Interior-Point Methods for PDE-Constrained Optimization

Inauguraldissertation

zur Erlangung der Würde eines Doktors der Philosophie
vorgelegt der Philosophisch-Naturwissenschaftlichen Fakultät
der Universität Basel

von

Johannes Huber

aus Oppenau, Baden-Württemberg, Deutschland

Basel, 2013

Originaldokument gespeichert auf dem Dokumentenserver der Universität Basel: **edoc.unibas.ch**.

Dieses Werk ist unter dem Vertrag "Creative Commons Namensnennung–Keine kommerzielle Nutzung–Keine Bearbeitung 2.5 Schweiz" lizenziert. Die vollständige Lizenz kann unter
`http://creativecommons.org/licences/by-nc-nd/2.5/ch`
eingesehen werden.

Attribution – NonCommercial – NoDerivs 2.5 Switzerland

You are free:

to Share — to copy, distribute and transmit the work

Under the following conditions:

Attribution — You must attribute the work in the manner specified by the author or licensor (but not in any way that suggests that they endorse you or your use of the work).

Noncommercial — You may not use this work for commercial purposes.

No Derivative Works —You may not alter, transform, or build upon this work.

With the understanding that:

Waiver — Any of the above conditions can be *waived* if you get permission from the copyright holder.

Public Domain — Where the work or any of its elements is in the *public domain* under applicable law, that status is in no way affected by the license.

Other Rights — In no way are any of the following rights affected by the license:

- Your fair dealing or *fair use* rights, or other applicable copyright exceptions and limitations;
- The author's *moral* rights;
- Rights other persons may have either in the work itself or in how the work is used, such as *publicity* or privacy rights.

Notice — For any reuse or distribution, you must make clear to others the license terms of this work. The best way to do this is with a link to the web page `http://creativecommons.org/licenses/by-nc-nd/2.5/ch`.

Disclaimer — The Commons Deed is not a license. It is simply a handy reference for understanding the Legal Code (the full license) – it is a human-readable expression of some of its key terms. Think of it as the user-friendly interface to the Legal Code beneath. This Deed itself has no legal value, and its contents do not appear in the actual license. Creative Commons is not a law firm and does not provide legal services. Distributing of, displaying of, or linking to this Commons Deed does not create an attorney-client relationship.

Genehmigt von der Philosophisch-Naturwissenschaftlichen Fakultät auf Antrag von

Prof. Dr. Marcus J. Grote
Prof. Dr. Olaf Schenk
Prof. Dr. Eldad Haber

Basel, den 23. April 2013

Prof. Dr. Jörg Schibler,
Dekan

Abstract

In applied sciences PDEs model an extensive variety of phenomena. Typically the final goal of simulations is a system which is optimal in a certain sense. For instance optimal control problems identify a control to steer a system towards a desired state. Inverse problems seek PDE parameters which are most consistent with measurements. In these optimization problems PDEs appear as equality constraints. PDE-constrained optimization problems are large-scale and often nonconvex. Their numerical solution leads to large ill-conditioned linear systems. In many practical problems inequality constraints implement technical limitations or prior knowledge.

In this thesis interior-point (IP) methods are considered to solve nonconvex large-scale PDE-constrained optimization problems with inequality constraints. To cope with enormous fill-in of direct linear solvers, inexact search directions are allowed in an inexact interior-point (IIP) method. This thesis builds upon the IIP method proposed in [24]. SMART tests cope with the lack of inertia information to control Hessian modification and also specify termination tests for the iterative linear solver.

The original IIP method needs to solve two sparse large-scale linear systems in each optimization step. This is improved to only a single linear system solution in most optimization steps. Within this improved IIP framework, two iterative linear solvers are evaluated: A general purpose algebraic multilevel incomplete $\boldsymbol{LDL}^\top$ preconditioned SQMR method is applied to PDE-constrained optimization problems for optimal server room cooling in three space dimensions and to compute an ambient temperature for optimal cooling. The results show robustness and efficiency of the IIP method when compared with the exact IP method.

These advantages are even more evident for a reduced-space preconditioned (RSP) GMRES solver which takes advantage of the linear system's structure. This RSP-IIP method is studied on the basis of distributed and boundary control problems originating from superconductivity and from two-dimensional and three-dimensional parameter estimation problems in groundwater model-

ing. The numerical results exhibit the improved efficiency especially for multiple PDE constraints.

An inverse medium problem for the Helmholtz equation with pointwise box constraints is solved by IP methods. The ill-posedness of the problem is explored numerically and different regularization strategies are compared. The impact of box constraints and the importance of Hessian modification on the optimization algorithm is demonstrated. A real world seismic imaging problem is solved successfully by the RSP-IIP method.

Contents

Contents i

1 Introduction **3**
1.1 Applications of PDE-Constrained Optimization 3
1.2 Numerical Methods for PDE-Constrained Optimization 6
1.2.1 Optimize-then-Discretize and Discretize-then-Optimize . 8
1.2.2 Reduced-Space Optimization 9
1.2.3 Unconstrained Optimization 9
1.2.4 Full-Space Optimization 12
1.3 Thesis Outline and Contribution 14

I Interior-Point Methods **17**

2 IP Framework and Exact Step Computation **19**
2.1 Interior-Point Framework 20
2.1.1 Optimality Conditions 20
2.1.2 Primal-Dual Interior-Point Algorithm 22
2.1.3 Barrier Parameter and Local Convergence 25
2.2 Exact Interior-Point Method 26
2.2.1 Computation of the Newton Direction 29
2.2.2 Filter Line-Search Methods 29
2.2.3 Second-Order Correction 30

3 Inexact Interior-Point Method **33**
3.1 SMART Tests for an Inexact Newton Method 34
3.2 SMART Test for a Stabilized Inexact Newton Method 36
3.3 Step Length Computation 40

II Interior-Point Methods for PDE-Constrained Optimization **43**

4 Problem Formulation and Inherent Structure **45**

4.1 Optimal Control and Parameter Estimation 46
4.2 Full-Space Approach and Reduced-Space Approach 47
4.2.1 Full-Space Approach 47
4.2.2 Reduced-Space Approach 50

5 Iterative Solution of the Newton System 55
5.1 Algebraic Multilevel Incomplete $LDL^\top$ Preconditioner 55
5.2 Reduced Space Preconditioned GMRES 57
5.2.1 Reduced-Space Preconditioning 60

6 Numerical Results 67
6.1 Software Implementation Aspects 67
6.2 Algebraic Multilevel Incomplete $LDL^\top$ Preconditioner 70
6.2.1 Standard Nonlinear Programming Test Sets 71
6.2.2 Optimal Boundary Control 72
6.2.3 Server Room Cooling 75
6.3 Numerical Results for RSP-IIP with Block Preconditioner 80
6.3.1 2D Distributed Control 82
6.3.2 2D Boundary Control Problem 84
6.3.3 2D and 3D Parameter Estimation 85
6.4 Conclusion 91

