method is automatically consistent, unlike some other upwind methods.

We will now look at this consistency-preserving feature in a more general framework of Galerkin least squares method (GLSFEM). The classical Galerkin method has the projection property and needs elements from $C^0(\Omega)$, whereas GLSFEM method can be applied to a larger class of problems.

Let us show the application of the classical least squares method to solve the problem

$$\mathcal{L}u = f \text{ in } \quad \Omega, \quad u|_{\partial\Omega} = 0, \tag{2.20}$$

where $\mathcal{L}$ is defined as

$$\mathcal{L} = -\varepsilon\Delta + \mathbf{b}\cdot\nabla + c, \tag{2.21}$$

and $f, \mathbf{b}$ and c are considered to be from the same spaces as in Section 1.2. Let us choose a space $W_h \subset H^2(\Omega)$, where $v_h|_{\partial\Omega} = 0$. Then we are seeking a solution $u_h \in W_h$ of the minimization problem

$$\|\mathcal{L}u_h - f\|^2_{0,\Omega} = \min_{v_h \in V_h} \|\mathcal{L}u_h - f\|^2_{0,\Omega}\,. \tag{2.22}$$

It can be easily shown that the problem (2.22) is equivalent to the problem of finding $u_h \in W_h$ such that for all $v_h \in W_h$ one has

$$(\mathcal{L}u_h - f, \mathcal{L}v_h) = 0 \ . \tag{2.23}$$

In comparison with the standard Galerkin finite element method, the assumption that $W_h \subset H^2(\Omega)$ requires the use of C^1 elements, which is quite important constraint. Other problem is, that the condition number of the matrix associated with the discrete problem (2.23) is larger than the condition number encountered in the standard Galerkin approach.

Among the benefits of the least squares method is that it does not have bad stability properties like the standard Galerkin finite element method in the singularly perturbed case and that the matrix associated with (2.23) is symmetric and positive definite.

Galerkin least squares finite element method (GLSFEM) aims to combine the best features of the Galerkin and least squares methods. Let us introduce a weighted residual

$$\sum_{T \in \mathcal{T}_h} \delta_T (\mathcal{L}u_h - f, \mathcal{L}v_h)_T \tag{2.24}$$

of the equation (2.20). As this is evaluated on a per-element basis it allows use of C^0 elements from any space $V_h \subset H_0^1(\Omega)$. The basic idea is to add the term (2.24) to Galerkin formulation, which leads to the discrete formulation

$$a(u_h, v_h) + \sum_{T \in \mathcal{T}_h} \delta_T (\mathcal{L}u_h, \mathcal{L}v_h)_T = (f, v_h) + \sum_{T \in \mathcal{T}_h} \delta_T (f, \mathcal{L}v_h)_T \ , \tag{2.25}$$

where a was defined in (1.4). Initial proposal of this GLSFEM method and further discussion about the value of δ_T is in Hughes et al. [1989]. The procedure used to get the value of δ_T is analogous to the one used to derive the parameter τ for SUPG method, the resulting definition is similar, too.

More generally, we can consider, instead of (2.24), the term

$$\sum_{T \in \mathcal{T}_h} \delta_T (\mathcal{L}u_h - f, \psi(v_h))_T \ , \tag{2.26}$$

where ψ is some user-chosen operator. We will call then this method *Generalized GLSFEM* as by choosing $\psi(v_h) = \mathcal{L}v_h$ we recover the GLSFEM method. By choosing $\psi(v_h) = \mathbf{b} \cdot \nabla v_h$ we get SUPG. There are other possible choices, e.g., $\psi(v_h) = \mathbf{b} \cdot \nabla v_h + \varepsilon \Delta v_h$ for problems with $c = 0$, see Franca et al. [1992].

2.4 SOLD methods

Since SUPG method (2.5) is not monotone, discrete SUPG solution generally still contains spurious oscillations localized in narrow regions along sharp layers. A possible remedy is to add an artificial diffusion term to the SUPG method to obtain the monotonicity at least in some model cases. According to Godunov [1959] linear monotone methods can be at most first order accurate, this means it is natural to look for terms which depend on the discrete solution in a nonlinear way. Such approach is often used in methods called in a general way as spurious oscillations at layers diminishing (SOLD) methods. Sometimes other names and

more detailed names for these methods are used, such as discontinuity capturing methods or shock capturing methods, many of them are compared in John and Knobloch [2007].

The artificial diffusion added by SOLD methods almost always depends on an unknown discrete solution of a given problem. This fact implies that these methods are nonlinear in nature. Recently, these methods were subject of a great interest, see John and Knobloch [2008]. It turned out that oscillation-free discrete solutions with sufficiently sharp layers can be generally obtained only using SOLD methods with free parameters, which can be further optimized. However, in general, it is not known how these parameters should be defined and according to which criterion should they be optimized.

A general parameter-dependent SOLD method for problem (1.1) based on the SUPG method can be written in the variational form: Find $u_h \in W_h$ such that $u_h - \tilde{u}_{bh} \in V_h$ and

$$a(u_h, v_h) + (R_h(u_h), \tau_h \mathbf{b} \cdot \nabla v_h) + z_h(y_h; u_h, v_h) = (f, v_h) + \langle g, v_h \rangle_{\Gamma^N} \quad \forall v_h \in V_h, \tag{2.27}$$

where the new stabilization term $z_h(y_h; u_h, v_h)$ depends on u_h and the new parameter y_h in a nonlinear way, in general, the dependence on v_h is linear. Usually, we choose y_h from the same space as the SUPG parameter τ_h, so $y_h \in Y_h$.

2.4.1 SOLD terms adding isotropic artificial diffusion

Hughes et al. [1986] came with the idea to change the upwind direction in the SUPG term of (2.5) by adding a multiple of the function

$$\mathbf{b}_h^{\parallel} = \begin{cases} \frac{(\mathbf{b} \cdot \nabla u_h) \nabla u_h}{|\nabla u_h|^2} & \text{if} \quad \nabla u_h \neq 0, \\ \mathbf{0} & \text{if} \quad \nabla u_h = 0, \end{cases} \tag{2.28}$$

which corresponds to the direction in which oscillations in SUPG solutions are observed. This leads to the term z_h from (2.27) in the form

$$z_h(\sigma_h; u_h, v_h) := (R_h(u_h), \sigma_h \, \mathbf{b}_h^{\parallel} \cdot \nabla v_h) \tag{2.29}$$

where R_h was defined in (2.4). Here σ_h is a non negative stabilization parameter. This term controls the derivatives in the direction of the solution gradient and so increases the robustness of the SUPG method in the vicinity of sharp layers. Since $\mathbf{b}_h^{\parallel}$ depends on the unknown discrete solution u_h the resulting method is nonlinear.

But how to choose σ_h? Various SOLD methods were proposed but there is just a small amount of theoretical research on them. Often the definition σ_h is related to the choice of τ in the SUPG stabilization. One could think of using just the value $\tau(\mathbf{b}_h^{\parallel})$ but this would necessarily lead to a doubling of the SUPG stabilization if $\mathbf{b}_h^{\parallel} = \mathbf{b}$. Therefore, Hughes et al. [1986] proposed to set

$$\sigma_h = \max\{0, \tau(\mathbf{b}_h^{\parallel}) - \tau(\mathbf{b})\}. \tag{2.30}$$

As (2.30) is just an ad hoc correction, Tezduyar and Park [1986] proposed to redefine $\tau(\mathbf{b}_h^{\parallel})$:

$$\sigma_h = \frac{h_T^{\parallel}}{2|\mathbf{b}_h^{\parallel}|} \eta \left(\frac{|\mathbf{b}_h^{\parallel}|}{|\mathbf{b}|} \right), \tag{2.31}$$

where

$$h_k^{\|} = \mathrm{diam}(T, \mathbf{b}_h^{\|}), \quad \eta(\alpha) = 2\alpha(1-\alpha), \tag{2.32}$$

where $\mathrm{diam}(T, \mathbf{b}_h^{\|})$ is the length of the element T in the direction given by vector $\mathbf{b}_h^{\|}$. This definition assures that the SUPG effect is not doubled if $\mathbf{b}_h^{\|} = \mathbf{b}$ and hence an ad hoc correction like (2.30) is not needed.

2.4.2 SOLD terms adding crosswind artificial diffusion

Since the SUPG method adds the diffusion in the streamline direction there is natural need to consider a method which adds diffusion only in the crosswind direction. A typical example of a term added to the SUPG method (2.5) is $(\tilde{\varepsilon} P\nabla u_h, P\nabla v_h)$ with the orthogonal projection P onto the plane orthogonal to $\mathbf{b}$. In that case of two dimensions it means the projection P can be defined as

$$(\tilde{\varepsilon} P\nabla u_h, P\nabla v_h) = (\tilde{\varepsilon}\mathbf{b}^{\perp}\cdot\nabla u_h, \mathbf{b}^{\perp}\cdot\nabla v_h), \text{ where } \mathbf{b}^{\perp} = \frac{(-b_2, b_1)}{|\mathbf{b}|}\ . \tag{2.33}$$

From the definition (2.33) of the projection P we can see the term z_h is this case

$$z_h(\tilde{\varepsilon}; u_h, v_h) := (\tilde{\varepsilon} P\nabla u_h, P\nabla v_h)\ . \tag{2.34}$$

The parameter $\tilde{\varepsilon}$ usually depends on the unknown solution u_h and hence the resulting formulation is nonlinear. In John and Knobloch [2007] and John and Knobloch [2008] many proposals for the value of parameter $\tilde{\varepsilon}$ can be found. One of the most successful formulas according to John and Knobloch [2013] is

$$\tilde{\varepsilon}|_T = \eta\frac{\mathrm{diam}(T)|R_h(u_h)|}{2|\nabla u_h|} \quad \forall T \in \mathcal{T}_h, \tag{2.35}$$

where η is a free parameter, which we will not consider in the framework of parameter optimization later. The value we choose as a default one is $\eta = 0.7$ for P_1 finite element space. See the details of the choice of the value of η in John and Knobloch [2008].

2.4.3 Edge stabilization methods

This stabilization strategy for simplicial finite elements was introduced by Burman and Hansbo [2004]. The SOLD term $z_h(y_h; u_h, v_h)$ is added to the left-hand side of (2.27). In the 2D case it will be:

$$z_h(y_h; u_h, v_h) = \sum_{T\in\mathcal{T}_h}\int_{\partial T} y_h \mathrm{sign}(\mathbf{t}_{\partial T}\cdot\nabla(u_h|_T))\mathbf{t}_{\partial T}\cdot\nabla(v_h|_T)\mathrm{d}\sigma, \tag{2.36}$$

where $\mathbf{t}_{\partial T}$ is a tangent vector to the boundary ∂T of T, and y_h can be defined as

$$y_h(u_h) = \mathrm{diam}(T)(C_1\varepsilon + C_2\mathrm{diam}(T))\max_{E\subset\partial T}|[|\mathbf{n}_E\cdot\nabla u_h|]_E|, \tag{2.37}$$

where $\mathbf{n}_E$ are normal vectors to edges E of T, $[|v|]_E$ denotes the jump of a function v across the edge E and C_1, C_2 are appropriate constants (C_2 is proportional to $|\mathbf{b}|$). One can see that this method adds diffusion in the direction of the edges. Amount of the diffusion added is directly proportional to the maximum jump of the gradient of the discrete solution across all edges of the respective element K. See Burman and Hansbo [2004] for more details.

2.4.4 Iteration

Since the SOLD methods are nonlinear, their solutions have to be computed iteratively. The iterations are stopped if the residue is sufficiently small. The simplest and often most efficient way is to apply fixed-point iterations in which the SOLD parameters are computed using the previous iterate. The initial approximation is computed using the SUPG method. Thus, the iteration procedure is as follows:

1. Use a previous solution u_h to calculate parameters of the chosen SOLD method. For the case of SOLD terms adding isotropic artificial diffusion it means to first calculate $\mathbf{b}_h^{\parallel}$ and then σ. In case of SOLD terms adding crosswind artificial diffusion the calculation of $\mathbf{b}^{\perp}$ is trivial (we use constant data in our test scenarios) so only the calculation $\tilde{\varepsilon}$ is carried out. We will not consider other free parameters in the numerical results section 6.

2. Compute residue $\overline{R_h}$ of the SOLD method we are working with. As we already denoted the term specific for each of SOLD method as z_h we can define the $\overline{R_h}$ by just one general relation

$$\begin{aligned}\overline{R_h}(\tau_h, y_h; u_h, v_h) = a(u_h, v_h) + (R_h(u_h), \tau_h \mathbf{b} \cdot \nabla v_h) \\ + z_h(y_h; u_h, v_h) - (f, v_h) - \langle g, v_h \rangle_{\Gamma^N}\end{aligned} \tag{2.38}$$

3. Check the value of the norm of $\overline{R_h}$ (take into account that $\overline{R_h}$ is in the FE space) and end with the current solution u_h and the current SOLD parameter y_h if the value of $\overline{R_h}$ is sufficiently small. Particularly, we consider the decrease of the L^2 norm of $\overline{R_h}$ each iteration. If the decrease of the value of $\overline{R_h}/(\overline{R_h})_{old}$ is lower than 0.001 we **stop** the iteration.

4. Compute the solution u_h with newest SOLD parameter y_h.

Note that as we use the initial value of a SOLD parameter y_h only as an upper bound for optimization procedures we do not need to compute it to any higher degree of precision. More details and results are provided in the Numerical results chapter.

3. Optimization of parameters

To improve the solutions obtained with the methods containing some free parameters in an automatic way we need an indicator which can tell us how good is the discrete solution computed with the given set of free parameters. In our work we focus on optimization of not only one the stabilization parameter τ_h from the SUPG method (2.5), but also on optimization of a parameter in different SOLD methods introduced in the previous chapter. The optimization in our case is formulated as minimization of a functional representing an error indicator. Concrete formulations will appear later in this chapter.

Suppose now that we have already some error indicator. Usually we have the free parameter defined on each triangle in our triangulation separately. So in fact we have to minimize function of very high number of variables. We will use nonlinear minimization methods which are based on computing gradients. To compute the gradient of a function of many variables can be the most important component in an algorithm from the point of view of time efficiency. That is why we start this chapter with a description of adjoint approach and duality formulation.

3.1 Adjoint approach for computing derivatives

3.1.1 Adjoint approach in general

Adjoint approach to design is very widely used in practice. In this subsection we show why this approach can be very effective in some cases.

Suppose we wish to evaluate the quantity $g^T u$, where g is a given vector and u is a solution of the linear system of algebraic equations

$$Au = f. \tag{3.1}$$

In addition suppose we want to evaluate $g^T u$ for many different right-hand sides f. Let us consider solving the following system of linear algebraic equations

$$A^T v = g. \tag{3.2}$$

Since

$$v^T f = v^T A u = (A^T v)^T u = g^T u \tag{3.3}$$

we can obtain the desired quantity $g^T u$ simply by

$$g^T u = v^T f. \tag{3.4}$$

Thus, instead of solving the linear system many times, it suffices to solve (3.2) only once. The corresponding variables in equations (3.1) and (3.2) are called adjoint variables and the problem (3.2) is called dual or adjoint to problem (3.1). In the next sections we will further develop the framework based on these relations.

3.1.2 General duality formulation

Our aim is to minimize a scalar objective function $J(\alpha) = I(U(\alpha), \alpha)$. For a nonlinear minimization method we need to compute the derivative $\frac{\mathrm{d}J}{\mathrm{d}\alpha}$. The minimization problem is subject to a constraint which U and α have to satisfy. This fact can be expressed in the most general implicit way as:

$$N(U, \alpha) \equiv 0. \tag{3.5}$$

Equation (3.5) can be interpreted as for any given α, U has to satisfy equation (3.5). For small changes in α we get

$$\frac{\mathrm{d}J}{\mathrm{d}\alpha} = \frac{\partial I}{\partial U}\frac{\mathrm{d}U}{\mathrm{d}\alpha} + \frac{\partial I}{\partial \alpha}. \tag{3.6}$$

We must not forget that $\frac{\mathrm{d}U}{\mathrm{d}\alpha}$ is subject to the constraint (3.5). We can compute the derivative of $N(U(\alpha), \alpha)$, which is identically zero, to obtain

$$\frac{\partial N}{\partial U}\frac{\mathrm{d}U}{\mathrm{d}\alpha} + \frac{\partial N}{\partial \alpha} = 0. \tag{3.7}$$

Let us see the correspondence between the current quantities and the framework with quantities u, A, g^T, and f introduced in Section 3.1.1:

$$A \sim \frac{\partial N}{\partial U},\ u \sim \frac{\mathrm{d}U}{\mathrm{d}\alpha},\ f \sim -\frac{\partial N}{\partial \alpha},\ g^T \sim \frac{\partial I}{\partial U}. \tag{3.8}$$

Now we can use what we have learned in Subsection 3.1.1 to cope with this constrained problem. We will view problem (3.6), (3.7) with (3.8) in mind, to get

$$g^T u \ \sim\ \frac{\partial I}{\partial U}\frac{\mathrm{d}U}{\mathrm{d}\alpha} \tag{3.9}$$

where for the constraint we see

$$Au = f \quad \sim \quad \frac{\partial N}{\partial U}\frac{\mathrm{d}U}{\mathrm{d}\alpha} = -\frac{\partial N}{\partial \alpha}. \tag{3.10}$$

If we had only a single variable α, it would not be sensible to use the adjoint approach. But often we have much more variables (let us identify them as $\alpha_i, i = 1, \ldots, d$), so using adjoint approach will decrease dramatically computational cost. To be more specific, we first solve

$$A^T v = g \quad \sim \quad \left(\frac{\partial N}{\partial U_i}\right)_j v_j = \frac{\partial I}{\partial U_i} \tag{3.11}$$

for v and then for every $i = 1, \ldots, d$ we can simply evaluate $\frac{\mathrm{d}J}{\mathrm{d}\alpha_i}$ by

$$\frac{\mathrm{d}J}{\mathrm{d}\alpha_i} = -v_j \left(\frac{\partial N}{\partial \alpha_i}\right)_j + \frac{\partial I}{\partial \alpha_i}\ . \tag{3.12}$$

At each step we have to solve only one linear system of equations (3.11) and not d linear systems of equations (3.10) which we would have to solve if we used a direct approach instead of the adjoint approach. The additional inner product $v^T f_i$ is much less resource-consuming to compute than solving a system of linear equations.

3.1.3 Duality formulation in FEM adaptive methods

The text in this subsection follows on Section 2.2 and the notation also follows that. We will optimize the parameters τ_h, σ_h and $\tilde{\varepsilon}_h$ from both the SUPG method and SOLD method from Chapter 1.

All free parameters from SUPG and SOLD methods (2.5) and (2.27) will be denoted not as τ_h, σ_h and $\tilde{\varepsilon}_h$, but rather as y_h, y_h^σ, and y_h^ε, respectively. This comes from the fact, that we do not use the original definitions of τ from (2.18), the same holds for other free parameters. Instead we have these ones as free parameters for minimizing the error indicator. For a non-specified SOLD method we will use the notation y_h^{SOLD} instead of particular notations y_h^σ and y_h^ε.

Let $D_h \subset Y_h$ be an open set such that, for any $y_h \in D_h$ ($y_h^{SOLD} \in D_h$), the SUPG (SOLD, respectively) method has a unique solution $u_h \in W_h$. To emphasize various dependencies we shall use parenthesis. If $I_h : W_h \to \mathbb{R}$ is an error indicator, we define

$$\Phi_h(y_h, y_h^{SOLD}) = I_h(u_h(y_h, y_h^{SOLD}))\,, \tag{3.13}$$

where always the dependency of u_h on y_h^{SOLD} naturally holds for SOLD methods only. Given parameters y_h and y_h^{SOLD}, the function Φ_h represents the error of the discrete solution $u_h(y_h, y_h^{SOLD})$. It is worth noting that Φ_h depends on y_h or y_h^{SOLD} only through u_h. This dependency will be important further when taking the derivative of Φ_h.

So, our objective is to minimize Φ_h with respect to y_h or y_h^{SOLD}. For this optimization we need to be able to effectively compute the Fréchet derivative $\mathrm{D}\Phi_h$.

Computing derivatives in an efficient way

We do not have to consider the space W_h but can work only with the space V_h, because we have $u_h(y_h^{SOLD}) = \tilde{u}_h(y_h^{SOLD}) + \tilde{u}_{bh}$ with $\tilde{u}_h : D_h \times D_h \to V_h$. Similarly, we define $\tilde{I}_h(w_h) = I_h(w_h + \tilde{u}_{bh})$. Now we can rewrite (3.13) in the following manner:

$$\Phi_h(y_h, y_h^{SOLD}) = \tilde{I}_h(\tilde{u}_h(y_h, y_h^{SOLD})). \tag{3.14}$$

Furthermore, we will alter the basic residue defined in (2.4) and define the residual operator $R_h : V_h \times Y_h \times Y_h \to V_h'$ which is for a SOLD method from Subsection 2.4.2 defined by

$$\begin{aligned}\langle R_h(w_h, y_h, y_h^\varepsilon), v_h\rangle = {} & a(w_h + \tilde{u}_{bh}, v_h) - (f, v_h) - \langle g, v_h\rangle_{\Gamma^N} \\ & + ((-\varepsilon\Delta_h + \boldsymbol{b}\cdot\nabla + c)(w_h + \tilde{u}_{bh}), y_h\boldsymbol{b}\cdot\nabla v_h) \\ & - (f, y_h\boldsymbol{b}\cdot\nabla v_h) \\ & + (y_h^\varepsilon P\nabla u_h, P\nabla v_h),\end{aligned} \tag{3.15}$$

where the term $(y_h^\varepsilon P\nabla u_h, P\nabla v_h)$ is not present in case of SUPG method.