III Inverse Medium Problem for the Helmholtz Equation 95

7 Formulation of an Inverse Problem 97
7.1 Derivation of the Helmholtz Equation 97
7.2 Finite Element Discretization of the Helmholtz Equation 98
7.3 Formulation as PDE-Constrained Problem 101

8 Numerical Studies 105
8.1 Ill-Posedness and Regularization 105
8.2 Nonconvexity, Hessian Modification, and Frequency Continuation 111
8.3 Model Parametrization 115

9 Real World Problem in Seismic Imaging 117

IV Conclusions & Future Work 123

10 Conclusion and Future Work 125
10.1 Conclusion 125
10.2 Future Work and Research Directions 127

Bibliography 131

Acknowledgment

This thesis was written at the Department of Mathematics and Computer Science at the University of Basel. I'd like to thank my supervisors Prof. Dr. Marcus J. Grote and Prof. Dr. Olaf Schenk for giving me the opportunity to do my PhD with them and for their research guidance, their support, inspirations, advice, and confidence. I would also like to thank Prof. Dr. Eldad Haber for kindly agreeing to act as a coreferee in the thesis committee and also for his very nice lectures at the mini course on PDE-constrained optimization in May 2011.

I'm very grateful to Prof. Dr. Andreas Wächter and Prof. Dr. Frank E. Curtis for the many fruitful and clarifying discussions. I also wish to thank the people from the T.J. Watson research center for welcoming me as an intern. I thank Prof. Dr. Ahmed Sameh, Prof. Dr. Faisal Saied, and Alicia Klinvex for the fruitful ideas about Newton-Krylov-Methods.

In my multifaceted career I learned from many different teachers and I'm grateful to all of them. Special thanks go to Prof. Dr. Axel Donges, Prof. Dr. Dieter Röschenthaler, and Prof. Dr. Andreas Kirsch for their inspiring lectures.

My sincere thanks also go to the colleagues at the Department of Mathematics and Computer Science at the University of Basel Dr. Madan Sathe, Dr. Loredana Gaudio, Dr. Teodora Mitkova, Dr. Marie Kray, Dr. Imbo Sim, Michaela Mehlin, Uri Nahum, and at the Advanced Computing Laboratory in Computational Science at the University of Lugano Dr. Matthias Christen, Dr. Drosos Kourounis, and Max Rietmann. Special thanks to my office mate Christian Stohrer for the helpful discussions and the inspiring and open working atmosphere.

I'd also like to thank the administrative staff at the Department, especially to Barbara Fridez, Heidi Karypidis, Robert Frank, and Philipp Georg.

Many thanks go to Judith Streckfuss, Gary Davidoff, and Uri Nahum for proofreading this thesis.

I would like to thank Judith for her love, understanding, support, endless patience, and warm encouragement when it was most needed. Finally, I am forever indebted to my parents, Annemarie and Albert Huber.

This thesis was partly supported by the Swiss National Science Foundation.

Chapter 1

Introduction

This thesis discusses interior-point (IP) methods for PDE-constrained optimization. While PDEs are used to simulate certain effects, the ultimate goal of simulations are often to find parameters which are optimal in a certain sense. In the first chapter of this thesis, we start by presenting some applications of PDE-constraint optimization to show possible practical impacts of this work. We then turn our focus on numerical methods to solve these optimization problems and end this chapter with an outline of the thesis.

1.1 Applications of PDE-Constrained Optimization

PDEs are ubiquitous when modeling continuum problems. They can be found in almost every area of science and engineering and are used to simulate systems of varying sizes and complexity. Their solution usually depends on parameters which adapt the PDE to an actual problem at hand. Often those quantities are not the final goal, but an intermediate result to find parameters leading to a “good“ simulation result.

PDE-constrained optimization can be considered as an automation of parameter searching, where the many interactively-modify-and-try iterations are replaced by a systematic computational procedure. The goal is modeled as an objective function and we aim to find the PDE parameters which yield a minimum. Typically, the objective depends only on the PDE’s solution, the *state variable* y, which can be affected only indirectly by variation of the parameters u via the PDE which appears as an equality constraint. To concretize this abstract point of view we now give some example applications of PDE-constrained optimization. Depending on how the PDE parameters influences the state variable, we can distinguish different types of PDE-constrained optimization problems.

In *shape optimization* problems the parameters affect the simulation domain and the goal is to find an optimal construction which still fulfills structural stability constraints and minimizes the objective. For instance, in the design of an aircraft the shape of the wings can be modeled by design parameters like twist or sweep angles, or its shape can be described more generally by splines. Given such a description a simulation yields the a drag or buoyancy. With shape optimization we can now design a wing with maximal buoyancy or minimal drag [60]. Another example for shape optimization is the design of structural elements with minimal weight which sustain a given mechanical load [4]. In medicine, shape optimization is used to design blood pumps or arterial grafts. Here, the goal is to minimize the shear rate to circumvent red blood cell damage or thrombus formation at device surfaces [2, 1]. In many of these examples the governing PDEs are nonlinear Navier-Stokes equations and inequality constraints prevent physically meaningless parameter settings, like a local negative mass for a structural element or a too small wings sweep angle.

The typical setting of *optimal control* is to steer a system toward a desired state. Here, the PDE parameter, called *control variable* u affects the PDEs' right-hand side and can be varied to achieve PDE solutions (state variables y) as close as possible to desired states $\hat{y}$. This is applied, e.g., in the glass industry to control glass cooling. The aim here is to cool the glass according to a desired temperature profile dependent on space and time in order to prevent cracks or to control chemical reactions in the glass. The underlying physical process, heat radiation, is modeled by a parabolic nonlinear PDE and can be influenced by the ambient temperature which is the control variable [29]. Here inequality constraints may prevent too large temperature gradients which lead to cracks in the glass. Another application of optimal control is the design of semiconductor devices. Here a doping profile accounts for concentrations of impurity atoms and is changed in order to maximize the current flow over contact regions [41]. Inequality constraints can avoid concentrations where the underlying model might become invalid. In weather forecasting large-scale PDEs are solved starting from initial conditions. At the initial time step atmospheric quantities like pressure, temperature, and wind velocity are necessary at each simulation mesh point. However, measurements are not available at all those points. The computation of these initial conditions, called data assimilation, can be stated as PDE-constrained optimization problem, where we seek a state based on a previous forecast and measurements. Here the governing nonlinear PDEs describe the atmosphere physics and the objective consists of a misfit of measurements and previous weather forecasts [25]. The enforcement of a maximal misfit then naturally leads to inequality constraints. The location of an earthquake hypocenter can also be stated as an optimal control problem. The problem is then constrained by the wave equation and the source term on the PDEs right-hand side takes the role of a control variable.

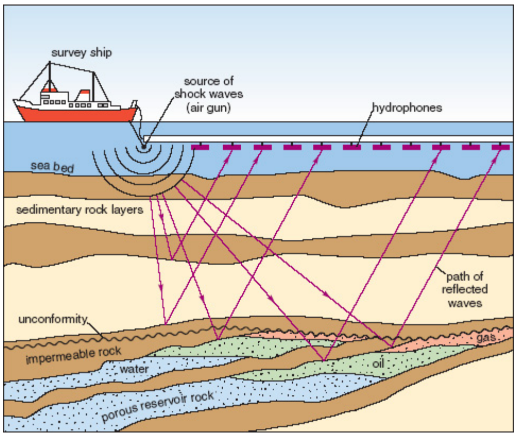


Figure 1.1: *Principle of seismic imaging for marine oil reservoir exploration: air guns towed behind a survey ship excite shock waves, which are reflected and scattered by the subsurface rock layers and recorded at hydrophones. With PDE-constrained optimization the seismic velocity of subsurface rock is reconstructed and helps geophysicists to locate potential oil reservoirs for exploration drills. Attribution to OpenLearn*[1].

In parameter estimation problems we want to solve an inverse problem. In this problem setting, we seek PDE parameters, (usually material parameters) which are most consistent with measurements. The parameters, called the *model*, affect the differential operator itself, and such problems are usually harder to solve than optimal control problems. Most methods in medical imaging are based on inverse problems. For instance, on electrical impedance tomography electrical current is supplied to the bodies' surface. This generates an electrical potential which depends in the conductivity inside the body. The location dependent conductivity is considered as a PDE parameter which is varied to minimize the misfit between measurement and simulation [50]. The same principal can be applied in DC resistivity estimation and groundwater modeling [38].

One of the most challenging applications of PDE-constrained optimization appears in oil reservoir exploration and we now discuss this application in more detail. As the era of "easy oil" has finished, in the recent decades it has become more and more difficult to find and produce oil. Today oil exploration is economically considered as an expensive and high-risk operation. An exploration oil drill can easily cost several million US dollars. Thus efficiently locating prospective oil reservoirs is of uttermost importance.

In the genesis of oil, it may migrate upwards until it accumulates beneath an impermeable sedimentary layer and gets trapped in certain geological formations. To identify these prospective oil reservoirs geophysicists survey the subsurface using seismic reflection experiments. Figure 1.1 shows a typical marine seismic imaging setup. A survey boat drags an air gun to excite pressure

---

[1] http://www.open.edu/openlearn

shock waves (“shots”). These waves propagate through water and sedimentary rock and are reflected and scattered at the stratigraphic boundaries. The reflected waves are then recorded at receivers (hydrophones) also towed by a survey boat.

During these experiments hundreds of terabytes of data may be collected and need to be processed. After a preprocessing step, numerical methods of seismic imaging including migration or inversion [19, 72] are applied. Due to the accurate physical model full waveform inversion is considered one of the most promising methods for high resolution imaging and material parameter quantification [71]. It consists of a PDE-constrained optimization problem, where the misfit is minimized subject to the constraining wave equation. To mitigate against false solutions of this nonconvex problem, filtering temporal frequencies is a common approach. This naturally leads to the frequency domain inversion governed by the Helmholtz equation instead of the wave equation [56]. Here different temporal frequencies are decoupled and can be solved separately, starting from low temporal frequencies with a coarser mesh but fewer local minima towards higher frequencies with more detailed features. In practice, low temporal frequencies are often not available [53]. This lack of information may be closed by the used of wide aperture data [67, 68, 69]. Again, inequality constraints can help to guide the algorithm and prevent physically unrealistic models.

Many of the above applications aim to minimize the distance of the achieved system to a desired or measured state. Often, with this goal the PDE parameters are not determined uniquely or the parameters do not depend continuously on the desired or measured data. Such ill-posed systems are highly susceptible to noise. To stabilize these methods, additional information is implemented in terms of restrictions on the search space or as an additional Tikhonov regularization term.

1.2 Numerical Methods for PDE-Constrained Optimization

In the previous section, we have seen how PDE-constrained optimization problems can arise in almost every area of science and engineering. In this section we survey numerical optimization algorithms to solve PDE-constrained optimization problems.

Many problems are nonconvex and may have multiple local solutions. Indeed, finding a global optimum in general is NP hard. Global optimization methods are classified into deterministic and stochastic ones [54, 26]. In deterministic methods error bounds are computed to exclude regions from the search space. Stochastic algorithms evaluate the function at points, which are found by heuristics and some random process. Both approaches may utilize local optimization algorithms [42]. In this thesis we concentrate on IP methods to find local solutions. They allow a search space restriction by inequality constraints, to improve stability of convergence to a global optimum.

Local numerical optimization algorithms are iterative procedures [55, 31, 32, 40]. Starting at an initial guess they generate a sequence of improved points, iterates, and aim to converge to a solution. To compute a new point, local information in terms of derivatives are used. In trust-region methods, a local model subproblem for the optimization step is solved in a neighborhood of the current iterate. In line-search methods, a model subproblem is solved to compute a search direction only. A line search algorithm then determines the step length to the next point.

There are three main important questions when considering numerical optimization algorithms.

1. Does the algorithm terminate? The first question considers the robustness of the algorithm, the global convergence properties. In this thesis we refer to *global convergence* as the guarantee to converge to a first-order critical point or a point of local infeasibility from each initial point. It should not be confused with guaranteed convergence to a global minimizer, which might be common in other scientific communities.

2. How many iterations does it need to find a solution? Far away from the solution global information is necessary for fast convergence. This is usually incorporated into the optimization problem formulation itself, e.g., in form of inequality constraints, and is usually not covered by numerical optimization algorithms. Close to the solution, however, derivative information can be used to fasten *local convergence*, i.e. to decrease the number of iterations to converge to a solution. This essentially depends on the approximation quality of the model subproblems which are solved in each iteration.

3. How expensive is each iteration? Most model subproblems are set up by a Taylor expansion. The approximation error can be reduced by taking more terms of this expansion. However, the computation of each step increases. Usually the Taylor approximation contains at most the quadratic term. Then a linear system needs to be solved. Indeed, in most modern numerical optimization algorithms, this is the most expensive step, and the question of fast solutions of the linear system appears.

When solving a PDE-constrained optimization problem, there are two possible approaches to follow [41, 39]. While in the optimize-then-discretize approach we first set up optimality conditions and then discretized the step computation operators, in the discretize-then-optimize approach we apply optimization algorithms to the discretized problem. This issue is discussed in the Subsection 1.2.1 first section of this chapter.