Because $\tilde{u}_h(y_h, y_h^{SOLD}) + \tilde{u}_{bh}$ is the exact solution for any y_h^{SOLD}, we have

$$R_h(\tilde{u}_h(y_h, y_h^{SOLD}), y_h, y_h^{SOLD}) \equiv 0. \tag{3.16}$$

For the following we will assume, that the mappings $R_h = R_h(y_h, y_h^{SOLD}, w_h)$, $\tilde{I}_h = \tilde{I}_h(w_h)$ and $\tilde{u}_h = \tilde{u}_h(y_h, y_h^{SOLD})$ are Fréchet-differentiable. We denote these derivatives as $\partial_w R_h, \partial_y R_h, \mathrm{D}\tilde{I}_h$ and $\mathrm{D}\tilde{u}_h$.

For the sake of simplicity we will assume only dependency on y_h in the rest of this chapter. We can express the Fréchet derivative of Φ_h explicitly as

$$\mathrm{D}\Phi_h(y_h) = \mathrm{D}\tilde{I}_h(\tilde{u}_h(y_h))\mathrm{D}\tilde{u}_h(y_h). \tag{3.17}$$

However, the computation of $\mathrm{D}\tilde{u}_h(y_h)$ requires the solution of dim Y_h systems of dim V_h linear equations (for every component of $(y_h)_i$ we have to compute solution $\tilde{u}_h((y_h)_i)$). If we take into account that in our case the dimension of Y_h is comparable with the dimension of V_h, we easily conclude that solving so many systems would be unacceptable even for smaller meshes.

We can circumvent this difficulty by using the adjoint approach as described in Subsection 3.1.2. It is not difficult to see the correspondence with a general adjoint approach. Instead of design variables denoted as α we now have y_h, $J(\alpha)$ corresponds to $\Phi_h(y_h)$, $U(\alpha)$ corresponds to $\tilde{u}_h(y_h)$, $\frac{\mathrm{d}J}{\mathrm{d}U}$ corresponds to $\mathrm{D}\tilde{I}_h$ and the constraint $N(U, \alpha)$ corresponds to residue $R_h(u_h, y_h)$. The last correspondence comes from the fact that the residue of the exact solution is identically equal to zero.

We will now use exactly the same procedure as in Subsection 3.1.2. We define an auxiliary mapping $\psi_h : D_h \to V_h$ which solves the adjoint problem

$$(\partial_w R_h)'(\tilde{u}_h(y_h), y_h)\psi_h(y_h) = \mathrm{D}\tilde{I}_h(\tilde{u}_h(y_h)), \tag{3.18}$$

where $(\partial_w R_h)'(\tilde{u}_h(y_h), y_h)$ is defined as the transposed operator and the behaviour of this operator is represented by

$$\langle(\partial_w R_h)'(w_h, y_h)v_h, \tilde{v}_h\rangle = \langle(\partial_w R_h)(w_h, y_h)\tilde{v}_h, v_h\rangle \quad \forall v_h, \tilde{v}_h \in V_h. \tag{3.19}$$

Analogously, we denote by $(\partial_y R_h)'(w_h, y_h)$ the operator acting as

$$\langle(\partial_y R_h)'(w_h, y_h)v_h, \tilde{y}_h\rangle = \langle(\partial_y R_h)(w_h, y_h)\tilde{y}_h, v_h\rangle \, \forall v_h \in V_h, \forall \tilde{y}_h \in Y_h. \tag{3.20}$$

Similarly as in (3.7) with N we can differentiate R_h around $(\tilde{u}_h(y_h), y_h)$:

$$0 = \mathrm{D}R_h = (\partial_w R_h)(\tilde{u}_h(y_h), y_h)\mathrm{D}\tilde{u}_h(y_h) + (\partial_y R_h)(\tilde{u}_h(y_h), y_h). \tag{3.21}$$

We can further rewrite (3.21) to obtain

$$\begin{aligned} (\partial_w R_h)(\tilde{u}_h(y_h), y_h)\mathrm{D}\tilde{u}_h(y_h) &= -(\partial_y R_h)(\tilde{u}_h(y_h), y_h) \\ \langle(\partial_w R_h)(\tilde{u}_h(y_h), y_h)\mathrm{D}\tilde{u}_h(y_h)\tilde{y}_h, \tilde{v}_h\rangle &= -\langle(\partial_y R_h)(\tilde{u}_h(y_h), y_h)\tilde{y}_h, \tilde{v}_h\rangle \\ &\qquad \forall \tilde{v}_h \in V_h, \forall \tilde{y}_h \in Y_h. \end{aligned} \tag{3.22}$$

Because $\mathrm{D}\tilde{u}_h(y_h)\tilde{y}_h \in V_h$, we can denote it v_h for now. Then we can transpose both sides of (3.22) to obtain step-wise

$$\langle(\partial_w R_h)(\tilde{u}_h(y_h), y_h)v_h, \tilde{v}_h\rangle = -\langle(\partial_y R_h)(\tilde{u}_h(y_h), y_h)\tilde{y}_h, \tilde{v}_h\rangle, \tag{3.23}$$

$$\langle(\partial_w R_h)'(\tilde{u}_h(y_h), y_h)\tilde{v}_h, v_h\rangle = -\langle(\partial_y R_h)'(\tilde{u}_h(y_h), y_h)\tilde{v}_h, \tilde{y}_h\rangle. \tag{3.24}$$

Changing now the test function $\tilde{v}_h$ with the solution ψ_h of adjoint problem (3.18), we immediately obtain from (3.18)

$$\begin{aligned} \langle (\partial_w R_h)'(\tilde{u}_h(y_h), y_h)\psi_h, v_h\rangle &= -\langle (\partial_y R_h)'(\tilde{u}_h(y_h), y_h)\psi_h, \tilde{y}_h\rangle \\ \langle \mathrm{D}I_h(\tilde{u}_h(y_h)), \mathrm{D}\tilde{u}_h(y_h)\tilde{y}_h\rangle &= -\langle (\partial_y R_h)'(\tilde{u}_h(y_h), y_h)\psi_h, \tilde{y}_h\rangle \\ \mathrm{D}I_h(\tilde{u}_h(y_h))\mathrm{D}\tilde{u}_h(y_h) &= -(\partial_y R_h)'(\tilde{u}_h(y_h), y_h)\psi_h \\ & \qquad \forall \tilde{y}_h \in Y_h. \end{aligned} \tag{3.25}$$

The whole process we have done in this subsection corresponds to the procedure described in Subsection 3.1.2. In the next chapter, the general relations derived above will be applied to the SUPG and SOLD methods formulated in Chapter 2.

4. Error estimators and indicators

Adaptive strategies are often used in practice. We can for example refine the grid in regions, where some local error estimator or error indicator is large, this we call h-adaptivity, see the original work of Babuška and Guo [1992]. Another possibility is p-adaptivity, which increases elements' polynomial degree, see Babuška et al. [1981]. There are also successful and well-known techniques which merge these two approaches together, for hp-FEM results see, e.g., Solin et al. [2003] Much more adaptive techniques are known.

Although we focus neither on refining grids nor on polynomial degree tuning, we will repeat probably all error indicators and estimators often used in other adaptive methods for the sake of completeness. We list them now briefly according to Roos et al. [2008]:

- estimators that are based on the solution of some local auxiliary problem for the discretization error.
- residual based error estimators or indicators
- estimators that use superconvergent approximations
- estimators that are based on an error representation formula involving the computed solution and the solution of an associated dual problem
- estimators that use complementary variational problems
- estimators based on hierarchical bases

In this work we use the second from the above list of possible error indicators and estimators, the residual based error indicators.

4.1 Residual based error indicator

We will follow the notation used throughout Chapter 3. In (3.13) we defined the function $\Phi_h(y_h) = I_h(u_h(y_h))$. If not explicitly defined otherwise, we will use the indicator $I_h^{L^2}$ in the form

$$I_h^{L^2}(w_h) = \sum_{T\in\mathcal{T}_h, \overline{T}\cap\overline{\Gamma^D}=\emptyset} h_T^2 \| -\varepsilon\Delta w_h + \mathbf{b}\cdot\nabla w_h + cw_h - f\|_{0,T}^2 \quad \forall w_h \in W_h. \quad (4.1)$$

Based on general definitions from Subsection 3.1.3 $\tilde{I}_h(w_h) = I_h(w_h + \tilde{u}_{bh})$, we can easily obtain the differential

$$\langle D\tilde{I}_h^{L^2}(\tilde{u}_h(y_h)), v_h\rangle = 2 \sum_{T\in\mathcal{T}_h, \overline{T}\cap\overline{\Gamma^D}=\emptyset} h_T^2(\mathcal{L}u_h(y_h) - f, \mathcal{L}v_h)_T \quad \forall v_h \in V_h, \quad (4.2)$$

where the linear operator $\mathcal{L}$ is defined in (2.21).

As we can see in (4.1) and (4.2), we exclude elements of $\mathcal{T}_h$ lying at the Dirichlet boundary, since there large errors of the discrete solution may occur

due to the approximation of a boundary layer on a coarse mesh. These errors cannot be significantly reduced by optimizing the stabilization parameter, but they dominate the errors on elements of $\mathcal{T}_h$ lying in the interior of Ω. This fact can deteriorate the overall minimization process.

4.2 Crosswind derivative control term

Oscillations that appear in a SUPG discrete solution can be characterized by large derivatives of a computed solution in crosswind direction. Possible remedy is to add a new term to the residual based error indicator (4.1) which should control overall crosswind derivative. For this reason, the following indicator was proposed in John et al. [2011]:

$$\begin{aligned} I_h^{cross}(w_h) = \sum_{T\in\mathcal{T}_h, \overline{T}\cap\overline{\Gamma^D}=\emptyset} \Big(& \| -\varepsilon\Delta w_h + \mathbf{b}\cdot\nabla w_h + c w_h - f\|_{0,T}^2 \\ & + \|\phi(|\mathbf{b}^\perp\cdot\nabla w_h|)\|_{0,1,T}\Big) \qquad \forall w_h \in W_h, \end{aligned} \tag{4.3}$$

where $\mathbf{b}^\perp$ is a unit vector in the crosswind direction defined by

$$\mathbf{b}^\perp(x) = \begin{cases} \frac{(b_2(x),-b_1(x))}{|\mathbf{b}(x)|} & \text{if } \mathbf{b}(x)\neq 0, \\ 0 & \text{if } \mathbf{b}(x) = 0, \end{cases} \quad \forall x\in\Omega, \tag{4.4}$$

and ϕ is a square root-like function defined by

$$\phi(t) = \begin{cases} \sqrt{t} & \text{if } t\geq 1, \\ 0.5(5t^2-3t^3) & \text{if } t<1. \end{cases} \tag{4.5}$$

We will call the term $\|\phi(|\mathbf{b}^\perp\cdot\nabla w_h|)\|_{0,1,T}$ in (4.3) crosswind derivative control term. The choice of the function $\phi(t)$ ensures that the function is Fréchet differentiable. We can show that $\phi(|t|)$ is differentiable by defining a new function $\psi(t) = \phi(|t|)$ so that

$$\int_T \phi(|\mathbf{b}^\perp\cdot\nabla w_h|)\,dx = \int_T \psi(\mathbf{b}^\perp\cdot\nabla w_h)\,dx\,. \tag{4.6}$$

We know already that for the original function $\phi(t)$ it holds that $\phi\in C^1(\mathbb{R})$, $\phi'(0)=0$, $\phi\in C^2((-\infty,1])$, $\phi\in C^2([1,\infty))$. By the definition of ψ we get

$$\psi(t) = \begin{cases} \phi(t) & t\geq 0 \\ \phi(-t) & t\leq 0 \end{cases}, \quad \psi'(t) = \left\{\begin{cases} \phi'(t) & t\geq 0 \\ -\phi'(-t) & t\leq 0 \end{cases}\right\} = \phi'(|t|)\,\mathrm{sgn}(t) \tag{4.7}$$

We immediately get $\psi\in C^1(\mathbb{R})$. From the second equation in (4.7) we infer

$$\psi''(t) = \big(\phi'(|t|)\,\mathrm{sgn}(t)\big)' = \phi''(|t|)\cdot\mathrm{sgn}(t)\cdot\mathrm{sgn}(t) \tag{4.8}$$

so $\psi\in C^2((-\infty,-1])$, $\psi\in C^2([-1,-1])$, and $\psi\in C^2([1,\infty))$.

Again, based on general definitions of the error indicators from Subsection 3.1.3, $\tilde{I}_h^{cross}(w_h) = I_h^{cross}(w_h + \tilde{u}_{bh})$, we can easily obtain the derivative of Indicator (4.3):

$$\langle D\tilde{I}_h^{cross}(\tilde{u}_h(\tau_h)), v_h\rangle = \sum_{T\in\mathcal{T}_h,\, \overline{T}\cap\overline{\Gamma^D}=\emptyset} \Big(2(\mathcal{L}u_h(\tau_h) - f, \mathcal{L}v_h)_T + \int_T \operatorname{sgn}(\mathbf{b}^\perp\cdot\nabla u_h(\tau_h))\phi'(|\mathbf{b}^\perp\cdot\nabla u_h(\tau_h)|)\,\mathbf{b}^\perp\cdot\nabla v_h\,dx\Big) \tag{4.9}$$

for all $v_h \in V_h$. In the above equation (4.9) we use the standard definition of $\mathcal{L}$ from (2.21).

4.3 Indicator with reduced residuals

Minimization of the error indicator I_h^{cross} leads to solutions with a small crosswind derivative. Thus, it is particularly suited for problems possessing solutions which, in regions away from layers, are nearly constant in the crosswind direction. In fact, this is the case for many of the test problems used for assessing numerical methods for convection–diffusion equations. For more general problems, however, the quality of the approximation may be then poor. Therefore, we introduce another indicator here. Let us start from the error indicator $I_h^{L^2}$. If the approximate solution possesses an interior layer, the largest residuals appear in the layer region and the minimization process tries to reduce them, causing a smearing of that layer. Thus, the idea is to reduce the influence of large residuals on the values of the error indicator. This can be achieved by setting

$$I_h^{lim}(w_h) = \sum_{T\in\mathcal{T}_h,\overline{T}\cap\overline{\Gamma^D}=\emptyset} \psi\Big(\| -\varepsilon\Delta w_h + \mathbf{b}\cdot\nabla w_h + cw_h - f\|_{0,T}^2\Big)\,, \tag{4.10}$$

where $\psi \in C^1(\mathbb{R}_0^+)$ is increasing and concave on $[0, t_0]$ and satisfies $\psi(t) = 1$ for $t > t_0$. For example, one may set

$$\psi(t) = \xi\left(\frac{t}{t_0}\right) \qquad \text{with} \qquad \xi(x) = \begin{cases} \frac{1}{2}\,x^4 - x^3 - \frac{1}{2}\,x^2 + 2\,x & \text{if } x \le 1,\\ 1 & \text{if } x > 1. \end{cases}$$

Of course, a crucial question is how to choose the value t_0. It is obvious that if t_0 is large, then I_h^{lim} will behave analogously as $I_h^{L^2}$ and hence will lead to a smearing of layers. On the other hand, if t_0 is very small, then ψ will remain equal 1 in large parts of Ω and the approximate solution after parameter optimization will be similar as the non-optimized one. In practice, the value of t_0 has to be chosen according to the magnitude of residuals in layer regions. We can obtain the derivative of Indicator (4.10), for all $v_h \in V_h$:

$$\langle D\tilde{I}_h^{lim}(\tilde{u}_h(\tau_h)), v_h\rangle = \sum_{T\in\mathcal{T}_h,\,\overline{T}\cap\overline{\Gamma^D}=\emptyset} \Big(\psi'\Big(\| -\varepsilon\Delta u_h(\tau_h) + \mathbf{b}\cdot\nabla u_h(\tau_h) + cu_h(\tau_h) - f\|_{0,T}^2\Big) \cdot 2\,(\mathcal{L}u_h(\tau_h) - f, \mathcal{L}v_h)_T\Big)\,. \tag{4.11}$$

4.4 Application to the SUPG method

Let us first define the linear operator $\mathcal{L}_h$ in a similar way as in (2.21)

$$\mathcal{L}_h = -\varepsilon\Delta_h + \mathbf{b}\cdot\nabla + c. \tag{4.12}$$

For the SUPG method (2.5) we obtain using definition (3.15) of the residual operator the following formula for the Fréchet derivatives $\partial_w R_h$ and $\partial_y R_h$:

$$\begin{aligned}\langle(\partial_w R_h)(w_h,y_h)\tilde{v}_h, v_h\rangle &= a(\tilde{v}_h,v_h) + (\mathcal{L}_h\tilde{v}_h, y_h\mathbf{b}\cdot\nabla v_h)\\ \langle(\partial_y R_h)(w_h,y_h)\tilde{y}_h, v_h\rangle &= (\mathcal{L}_h(w_h+\tilde{u}_{bh}), \tilde{y}_h\mathbf{b}\cdot\nabla v_h)\\ &\quad - (f, \tilde{y}_h\mathbf{b}\cdot\nabla v_h)\end{aligned} \tag{4.13}$$

for all $y_h, \tilde{y}_h \in Y_h$ and $v_h, \tilde{v}_h, w_h \in V_h$. According to (3.18) and (3.19), we obtain the relation for the auxiliary function $\psi_h(y_h) \in V_h$:

$$a(v_h,\psi_h(y_h)) + (\mathcal{L}_h v_h, y_h\mathbf{b}\cdot\nabla\psi_h(y_h)) = \langle \mathrm{D}\tilde{I}_h(\tilde{u}_h(y_h)), v_h\rangle \quad \forall v_h \in V_h. \tag{4.14}$$

For the Fréchet derivative of Φ_h we obtain using (3.25) the following:

$$\langle \mathrm{D}\Phi_h(y_h), \tilde{y}_h\rangle = (-\mathcal{L}_h u_h(y_h) + f, \tilde{y}_h\mathbf{b}\cdot\nabla\psi_h(y_h)) \quad \forall \tilde{y}_h \in Y_h. \tag{4.15}$$

4.5 Application to SOLD methods

We will use again the linear operator $\mathcal{L}_h$ from (4.12). One should take into account that for the sake of simplicity we use the same finite element spaces for all free parameters ($y_h, y_h^\sigma, \tilde{y}_h, \tilde{\tilde{y}}_h \in Y_h$). For SOLD method with the isotropic diffusion from (2.29) we obtain for the Fréchet derivatives $\partial_w R_h$ and $\partial_y R_h$ the following relations, assuming $\mathbf{b}_h^{\parallel}$ is taken as a constant:

$$\begin{aligned}\langle(\partial_w R_h)(w_h,y_h,y_h^\sigma)\tilde{v}_h, v_h\rangle &= a(\tilde{v}_h,v_h) + (\mathcal{L}_h\tilde{v}_h, y_h\mathbf{b}\cdot\nabla v_h) + (\mathcal{L}_h\tilde{v}_h, y_h^\sigma\mathbf{b}_h^{\parallel}\cdot\nabla v_h)\\ \langle(\partial_y R_h)(w_h,y_h,y_h^\sigma)\tilde{y}_h, v_h\rangle &= (\mathcal{L}_h(w_h+\tilde{u}_{bh}), \tilde{y}_h\mathbf{b}\cdot\nabla v_h)\\ &\quad - (f, \tilde{y}_h\mathbf{b}\cdot\nabla v_h)\\ \langle(\partial_{y_\sigma} R_h)(w_h,y_h,y_h^\sigma)\tilde{y}_h, v_h\rangle &= (\mathcal{L}_h(w_h+\tilde{u}_{bh}), \tilde{y}_h\mathbf{b}_h^{\parallel}\cdot\nabla v_h)\\ &\quad - (f, \tilde{y}_h\mathbf{b}_h^{\parallel}\cdot\nabla v_h)\end{aligned} \tag{4.16}$$

for all $y_h, y_h^\sigma, \tilde{y}_h \in Y_h$ and $v_h, \tilde{v}_h, w_h \in V_h$. According to (3.18) and (3.19), we obtain the relation for the auxiliary function $\psi_h(y_h, y_h^\sigma) \in V_h$:

$$\begin{aligned}a(v_h,\psi_h(y_h,y_h^\sigma)) + (\mathcal{L}_h v_h, y_h\mathbf{b}\cdot\nabla\psi_h(y_h,y_h^\sigma)) + (\mathcal{L}_h v_h, y_h^\sigma\mathbf{b}_h^{\parallel}\cdot\nabla\psi_h(y_h,y_h^\sigma))\\ = \langle \mathrm{D}\tilde{I}_h(\tilde{u}_h(y_h,y_h^\sigma)), v_h\rangle\end{aligned} \tag{4.17}$$

for all $v_h \in V_h$. For the Fréchet derivative of Φ_h we obtain using (3.25) the following:

$$\begin{aligned}\langle \mathrm{D}\Phi_h(y_h,y_h^\sigma), (\tilde{y}_h,\tilde{\tilde{y}}_h)\rangle &= (-\mathcal{L}_h u_h(y_h,y_h^\sigma) + f, \tilde{y}_h\mathbf{b}\cdot\nabla\psi_h(y_h,y_h^\sigma))\\ &\quad + (-\mathcal{L}_h u_h(y_h,y_h^\sigma) + f, \tilde{\tilde{y}}_h\mathbf{b}_h^{\parallel}\cdot\nabla\psi_h(y_h,y_h^\sigma))\end{aligned} \tag{4.18}$$

for all $\tilde{y}_h, \tilde{\tilde{y}}_h \in Y_h$. We can easily get similar results for SOLD method with the crosswind diffusion term from Subsection (2.4.2).

5. Numerical methods of minimizing error indicators

Error indicators introduced in Section 4 are in fact functions dependent on many variables. Therefore, we have to cope with a minimization problem in $\mathbb{R}^d$, where d is typically proportional to the number of finite elements. For example, if we use the piecewise constant discontinuous finite element space for the parameter y_h, then d is the number of elements of $\mathcal{T}_h$ (it can be a very large number).

As the number of variables is large, we solve problem, whose Hessian matrix cannot be computed at a reasonable cost. This fact affects the choice of methods we can use for the minimization of our error indicators. Particularly this means that we cannot use Newton's method, which needs a full Hessian. This computation would be too expensive.

In this section we introduce algorithms we used for minimizing error indicators from Section 4. We begin with the description of line search algorithms, which are used throughout many other algorithms. Then we turn our attention to four algorithms we used in our numerical tests. All used algorithms must not store the full Hessian. This fact prevents us from using other algorithms than those from the following sets of algorithms: steepest descent algorithms, conjugate gradient algorithms and quasi-Newton algorithms. Quasi-Newton algorithms then comprise particularly L-BFGS and L-SR1 algorithms.