Another choice needs to be taken whether as to we want to optimize an intricate objective function on an easy manifold (often a box or even the whole space) or we solve for a simpler objective function but on an intricate manifold. The first choice is realized in the reduced-space approach while the latter is carried out in the full-space approach. Both are discussed in Section 1.2.2 and 1.2.4

1.2.1 Optimize-then-Discretize and Discretize-then-Optimize

PDE-constrained problems are infinite dimensional and to make them tractable by computers, they need to be discretized. There are two school considering when to discretize the appearing functions and operators. In the optimize-then-discretize approach, we first set up optimality conditions and optimization method in function space and discretize the algorithm afterwards. However, implemented derivatives many then not be the derivatives of implemented functions, which slows down local convergence.

On the other hand, we can discretize the PDE and objective function first, turning the infinite-dimensional optimization problem into a finite-dimensional one. Then, we set up the optimality conditions and optimization algorithm in the finite dimensional space. Now commonly used optimization methods can be applied. Furthermore, automatic differentiation techniques may generate source code to compute derivatives of the objective and constraint functions [43].

In general, both approaches do not lead to the same algorithm. For example, for a parameter estimation problem discretized on a staggered grid for state and control variable we find optimality conditions which are linear in the model parameter when following the optimize-then-discretize approach while the discretize-then-optimize approach leads to nonlinear optimality conditions [38]. Another difference can be observed when considering PDE constraints with initial conditions. In the optimality conditions, the adjoint operator appears. However, the discretized adjoint operator is not necessarily the transposed of the discretized operator. As an example we consider an initial value problem. After discretization we can think of a block vector containing one time in each block. The time stepping scheme can then be written as a matrix where the initial condition enters in the first row blocks. Thus they differ from the blocks corresponding to later time steps. When transposing this matrix, the initial condition block will stay in the first block. However, when we first set up the optimality condition, we find a final value problem as adjoint problem and the final condition block will appear in the last row block. From this example we see that the discretization of the adjoint operator may result in another matrix and thus in another method than the transpose of the discretized operator [52].

1.2.2 Reduced-Space Optimization

One characteristic of PDE-constrained optimization is the large number of variables. Indeed, nowadays simulations can have several millions or even billions of unknowns. These simulation variables appear in the optimization problem as state variables. Assuming unique solvability of the PDE, they are determined as a solution of the PDE for given parameters. Thus, the state variables can be eliminated from the optimization problem and only the parameters appear as optimization variables. The resulting optimization problem is formulated in the *reduced-space*.

The number of parameters varies in a wide range, starting from just a few ones up to the order of the state variable. For instance, in the shape optimization problem in 1.1, a wing can be modeled with a fixed number of only a few parameters. On the other extreme a parametrization of a wing using splines drastically increasing the number of parameters. Also in distributed control or distributed parameter estimation, we have a function as parameter which needs to be discretized. Then the number of optimization variables depends on the discretization of the parameter function, which may yield a parameter vector as large as the state variable.

1.2.3 Unconstrained Optimization

In the reduced-space we eliminate the state variables and the PDE constraints disappear. If there are no further constraints, the reduced-space problem is unconstrained and has fewer variables than the full-space problem. However, to evaluate the objective, forward simulations are necessary, which are expensive, especially when considering nonlinear PDEs, where the solution procedure incorporates a Newton-type method itself.

Optimization methods use local information, i.e., derivatives, to improve the current solution estimate. Trust-region methods need to control the size of the trust region to ensure global convergence. To reach the same goal, line search methods need to ensure that the objective function descends in the search direction and that the step length actually yields a sufficient objective function decrease. For the latter, step length algorithms are used including the Armijo condition, the Wolfe condition, and the strong Wolfe condition. Within these step length algorithms the objective function needs to be evaluated repeatedly. The number of function evaluations can be reduced by polynomial interpolation.

The basic optimization method is steepest descent, where the search direction is the negative gradient. Steepest descent possesses global convergence guaranties, but only slow linear local convergence. For gradient evaluation, the linearized PDE and its adjoint need to be solved.

Nonlinear conjugate gradient methods generalize the linear CG method and aim to increase the local convergence of steepest descent while still avoiding higher derivative evaluations. There are a number of nonlinear CG methods, and among them, Fletcher-Reeves and Polak-Ribière are the most important ones. While Fletcher- Reeves in combination with a strong Wolfe step length algorithm leads to descent direction and thereby achieves global convergence, Polak-Ribère needs to be safeguarded. Even though it has been shown that both methods with reinitialization and exact step length algorithm achieve n-step quadratical convergence [21] (i.e. if the algorithm is restarted after more than n steps, the residuals decrease quadratically each n steps), quadratic convergence in practice is not observed due to large n. However, the number of iterations needed by nonlinear CG algorithms is usually much smaller than with steepest descent [59].

To improve local convergence, the subproblem can be set up from a quadratic model. This leads to Newton's method which is the primary starting point for algorithm development in numerical optimization. The local convergence rate is here improved to quadratic, but at an increased cost of each iteration: Here, a linear system, the *Newton system* with the Hessian matrix needs to be solved in each optimization iteration. For optimization approaches in the reduced-space, this reduced Hessian matrix has the size of the PDE parameters and it is dense. Thus, it is usually not possible to form explicitly, not to mention its factorization.

Trust-regions methods define a region where the model is assumed to be reasonably accurate. Within this region, the quadratic model is minimized. Often, the trust region radius is adapted depending on the ratio of actual and predicted reduction of the objective function. In this thesis, trust region algorithms are applied to a feasibility problem, where the objective function is hopefully reduced to 0. There, the trust region radius is set to a multiple of the infeasibility, i.e., dependent on the objectives current iterate. Within trust-region methods, the subproblems are often solved inexactly. To rate the current approximation to a subproblem's solution, it is compared to the Cauchy point, that is a steepest descent step with exact step length. This rule plays a similar role as the conditions in the step length algorithms for line search methods. Indeed, it suffices to guarantee global convergence. One approach to solve the subproblems inexactly is the dogleg method. Here the solution of the subproblem is constructed on the polygon defined by the current iterate, the Cauchy point, and the solution of an unconstrained quadratic model problem.

Similarly, in Newton line search methods, a quadratic model is formed to compute a search direction. For nonconvex objective functions the quadratic model may represent a saddle instead of a bowl and the objective function might increase in the search direction. To overcome this difficulty, the Hessian matrix is modified yielding a descent direction. With this, a step length algorithm can determine the next iterate, thereby guaranteeing global convergence. In this step length algorithm, the objective function needs to be evaluated several times, which includes repeated assemblies and solutions of PDEs. Step length algorithms are designed to generate full Newton steps close to the solution.

With an exact solution of the Newton system, both Trust-region and line search methods inherit the fast local quadratic convergence from Newton's method. However, in the context of PDE-constrained reduced-space optimization, a large dense linear system needs to be solved. One possible approach to overcome the drawback of forming the dense Hessian is to approximate the Hessian by finite differences of the previous optimization steps. This led to the development of Quasi-Newton methods. Since in each optimization step knowledge about the gradients change in one direction only is contained, there are several methods to update a current Hessian estimate in order to reflect this new information. Among Quasi-Newton methods, BFGS and DFP are the most important ones. Remarkably, it is also possible not only to construct Hessian approximations but also approximations to its inverse. Thus, it is not necessary to solve the approximation to the Newton system, but only apply the current approximation to a vector. Furthermore, since the approximations are created by low rank updates in each iteration, it is not necessary to save the dense approximation matrix, but only the previous vectors. In combination with Wolfe or strong Wolfe step length algorithms, the search directions are descent directions and global convergence is achieved. Furthermore, superlinear convergence has been proved. However, these methods suffer from reduced Hessian approximation efficiency with increasing problem size.

In most PDE-constrained optimization problems, the objective function is quadratic and a Gauss-Newton approximation to the Hessian matrix is suitable. Here, in the objective function a first-order approximation is applied leading to a positive semidefinite Hessian. Thus, accent search directions cannot appear and Hessian modification is usually not performed. Since the approximate Hessian is also symmetric, the CG method can be applied to solve the linear systems. With appropriately controlled inexact solutions, the Gauss-Newton method achieves superlinear convergence rates.

Another possibility for tackling the drawbacks of Newton's method is to solve the Newton system inexactly via Krylov methods. These methods do not require to form or store the actual matrix but only its application to a vector is needed. The resulting optimization problems are also referred to Newton-Krylov methods. A prototype of this family is the Newton-CG algorithm by Steihaug [70]. Global convergence can be assured in a trust-region framework in combination with a step length algorithm and Hessian modification as described above. When the inexact solution of the Newton system is controlled suitably, Newton-Krylov methods converge superlinearly or even quadratically.

The most expensive part in inexact Newton methods is the iterative solution of the KKT system. For efficient iterative solvers preconditioning is the key component. Since the large-scale dense Hessian cannot be formed explicitly, incomplete factorization methods are out of scope. However, one can apply Quasi-Newton techniques like BFGS for preconditioning. There are two major drawbacks. First, Quasi-Newton methods often try to resemble positive definite matrices, but for nonconvex optimization problems positive definiteness of the reduced Hessian is not guaranteed. Second, Quasi-Newton approximations often work well only in certain directions but they do not approximate the whole matrix, as would be desirable for a preconditioner.

1.2.4 Full-Space Optimization

While, in the reduced-space approach, the PDE constraints are used to eliminate the state variables, in the full-space approach we keep the state variables and PDE constraints. This has several advantages:

- Derivatives can be evaluated exactly and stored explicitly. The linear systems, i.e., the constraints' Jacobian and the Lagrange's Hessian, are sparse and their exact evaluation circumvents additional errors from approximate evaluation.

- Inexact solutions of the PDEs are acceptable. Since we do not assume feasibility of the state variables, the algorithm may also visit infeasible points and gain feasibility and optimality simultaneously.

- The evaluation of the functional is very cheap. In contrast, in the line search of reduced-space methods a change of the control variable necessitates a reassembly and possibly a refactorization to solve PDEs for objective function evaluation.

This advantages come at the cost of many more optimization variables and appearance of equality constraints.

Equality constrained optimization

First-order optimal points for equality constrained problems are saddle points of the Lagrangian. To compute them, in each iteration a model subproblem is formed and solved.

In penalty methods these subproblems are unconstrained and the constraint violation is added as a penalty term to the objective function. The weight of the penalty term is then increased to infinity, which leads to highly ill-conditioned matrices when solving the unconstrained subproblems. In contrast, for exact penalty functions a finite penalty parameter already yields a solution of the constrained problem. However, exact penalty functions are nonsmooth or quite complicated and they are not well suited for large-scale optimization. A loophole for this situation are augmented Lagrangian methods, where a smooth penalty term is added to the Lagrangian. Then, a finite penalty parameter suffices. Since the subproblem of penalty methods is unconstrained, global convergence and the local convergence rate comply with global convergence of the subproblem solver and an appropriate estimation of the required penalty parameter.

In SQP methods the subproblem is generated by a quadratic approximation of the objective and linearized constraints. While for the unconstrained problems, global convergence can be achieved by a sufficient decrease in each step, the situation for constrained optimization is more involved. Here, we seek two goals: finding a feasible point and at the same time reducing the objective. Trust-region methods achieve global convergence by a suitable choice of the trust-region radius and a strategy to handle inconsistent subproblems in case the linearized constraints are not fulfilled in the trust region. Line search methods utilize filters or merit functions to cope with the two goals. Since the SQP method can be motivated by Newton's method, local quadratic convergence can be achieved. Like for unconstrained optimization problems, it is often practical to approximate Hessians by Quasi-Newton methods or inexact solutions. These approximations usually yield superlinear local convergence.

Inequality constraints

Inequalities are very useful in practical applications as they can be used to model technical limitations or to keep the algorithm in a basin of attraction of a good local minimum. Despite these attractive features, they constitute serious algorithmic problems. The underlying problem is to determine the set of active constraints, i.e., the set of inequality constraints which hold with an equality sign at the optimal point. This problem owns combinatorial complexity.

In reduced-space methods the PDE equality constraints are eliminated. Thus only constraints involving PDE parameters can be treated efficiently. Four types of methods are well established in this context.

In projection methods, search directions are first computed from an unconstrained subproblem and afterwards iterates and search directions are projected into the admissible set and feasible directions, respectively. For steepest descent algorithms, the projected search direction yields a descent direction. For other search directions like a Newton direction, even the projection of descent directions can yield an ascent direction. This can be corrected by varying the Newton system and still superlinear convergence can be achieved [10].

Active set SQP methods hold an estimate for the set of active inequality constraints. In each iteration they impose the active set as equality constraints and update this set based on the solution of the subproblem. Once the set of active inequalities has been identified correctly, these methods converge superlinearly or even quadratically. The mayor problem of active set SQP methods is to identify the active set, especially if there are many almost active inequalities. Active set SQP methods tend to change the active set by only a few single inequalities. By these updates, the number of equality constraints varies which often necessitates reinitialization of internal data structures.

In this thesis we consider IP methods to solve nonconvex PDE-constrained optimization methods with inequality constraints. These methods add a logarithmic barrier term to the objective and solve a series of equality constrained problems. The methods treated in this thesis have global convergence guaranties and converges superlinearly. They are discussed in more detail in part I.