The function which we are supposed to minimize will be denoted by f throughout this chapter. The gradient of f is defined as $\nabla f = (\frac{\partial f}{\partial x_1}, \cdots, \frac{\partial f}{\partial x_d})$, where d is the number of variables (dimension).

5.1 Line search algorithms

Each iteration of a line search method computes a *search direction* p_k and then decides how far to move along that direction. The iteration is given by

$$x_{k+1} = x_k + \alpha_k p_k, \tag{5.1}$$

where $\alpha_k > 0$ is called *step length*.

5.1.1 Search direction

Most line search algorithms require p_k to be a descent direction, which means $p_k^T \nabla f_k < 0$, where $\nabla f_k = \nabla f(x_k)$. This property guarantees that the function f can be reduced along p_k.

In most algorithms the search direction has the following form

$$p_k = -B_k^{-1} \nabla f_k, \tag{5.2}$$

where B_k is a symmetric nonsingular matrix. In the steepest descent method B_k is the identity matrix I, in Newton's method B_k is the exact Hessian $\nabla^2 f_k$ and in quasi-Newton methods B_k is an approximation to the Hessian. If B_k is in addition positive definite, we have

$$p_k^T \nabla f_k = -\nabla f_k^T B_k^{-1} \nabla f_k < 0,$$

which proves that p_k is a descent direction.

5.1.2 Step length

In the process of minimizing some function, choosing right step length is just as important as choosing right direction. To choose a quasi-optimal step length many different types of so called "line search algorithms" are used. The word "line" stands there because we search in only one direction p_k which is for any line search method given as a parameter. We suppose that p_k is a descent direction, so that $p_k^T \nabla f_k < 0$ holds. The desired step length $\alpha_k > 0$ would then be a minimizer of:

$$\phi(\alpha_k) = f(x_k + \alpha_k p_k). \tag{5.3}$$

The line search is often done in two stages: A bracketing phase finds an interval containing desirable step lengths and a bisection or interpolation phase computes a good step length within this interval.

A simplest possible condition we could impose on a step length α_k is to require a reduction of the function value for the given α_k:

$$f(x_k + \alpha_k p_k) < f(x_k). \tag{5.4}$$

See Algorithm 1 which uses the simple condition (5.4). In some special cases this simple method can behave better than some more sophisticated algorithms. Nevertheless, as shown by Nocedal and Wright [2006], this requirement is usually not enough to produce an effective convergence to a local minimum.

Algorithm 1 Simple Line Search

Choose a starting point x_0, initial step length α and search direction p
$l \leftarrow 0$
loop
  $x_1 \leftarrow x_0 + \alpha p$
  if $f(x_1) < f(x_0)$ **then**
    if $l = 1$ **then**
      $\alpha_* \leftarrow \alpha$ and **stop**
    end if
    $\alpha \leftarrow 2\alpha$
  else
    $\alpha \leftarrow \frac{\alpha}{2}$
    $l \leftarrow 1$
  end if
  if $|\alpha|$ is too small **then**
    $\alpha_* \leftarrow 0$ and **stop**
  end if
end loop
return optimal step length α_*

More effective algorithms than the simple line search (means more effective in most cases, not always) typically use more conditions imposed on the line search procedure. These conditions must be fulfilled to accept a step length α. If all

conditions are fulfilled, the algorithm stops with a desired step length, that is why these conditions are known as "termination conditions" for line search methods. Frequently used termination conditions for line search algorithms are the *strong Wolfe conditions*:

$$f(x_k + \alpha_k p_k) \leq f(x_k) + c_1 \alpha_k \nabla f_k^T p_k, \tag{5.5a}$$

$$|\nabla f(x_k + \alpha_k p_k)^T p_k| \leq c_2 |\nabla f_k^T p_k|, \tag{5.5b}$$

where the adjective "strong" comes from the presence of absolute values in the equation (5.5b). An enhanced method finding a step length satisfying the strong Wolfe conditions (5.5) is described in Algorithm 2.

Algorithm 2 Enhanced Line Search

Choose a starting point x_0, maximal step length $\alpha_{\max} > 0$, $\alpha_1 \in (0, \alpha_{\max})$ and search direction p. In addition we define ϕ as $\phi(\alpha) = f(x_0 + \alpha p)$ to simplify the notation.
$\alpha_0 \leftarrow 0$
loop
  if $\phi(\alpha_i) > \phi(0) + c_1 \alpha_i \phi'(0)$ or $[\phi(\alpha_i) \geq \phi(\alpha_{i-1})$ and $i \geq 1]$ **then**
    $\alpha_* \leftarrow$ **zoom**(α_{i-1}, α_i) and **stop**
  end if
  if $|\phi'(\alpha_i)| \leq -c_2 \phi'(0)$ **then**
    $\alpha_* \leftarrow \alpha_i$ and **stop**
  end if
  if $\phi'(\alpha_i) \geq 0$ **then**
    $\alpha_* \leftarrow$ **zoom**$(\alpha_i, \alpha_i - 1)$ and **stop**
  end if
  Choose $\alpha_{i+1} \in (\alpha_i, \alpha_{\max})$
  $i \leftarrow i + 1$
end loop
return optimal step length α_*

Enhanced line search (Algorithm 2) has two stages. This first stage begins with a trial estimate α_1 and keeps increasing it until it finds either an acceptable step length or an interval that brackets the desired step lengths. In the latter case, the second stage is invoked by calling a function called *Zoom*. This function is described in Algorithm 3. The Zoom function successively decreases the size of the interval until an acceptable step length is identified.

5.2 Trust region methods

Trust region algorithms also generate steps with the help of a quadratic model of the objective function in a neighborhood of a point, but they use this model in different ways than line search algorithms. Line search methods use it to generate a search direction and then focus their efforts on finding a suitable step length α along this direction. Trust region methods define a region around the current point, within which they trust the model to be an adequately precise representation of the objective function. Afterwards trust region algorithms choose the step to be the approximate minimizer of the model in this region.

Algorithm 3 Zoom(α_{lo}, α_{hi})

loop
  Interpolate (using quadratic, cubic, or bisection) to find a trial step length α_j between α_{lo} and α_{hi}. Again we define ϕ as $\phi(\alpha) = f(x_0 + \alpha p)$.
  if $\phi(\alpha_j) > \phi(0) + c_1\alpha_j\phi'(0)$ or $\phi(\alpha_j) \geq \phi(\alpha_{\text{lo}})$ **then**
    $\alpha_{\text{hi}} \leftarrow \alpha_j$
  else
    if $|\phi'(\alpha_j)| \leq -c_2\phi'(0)$ **then**
      $\alpha_* \leftarrow \alpha_j$ and **stop**
    end if
    if $\phi'(\alpha_j)(\alpha_{\text{hi}} - \alpha_{\text{lo}}) \geq 0$ **then**
      $\alpha_{\text{hi}} \leftarrow \alpha_{\text{lo}}$
    end if
    $\alpha_{\text{lo}} \leftarrow \alpha_j$
  end if
end loop
return optimal step length α_*

Assume the quadratic model function m_k of f near some point x_k is a Taylor series expansion of f near this point. Then we can write

$$m_k(p) = f_k + (\nabla f_k)p^T + \frac{1}{2}p^T B_k p, \tag{5.6}$$

where the index $\cdot_k$ means we evaluate a given quantity at the point x_k and B_k is some symmetric matrix which stands for the Hessian. So we seek for the solution of the following subproblem:

$$\min_{p \in R^n} m_k(p) = f_k + (\nabla f_k)p^T + \frac{1}{2}p^T B_k p \quad \text{subject to} \quad \|p\| \leq \Delta_k, \tag{5.7}$$

where $\Delta_k > 0$ is trust region radius.

When B_k is positive definite and $\|B_k^{-1}\nabla f_k\| \leq \Delta_k$, the solution of (5.7) is the unconstrained minimum $p_k = -B_k^{-1}\nabla f_k$. In this case, we call p_k the *full step*. The solution is not so obvious in other cases of B_k, but we only need an approximate solution to obtain a good convergence.

One of the key ingredients in a trust region algorithm is the strategy for choosing the trust region radius Δ_k at each iteration. This choice is usually based on the agreement between the quadratic model m_k and the objective function f at previous iterations. Given a step p_k we can define the ratio

$$\rho_k = \frac{f(x_k) - f(x_k + p_k)}{m_k(0) - m_k(p_k)} = \frac{\text{ared}}{\text{pred}}\,. \tag{5.8}$$

The numerator is called the *actual reduction* and the denominator is called *predicted reduction*. We will use this later in this chapter. The algorithm which use equation (5.8) for adjusting Δ_k is used later in the L-SR1 algorithm (Algorithm 10).

We now turn our attention to the solution of the minimization subproblem (5.7). We will describe the so-called Cauchy point algorithm. The "Cauchy point"

is simply the minimizer of m_k along the steepest descent direction $-\nabla f_k$. We will minimize $m_k(\tau p_k)$ as a function of τ, where we choose the size of the vector p_k to be Δ_k, so we can write

$$p_k = \frac{\Delta_k}{\|\nabla f_k\|} \nabla f_k. \tag{5.9}$$

To obtain τ_k in an explicit form we consider the cases of $\nabla f_k^T B_k \nabla f_k \leq 0$ and $\nabla f_k^T B_k \nabla f_k \geq 0$ separately. For the former case, the function $m_k(\tau p_k)$ decreases monotonically so τ_k is simply the largest value that satisfies the trust region bound, $\tau_k = 1$.

For the case $\nabla f_k^T B_k \nabla f_k \geq 0$, $m_k(\tau p_k)$ is a convex quadratic in τ, so τ_k is either the unconstrained minimizer of this quadratic or the boundary value 1, whichever comes first. For the constrained minimizer we compute τ_k as solution of $\frac{\partial m_k(\tau p_k)}{\partial \tau} = 0$. In summary we have

$$\tau_k = \begin{cases} 1 & \text{if } \nabla f_k^T B_k \nabla f_k \leq 0, \\ \min\left(\frac{\|\nabla f_k\|^3}{\Delta_k \nabla f_k^T B_k \nabla f_k}, 1\right) & \text{otherwise.} \end{cases} \tag{5.10}$$

The Cauchy point algorithm is inexpensive to calculate, because no matrix factorizations are required. However, a better convergence speed might be obtained if B_k is used not only to compute the step length but also to identify a better search direction p_k.

5.3 Steepest descent methods

The most simplest implementation of the steepest descent method is represented in Algorithm 4. In the steepest descent algorithm we use the line search method defined in Algorithm 1 or 2. The choice of the line search method does not have a fundamental effect on obtained results. In general an enhanced line search method should be faster but in some cases it returns a worse minimum than a simple line search method.

Algorithm 4 Steepest descent algorithm

Choose a starting point x_0 and initial step length α
$k \leftarrow 0$
$\mathrm{d}\phi_{old} \leftarrow -\nabla f^T(x_0)\nabla f(x_0)$
while value of $|\nabla f(x_k)|$ is sufficiently high **do**
  $p \leftarrow -\nabla f(x_k)$
  $\mathrm{d}\phi \leftarrow \nabla f^T(x_k)p$
  $\alpha \leftarrow \alpha \frac{\mathrm{d}\phi_{old}}{\mathrm{d}\phi}$
  Compute α using Algorithm 1 or Algorithm 2 with initial step length α
  $x_{k+1} \leftarrow x_k + \alpha p$
  $\mathrm{d}\phi_{old} \leftarrow \mathrm{d}\phi$
  $k \leftarrow k + 1$
end while

5.4 Nonlinear conjugate gradient methods

Fletcher and Reeves [1964] showed how to extend the classical linear conjugate gradient method, which is used to solve a linear system of equations $Ax = b$ (or equivalently minimizing $\phi(x) = \frac{1}{2}x^T Ax - b^T x$), where A is a positive semidefinite matrix, to nonlinear functions by making two simple changes.

First, in the formula for the step length α_k (which minimizes ϕ along the search direction p_k), we need to perform a line search that identifies an approximate minimum of the nonlinear function f along p_k. Second, the residual $Ax - b$, which corresponds to the gradient of ϕ in the linear conjugate gradient algorithm, must be replaced by the gradient of the nonlinear function ∇f. These changes give rise to the conjugate gradient Algorithm 5 for nonlinear optimization.

Algorithm 5 Nonlinear conjugate gradient algorithm

Choose a starting point x_0
$k \leftarrow 0$, $p_0 \leftarrow -\nabla f(x_0)$
while $|\nabla f(x_k)| > 0$ **do**
  Compute α_k using line search method
  $x_{k+1} \leftarrow x_k + \alpha_k p_k$

$$\beta_{k+1} \leftarrow \frac{\nabla f^T(x_{k+1})(\nabla f(x_{k+1}) - \nabla f(x_k))}{\|\nabla f(x_k)\|^2} \tag{5.11}$$

$$p_{k+1} \leftarrow -\nabla f(x_{k+1}) + \beta_{k+1} p_k \tag{5.12}$$

  $k \leftarrow k + 1$
end while

Algorithm 5 is appealing for large nonlinear optimization problems since each iteration requires only the evaluation of the objective function and its gradient. No matrix operations are required for the step computation and just a few vectors of storage are required.

To make the specification of Algorithm 5 complete, we need to be more precise about the choice of the line search parameter α_k. Because of the second term in (5.12), the search direction p_k may fail to be a descent direction unless α_k satisfies certain conditions. It can be simply shown (see Nocedal and Wright [2006]) that if α_k satisfies strong Wolfe conditions (5.5) with $0 < c_1 < c_2 < \frac{1}{2}$, then all directions p_k are descent directions.

It is possible to use another definition of (5.11). The definition we use is known as the *Polak–Ribiére* formula for β. Other possible formulas for β are the Fletcher–Reeves formula and the Hestenes–Stiefel formula. However, according to Nocedal and Wright [2006], the Polak–Ribiére formula performs slightly better than other formulas for β in most cases. According to our tests, the differences in performance among these formulas are negligible so we do not list tests for every single formula for β as a standalone test.

As explained in Nocedal and Wright [2006], restarting can be added to obtain a better performance when we have a modest dimension d (say $d \sim 50$). Then, we restart every d iterations our algorithm by setting $\beta_{k+1} = 0$ in (5.11). Since we have much larger dimension, no restarting strategy is applied.

5.5 Limited-memory quasi-Newton methods

Quasi-Newton methods are also known as variable metric methods. They are based on Newton's method to find the stationary point of a function. Newton's method assumes that the function can be locally approximated as a quadratic in the region around the optimum and uses gradient and Hessian (matrix of second derivatives) to find the stationary point.

We need to solve a problem whose Hessian matrix cannot be computed at a reasonable cost. In Newton's method, B_k is the exact Hessian $\nabla^2 f(x_k)$. In quasi-Newton methods, B_k is an approximation to the Hessian that is updated at every iteration by means of a low-rank formula. A possible simplistic interpretation is that they determine the descent direction by preconditioning the gradient with a curvature information.

The most common quasi-Newton algorithms are currently the SR1 formula (symmetric rank one), the widespread BFGS method (suggested in 1970 independently by Broyden, Fletcher, Goldfarb, and Shanno), then described in a more complete way in Fletcher [1987] together with some limited-memory extensions.

According to Nocedal and Wright [2006], limited-memory quasi-Newton methods are most effective to solve large problems. These methods maintain simple and compact approximations of Hessian matrices: Instead of storing fully dense $d \times d$ approximations, they save only a few vectors of length d that represent the approximations implicitly. Despite of low storage requirements, they often yield an acceptable (albeit linear) rate of convergence.

Various limited-memory methods have been proposed, but we focused on the algorithm known as the L-BFGS algorithm and our proposed L-SR1 algorithm. The "L" stands in the name, because of limited memory cost of the method. The main idea of both methods is to use curvature information from only the most recent iterations to construct the Hessian approximation. Curvature information from earlier iterations, which is less likely to be relevant to the behavior of the Hessian at the current iteration, is discarded in the interest of saving storage.

So instead of computing B_k at every iteration, we update it in a simple manner to account for the curvature measured during the most recent step. Suppose that we have somehow generated a new iterate x_{k+1} and wish to construct a new quadratic model of f of the form

$$m_{k+1}(p) = f_{k+1} + \nabla f_{k+1}^T p + \frac{1}{2} p^T B_{k+1} p. \tag{5.13}$$

We now want that m_{k+1} matches gradient of f at the latest two iterates x_k and x_{k+1}. Since $\nabla m_{k+1}(0)$ is precisely ∇f_{k+1}, the second of these conditions is satisfied automatically. The first condition can be written as

$$\nabla m_{k+1}(-\alpha_k p_k) = \nabla f_{k+1} - \alpha_k B_{k+1} p_k = \nabla f_k. \tag{5.14}$$

By rearranging we obtain

$$B_{k+1}\alpha_k p_k = \nabla f_{k+1} - \nabla f_k. \tag{5.15}$$

To simplify the notation we will use the following abbreviations

$$s_k = x_{k+1} - x_k = \alpha_k p_k, \quad y_k = \nabla f_{k+1} - \nabla f_k. \tag{5.16}$$

We can now rewrite (5.15) as

$$B_{k+1}s_k = y_k. \tag{5.17}$$

We will refer to equation (5.17) as the *secant equation.*

5.5.1 Limited-memory BFGS method

We begin our description of the L-BFGS method by a description of the BFGS method. Then we will explain how to avoid assembling of the full Hessian. Each step of the BFGS method has the following form

$$x_{k+1} = x_k - \alpha_k H_k \nabla f_k, \tag{5.18}$$

where α_k is the step length and the inverse Hessian approximation H_k is updated every step using the following formula

$$H_{k+1} = V_k^T H_k V_k + \rho_k s_k s_k^T, \tag{5.19}$$

where V_k, ρ_k, and s_k are defined by

$$s_k = x_{k+1} - x_k, \quad y_k = \nabla f_{k+1} - \nabla f_k, \quad \rho_k = \frac{1}{y_k^T s_k}, \quad V_k = I - \rho_k y_k s_k^T. \tag{5.20}$$

Since the inverse Hessian approximation H_k will be dense, the cost of storing and manipulating it is prohibitive when we have large dimensions. Optimizing parameters in FEM takes place in large dimensions, so we have to circumvent somehow this problem.

We will store a modified version of H_k only implicitly, by storing a certain number m of the vector pairs $\{s_i, y_i\}$ used in formula (5.19). The Hessian H_k is generally used only through the product $H_k \nabla f_k$. The product can be obtained by performing a sequence of inner products and vector summations involving f_k and the pairs $\{s_i, y_i\}$.

After the new iterate is computed, if there are already stored m pairs $\{s_i, y_i\}$, the oldest vector pair in the set of pairs is replaced by the new pair $\{s_k, y_k\}$ obtained from the current step. This means that the set of vector pairs includes curvature information only from the m most recent iterations. Practical experience has shown that lower values of m (lower than 100) produce good results. At this place it is necessary to remember that we in fact collect curvature information from more different points in the multidimensional space, so larger values of m really have no relevance to reproduce a good approximation of the Hessian at a given point.

Suppose that we did k iterations, so we have the vector pair $\{s_k, y_k\}$. Further suppose that we already have vector pairs $\{s_i, y_i\}$ for $i = k - m, \ldots, k - 1$.

For computing $H_k \nabla f_k$ we choose some initial inverse Hessian approximation H_k^0. Then we apply formula (5.19) repeatedly, so that the L-BFGS approximation of H_k satisfies the following formula, where we used the notation introduced in (5.20)

$$\begin{aligned} H_k = & (V_{k-1}^T \cdots V_{k-m}^T) H_k^0 (V_{k-m} \cdots V_{k-1}) \\ & + \rho_{k-m} (V_{k-1}^T \cdots V_{k-m+1}^T) s_{k-m} s_{k-m}^T (V_{k-m+1} \cdots V_{k-1}) \\ & + \rho_{k-m+1} (V_{k-1}^T \cdots V_{k-m+2}^T) s_{k-m+1} s_{k-m+1}^T (V_{k-m+2} \cdots V_{k-1}) \\ & + \cdots + \rho_{k-1} s_{k-1} s_{k-1}^T. \end{aligned} \tag{5.21}$$

From the formula (5.21) we can derive an algorithm for computing the product $H_k \nabla f_k$. According to Nocedal and Wright [2006], the most effective version of such an algorithm is that in Algorithm 6.

The Algorithm 6 needs only $4mn + n$ multiplications. And because the multiplication by H_k^0 is isolated from other multiplications, we can use a different H_k^0 in each iteration.

Algorithm 6 L-BFGS two-loop recursion algorithm to compute $H_k \nabla f_k$

$q \leftarrow \nabla f_k$
for $i = k-1, \cdots, k-m$ **do**
  $\alpha_i \leftarrow \rho_i s_i^T q$
  $q \leftarrow q - \alpha_i y_i$
end for
$r \leftarrow H_k^0 q$
for $i = k-m, \cdots, k-1$ **do**
  $\beta \leftarrow \rho_i y_i^T r$
  $r \leftarrow r + s_i(\alpha_i - \beta)$
end for
return r (in r is now the value of $H_k \nabla f_k$)

According to Nocedal and Wright [2006], we choose H_k^0 as the diagonal matrix $H_k^0 = \gamma_k I$, where

$$\gamma_k = \frac{s_{k-1}^T y_{k-1}}{y_{k-1}^T y_{k-1}}. \tag{5.22}$$

A full algorithm of the L-BFGS method is now quite straightforward to write. We can see it in Algorithm 7. In every iteration we use some line search method to obtain the step length α_k. We can choose from both line search methods introduced in Section 5.1, although only the enhanced line search method guarantees the linear rate of convergence, according to Nocedal and Wright [2006]. In our tests we use the simple line search method, which proved to be more robust in practice than the enhanced line search method, although, according to Nocedal and Wright [2006], the theoretical convergence results are only for the enhanced line search method (the enhanced line search method guarantees H_k is positive-definite).