1.3 Thesis Outline and Contribution

This thesis is organized in three parts. In part I interior-point methods are discussed and major convergence results are reviewed. Particularly, in Chapter 3 an IP algorithm which allows for inexact step computations is covered. In Part II these IP methods are applied to PDE-constrained optimization algorithms. First, a general structure induced by the distinction of state variables and PDE parameters is discussed in the context of full-space and reduced-space. On the basis of several PDE-constrained optimization problems in two and three dimensions, a general purpose algebraic multilevel incomplete $\boldsymbol{LDL}^\top$ preconditioner and a reduced-space preconditioned Krylov subspace method are evaluated and compared to an exact IP algorithm. In Part III IP algorithms are applied to an inverse medium problem for the Helmholtz equation. The influence of inequality constraints and Hessian modification are evinced numerically and different regularization strategies are compared. The part closes with numerical results for a real world benchmark problem in seismic imaging. Finally, Part IV concludes this thesis and points out possible directions of future research.

The main contribution of the thesis are the following:

- An exact and an inexact IP algorithm with global convergence properties are considered for large-scale nonconvex PDE-constrained optimization problems with inequality constraints.

- The IIP algorithm with a general purpose preconditioner is evaluated on large-scale PDE-constrained optimization problems.

- A reduced-space preconditioned GMRES (RSP-GMRES) solver is introduced in the IIP method. This RSP-GMRES method was implemented building upon `IPOPT` and several preconditioners are assessed. The

RSP-IIP method is assessed on the basis of large-scale nonconvex PDE-constrained optimization problems with inequality constraints.

- For an inverse medium problem for the Helmholtz equation, model parameter box constraints are applied. Their influence as well as the need for Hessian modification is demonstrated. Finally real world seismic imaging problem is solved by the RSP-IIP method.
- An interface for a parallel version of `IPOPT` was contributed to the open source project

Part I

Interior-Point Methods

Chapter 2

Interior-Point Framework and Exact Step Computation

In the first chapter we have seen how PDE-constrained optimization problems appear and that they are large-scale and often the matrices are ill-posed. Inequality constraints Especially pose a serious algorithmic challenge. In this part, we will present two IP methods to solve large-scale nonconvex PDE-constrained problems with inequality constraints.

First, we will introduce an IP framework and discuss local convergence properties. In the framework, a sequence of barrier subproblems is solved and its solutions converge to the solution of the original PDE-constrained problems. To solve the subproblems, sparse, large-scale linear systems, the Karush-Kuhn-Tucker (KKT) systems, need to be solved in each iteration. Two options to solve them will be presented.

In the first method a sparse direct solver is used. This will be discussed in Section 2.2. There, additional knowledge about an KKT system's inertia is available almost for free and the model problem can be varied by Hessian modification. This in combination with a filter line search then guarantees global convergence under mild assumptions.

The KKT system can also be solved iteratively by Krylov subspace methods. There, the systems are solved only inexactly and the question appears, how to control the error. More importantly, to guarantee global convergence, Hessian modification needs to be controlled in the absence of knowledge about the KKT matrix inertia. These issues are handled by the SMART tests in chapter 3.

2.1 Interior-Point Framework

In this thesis we solve large-scale nonconvex PDE-constrained optimization problems with inequality constraints. More precisely, we follow the discretize-than-optimize approach and use IP methods to solve problems of the form

$$\begin{aligned} &\min_{x\in\mathbb{R}^n} \boldsymbol{F}(\boldsymbol{x}) \\ &\text{s.t. } \boldsymbol{c}_{\boldsymbol{\mathcal{E}}}(\boldsymbol{x}) = \boldsymbol{0}, \\ &\quad\; \boldsymbol{c}_{\boldsymbol{\mathcal{I}}}(\boldsymbol{x}) \geqslant \boldsymbol{0}, \end{aligned} \tag{2.1}$$

where the equality constraints $\boldsymbol{c}_{\boldsymbol{\mathcal{E}}} : \mathbb{R}^n \to \mathbb{R}^{n_{\mathcal{E}}}$, inequality constraints $\boldsymbol{c}_{\boldsymbol{\mathcal{I}}} : \mathbb{R}^n \to \mathbb{R}^{n_{\mathcal{I}}}$, as well as the optimization variable $\mathbb{R}^n \ni \boldsymbol{x} = (\boldsymbol{y}\,\boldsymbol{u})$, consisting of state variables $\boldsymbol{y} \in \mathbb{R}^{n_y}$ and decision variables $\boldsymbol{u} \in \mathbb{R}^{n_u}$, are high dimensional, typically on the order of hundreds of thousands or millions.

Notation

Throughout this thesis vector-valued inequalities are meant componentwise. Vectors and matrices are denoted in bold letters and their components with a parenthesized superscript. We denote a vector consisting of stacked subvectors by $[a\,b] = (a^\top\, b^\top)^\top$. Function dependencies are often dropped once they are defined.

2.1.1 Optimality Conditions

In this section, we give a very rough overview of optimality conditions and follow the line from [55]. Solutions of problem (2.1) are characterized as saddle points of the Lagrangian

$$\boldsymbol{\mathcal{L}}(\boldsymbol{x}, \boldsymbol{\lambda}_{\mathcal{E}}, \boldsymbol{\lambda}_{\mathcal{I}}) := \boldsymbol{F}(\boldsymbol{x}) + \boldsymbol{\lambda}_{\mathcal{E}}^\top \boldsymbol{c}_{\boldsymbol{\mathcal{E}}}(\boldsymbol{x}) + \boldsymbol{\lambda}_{\mathcal{I}}^\top \boldsymbol{c}_{\boldsymbol{\mathcal{I}}}(\boldsymbol{x}),$$

with Lagrange multipliers $\boldsymbol{\lambda}_{\mathcal{E}} \in \mathbb{R}^{n_{\mathcal{E}}}, \boldsymbol{\lambda}_{\mathcal{I}} \in \mathbb{R}^{n_{\mathcal{I}}}$.

To state optimality conditions we need the concept of constraint qualifications:

Definition 2.1 (Active inequality, LICQ). *Given a point $\boldsymbol{x} \in \mathbb{R}^n$ the set of active inequalities is defined as*

$$\mathcal{A}(\boldsymbol{x}) = \{i \in 1, \dots n_{\mathcal{I}} : {\boldsymbol{c}_{\boldsymbol{\mathcal{I}}}}^{(i)} = 0\}.$$

We say, that the linear independence constraint qualification, (LICQ) *hold in $\boldsymbol{x}$ if the gradients of all equality and active inequality constraints are linearly independent.*

With this, we can formulate the first-order optimality conditions, see also e.g. Section 12.3 in [55]:

Theorem 2.2 (Karush-Kuhn-Tucker conditions). *Suppose that $\boldsymbol{x}^*$ is a local solution of (2.1), that $\boldsymbol{F}$, $\boldsymbol{c}_{\mathcal{E}}$, and $\boldsymbol{c}_{\mathcal{I}}$ are continuously differentiable, and that the LICQ holds at $\boldsymbol{x}^*$. Then there exist Lagrange multipliers $\boldsymbol{\lambda}^* = (\boldsymbol{\lambda}^*_{\mathcal{E}}, \boldsymbol{\lambda}^*_{\mathcal{I}}), \boldsymbol{\lambda}^*_{\mathcal{E}} \in \mathbb{R}^{n_{\mathcal{E}}}, \boldsymbol{\lambda}^*_{\mathcal{I}} \in \mathbb{R}^{n_{\mathcal{I}}}$, such that the following conditions are satisfied:*

$$\begin{aligned} \nabla_x \boldsymbol{\mathcal{L}}(\boldsymbol{x}^*, \boldsymbol{\lambda}^*) &= \boldsymbol{0}, && \qquad (2.2) \\ \boldsymbol{c}_{\mathcal{E}}(\boldsymbol{x}^*) &= \boldsymbol{0}, \\ \boldsymbol{c}_{\mathcal{I}}(\boldsymbol{x}^*) &\geqslant \boldsymbol{0}, \\ \boldsymbol{\lambda}^*_{\mathcal{I}} &\leqslant \boldsymbol{0}, \\ \boldsymbol{\lambda}^{*(i)}_{\mathcal{I}} \boldsymbol{c}_{\mathcal{I}}(\boldsymbol{x}^*)^{(i)} &= \boldsymbol{0} \qquad \textit{for all } i \in 1, \ldots, n_{\mathcal{I}}. && \qquad (2.3) \end{aligned}$$

Equation (2.3) is called the *complementary condition*. It is the reason why optimization problems with many inequality constraints are challenging: To solve this system, the distinct cases for the complementarity condition need to be considered, which number $2^{n_{\mathcal{I}}}$. Active set SQP methods seek the set of active inequality constraints. In each iteration they consider an estimate on the active set and update it on the basis of the solution of a subproblem. Usually in this update, only a few inequalities can be added or removed from their current estimate of the active set. Therefore, active set SQP methods are not favored when optimization problems with many inequality constraints need to be solved. Here, we treat inequality constraints by a barrier method which allows fast local convergence.

In unconstrained optimization, a second-order necessary condition is that the Hessian matrix is positive semidefinite. To state a similar theorem in constrained optimization, we need to define the critical cone.

Definition 2.3 (Strongly active constraints, critical cone). *Given a feasible point $\boldsymbol{x}$ and a Lagrange multiplier $\boldsymbol{\lambda}$ satisfying the KKT conditions (2.2), the set of* strongly active inequality constraints *is defined as*

$$\mathcal{A}^*(\boldsymbol{x}) = \{i \in \mathcal{A}(\boldsymbol{x}) : \boldsymbol{\lambda}_{\mathcal{I}}{}^{(i)} < 0\}.$$

The critical cone $\mathcal{C}(\boldsymbol{x}, \boldsymbol{\lambda})$ *is the set of vectors*

$$\mathcal{C}(\boldsymbol{x}, \boldsymbol{\lambda}) = \left\{ \boldsymbol{v} \in \mathbb{R}^n : \begin{cases} \nabla \boldsymbol{c}_{\mathcal{E}}(\boldsymbol{x})\boldsymbol{v} = \boldsymbol{0} & \textit{and} \\ \nabla \boldsymbol{c}_{\mathcal{I}}(\boldsymbol{x})^{(i,:)}\boldsymbol{v} = 0 & \textit{for all } i \in \mathcal{A}^*(\boldsymbol{x}) \textit{ and} \\ \nabla \boldsymbol{c}_{\mathcal{I}}(\boldsymbol{x})^{(i,:)}\boldsymbol{v} \geqslant 0 & \textit{for all } i \in \mathcal{A}(\boldsymbol{x}) \backslash \mathcal{A}^*(\boldsymbol{x}) \end{cases} \right\}.$$

Here $\nabla \boldsymbol{c}_{\mathcal{E}}$ denotes the equality constraint Jacobian and $\nabla \boldsymbol{c}_{\mathcal{I}}{}^{(i,:)}$ the ith line of the inequality constraint Jacobian. We now can give second-order necessary and sufficient optimality conditions, e.g., from Section 12.5 in [55].

Theorem 2.4 (Second-order necessary conditions)**.** *Suppose that $\boldsymbol{x}^*$ is a local solution of* (2.1) *and that the LICQ condition is satisfied. Let $\boldsymbol{\lambda}^*$ be the Lagrange multiplier vector, for which the KKT conditions* (2.2) *hold. Then*

$$\boldsymbol{v}^\top \nabla^2_{x,x}\boldsymbol{\mathcal{L}}(\boldsymbol{x}^*,\boldsymbol{\lambda}^*)\,\boldsymbol{v} \geqslant 0 \text{ for all } \boldsymbol{v} \in \mathcal{C}(\boldsymbol{x}^*,\boldsymbol{\lambda}^*).$$

Theorem 2.5 (second-order sufficient conditions)**.** *Suppose that for some feasible $\boldsymbol{x}^* \in \mathbb{R}^n$ there is a Lagrange multiplier vector $\boldsymbol{\lambda}^*$ such that the KKT conditions* (2.2) *are satisfied. Suppose also that*

$$\boldsymbol{v}^\top \nabla^2_{x,x}\boldsymbol{\mathcal{L}}(\boldsymbol{x}^*,\boldsymbol{\lambda}^*)\,\boldsymbol{v} > 0 \text{ for all } \boldsymbol{v} \in \mathcal{C}(\boldsymbol{x}^*,\boldsymbol{\lambda}^*), \boldsymbol{v} \neq \mathbf{0}.$$

Then $\boldsymbol{x}^*$ is a strict local solution of (2.1)

2.1.2 Primal-Dual Interior-Point Algorithm

We now describe the IP framework from [80, 24]. By introduction of slack variables $\boldsymbol{s}$, inequality constraints turn into equality constraints and new, possibly easier, inequality constraints,

$$\boldsymbol{s} \geqslant \mathbf{0} \qquad \text{with} \qquad \boldsymbol{c}_{\mathcal{I}}(\boldsymbol{x}) - \boldsymbol{s} = \mathbf{0},$$

are created.