Algorithm 7 L-BFGS algorithm

1: Choose a starting point x_0 , integer $m > 0$
2: $k \leftarrow 0$
3: **repeat**
4: Choose H_k^0
5: Compute $p_k \leftarrow -H_k \nabla f_k$ using two-loop recursion (Algorithm 6)
6: Compute $x_{k+1} \leftarrow x_k + \alpha_k p_k$ with α_k given by Algorithm 1
7: **if** $k > m$ **then**
8: Discard the pair $\{s_{k-m}, y_{k-m}\}$
9: **end if**
10: Compute $s_k \leftarrow x_{k+1} - x_k, y_k \leftarrow \nabla f_{k+1} - \nabla f_k$
11: $k \leftarrow k + 1$
12: **until** convergence

5.5.2 Limited-memory SR1 method

In the BFGS updating formula (5.19), the matrix B_{k+1} differs from its predecessor B_k by a rank-2 matrix. We now show that there is a simpler rank-1 update

that maintains symmetry of the matrix and allows it to satisfy the secant equation. Unlike the rank-2 update formula, this symmetric-rank-1, or SR1, update does not guarantee that the updated matrix maintains positive definiteness. The symmetric rank-1 update has the general form

$$B_{k+1} = B_k + \sigma v v^T, \tag{5.23}$$

where σ is either $+1$ or -1 and σ and v are chosen so that B_{k+1} satisfies the secant equation (5.17), that is $y_k = B_{k+1}s_k$. By substituting (5.23) into the secant equation, we obtain

$$y_k = B_k s_k + [\sigma v^T s_k]v. \tag{5.24}$$

From (5.24) we deduce, because the term in the brackets is scalar, that v is a multiple of $(y_k - B_k s_k)$. By substituting $v = \delta(y_k - B_k s_k)$ into (5.24), we obtain the following equation

$$y_k - B_k s_k = \sigma\delta^2[s_k^T(y_k - B_k s_k)](y_k - B_k s_k). \tag{5.25}$$

Equation (5.25) tells us how δ and σ should be defined:

$$\sigma = \operatorname{sgn}[s_k^T(y_k - B_k s_k)], \quad \delta = \pm|s_k^T(y_k - B_k s_k)|^{-1/2}. \tag{5.26}$$

Now we can rewrite (5.23) to obtain an explicit update relation for B

$$B_{k+1} = B_k + \frac{(y_k - B_k s_k)(y_k - B_k s_k)^T}{(y_k - B_k s_k)^T s_k}, \tag{5.27}$$

from which we can see possible errors connected with the fact that the denominator of (5.27) may happen to be zero or close to zero.

It is easy to see that even if B_k is positive definite, B_{k+1} may not have the same property. This observation was considered a major drawback in the early days of nonlinear optimization when only line search iterations were used. However, with the advent of trust region methods, the SR1 updating formula has proved to be useful and its ability to generate indefinite Hessian approximations, which are indeed better approximations to the true Hessian than those from the BFGS method, can actually be regarded as one of its chief advantages.

At first glance we may see a possible drawback of the SR1 updating. The denominator in (5.27) can vanish. We see that there are three possible cases:

1. $(y_k - B_k s_k)^T s_k \neq 0$ – there is a unique rank-1 updating formula satisfying the secant equation (5.17), we can use (5.27).

2. $y_k = B_k s_k$ – only updating formula satisfying (5.17) is $B_{k+1} = B_k$.

3. $y_k \neq B_k s_k$ and $(y_k - B_k s_k)^T s_k = 0$ – there is no symmetric rank-1 updating formula satisfying (5.17). As we shall see later, this case can be relatively simply fixed.

The last case suggests that rank-1 updating does not provide enough freedom to develop a matrix with all the desired characteristics.

Nevertheless, we are interested in the SR1 method for the following reasons:

1. A simple safeguard can adequately prevent SR1 from breakdown.

2. The matrices generated by the SR1 are in general better approximations to true Hessian than the BFGS approximations.

3. In quasi-Newton methods for constrained problems, it may not be possible to impose the *curvature condition* $y_k s_k > 0$ and thus BFGS updating is not recommended. Indeed, in these two settings, indefinite Hessian approximations are desirable insofar as they reflect indefiniteness in the true Hessian.

A possible strategy to prevent the SR1 method from breaking down is skipping the update if the denominator in (5.27) is small. More specifically, the update (5.27) is applied only if

$$|s_k^T(y_k - B_k s_k)| \geq r\|s_k\|\|y_k - B_k s_k\|, \tag{5.28}$$

where $r \in (0, 1)$ is a small number (e.g., $r = 10^{-5}$). If (5.28) does not hold, we set $B_{k+1} = B_k$.

In Algorithms 8, 9, and 10 we can see the L-SR1 trust region method we used in our computations, the first two algorithms are used in Algorithm 10. We prefer the trust region framework over a line search framework, because it can accommodate an indefinite Hessian much more easily (although making L-SR1 method with line searching is possible). In the trust region framework we use the notation from Section 5.2.

Algorithm 8 Update of B in the L-SR1 algorithm

$v_k \leftarrow g s_k$
for $i = k - m, k - m + 1, \ldots, k - 1$ **do**
  $q \leftarrow \sigma_i s_k^T v_i$
  $v_k \leftarrow q v_i^T v_k$
end for
$v_k \leftarrow y_k - v_k$
$\delta \leftarrow \frac{1}{\sqrt{|s_k^T v_k|}}$
$v_k \leftarrow \delta v_k$
if $s_k^T v_k > 0$ **then**
  $\sigma_k \leftarrow 1$
else if $s_k^T v_k < 0$ **then**
  $\sigma_k \leftarrow -1$
else
  $\sigma_k \leftarrow 0$
end if
return σ_k and v_k

5.5.3 Restarting, termination criterion, and remarks on quasi-Newton methods

Since the function that we minimize not only depends on many variables, but it is also locally rugged, it is difficult to resolve its minimum properly. The strategy

Algorithm 9 Computation of $B_k z$ in the L-SR1 algorithm

$w \leftarrow gz$
for $i = k - m, k - m + 1, \ldots, k - 1$ **do**
  $q \leftarrow \sigma_i z_i^T v_i$
  $w \leftarrow q v_i^T + w$
end for
return w is the resulting matrix-vector product we want ($w = B_k z$)

Algorithm 10 L-SR1 algorithm

Choose a starting point x_0, initial Hessian approximation B_0, trust region radius Δ_0, number of stored vectors m, convergence tolerance $\epsilon > 0$, parameters $\eta \in (0, 10^{-3})$ and $r \in (0, 1)$
$k \leftarrow 0$
while $\|\nabla f_k\| > \epsilon$ **do**
  Compute s_k by solving the minimization subproblem

$$\min_s \left(\nabla f_k^T s + \frac{1}{2} s^T B_k s \right) \quad \text{subject to} \quad \|s\| \leq \Delta_k \tag{5.29}$$

  using Cauchy point trust region method, (5.9), (5.10) and using Algorithm 9 with m as the parameter
  $y_k \leftarrow \nabla f(x_k + s_k) - \nabla f_k$
  ared $\leftarrow f_k - f(x_k + s_k)$ (actual reduction)
  pred $\leftarrow -\left(\nabla f_k^T s_k + \frac{1}{2} s_k^T B_k s_k \right)$ (predicted reduction)
  $\rho_k \leftarrow \frac{\text{ared}}{\text{pred}}$ (from (5.8))
  if $\rho_k > \eta$ **then**
    $x_{k+1} \leftarrow x_k + s_k$
  else
    $x_{k+1} \leftarrow x_k$
  end if
  if $\rho_k > 0.75$ **then**
    if $\|s_k\| \leq 0.8\Delta_k$ **then**
      $\Delta_{k+1} = \Delta_k$
    else
      $\Delta_{k+1} = 2\Delta_k$
    end if
  else if $0.1 \leq \rho_k \leq 0.75$ **then**
    $\Delta_{k+1} = \Delta_k$
  else
    $\Delta_{k+1} = 0.5\Delta_k$
  end if
  if (5.28) holds **then**
    Use Algorithm 8 and m to compute B_{k+1} (do that even if $x_{k+1} = x_k$)
  else
    $B_{k+1} \leftarrow B_k$
  end if
end while

we propose is often used in many numerical algorithms. When the decrease of the function is not sufficient, then restart comes on. The restarting is used in a slightly different way in the L-BFGS method and in the L-SR1 method.

In our case of the L-BFGS method, restarting occurs when decrease of function is lower than some constant during the last ten iterations. This constant is close to zero (e.g., 10^{-4}) and does not have any important impact on performance. If this happens consecutively, the algorithm terminates.

The L-SR1 method restarts itself in co-operation with the parameter $\rho_k = \frac{\text{ared}}{\text{pred}}$ from (5.8), which is equivalent to the likelihood of the model function m_k from (5.6) to the function f. If this parameter is two times consecutively lower than some constant and at least three iterations were performed, we restart the algorithm. In effect this means to alter the IF statement in Algorithm 10 on the line 22, where we add this simple conditional statement for restarting. The parameter Δ_0 decreases during each restart, too.

Termination criterion in L-SR1 method is based on decrease in last $5m$ iterations, where m is number of stored vectors from Algorithm 10. If any decrease has occurred, the algorithm continues, otherwise the algorithm terminates and returns the *point with the lowest obtained value* of the function f. This strategy is the best one particularly for the L-SR1 method, since after the L-SR1 method reaches a local minimum a restart occurs, which almost always deteriorates the solution in a few following iterations.

In all tested algorithms including L-BFGS and L-SR1 algorithms a bound for parameter y_h was imposed in order to prevent algorithm from reaching another local minimum far from the global minimum. In case of the SUPG method this bound was defined as a multiple of τ_h from initial SUPG method. We use the bound

$$0 \leq |y_h| < 10|\tau_h|, \tag{5.30}$$

where τ_h is defined in (2.19). This bound also allows us to use theoretical results for the SUPG method. For SOLD methods we use a richer range of values, but unless said otherwise, for any SOLD parameter $\tilde{y}_h$ we use a similar bound

$$0 \leq |\tilde{y}_h| < 10|y_h^{SOLD}|, \tag{5.31}$$

where y_h^{SOLD} is the value of the definition of a SOLD parameter from Section 2.4.

5.6 On tuning of the parameters

In this section we will briefly discuss how we tuned some parameters of mentioned methods. In the L-SR1 method there are many constants which can be treated separately and fine-tuned to obtain the best possible results in the best possible time in our tests. The following work stems from the diploma thesis of the author.

In other methods we do not have so many possibilities. This can be used to optimize our L-SR1 method for any given example, but also could lead to lower robustness of the method if we tuned the L-SR1 algorithm exactly for some particular test and used that setup to other ones. We tried to find a setup of the constants performing well in large variety of tests.

In Subsection 5.5.3 we introduced a strategy for restarting. Constants used here are number of stored vectors m, upper bound for the parameter y_h 5.30, initial trust region radius Δ_0, and κ defined as

$$\kappa = \frac{\Delta_0}{\Delta_{0,old}}, \tag{5.32}$$

where $\Delta_{0,old}$ is the initial trust region radius in the last restart-loop and Δ_0 is the radius in the current loop. Our numerical tests suggest there is a relation among these constants.

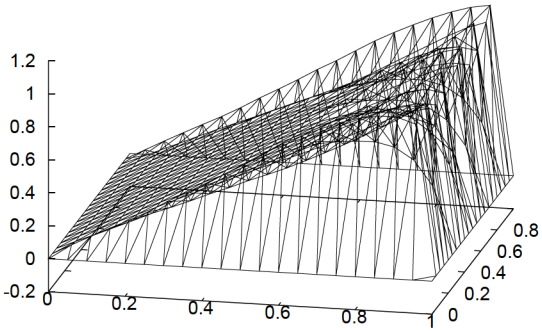


Figure 5.1: Example of improper upper safety bound for parameter y_h (particularly $y_h < 3|\tau_h|$) for Example 1 from Subsection 6.1 when optimizing according to indicator (4.9).

One relation of this type is already mentioned in Subsection 5.5.3. It is the termination strategy, where the algorithm ends when there is no decrease in the last $5m$ iterations. The fact that this number depends on m is natural as the algorithm simply needs more than m iterations to establish a new good rank-1 approximation of the Hessian with m vectors and in addition reduce the trust region radius. This heuristic direct proportionality was tested successfully on our examples.

Similar but weaker heuristics can be done when seeking for optimal $\kappa \in (0, 1)$. It is natural that there must be a relation between the initial trust region radius Δ_0 and the constant κ. Particularly, when we have a large initial Δ_0 then we can decrease in each restart-loop the trust region more rapidly. In Figure 5.2 a graph of typical dependence of achieved minimum by the L-SR1 algorithm on the constant κ is depicted. This suggests that we should choose κ lower than 0.9. For our tests we use $\kappa = 0.5$.

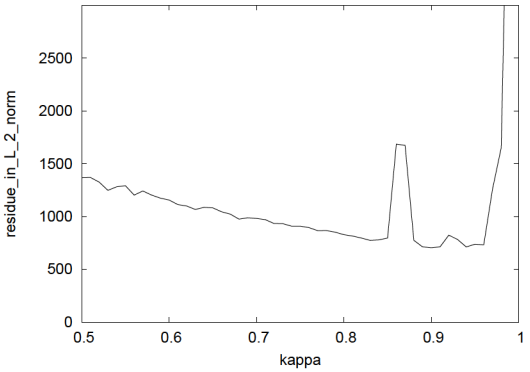


Figure 5.2: Graph of achieved minimum by the L-SR1 algorithm on κ from (5.32).

The most important among all parameters for the speed of convergence of the L-SR1 algorithm is the proper choice of the *initial trust region* Δ_0. In fact, when we have good safety bound (5.30), it is possible to choose any sensible initial trust region. From our tests it comes out that larger values produce faster convergence. The safety bound must be set up properly when using larger values, otherwise this can lead to deterioration of the solution as we can see in Figure 5.1, where a lower "upper bound" for y_h than in (5.30) was used, but not sufficiently small to obtain a good result. It was possible to solve a few problems which occurred during some tests of the L-SR1 algorithm by lowering the safety bound or by a change of the initial trust region Δ_0, we can choose to change one or the other.

How the obtained minimum depends on the choice of the initial trust region Δ_0 in two different tests is shown in graph in Figure 5.3. In the first case, the safety bound was properly chosen while in the second case, the safety bound

was too large. In the latter case the L-SR1 algorithm converges somewhere far from the global minimum if in addition a too large initial trust region is set. This graph also allows us the conclusion that the upper safety bound for the latter test should be somewhat lower. If the upper safety bound was sufficiently small we could use large initial trust region radius.

If we already have a good solution, our tests suggest that an improvement may be achieved by slightly increasing the safety bound (5.30) for the parameter y_h. If we increase it too much, a deterioration of the solution in a way similar to what we described in the previous paragraph occurs. This process can be done also for the L-BFGS algorithm. However, increasing of the safety bound has generally only a negligible positive effect on the overall obtained solution. Even a noticeable speedup of algorithms cannot overcome stability issues connected with a large safety bound.

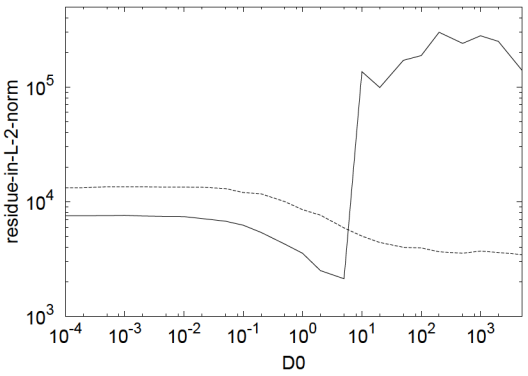


Figure 5.3: Comparison between computations with proper and improper upper safety bound for parameter y_h, D0 stands for Δ_0. Computations with proper safety bound are visualized using a dashed line and those with improper safety bound with a solid line.

The number of parameters that need to be tweaked in order to have a quality of obtained discrete solution similar to that from the L-BFGS method is large and the uncertainty leads us to the conclusion of using the L-BFGS method in a general case. It is clearly a more robust solution which is more documented and applicable to larger set of test cases.

6. Numerical results

6.1 Examples

The following examples were used in our tests. Projection of the exact solution to the 30×30 rectangular isotropic mesh is always next to the definition of each example.

Example 1

We consider the convection–diffusion–reaction equation (1.1) in $\Omega = (0,1)^2$ with $\Gamma^D = \partial\Omega$, $\varepsilon = 10^{-8}$, $\mathbf{b} = (1,0)^T$, $c = 0$, $f = 1$, and $u_b = 0$. The solution $u(x,y)$ of this problem is depicted in Figure 6.1. The solution possesses an exponential boundary layer at $x = 1$ and parabolic boundary layers at $y = 0$ and $y = 1$. In the interior grid points, the solution is close to $u(x,y) = x$. Péclet number from (2.17) in this example is approximately $2.6 \cdot 10^6$. This example is well known and it was used by Mizukami and Hughes [1985].

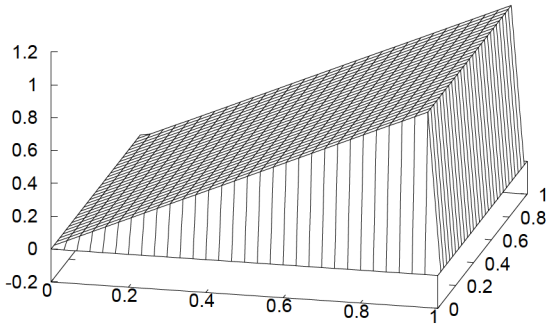


Figure 6.1: Solution of Example 1

Example 2

The convection–diffusion–reaction equation (1.1) is considered in $\Omega = (0,1)^2$ with $\Gamma^D = \partial\Omega$, $\varepsilon = 10^{-8}$, $\mathbf{b} = (\cos(-\pi/3), \sin(-\pi/3))^T$, $c = 0$, $f = 0$, and

$$u_b(x,y) = \begin{cases} 0 & \text{for } x = 1 \text{ or } y \leq 0.7, \\ 1 & \text{otherwise.} \end{cases} \tag{6.1}$$

The solution $u(x,y)$ of this example is depicted in Figure 6.2. The solution possesses an interior characteristic layer in the direction of the convection starting at $(0, 0.7)$ and exponential boundary layers at $x = 1$ and $y = 0$. The Péclet number from (2.17) in this example is approximately $2.6 \cdot 10^6$. This example appeared in Hughes et al. [1986].

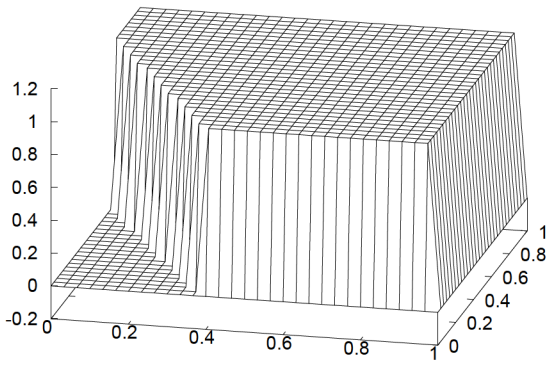


Figure 6.2: Solution of Example 2

Example 3

The convection–diffusion–reaction equation (1.1) is considered in $\Omega = (0,1)^2$ with $\Gamma^N = \{0\} \times (0,1)$, $\Gamma^D = \partial\Omega \setminus (\{0\} \times (0,1))$, $\varepsilon = 10^{-8}$, $\mathbf{b}(x,y) = (-y,x)^T$, $c = 0$, $f = 0$, $g = 0$, i.e., Neumann condition $\frac{\partial u}{\partial n} = 0$ on $x = 0$, and

$$u_b(x,y) = \begin{cases} 1 & \text{if } 1/3 \le x \le 2/3 \text{ and } y = 0, \\ 0 & \text{otherwise.} \end{cases} \tag{6.2}$$

The solution $u(x,y)$ of this problem is depicted in Figure 6.3. The solution possesses two interior characteristic layers in the direction of the convection starting at $(1/3,0)$ and $(2/3,0)$. One can see that the initial boundary condition is transferred towards the outflow boundary at $x = 0$. The Péclet number from (2.17) in this example is in the interval from $8.6 \cdot 10^5$ to $1.74 \cdot 10^6$. This example was used, e.g., by Knopp et al. [2002].

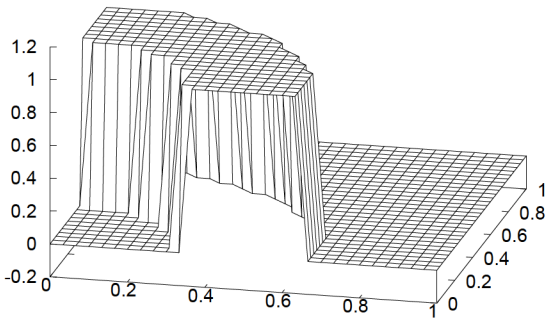


Figure 6.3: Solution of Example 3

Example 4

The convection–diffusion–reaction equation (1.1) is considered in $\Omega = (0,1)^2$ with $\Gamma^D = \partial\Omega$, $\varepsilon = 10^{-8}$, $\mathbf{b} = (1,0)^T$, $c = 0$,

$$f(x,y) = \begin{cases} 0 & \text{if } |x - 0.5| \ge 0.25 \text{ or } |y - 0.5| \ge 0.25, \\ -32(x - 0.5) & \text{otherwise,} \end{cases} \tag{6.3}$$

and $u_b = 0$. The solution $u(x,y)$ of this problem is depicted in Figure 6.4. The solution possesses two interior characteristic layers in the direction of the convection starting at $(0.25, 0.25)$ and $(0.25, 0.75)$. The Péclet number from (2.17) in this example is approximately $2.6 \cdot 10^6$. This example was first considered in John and Knobloch [2008]. Then, it was subsequently used, e.g., in Lukáš and Knobloch [2018].

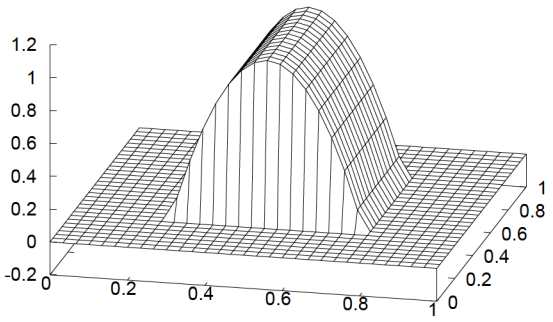


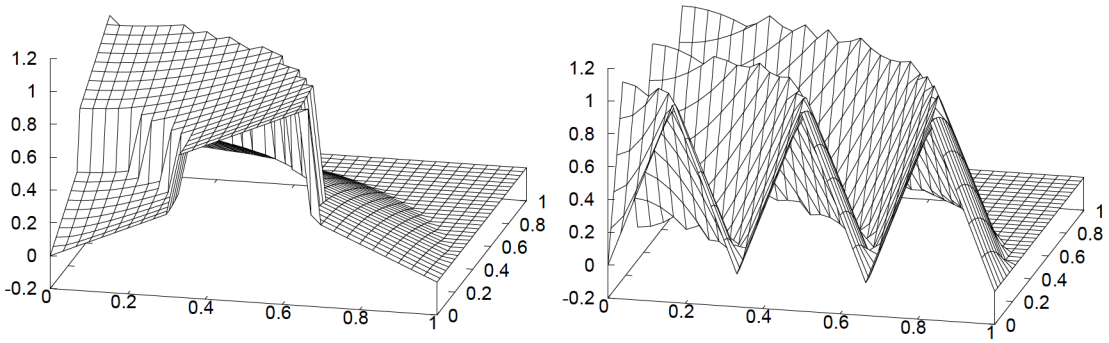
Figure 6.4: Solution of Example 4

Example 5

We consider the problem (1.1) with $\Omega = (0,1)^2$, $\Gamma^N = \{0\} \times (0,1)$, $\Gamma^D = \partial\Omega \setminus \overline{\Gamma^N}$, $\varepsilon = 10^{-8}$, $\mathbf{b}(x,y) = (-y,x)^T$, $c = f = 0$, $g = 0$,

$$u_b(x,0) = \begin{cases} x & \text{if } 0 \le x \le \frac{1}{3}\,, \\ \frac{1}{3} + x & \text{if } \frac{1}{3} < x < \frac{2}{3}\,, \\ 1 - x & \text{if } \frac{2}{3} \le x \le 1\,, \end{cases} \qquad x \in [0,1]\,,$$

and $u_b = 0$ elsewhere on Γ^D. This example was inspired by Example 3 but it has a different boundary condition which has three parts which are not piecewise constant. One can see it in Figure 6.5a.



(a) Solution of Example 5 **(b)** Solution of Example 6

Figure 6.5: Solutions of Examples 5 and 6

Example 6

This example is similar to Examples 3 and 5. In this case we slightly change the boundary condition at $y = 0$ which now reads:

$$u_b(x,0) = \begin{cases} 6\,x & \text{if } 0 \le x \le \frac{1}{6}\,, \\ 2 - 6\,x & \text{if } \frac{1}{6} \le x \le \frac{1}{3}\,, \\ -2 + 6\,x & \text{if } \frac{1}{3} \le x \le \frac{1}{2}\,, \\ 4 - 6\,x & \text{if } \frac{1}{2} \le x \le \frac{2}{3}\,, \\ -4 + 6\,x & \text{if } \frac{2}{3} \le x \le \frac{5}{6}\,, \\ 6 - 6\,x & \text{if } \frac{5}{6} \le x \le 1\,. \end{cases} \qquad x \in [0,1]\,.$$

In this example, there are no layers in the solution, one can see it in Figure 6.5b. An accurate approximation of the solution to this example is challenging for all the methods.

Example 7

We consider the problem (1.1) with $(0,1) \times \Gamma^N = \{0\}$, $\Gamma^D := \partial\Omega \setminus \overline{\Gamma^N}$. In this case the boundary condition at $y = 0$:

$$u_b(x,0) = \begin{cases} 0 & \text{if} \quad 0 \le x \le \frac{1}{6}\,, \\ \sqrt{-x^2 + \frac{4}{3}x - \frac{7}{36}} & \text{if} \quad \frac{1}{6} \le x \le \frac{2}{3}\,, \\ 0 & \text{if} \quad \frac{2}{3} < x \le 1\,, \end{cases} \qquad x \in [0,1]\,. \tag{6.4}$$

Diffusion $\varepsilon = 10^{-8}$, convection is directed in the y-axis direction $\mathbf{b}(x,y) = (0,1)^T$, $c = f = 0$, $g = 0$, In this example a smooth boundary condition which is not piecewise linear is considered. There is one steep layer in the exact solution which can be seen in Figure 6.6.

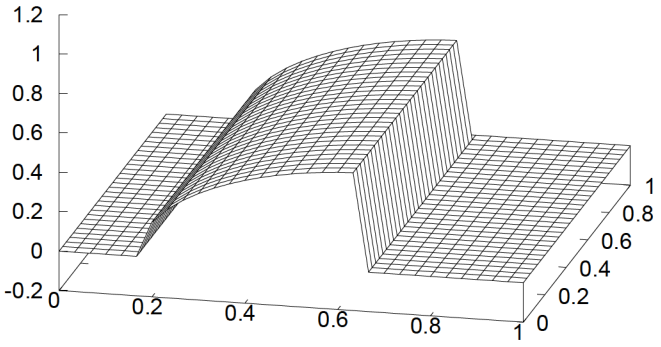


Figure 6.6: Solution of Example 7

Let us note that in Examples 2, 3, 5, and 7, the function u_b does not satisfy the assumption $u_b \in H^{1/2}(\Gamma^D)$ but it can be regularized in such a way that the assumption holds and the numerical results presented here do not change.

6.2 Numerical methods of minimizing error indicators

In this section we try to compare numerical methods of minimizing error indicators introduced in Section 4 on four basic examples introduced in Section 6.1, these are in the two-dimensional domain and are discretized by conforming piecewise linear finite elements. We will optimize the parameter τ_h from the SUPG method in this section. Abbreviations of used methods are listed in Table 6.1 and will be used in the following figures and tables. The criterion for the evaluation of compared methods will be introduced later in Subsection 6.2.1.

For maximum efficiency of vector and matrix operations we used the UMFPACK library, which is part of the SuiteSparse library, see Davis [2006]. All tasks in this section were carried out on the netbook Asus eee 901 with a single Intel Atom N270 processor and 1GB of DDR2 memory. This computer has a low performance, from our experience the computation on a common notebook is more than twice faster. We used the operating system Ubuntu 11.04, which is a Linux distribution based on the kernel 2.6.38-8. No other applications were in use on the computer during the testing and the computer was in no saving mode.

Name	*Method description*	*Related algorithms*
SD	Steepest descent method	4, 1
NCG	Nonlinear conjugate gradient	5, 1
L-BFGS	Limited-memory BFGS	7, 6, 1
L-SR1	Limited-memory SR1	10, 8, 9

Table 6.1: List of methods used to optimization of parameter y_h

In all numerical algorithms a global bound for iteration count was imposed but in fact never reached in practice by the L-BFGS or L-SR1 algorithms. Only the steepest descent algorithm and the conjugate gradient algorithm reached this bound which was 3000 iterations in the case of the steepest descent algorithm and 1000 iterations in the case of the conjugate gradient algorithm. However, the runtime of both of these algorithms was more than ten times longer than the runtime of both the L-BFGS algorithm and the L-SR1 algorithm in any of our examples.

So when we will talk about the best minimum reached by a given method we mean the uncut minimum in the case of quasi-Newton methods and the result after the large number of iterations in the case of the steepest descent method or the nonlinear conjugate gradient method.

The safety bound from (5.30) was used in all methods, in this section we only considered SUPG method. While the safety bound in the presented tests is left constant among all used tasks, the initial trust region in the L-SR1 method Δ_0 was in two tests (tests 1 and 2) reduced from 100 to 0.5. This prevents the L-SR1 algorithm from converging to some local minimum far from the global minimum. In this way, the L-SR1 method can be in fact used to identify whether the safety bound for the given example is not chosen unproperly. As it is shown later, the L-BFGS method performs always well unless the safety bound is very large, which is not our case.

6.2.1 Evaluating tests

In this subsection we will introduce the testing methodology we use for evaluating the numerical tests. The best obtained minimum Φ_{best} among all minimization methods from Table 6.1 for the given minimization task is selected as the origin for the given task and then we calculate the quantities

$$\Delta_{tot}(\Phi) = \log \Phi - \log \Phi_{best} \ , \tag{6.5}$$

where Φ is the value obtained by a given minimization method, and

$$\Delta_{20s}(\Phi) = \log \Phi_{20s} - \log \Phi_{best,20s} \ , \tag{6.6}$$

where $\Phi_{best,20s}$ is the best minimum among all minimization methods for the given minimization task obtained in the first 20 seconds and Φ_{20s} is the value obtained by a given minimization method in the first 20 seconds.

Quantities Δ_{tot} and Δ_{20s} are motivated by the fact that we have very different order of magnitude of Φ for different tasks and relations (6.5) and (6.6) for

evaluating the tests are robust enough to cope with this. So the quantities Δ_{tot} and Δ_{20s} have the same meaning in all examples and tasks and we can compare them directly among all the tasks.

6.2.2 Results of numerical tests

In this subsection we present the results from the point of view of minimization of functions of many variables. A graph of such a minimization in time is given in Figure 6.7

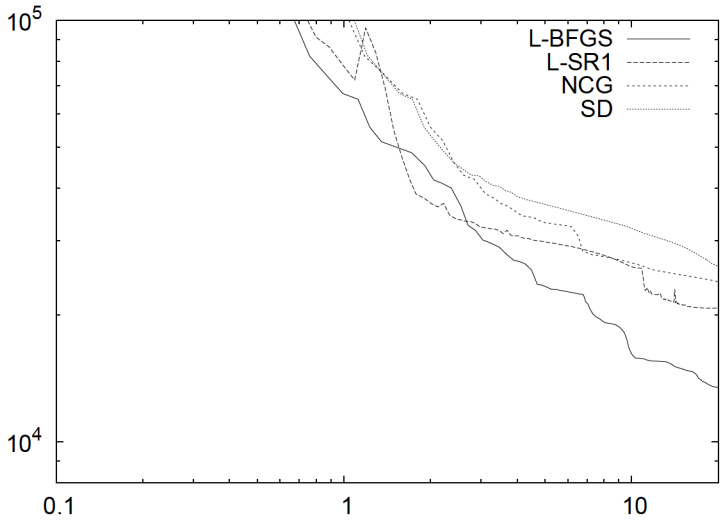


Figure 6.7: Process of minimization according to indicator (4.1) on isotropic mesh for investigated Example 2. On the vertical axis there is L^2 residue and on the horizontal axis there is time in seconds.

In the graph of minimization according to time in Figure 6.7 we do not display the best minimum obtained by a method to that time but the value in an actual iteration. This can be nicely seen particularly in the case of the L-SR1 method, where a run off the global minimum can occur by successive restarts.

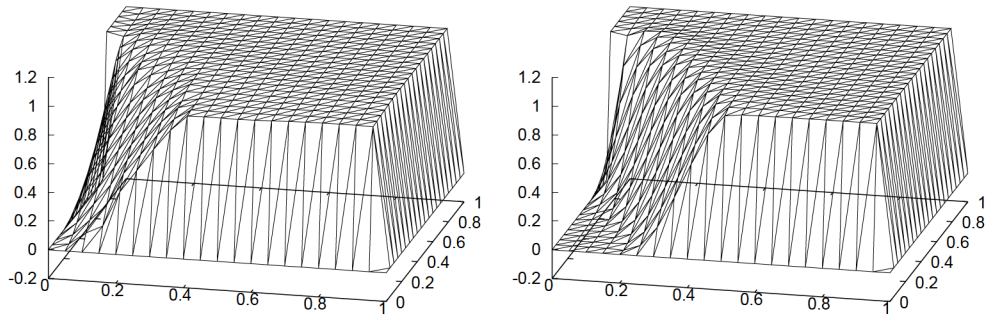
Calculations of Δ_{tot} in the case of SD method lasted more than ten times longer than similar calculations in the case of L-SR1 or L-BFGS algorithms. Almost always the L-SR1 algorithm ended a few seconds before the L-BFGS algorithm, NCG algorithm ran much longer but not as long as SD method. Calculations show that different nonlinear minimization methods are suited for different minimization tasks.

One of the crucial parts of theoretical research is the proper choice of the upper bound for y_h from (5.30). Whether it is theoretically possible to find an optimal upper bound dependent on a given example, which would guarantee a "correct" result, is not known. We see an example of an improper (too large) upper safety bound for y_h in Figure 6.8, where the L-BFGS method found an overdiffuse solution. This is an important issue in the case of all minimization methods.

The initial choice of the parameter y_h is "correct" in the sense that it is the parameter from the SUPG method (2.5), for which we have theoretical results. When too large upper bound for y_h is applied, a solution with a lower value of

Task to solve	SD		NCG		L-BFGS		L-SR1	
	Δ_{20s}	Δ_{tot}	Δ_{20s}	Δ_{tot}	Δ_{20s}	Δ_{tot}	Δ_{20s}	Δ_{tot}
Example 1	0.46	0.36	0.28	0.16	**0**	**0**	0.41	0.42
Example 2	0.21	0.01	0.20	0.04	**0**	**0**	0.28	0.30
Example 3	0.44	0.44	0.45	0.44	0.45	0.45	**0**	**0**
Example 4	0.03	0.02	0.02	0.03	0.02	0.03	**0**	**0**
Example 1 A	1.60	0.96	1.50	1.36	**0**	**0**	0.95	1.40
Example 2 A	0.66	0.20	0.58	0.36	**0**	**0**	0.43	0.55
Example 3 A	0.39	0.39	0.39	0.39	0.41	0.40	**0**	**0**
Example 4 A	0.01	0.02	0.01	0.02	0.02	0.03	**0**	**0**
Example 1 C	0.59	0.34	0.52	0.24	**0**	**0**	0.55	0.59
Example 2 C	0.09	0.02	0.09	0.04	**0**	**0**	0.09	0.10
Example 3 C	0.01	**0**	0.02	0.01	**0**	0.01	0.01	0.02
Example 4 C	0.14	0.14	0.15	0.15	0.12	0.15	**0**	**0**
Example 1 AC	1.24	0.81	1.22	1.13	**0**	**0**	0.87	1.20
Example 2 AC	0.25	0.10	0.19	0.15	**0**	**0**	0.18	0.13
Example 3 AC	0.02	**0**	0.02	0.01	**0**	0.01	0.02	0.01
Example 4 AC	0.05	0.03	0.04	0.04	0.04	0.04	**0**	**0**
Σ	1.14	0.83	0.95	0.65	**0.47**	**0.47**	0.69	0.73
Σ_A	2.68	1.57	2.50	2.15	**0.43**	**0.42**	1.38	1.96
Σ_C	0.83	0.50	0.77	0.44	**0.12**	**0.15**	0.65	0.69
Σ_{AC}	1.53	0.95	1.45	1.32	**0.04**	**0.04**	1.06	1.34

Table 6.2: Comparison of used methods. "C" means we minimize according to the indicator I_h^{cross} (4.3), otherwise we minimize according to the indicator I_h (4.1). "A" means anisotropic mesh is used, otherwise isotropic mesh is used. Δ_{20s} and Δ_{tot} are defined in (6.6) and (6.5), respectively. The symbol Σ stands for the sum of either Δ_{20s} or Δ_{tot} for Examples 1–4 with indicator I_h on isotropic mesh. Analogously, Σ_A is the sum of either Δ_{20s} or Δ_{tot} for Examples 1–4 with indicator I_h on anisotropic mesh, Σ_C is the sum with indicator I_h^{cross} on isotropic mesh and Σ_{AC} is the sum of results obtained with indicator I_h^{cross} on anisotropic mesh. The best results for any given row in this table are emphasized by using the bold face.



(a) $I_h^{cross} = 0.28$, upper bound $y_h \leq 500\tau_h$ (b) $I_h^{cross} = 0.44$, upper bound $y_h \leq 5\tau_h$

Figure 6.8: An overdiffuse solution according to the indicator with crosswind derivative control term (4.3) when too large upper bound for y_h is used. In Figure 6.8a we can see the solution with a lower value of the error indicator I_h^{cross} than in Figure 6.8b. However, the solution with the higher value of the error indicator is better due to correctly chosen upper safety bound. L-BFGS method was used to get the minimum.

the error indicator I_h can be worse from the physical point of view, even worse than the solution of the SUPG method with the parameter from (2.19). Such a nonphysical solution found by the L-BFGS method is depicted in Figure 6.8, where we can compare it with the solution with a "correct" upper bound.

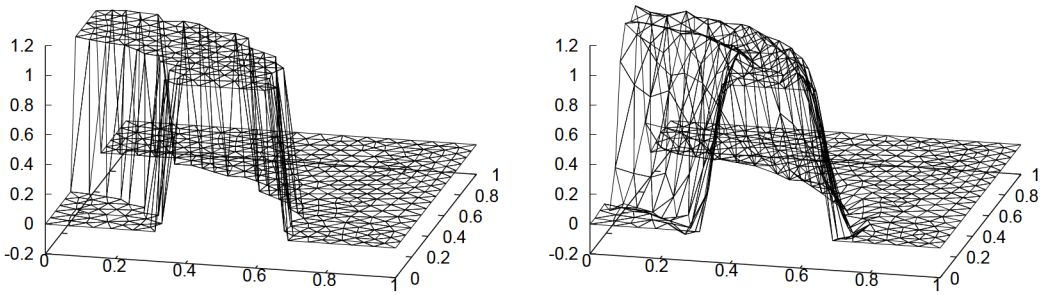
6.3 Results on anisotropic meshes

In this section we will present numerical results for Example 3 on anisotropic meshes. These numerical results come mostly from Lukáš [2012]. Anisotropic meshes were generated by ANGENER numerical software written by Dolejší [1998].

We use linear conforming (P_1) type of finite elements. This defines the space W_h. The solution possesses two interior characteristic layers in the direction of the convection starting at $(\frac{1}{3}, 0)$ and $(\frac{2}{3}, 0)$. We provide results on an unstructured mesh with 858 elements. We use Indicator (4.3) in the following.

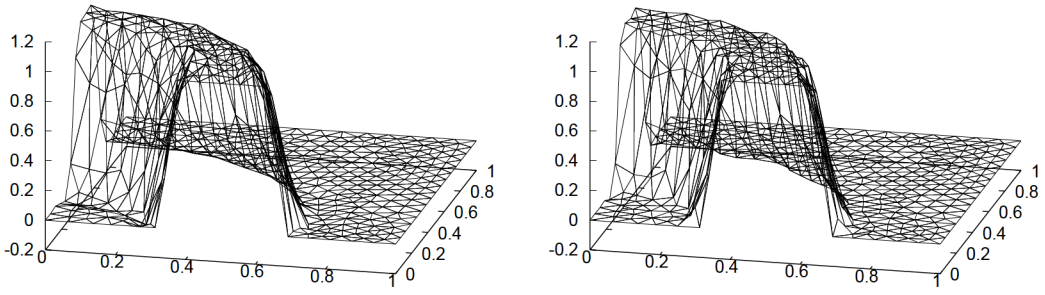
We use three different finite element spaces for the optimized parameter τ_h in this section. The first one is the space of discontinuous piecewise constant functions P_0^{disc}. The second one is the space of discontinuous piecewise linear functions P_1^{disc} and the last one is the space of continuous piecewise linear functions P_1^{C}.

The Lagrange interpolation in W_h of the exact solution $u(x, y)$ of this problem is depicted in Figure 6.9a. In Figure 6.9b we can see the solution of the SUPG method (2.5). In Figure 6.9c we show the discrete solution obtained by minimization of τ_h from P_1^{C}. The discrete solutions obtained by minimization of parameter τ_h from P_0^{disc} and P_1^{disc} are very similar. Therefore, we provide only the figure of discrete solution with piecewise linear discontinuous parameter τ_h ($\tau_h \in P_0^{\text{disc}}$), see Figure 6.9d.



(a) Interpolation of exact solution

(b) SUPG solution

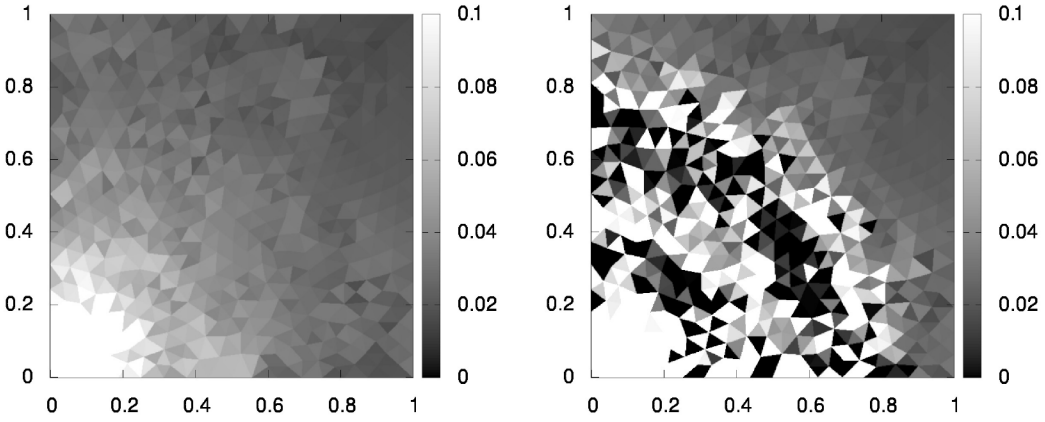


(c) Minimizing with $\tau_h \in P_1^{\text{C}}$

(d) Minimizing with $\tau_h \in P_0^{\text{disc}}$

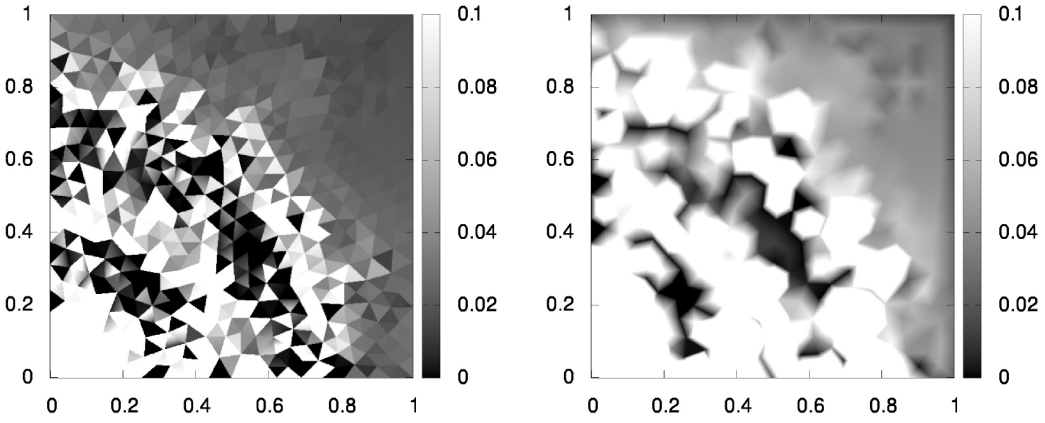
Figure 6.9: In (a), (b), (c), and (d), there is a comparison of different discrete solutions of Example 3 on anisotrop meshes.

In Figure 6.10a there is the gray-scale graph of values of the parameter τ_h from the SUPG method (2.5), $\tau_h \in P_0^{\text{disc}}$ defined in (2.19), which is also the



(a) SUPG - $\tau_h \in P_0^{\text{disc}}$ from (2.19)

(b) $\tau_h \in P_0^{\text{disc}}$ optimized



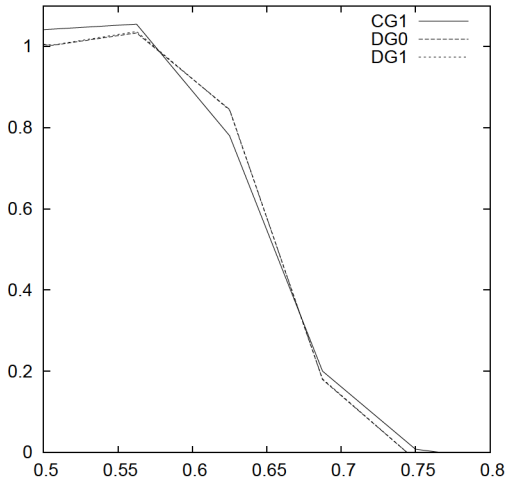
(c) $\tau_h \in P_1^{\text{disc}}$ optimized

(d) $\tau_h \in P_1^{\text{C}}$ optimized

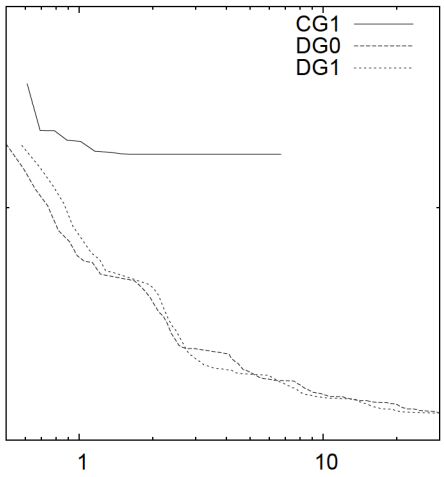
Figure 6.10: Gray-scale maps of values of τ_h for various spaces of τ_h

initial setup for the L-BFGS minimization method. The fact that the results with piecewise linear discontinuous and piecewise constant parameters are very similar can be seen also in Figure 6.10b and 6.10c, where the values of parameter τ_h for these spaces are shown. We provide in Figure 6.10d also the values of τ_h after the minimization process with piecewise linear continuous parameter τ_h.

In Figure 6.11a are depicted the cross-sections of resulting discrete solutions for all spaces of parameter τ_h. In Figure 6.11b, we can see the fact that even the minimization speed is almost equivalent for piecewise linear discontinuous and piecewise constant parameters τ_h.



(a) Cross-sections at $x = 0$



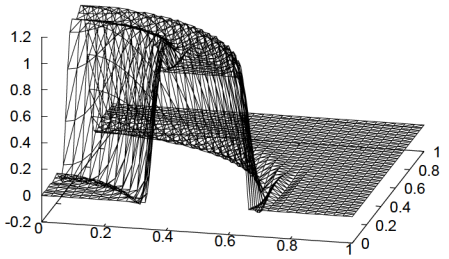
(b) Process of minimizing

Figure 6.11: In (a) there are the cross-sections of Example 3 at $x = 0$. (b) Minimizing process according to Indicator (4.1) for various spaces of τ_h, on the vertical axis there is the L^2 residue according to Indicator (4.1) and on the horizontal axis there is time in seconds. CG1 stands for P_1^{C}, DG0 stands for P_0^{disc}, and DG1 stands for P_1^{disc}

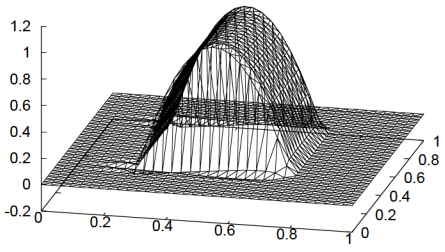
6.4 Higher degree FE spaces

We consider piecewise polynomial spaces up to degree 5 which are globally continuous (Lagrange finite elements) for the discrete solution. In the following text, we use abbreviations for these spaces derived from the FEniCS Alnaes et al. [2012] software `CG1` – `CG5`. We denoted the space CG1 as P_1^{C} in the previous section.

Based on our former observations in numerical experiments, we have chosen the spaces Y_h for the parameter τ_h as piecewise discontinuous, to name these spaces we use again the abbreviations from the FEniCS software `DG0` – `DG5`. We denoted the spaces DG0 and DG1 as P_0^{disc}, and P_1^{disc}, respectively. So in addition to the classical choice for the space of parameter τ_h with $P_0(K)$ for all $K \in \mathcal{T}_h$ (which is the space `DG0` in our notation), we consider spaces of discontinuous piecewise P_k functions, $k = 1, \ldots, 5$.



(a) Example 3 - SUPG solution



(b) Example 4 - SUPG solution

Figure 6.12: Solutions of the SUPG method, τ_h not optimized.

The SUPG solution of Example 3 in the `CG1` space is depicted in Figure 6.12a. The approximate solution of Example 4 in the `CG1` space is in Figure 6.12b. The Lagrange interpolation in W_h of the exact solution $u(x, y)$ of a problem under consideration is denoted as u_e where necessary.

The solution of Example 3 possesses two interior characteristic layers in the direction of the convection starting at $(\frac{1}{3}, 0)$ and $(\frac{2}{3}, 0)$. These interior layers are generally not aligned with the direction of elements' sides.

The solution of Example 4 possesses two interior characteristic layers in the direction of the convection starting at $(0.25, 0.25)$ and $(0.25, 0.75)$. This also means that the resulting discrete solution can be strongly influenced by the choice of the mesh, particularly by alignment of elements' sides.

If not said otherwise we provide results on a structured mesh of Friedrichs–Keller type with 34 nodes in each direction.

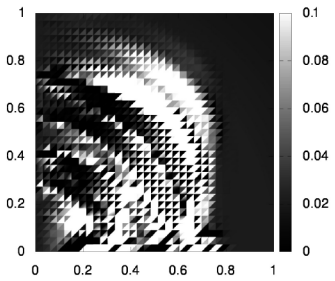
In both examples we consider in this section the Péclet number from (2.18) is of the order 10^6. We use Indicator (4.3) in all of the following numerical tests.

In Figures 6.12a and 6.12b we show solutions of the SUPG method for the `CG1` FE space where the parameter τ_h is defined in (2.19) and in this case we choose τ_h from the space `DG0`.

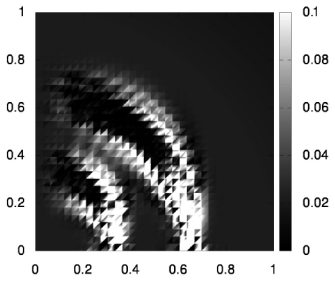
The parameter τ_h is optimized by the L-BFGS-B nonlinear minimization method described in chapter 5 using the default setup from `scipy` library with `gtol: 1e-14` and `ftol: 1e-14`. The method starts with the values given by (2.19).

We do not provide any figure which would show just the values of the parameter τ_h from (2.19), which is the starting point of the minimization procedure. Such an image would be not interesting as the values of τ_h for both examples we use are (almost) constant on the whole domain. The quality of discrete solutions after the minimization procedure is so good that images of such a discrete solution would be very similar to images 6.3 and 6.4 so we do not involve them right now. The quality of discrete solutions after the minimization procedure with Indicator (4.3) has been studied in John et al. [2011] and Lukáš [2012].

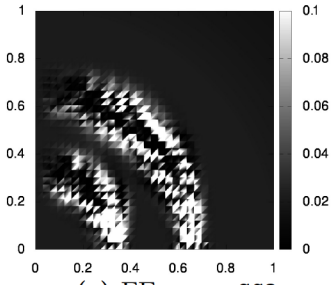
The values of the parameter τ_h after the whole minimization procedure are in Figures 6.13 and 6.14. In Figure 6.13 we fix the FE space for the parameter τ_h and change the FE space of the discrete solution. On the other hand we fix the FE space of the discrete solution in Figure 6.14 and change the FE space of the parameter τ_h. We can see that higher values of the parameter τ_h are at places where oscillations in the SUPG method with τ_h from (2.19) appear.



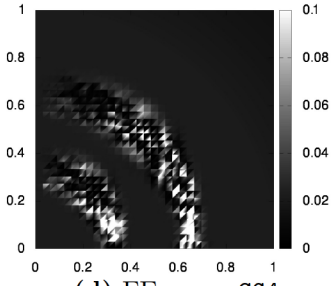
(a) FE space `CG1`



(b) FE space `CG2`

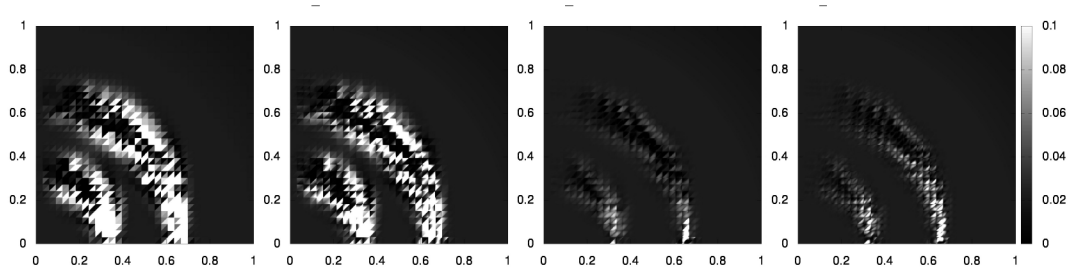


(c) FE space `CG3`



(d) FE space `CG4`

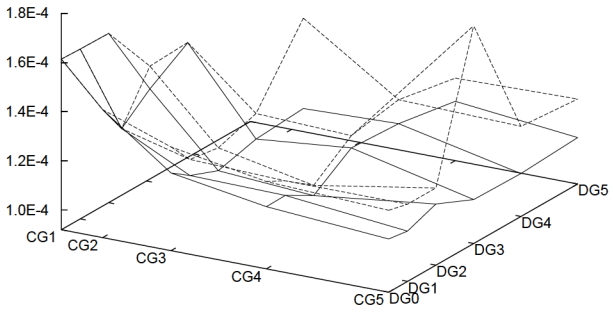
Figure 6.13: Optimized parameter τ_h for different FE spaces, τ_h is from the space `DG1`, Ex. 3



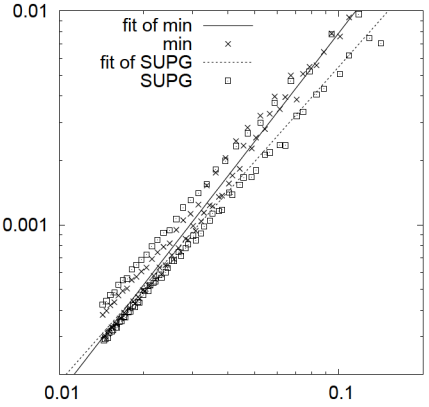
(a) τ_h from `DG0` **(b)** τ_h from `DG1` **(c)** τ_h from `DG2` **(d)** τ_h from `DG3`

Figure 6.14: Changing the space for the parameter τ_h and preserving the finite element space `CG3` for the discrete solution of Example 3.

We provide in Figure 6.15a the value of the Indicator (4.3) after the 30 seconds run of the minimization procedure and at the end of the whole minimization procedure. We can see from the image that using finer FE spaces is not efficient in this case, and we do not obtain a lower value of the indicator by applying this time constraint.



(a) - Minimization after 30s and at the end On the vertical axis there is the value of Indicator (4.3), on the front axis there is the number of degrees of freedom of the Lagrange FE space (data points marked by the `CG` space) of u_h, and on the last axis there is the number of DOFs of the `DG` space of parameter τ_h (again, denoted by the space name). Results after 30 seconds (dashed line) and at the end of the program run (full line) are provided.



(b) - Convergence rate Logarithmic scale, on the vert. axis there is the value of the error $\|u_h - u_e\|_{H^1(\Omega)}$, on the horiz. axis there is the mesh par. h, compare with SUPG (τ_h from (2.19)).

Figure 6.15: Minimization after 30s and at the end **(a)** and convergence results **(b)** for Example 3

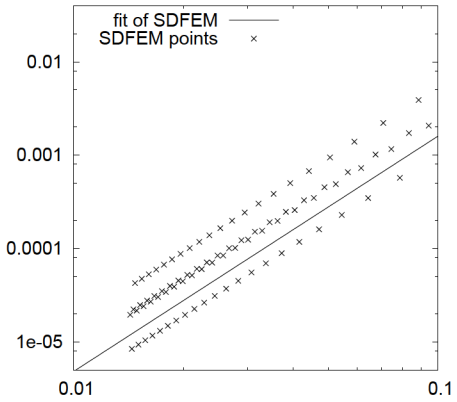
We provide numerical convergence results in Figure 6.15b where we changed the mesh step by step from 8 elements in each direction to 100 elements in each direction and run every time the whole optimization process. We see that in this case the convergence rate of optimized solutions (labeled `min`) is approximately the same as of the SUPG method with parameter from (2.19) (labeled `SUPG`).

To obtain precisely the convergence rates we use a different technique than it is usual among the community of the numerical mathematicians. We use here a technique of curve fitting which is well known to physicists. This approach is justified by having a lot of data for such a "physical" fitting. The objective function $f(a, b)$, whose parameters a and b are fitted, has the form

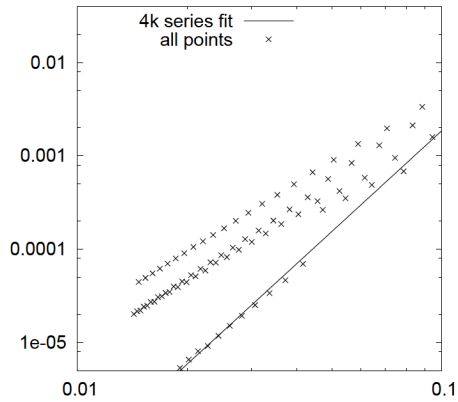
$$f(a, b) = a \cdot h^b. \tag{6.7}$$

Python `scipy` library uses the Levenberg–Marquardt algorithm in `curve_fit`

function of `scipy` module. Paper Moré [1978] describes how this algorithm is implemented in the python library. An important fact about this approach is that we obtain also a rigorous uncertainty or standard deviation of fitted coefficients.



(a) On the vertical axis there is the value of the error $\|u_h - u_e\|_{H^1(\Omega)}$, u_h solution of SUPG with τ_h from (2.19).



(b) On the vertical axis there is the value of the error $\|u_h - u_e\|_{H^1(\Omega)}$, u_h is the solution of the minimization problem.

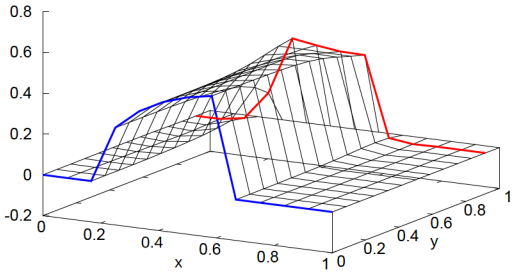
Figure 6.16: Comparison of the results from the SUPG method with parameter τ_h from (2.19) and the adaptive method. On the horizontal axis there is the mesh parameter h. On the right hand side the fit is only for the points corresponding to the `4k` mesh. Results are for Example 4.

In Figure 6.16a we can see the convergence results of the SUPG method for Example 4 and in Figure 6.16b we can see the convergence results obtained by minimizing according to the Indicator (4.3). As we see, if the meshes are chosen properly we are able to obtain a higher order convergence using our technique. A properly chosen mesh in Example 4 is apparently, regarding the exact solution or regarding the convergence graphs, a structured square mesh which has $2 + 4k$ edges (or equivalently $3 + 4k$ nodes), $k \in \mathbb{N}$ at $x = 0$. We will refer to such a mesh as `4k` mesh. The `4k` mesh has the inner layers well resolved by an element's side.

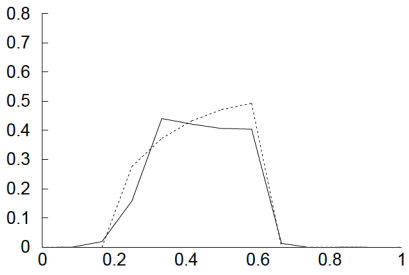
The resulting coefficients of fitted function for the SUPG method with τ_h defined in (2.19) are, together with their standard deviation: $f(h) = (0.5 \pm 0.3) \cdot h^{(2.5 \pm 0.1)}$. The values of the coefficient a naturally differ slightly among the types of meshes with different numbers of elements, the alignment of elements' sides with the direction of the convection is better than in case of Example 3. The values of parameter b were (2.5 ± 0.1). So we have $h^{2.5}$ convergence rate for the SUPG method for Example 4 on our structured grid.

But how do the solutions of our adaptive method behave? The Figure 6.16 suggests a higher rate of convergence than the SUPG method has for a special mesh (every fourth point in the graph has apparently a different order of convergence). After all, for the `4k` mesh defined earlier the fitted function is $f(h) = (7.0 \pm 0.3) \cdot h^{(3.5 \pm 0.1)}$. This means that the convergence rate is $h^{3.5}$. The gain in convergence rate in this setting is 1 in comparison with SUPG method. It is a substantial improvement in this setup and shows us the potential of our adaptive techniques.

6.5 Behavior of indicators I_h^{cross} and I_h^{lim}



(a) the discrete solution u_h after optimization using I_h^{cross}.



(b) profile at $y = 1$ after optimization using I_h^{cross} and nodally exact solution (dotted).

Figure 6.17: Optimized discrete solution u_h of Example 7 using the indicator I_h^{cross}.

We are going to present one numerical example now to show how indicators influence the quality of the numerical solution. We are going to show the typical drawbacks of Indicator I_h^{cross} on Example 7, optimizing the parameter τ_h from the SUPG method (2.19). Let us remind that the initial condition is transferred in the direction of the convection (see the definition of Example 7 definition for further details). The solution possesses an exponential boundary layer at $y = 1$ and an inner layer at $x = \frac{2}{3}$. Péclet number from (2.18) in this illustration is approximately $2.6 \cdot 10^6$.

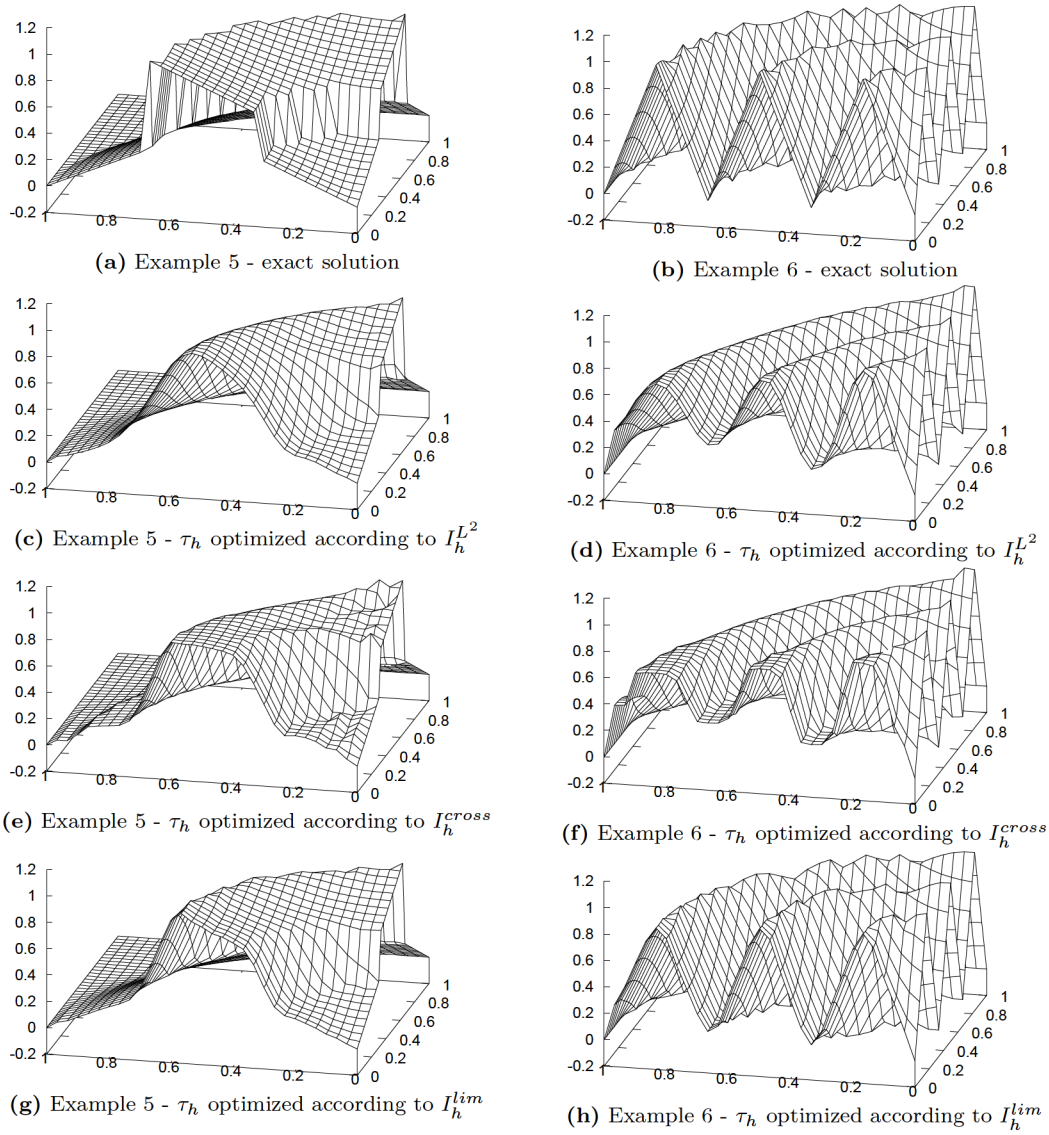
One can see in Figure 6.17 the tendency of the indicator I_h^{cross} (4.3) to artificially create steep layers and to flatten the solution. It comes from its nature to favour one steep layer over smooth ones. The presence of the exponential boundary layer along the penultimate row of nodes ($y = \frac{12}{13}$) depicted with the red color in Figure 6.17 is crucial. If the outflow Dirichlet boundary condition was not prescribed the behavior would be the same for all indicators and the discrete solution would be a "physical", or "correct", one.

Based on our former observations in numerical experiments (Lukáš [2015]), we have chosen the space Y_h for the parameter τ_h to be the space of piecewise constant functions (space `DG0` in our notation based on Alnaes et al. [2012]). This choice of the FE space for the parameter τ_h led always to a satisfactory optimization result (the choice of the actual discrete solution space has no significant impact on the quality and result of the optimization process).

We will now consider two other examples, Example 5 and Example 6. We will optimize still the parameter τ_h from the SUPG method (2.19). The solutions of both examples possess layers or steep parts which are not aligned with the mesh. Later on, outflow profiles at $x = 0$ will enable an easy evaluation of various approaches for the numerical solution of (1.1).

We consider a structured triangulation $\mathcal{T}_h$ of Ω of Friedrichs–Keller type containing 31×31 vertices which means 30 triangular cells along each side. For both examples, the Péclet number defined in (2.17) is then of the order 10^6. The space W_h consists of continuous piecewise linear functions (space `CG1` in our notation based on Alnaes et al. [2012]). Lagrange interpolations in W_h of the exact solutions of Examples 5 and 6 are depicted in Figures 6.18a and 6.18b.

Parameter τ_h is optimized by the L-BFGS-B nonlinear minimization method



(a) Example 5 - exact solution

(b) Example 6 - exact solution

(c) Example 5 - τ_h optimized according to $I_h^{L^2}$

(d) Example 6 - τ_h optimized according to $I_h^{L^2}$

(e) Example 5 - τ_h optimized according to I_h^{cross}

(f) Example 6 - τ_h optimized according to I_h^{cross}

(g) Example 5 - τ_h optimized according to I_h^{lim}

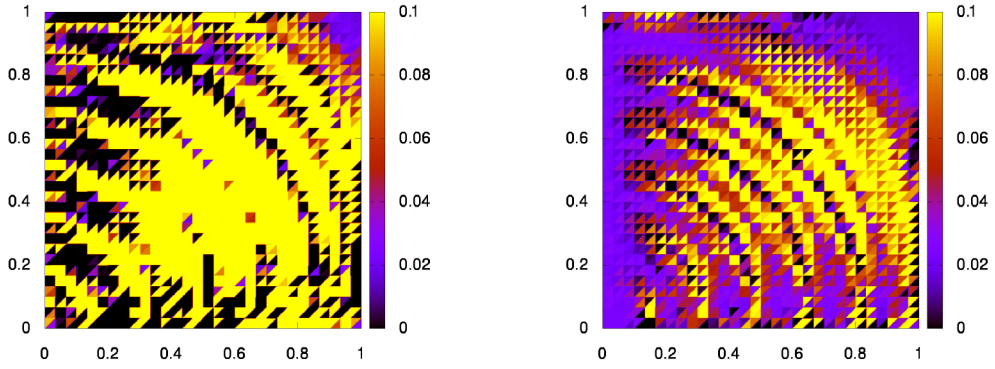
(h) Example 6 - τ_h optimized according to I_h^{lim}

Figure 6.18: Interpolations of exact solutions and results after optimization according to $I_h^{L^2}$, I_h^{cross}, and I_h^{lim}.

described in chapter 5, the `scipy` scientific library is used, with `gtol: 1e-16`, `ftol: 1e-16`, maximum number of iterations is 300. The method starts with the values given by (2.19). We use the FEniCS finite element library for all FE computations Alnaes et al. [2012]. The value of parameter t_0 in the definition of I_h^{lim} is 0.03 for both test cases but values up to approximately $t_0 = 1$ still provide qualitatively better results than those obtained by minimizing the error indicator with added crosswind derivative control term (4.3).

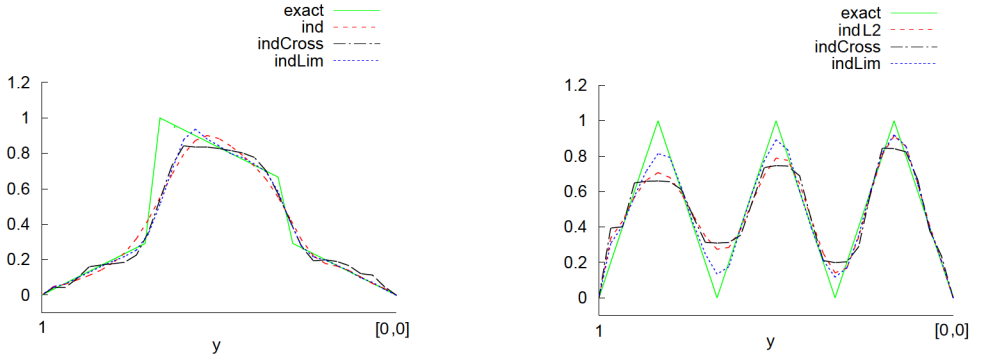
Approximate solutions obtained after parameter optimization can be seen in Fig. 6.18. In addition, Fig. 6.20 shows the outflow profiles corresponding to all three error indicators and the exact solution. It can be clearly seen that the error indicator I_h^{lim} leads to the best results. In particular, we see that the indicator I_h^{lim} helps to prevent flattening and creation of cube-like patterns in the approximate solution in some sub-regions observed with the error indicator I_h^{cross} and does not introduce any significant smearing.

If we compare the quality of actual approximate solutions in Fig. 6.18, and



(a) τ_h after optimization using I_h^{cross}. **(b)** τ_h after optimization using I_h^{lim}.

Figure 6.19: Comparison of the values of computed parameters τ_h for Example 6 after the optimization procedure using the indicators I_h^{cross} and I_h^{lim}, respectively.



(a) Example 5, outflow profile at $x = 0$. **(b)** Example 6, outflow profile at $x = 0$.

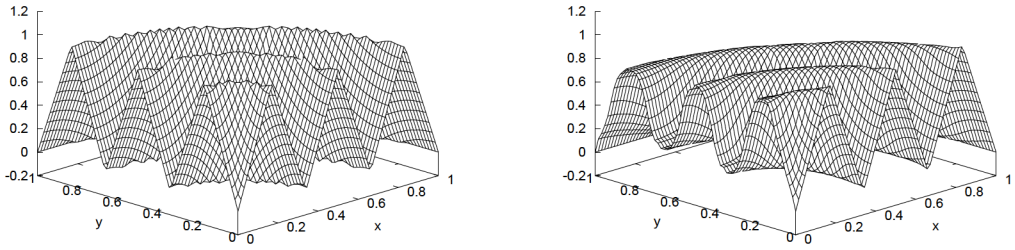
Figure 6.20: Outflow profiles of the optimized solutions obtained using all three indicators, comparison with the interpolation of the exact solution (`ind`/`indL2` stands for optimization using $I_h^{L^2}$, `indCross` stands for I_h^{cross}, and `indLim` stands for I_h^{lim}). The value of t_0 for Example 5 is 0.03, for Example 6 the value is 1.0 in this case.

move on to see values of parameter τ_h in the whole domain (Figure 6.19), we can say that the lower values of τ_h produced by I_h^{lim} can lead to a more satisfactory (physical) approximate solution.

6.6 Indicator with reduced residuals and SOLD method

In this section we present numerical results obtained using the above-described approaches for Examples 5 and 6. In Example 5 interior layers are present.

The numerical results presented in this section were computed using FEniCS Alnaes et al. [2012] software. In contrast to some of our papers, we considered only piecewise polynomial spaces of degree one for the discrete solution. Based on our former observations in numerical experiments, the set Y_h for the parameter τ_h and also $\tilde{\varepsilon}_h$ from Subsection 2.4.2 was defined using piecewise constant functions. If not said otherwise, the results were computed on structured meshes of Friedrichs–Keller type. We used Indicator I_h^{lim} (4.10) with $t_0 = 0.02$.

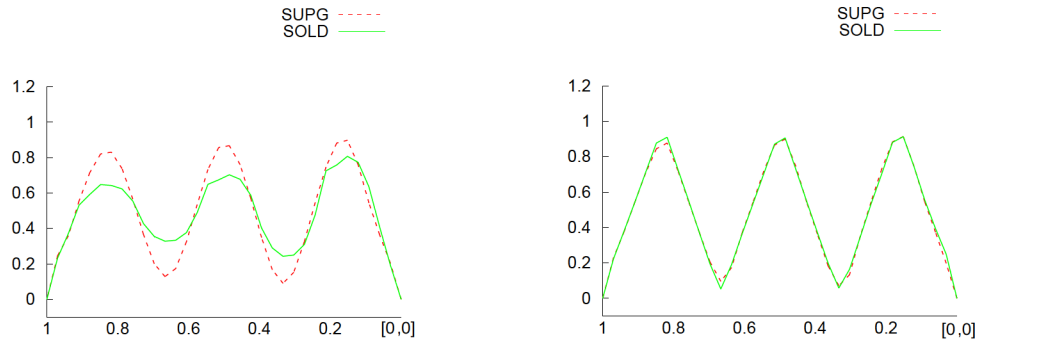


(a) Exact solution.

(b) Non-optimized SOLD method.

Figure 6.21: Exact and approximate solutions of Example 6.

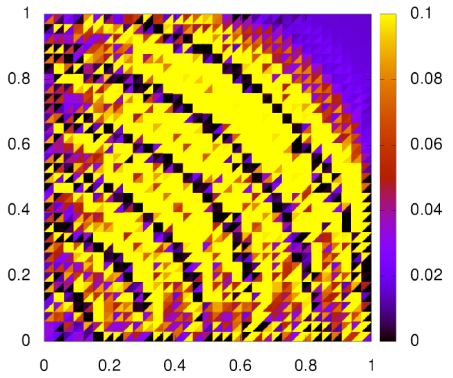
It turns out that preventing smearing in the numerical solution of the considered Example 6 is challenging for most SOLD methods. This is illustrated by Fig. 6.21b where the discrete solution obtained using the SOLD method from Section 2.4.2 with $\eta = 0.7$ is shown. In this case, a Friedrichs–Keller mesh with 34 nodes in each direction was used so that the Péclet number is of the order 10^6. The corresponding outflow profile is shown in Fig. 6.22a and compared with the outflow profile of the solution of the SUPG method with τ_h defined by (2.19). It can be observed that the solution of the SOLD method is significantly more smeared than the SUPG solution. Fig. 6.22b shows the outflow profiles after parameter optimization. The solutions of both methods are now comparable and significantly better than for non-optimized stabilization parameters.



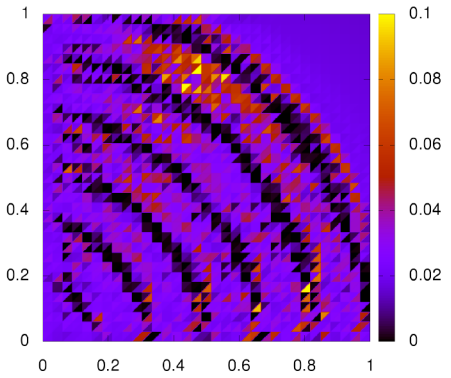
(a) Outflow profile at $x = 0$ for SUPG and SOLD methods.

(b) Profile after optimization of SUPG and SOLD methods.

Figure 6.22: Outflow profiles of non-optimized *(left)* and optimized *(right)* approximate solutions of Example 6.

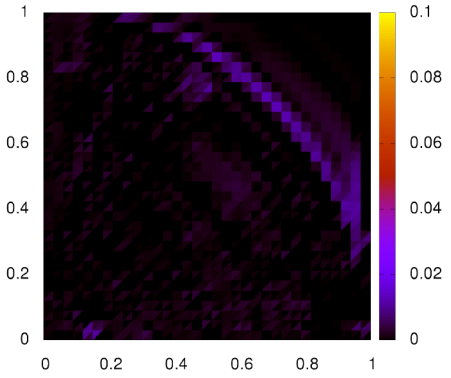


(a) SUPG: τ_h after optimization

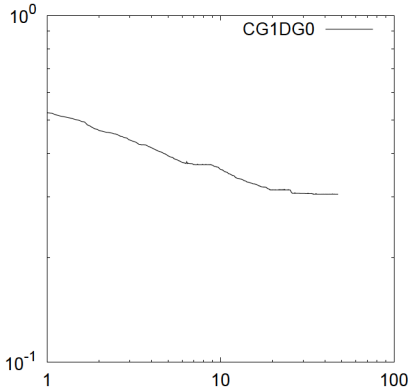


(b) SOLD: τ_h after optimization

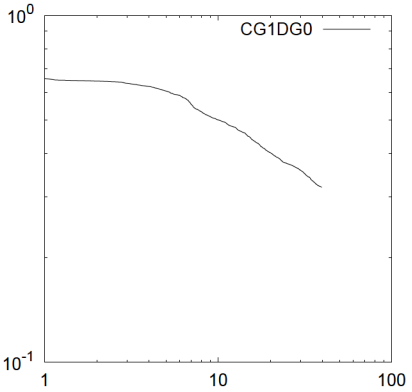
Figure 6.23: Values of free parameters after the optimization according to the Indicator (4.10). For SUPG method we optimize the τ_h parameter and for SOLD method we optimize both τ and $\tilde{\varepsilon}_h$ parameters. We bound both parameter values by $10 \cdot \tau$ and $10 \cdot \tilde{\varepsilon}$, respectively, where τ is defined by (2.19) and $\tilde{\varepsilon}$ is defined in (2.35). In (d) and (e) we show how the L-BFGS minimization proceeds. On the horizontal axis there is computational time and on the vertical axis there is value of Ind. (4.10).



(c) SOLD: $\tilde{\varepsilon}_h$ after optimization

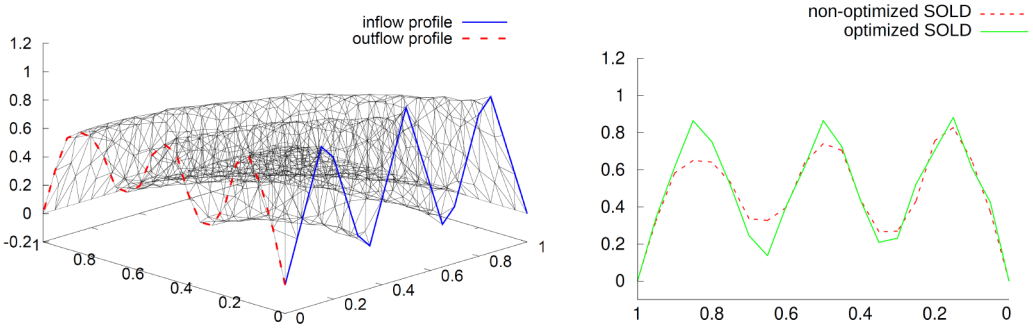


(d) SUPG: Minimization procedure



(e) SOLD: Minimization procedure

We performed analogous computations also on both coarser and finer meshes of Friedrichs–Keller type leading to similar observations. On finer meshes, the results for methods with optimized parameters were more accurate and the difference to the non-optimized case was even larger than shown above. On coarser meshes, the differences were not so significant but the non-optimized SOLD solution was always clearly worse than the optimized one. This was also observed for unstructured meshes, see Fig. 6.24 for results obtained on a mesh with 1054 elements.



(a) Non-optimized SOLD method.

(b) Outflow profiles at $x = 0$.

Figure 6.24: Approximate solutions of Example 6 computed on an unstructured mesh: non-optimized SOLD method *(left)* and comparison of its outflow profile with the optimized SOLD method *(right)*.

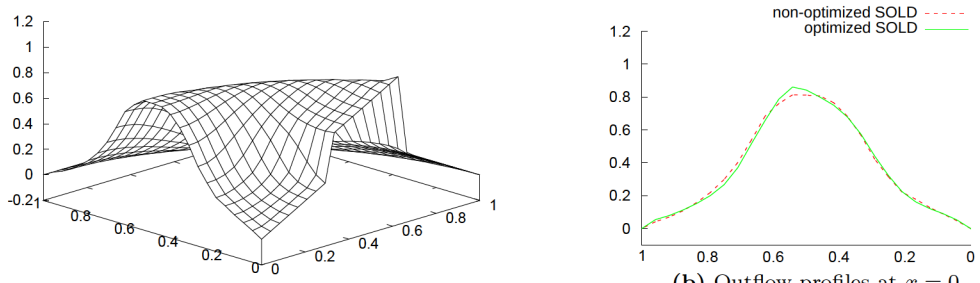
We also tested the methods for other values of the diffusion parameter ε and observed analogous properties of the methods as for $\varepsilon = 10^{-8}$. Of course, for large values of ε, the outflow profile of the exact solution is smeared and hence the differences among the various approximate solutions are less pronounced. In all these computations it was also observed that a parameter optimization for a SOLD method led to a lower value of the error indicator than a parameter optimization in the SUPG method.

Our results show that in cases, when the application of a SOLD method is needed to suppress spurious oscillations in layer regions, the parameter optimization is able to eliminate the negative influence of the SOLD term in regions where the solution is varying but no layers occur.

The parameters τ_h and $\tilde{\varepsilon}_h$ were optimized by the L-BFGS-B nonlinear minimization method from Chapter 5 using the default setup from `scipy` library with `gtol: 1e-14` and `ftol: 1e-14`. We restrict the number of iterations to 500 but this restriction has virtually no effect as the method uses less iterations before reaching a minimum. Like in our other computations, the SUPG parameter τ_h was initialized by (2.19) and the SOLD parameter $\tilde{\varepsilon}_h$ by 0. Other ways to initialize the SOLD parameter led to worse results of the optimization process. Bounds on both parameters were given as:

- $\tau_h : 0 < \tau_h < 10 * \tau$, where τ is the value from (2.19)
- $\tilde{\varepsilon}_h : 0 < \tilde{\varepsilon}_h < 10 * \tilde{\varepsilon}$, where $\tilde{\varepsilon}$ is the value from (2.35)

By comparing values of the parameter τ_h for SUPG and SOLD methods in Figure 6.23 we see that significantly lower values of τ_h are used in case of the SOLD method. Of course, there is a second part in case of SOLD, the $\tilde{\varepsilon}_h$ parameter



(a) Non-optimized SOLD method.

(b) Outflow profiles at $x = 0$.

Figure 6.25: Approximate solutions of Example 6 computed on a Friedrichs–Keller mesh with 25 nodes in each direction: non-optimized SOLD method *(left)* and comparison of its outflow profile with the optimized SOLD method *(right)*.

that needs to be taken into account. The calculated values of $\tilde{\varepsilon}_h$ are much lower than those of the SUPG part. We can interpret the lower values of parameters as a lower impact on the original formulation of the discrete problem. It is an illustration of how optimization helps in the SOLD method which may remove spurious oscillations but produces a smeared solution without the optimization.

We obtained slightly better minima of the error indicator for the SOLD method in our tests in comparison with the SUPG method. This lower minimum is naturally always obtained if we use 0 as the starting value for the SOLD parameter (on all elements). For presented Example 6 the obtained minima of I_h^{lim} from (4.10) were within 5 % of each other (SUPG-only and SOLD) and the domain clearly contains many different local minima. To be more specific, the highest difference in the value of I_h^{lim} between results computed by SUPG and SOLD optimization was in case of the unstructured mesh in Example 6 where the SOLD method led to 10% lower value of I_h^{lim} than the SUPG method.

In earlier papers we have already shown examples where a minimum which is further from the SUPG solution (but has a lower value of an indicator) can be a non-physical one. This solution is then practically worse than a solution which is closer to the SUPG solution, see, e.g., Lukáš [2015]. Bounds on the parameters are crucial and need to be set carefully.

Both examples were tested in Knobloch et al. [2019], for a slightly different value of $\varepsilon = 10^{-8}$, the SUPG method, and various error indicators. It turns out that the SUPG method optimized using I_h^{lim} already successfully removes spurious oscillations without smearing the layers and that the optimized SOLD method leads to very similar results. Since we want to develop further the results of Knobloch et al. [2019] here, we consider a different setting: $\varepsilon = 10^{-4}$ instead of $\varepsilon = 10^{-8}$. We use also a coarser mesh than in Knobloch et al. [2019]. In this case, the solutions are more smeared but Figure 6.25 shows that the optimized SOLD method again leads to better results than the non-optimized one.

We tested also residual-based error indicator $I_h^{L^2}$ from (4.1) in Knobloch et al. [2019]. For uniform meshes, it is equivalent to I_h^{lim} with $\psi(t) = t$. This indicator does not prevent smearing and is not considered as a suitable one for neither Example 5, nor Example 6, one can see the solutions in Figure 6.18.

6.7 Isotropic diffusion term and higher order FE spaces

As the stability of SOLD methods adding isotropic diffusion is high we were able to employ the SOLD method form Section 2.4, particularly (2.29) to get results for even higher-order FE spaces where a lot of instabilities usually appear during the optimization procedure. We not only tested higher order spaces for the discrete solution itself but also for the space from which the free parameters are chosen. In the following we present results for Example 3.

In our tests the SOLD method adding isotropic diffusion (2.29) has discrete solutions for higher order finite element spaces very similar qualitatively to the interpolations of nodally exact ones depicted in Figure 6.3. Also, the reason why we incorporated this Section is rather to show how the values of optimized parameter τ_h and SOLD parameter σ_h look like. The graphs of the optimized σ_h parameter from the SOLD method with isotropic diffusion are compared to the values of parameter τ_h, too.

Particularly, we provide in Figure 6.26 the table with all the parameter value maps of τ_h and σ_h after optimization according to Indicator with crosswind derivative control term (4.3) (4.3). Discrete solution u_h, under which all the important parameter optimizations ran, is always from `CG3`. The same spaces are always used for both τ_h and σ_h parameters, the results provided here are for finite element spaces `DG0 - DG3`.

We see how higher-degree FE spaces influence the size and shape of regions with higher value of optimized σ_h, we also see the results for different FE spaces used for the parameters τ_h and σ_h. The starting value to compute σ_h in this case is given by the equation (2.28). We set the bound on the values of both parameters to be tenfold the value of τ_h and σ_h defined in Section 2.4.1.

One can observe a problem that typically appears for a choice of a higher order FE space for the free parameters in the last image (h) of Figure 6.26. The problem lies in the fact that number of degrees of freedom is too high for the optimization procedure to choose a physically meaningful minimum, for other SOLD methods we encountered this instability in lower dimension spaces already. You see that some values of the parameter σ_h are too large in comparison with the predecessors. This problematic behavior appears especially in places with steep layers or just near to such places. Usually this means near the region with prescribed inflow boundary conditions.

As the optimized discrete solutions in these test cases are always similar to a nodally exact solution, which means almost perfect, we can tell from the computed values of free parameters in Figure 6.26 how an optimal choice of a free parameter should look, based, e.g., on the position of steep layers and other artifacts. We can further inspect this in the future. Parameters like, e.g., vicinity of a layer, distance from an inflow boundary condition, etc., may easily be defined, incorporated into a future theoretical research, and then their influence can be examined.

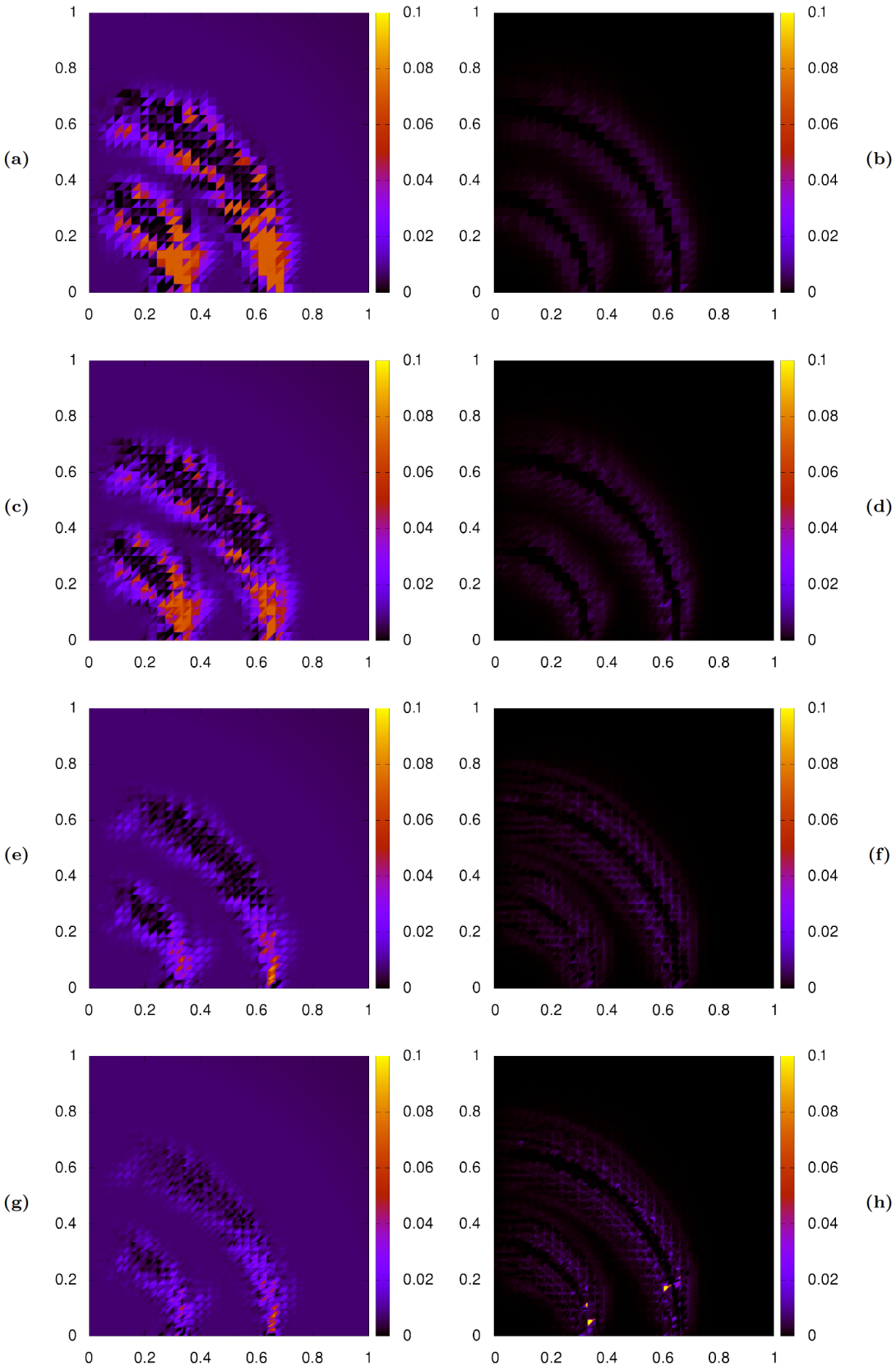


Figure 6.26: SOLD method with isotropic diffusion term (2.29), values of τ_h and σ_h after optimization according to Indicator with crosswind derivative control term (4.3). Discr. sol. u_h is always from `CG3`. The same space is always used for τ_h and σ_h, `DG0` - `DG3` from top to bottom, τ_h on the left (**a, c, e, g**), σ_h on the right (**b, d, f, h**).

Conclusion

We considered the standard scalar convection–diffusion–reaction problem (1.1) in the thesis. A very important aspect of the numerical solution of (1.1) are spurious oscillations that appear in the discrete solution when convection dominates diffusion and standard non-adaptive discretizations are used. Various stabilized methods have been proposed in the literature which often depend on free parameters. Optimal choice of these parameters is usually not known. Theoretical bounds for the values of these parameters are derived in most cases. In this thesis we introduced and explored also other stabilized methods than only the streamline upwind/Petrov–Galerkin (SUPG) method.

Among the goals we have accomplished were: The introduction of the L-SR1 method and comparison of methods of minimizing functions with large number of variables, the introduction and comparison of adaptive methods which are based on minimizing error indicators, the introduction and testing of new types of error indicators, and study of higher-order degree finite element spaces for discrete solution and free parameters. We also provided many interesting insights, including experimental convergence rates for specially chosen meshes.

Although a nodally exact solution of the convection–diffusion–reaction equation (1.1) is piecewise flat and the interior layer follows a smooth curve, the oscillations in the SUPG solution still appear along the sharp layers of the solution of (1.1) if we use the standard SUPG stabilized method (2.5).

The optimization method chooses the stabilization parameter τ_h so that it minimizes an error indicator. It is natural that the parameter itself is not smooth anymore as it can change even inside one element quite rapidly. To get an insight into this behaviour is not easy as the value of the parameter τ_h is a product of the process of a nonlinear optimization. On the other hand, we observed in Figures 6.13 and 6.14 that higher values of optimized parameter τ_h appear where the oscillations initially appeared, which means in the vicinity of sharp layers. We observed this fact holds for higher-order finite elements in Figure 6.26.

We introduced a new error indicator (4.10) for optimizing parameters in stabilization methods for the numerical solution of convection-dominated problems. Numerical results showed that this new indicator behaves better than indicators applied so far when, in regions away from layers, the exact solution is not constant in the crosswind direction. In addition, in contrast with the indicator adding crosswind derivative control term (4.3), the new indicator is consistent with the approximated problem. We tested this new Indicator (4.10) on both anisotropic and isotropic meshes, the results were presented especially in Knobloch et al. [2021].

We showed in Section 3.1 how to effectively minimize a general function with many variables, particularly how to compute the gradient of such a function effectively. This is an important algebraic part as all nonlinear minimization methods work with the gradient and its evaluation is the most time-consuming part of these algorithms.

We introduced the L-SR1 method and tuned its constants, most of which play a role in an SR1 method. The most important constants in L-SR1 method are the upper safety bound for parameter y_h and the initial trust region radius Δ_0.

We also explored a relation between these constants. We presented the constant κ defined in (5.32).

As we can see in Algorithm 10, the L-SR1 method now uses the limited-memory Hessian approximation only to compute the optimal step length. A better convergence speed of the L-SR1 method might theoretically be obtained if we used B_k not only to compute the step length but also to identify a better search direction p_k.

We then used the nonlinear minimization methods, including the steepest descent method, the nonlinear conjugate gradient method, the L-BFGS method, and the L-SR1 method in our computations. We showed the process of minimization in the graph in Figure 6.11. The L-BFGS method behaved best in total. However, the L-SR1 method was better in 12 of 32 test cases. All results are in Table 6.2.

From the investigated examples we found out that all error indicators (4.1), (4.3), and (4.10) allow us to obtain a high-quality solution of the scalar convection-diffusion equation with dominant convection even on a relatively coarse mesh. The reason that it works is that the spurious oscillations in the crosswind direction remaining from the SUPG method (2.5) are cured by employing an adaptive method based on an error indicator.

The indicator with the best results according to our tests was the indicator with the crosswind derivative control term (4.3) for examples with piecewise constant function as the boundary condition, and the indicator (4.10) for a broader range of examples with non-piecewise constant data. The crosswind derivative control term favours a steep layer with no spurious oscillations over more gentle layers or oscillations.

One of the other crucial parts of parameter optimization is the proper choice of the upper bounds for the free parameters. For the parameter τ_h this is proposed in (5.30). Whether it is theoretically possible to find an optimal upper bound dependent on a given example, which would guarantee a "correct" result, is not known. We have already seen an example of improper (too large) upper safety bound for τ_h in Figure 6.8, where the L-SR1 method found a narrow local minimum which resulted in a nonphysical solution.

This is an important issue in general, all minimization methods suffer more or less from finding a local nonphysical minimum. The initial choice of the parameter τ_h is "correct" in the sense that it is the parameter from the SUPG method (2.5), for which we have theoretical results. When a too large upper bound for τ_h is applied, a solution with a lower value of the error indicator I_h can be worse from the physical point of view than the solution of the SUPG method with the parameter defined by 2.19. Such a nonphysical solution found by the L-BFGS method is depicted in Figure 6.8, where we can compare it with the solution with a lower upper bound set for τ_h.

From the numerical tests we have done so far it comes out that using higher order finite elements or higher order discontinuous finite element spaces for the parameter τ_h has almost no positive effect on the resulting discrete solution of the proposed adaptive technique. By a higher order finite element is meant an element with the polynomial degree higher than 3. This holds also for all indicators we have tested in our adaptive framework. From the actual computed values of free parameters in Figure 6.26 we see how an optimal choice of the values of free

parameters should be.

It seems to be to useful to consider the adaptive method together with a carefully chosen mesh. In such a setup the adaptive method can give us satisfactory results since the rate of convergence of the SUPG method can be improved by the order of 1.

We can tell from the computed values of free parameters, e.g. in Figure 6.26 how an optimal choice of a free parameter should be, based, e.g., on the position of steep layers and other artifacts. We suggest to further inspect this in the future. Parameters like, e.g., vicinity of a layer, distance from an inflow boundary condition, etc., may easily be defined, considered into a future theoretical research, and then their influence can be examined.

Further theoretical research needs to be done to explain the gain of order one which we observed by using a mesh which is aligned with the layers in a specific way, one can see the results in Section 6.4.

We believe that the global optimization of free parameters can be beneficial in different ways. As Godunov's theorem (Godunov [1959]) holds, nonlinear methods are important and used to get a higher-order method with a higher accuracy. This natural need for nonlinear methods means that the models we are solving are often already nonlinear. So the computational overhead of using a global optimization, in general, may be relatively less important.

Bibliography

M.S. Alnaes, J. Blechta, J. Hake, A. Johansson, B. Kehlet, A. Logg, C. Richardson, J. Ring, M.E. Rognes, and G.N. Wells. The FEniCS Project Version 1.5, Automated Solution of Differential Equations by the Finite Element Method. *Archive of Numerical Software*, 3, 2012.

I. Babuška and B.Q. Guo. The h, p and h-p version of the finite element method: basis theory and applications. *Advances in Engineering Software, Issue 3-4*, 15, 1992.

I. Babuška, B.A. Szabó, and I.N. Katz. The p-version of the finite element method. *Journal on Numerical Analysis*, 18:515–545, 1981.

Alexander N. Brooks and Thomas J. R. Hughes. Streamline upwind/Petrov-Galerkin formulations for convection dominated flows with particular emphasis on the incompressible Navier-Stokes equations. *Comput. Methods Appl. Mech. Engrg.*, 32(1-3):199–259, 1982. ISSN 0045-7825.

Erik Burman and Peter Hansbo. Edge stabilization for Galerkin approximations of convection-diffusion-reaction problems. *Comput. Methods Appl. Mech. Engrg.*, 193(15-16):1437–1453, 2004. ISSN 0045-7825.

P.G. Ciarlet. *The Finite Element Method for Elliptic Problems*. North-Holland, NY, 1978.

T. A. Davis. *Direct Methods for Sparse Linear Systems*. SIAM, Philadelphia, 2006.

V. Dolejší. Anisotropic mesh adaptation for finite volume and finite element methods on triangular meshes. *Computing and Visualisation in Science*, 1: 165–178, 1998.

L. Evans. *Partial Differential Equations*. Graduate studies in mathematics. American Mathematical Society, 1998.

R. Fletcher. *Practical Methods of Optimization (2nd ed.)*. John Wiley & Sons, New York, 1987. ISBN 978-0-471-91547-8.

R. Fletcher and C.M. Reeves. Function minimization by conjugate gradients. *The Computer Journal*, 7:149–154, 1964.

L.P. Franca, S.R. Frey, and T.J.R. Hughes. Stabilized finite element methods: I. application to the advective-diffusive model. *Comput. Methods Appl. Mech. Engrg.*, 95:253–276, 1992.

Sergei Konstantinovich Godunov. A difference method for numerical calculation of discontinuous solutions of the equations of hydrodynamics. *Mat. Sb. (N.S.)*, 47 (89):271–306, 1959.

C. Grossmann, H.-G. Roos, and M. Stynes. *Numerical treatment of partial differential equations*. Springer-Verlag Berlin, Heidelberg, 2007.

T.J.R. Hughes, M. Mallet, and A. Mizukami. A new finite element formulation for computational fluid dynamics: Ii. Beyond SUPG. *Comput. Methods Appl. Mech. Engrg.*, 54:341–355, 1986.

T.J.R. Hughes, L.P. Franca, and G.M. Hulbert. A new finite element formulation for computational fluid dynamics: Viii the Galerkin/least-squares method for advective-diffusive equations. *Comput. Methods Appl. Mech. Engrg.*, 73:173–189, 1989.

V. John, P. Knobloch, and J. Novo. Finite elements for scalar convection-dominated equations and incompressible flow problems: a never ending story? *Computing and Visualization in Science*, 19:47–63, 2018. doi: 10.1007/s00791-018-0290-5. URL `https://doi.org/10.1007/s00791-018-0290-5`.

Volker John and Petr Knobloch. On spurious oscillations at layers diminishing (SOLD) methods for convection-diffusion equations. I. A review. *Comput. Methods Appl. Mech. Engrg.*, 196(17-20):2197–2215, 2007. ISSN 0045-7825.

Volker John and Petr Knobloch. On spurious oscillations at layers diminishing (SOLD) methods for convection-diffusion equations. II. Analysis for P_1 and Q_1 finite elements. *Comput. Methods Appl. Mech. Engrg.*, 197(21-24):1997–2014, 2008. ISSN 0045-7825.

Volker John and Petr Knobloch. Adaptive computation of parameters in stabilized methods for convection-diffusion problems. *Computer Methods in Applied Mechanics and Engineering 2011*, pages 275–283, 2013.

Volker John, Petr Knobloch, and Simona B. Savescu. A posteriori optimization of parameters in stabilized methods for convection–diffusion problems – part i. *Comput. Methods Appl. Mech. Engrg.*, 200:2916–2929, 2011. ISSN 0045-7825. doi: 10.1016/j.cma.2011.04.016.

P. Knobloch, P. Lukáš, and P. Solin. On error indicators for optimizing parameters in stabilized methods. *Adv. Comput. Math.*, pages 24–30, 2019. doi: 10.1007/s10444-019-09662-4.

P. Knobloch, P. Lukáš, and P. Solin. Importance of parameter optimization in a nonlinear stabilized method adding a crosswind diffusion. *Journal of Computational and Applied Mathematics*, 393:—, 2021. doi: 10.1016/j.cam.2021.113527.

Petr Knobloch. On the choice of the supg parameter at outflow boundary layers. *Adv. Comput. Math.*, 31:369–389, 2009.

T. Knopp, G. Lube, and G. Rapin. Stabilized finite element methods with shock capturing for advection-diffusion problems. *Comput. Methods Appl. Mech. Engrg.*, 191(27-28):2997–3013, 2002. ISSN 0045-7825.

P. Lukáš. Adaptive Choice of Parameters in Stabilization Methods for Convection-Diffusion Equations. *WDS 2012*, pages 54–59, 2012.

P. Lukáš. Optimization of parameters in SDFEM for different spaces of parameters. *Appl. Math. and Comp.*, 267:711–715, 2015.

P. Lukáš and P. Knobloch. Adaptive techniques in SOLD methods. *Appl. Math. and Comp.*, 319:24–30, 2018.

Akira Mizukami and Thomas J. R. Hughes. A Petrov-Galerkin finite element method for convection-dominated flows: an accurate upwinding technique for satisfying the maximum principle. *Comput. Methods Appl. Mech. Engrg.*, 50 (2):181–193, 1985. ISSN 0045-7825. doi: 10.1016/0045-7825(85)90089-1. URL `http://dx.doi.org/10.1016/0045-7825(85)90089-1`.

J.J. Moré. The Levenberg–Marquardt algorithm: implementation and theory. *Numerical analysis*, 630:105–116, 1978.

J. Nocedal and S.J. Wright. *Numerical Optimization*. Springer, NY, 2006.

Hans-Görg Roos, Martin Stynes, and Lutz Tobiska. *Robust numerical methods for singularly perturbed differential equations. Convection-diffusion-reaction and flow problems*, volume 24 of *Springer Series in Computational Mathematics*. Springer-Verlag, Berlin, second edition, 2008. ISBN 978-3-540-34466-7.

P. Solin, K. Segeth, and I. Dolezel. *Higher-Order Finite Element Methods*. Chapman & Hall/CRC Press, 2003.

T.E. Tezduyar and Y.J. Park. Discontinuity-capturing finite element formulations for nonlinear convection-diffusion-reaction equations. *Comput. Methods Appl. Mech. Engrg.*, 59:307–325, 1986.

List of Figures

List of Tables

List of Abbreviations

- BFGS - Broyden–Fletcher–Goldfarb–Shanno algorithm for nonlinear optimization
- CG - Conjugate gradient method
- CG1–CG3 - Continuous Galerkin FE spaces, from piecewise linear to polynomials of degree 3
- DG0–DG3 - Discontinuous Galerkin FE spaces, from piecewise constant to polynomials of degree 3
- DOF - Degree of freedom, usually used in plural
- FE - Finite Element
- FEniCS - FE representing finite element, CS representing computational software, a software used in this Thesis (fenicsproject.org)
- GLSFEM - Galerkin least squares finite element method
- L-BFGS - limited-memory versions of BFGS or SR1 gradient methods
- SDFEM - Streamline-diffusion finite element method
- SOLD - Spurious oscillations at layers diminishing methods
- SR1 - Symmetric Rank 1 method is a quasi-Newton method for nonlinear optimization
- SUPG - Streamline upwind Petrov-Galerkin method

List of publications

- **P. Lukáš**, Adaptive Choice of Parameters in Stabilization Methods for Convection-Diffusion Equations, *Week of Doctoral Students 2012* (2012), 54–59, ISBN = 978-80-7378-224-5.
- **P. Lukáš**, Optimization of parameters in SDFEM for different spaces of parameters, *Applied Mathematics and Computation* 267 (2015), 711–715.
- **P. Lukáš**, A Posteriori Optimization of Parameters in the SUPG Method for Higher Degree FE Spaces, *Lecture Notes in Computational Science and Engineering* 108 (2015), 171–182.
- **P. Lukáš, P. Knobloch**, Adaptive techniques in SOLD methods, *Applied Mathematics and Computation* 319 (2018), 24–30.
- **P. Knobloch, P. Lukáš, P. Solin**, On error indicators for optimizing parameters in stabilized methods, *Advances in Computational Mathematics* (2019), 24–30, doi = 10.1007/s10444-019-09662-4.
- **P. Knobloch, P. Lukáš, P. Solin**, Importance of parameter optimization in a nonlinear stabilized method adding a crosswind diffusion, *Journal of Computational and Applied Mathematics 393* (2021), 113527, doi = 10.1007/s10444-019-09662-4.