Next, a logarithmic barrier term is added to the objective and the *barrier subproblem*

$$\begin{aligned} \min_{\boldsymbol{x}\in\mathbb{R}^n,\boldsymbol{s}\in\mathbb{R}^{n_{\mathcal{I}}}} \varphi(\boldsymbol{x},\boldsymbol{s};\mu) &:= \boldsymbol{F}(\boldsymbol{x}) - \mu \sum_{i=1}^{n_{\mathcal{I}}} \ln \boldsymbol{s}^{(i)} \\ \text{s.t. } \boldsymbol{c}(\boldsymbol{x},\boldsymbol{s}) &:= \begin{bmatrix} \boldsymbol{c}_{\mathcal{E}}(\boldsymbol{x}) \\ \boldsymbol{c}_{\mathcal{I}}(\boldsymbol{x}) - \boldsymbol{s} \end{bmatrix} = \mathbf{0} \end{aligned} \tag{2.4}$$

is formed. Note that due to the slack variables, the objective function and constraints can be evaluated in points, which do not satisfy the inequality constraints. This is of practical importance when it is hard to find an initial guess which fulfills the inequality constraints.

For a sequence of barrier parameters $\mu \searrow 0$, problem (2.1) is solved through an inexact solution of a sequence of barrier subproblems of the form (2.4). If $\boldsymbol{F}$ and $\boldsymbol{c}$ are continuously differentiable, then first-order KKT conditions for (2.4) are

$$\begin{pmatrix} \gamma(\boldsymbol{x},\boldsymbol{s};\mu) + \boldsymbol{J}(\boldsymbol{x})^\top\boldsymbol{\lambda} \\ \boldsymbol{c}(\boldsymbol{x},\boldsymbol{s}) \end{pmatrix} = \mathbf{0} \tag{2.5}$$

along with $\boldsymbol{s} \geqslant 0$, where $\boldsymbol{\lambda} = (\boldsymbol{\lambda}_{\mathcal{E}}, \boldsymbol{\lambda}_{\mathcal{I}})$, $\boldsymbol{e} \in \mathbb{R}^{n_{\mathcal{I}}}$ is a vector of ones,

$$\gamma(\boldsymbol{x},\boldsymbol{s};\mu) := \begin{pmatrix} \nabla\boldsymbol{F}(\boldsymbol{x}) \\ -\mu\boldsymbol{S}^{-1}\boldsymbol{\Sigma}\boldsymbol{e} \end{pmatrix}, \quad \text{and} \quad \boldsymbol{J}(\boldsymbol{x}) := \begin{pmatrix} \nabla\boldsymbol{c}_{\mathcal{E}}(\boldsymbol{x}) & 0 \\ \nabla\boldsymbol{c}_{\mathcal{I}}(\boldsymbol{x}) & -\boldsymbol{\Sigma} \end{pmatrix}, \tag{2.6}$$

where $\boldsymbol{\Sigma}$ denotes a diagonal scaling matrix, $\boldsymbol{S} = \operatorname{diag}(\boldsymbol{s})$ and $\nabla \boldsymbol{c}_{\mathcal{E}}, \nabla \boldsymbol{c}_{\mathcal{I}}$ denote the Jacobian matrix of the equality and inequality constraints, respectively.

In situations where (2.1) is infeasible, (2.5) has no solution, so the algorithms are designed to transition automatically from solving (2.4) to solving the unconstrained feasibility problem

$$\min_{x \in \mathbb{R}^n} \frac{1}{2} \|\boldsymbol{c}_{\mathcal{E}}(\boldsymbol{x})\|_2^2 + \frac{1}{2} \| \max\{-\boldsymbol{c}_{\mathcal{I}}(\boldsymbol{x}), \boldsymbol{0}\}\|_2^2 \tag{2.7}$$

as a certificate of infeasibility. Here, the "max" of vector quantities is to be understood componentwise. A solution to (2.7) that does not satisfy the constraints of problem (2.1) is known as an infeasible stationary point of the optimization problem. It satisfies the KKT conditions

$$\boldsymbol{J}(\boldsymbol{x})^\top \boldsymbol{c}(\boldsymbol{x}, \boldsymbol{s}) = 0 \tag{2.8}$$

along with $\boldsymbol{s} \geqslant 0$ and $\boldsymbol{c}_{\mathcal{I}}(\boldsymbol{x}) - \boldsymbol{s} \leqslant 0$. In fact, the algorithms maintain $\boldsymbol{s} \geqslant 0$ and $\boldsymbol{c}_{\mathcal{I}}(\boldsymbol{x}) - \boldsymbol{s} \leqslant 0$ during each iteration by increasing $\boldsymbol{s}$ when necessary. Thus, convergence to a solution of the barrier subproblem (2.4) or an infeasible stationary point of (2.1) is achieved once (2.5) or (2.8), respectively, is satisfied.

More precisely, an iterate $(\boldsymbol{x}\boldsymbol{s}\boldsymbol{\lambda})$ is considered an inexact solution if the error

$$E_\mu(\boldsymbol{x}, \boldsymbol{s}, \boldsymbol{\lambda}) \leqslant \kappa_\varepsilon \mu, \tag{2.9}$$

with some constant $\kappa_\varepsilon > 0$, where

$$\begin{aligned} E_\mu(\boldsymbol{x}, \boldsymbol{s}, \boldsymbol{\lambda}) := \max \Bigg(& \frac{\|\nabla \boldsymbol{F}(\boldsymbol{x}) + \boldsymbol{\lambda}_{\mathcal{E}}^\top \nabla \boldsymbol{c}_{\mathcal{E}}(\boldsymbol{x}) + \boldsymbol{\lambda}_{\mathcal{I}}^\top \nabla \boldsymbol{c}_{\mathcal{I}}(\boldsymbol{x}, \boldsymbol{s})\|_\infty}{s_d}, \\ & \|\boldsymbol{c}(\boldsymbol{x}, \boldsymbol{s})\|_\infty, \frac{\| \operatorname{diag}(\boldsymbol{\lambda}_{\mathcal{I}}) \boldsymbol{c}_{\mathcal{I}}(\boldsymbol{x}, \boldsymbol{s}) - \mu \boldsymbol{e}\|_\infty}{s_c} \Bigg) \end{aligned} \tag{2.10}$$

denotes the error. The scaling factors

$$s_d = \max \left(1, \frac{\|\boldsymbol{\lambda}_{\mathcal{E}}\|_1 + \|\boldsymbol{\lambda}_{\mathcal{I}}\|_1}{100 \cdot (n_{\mathcal{E}} + n_{\mathcal{I}})}\right) \text{ and } s_c = \max \left(1, \frac{\|\boldsymbol{\lambda}_{\mathcal{I}}\|_1}{100 \cdot n_{\mathcal{I}}}\right)$$

handle situations where gradients of active constraints are almost linearly dependent, resulting in very large multipliers. Once an inexact solution of a barrier subproblem is found, the barrier parameter μ is updated as

$$\mu \leftarrow \max \left(\frac{\varepsilon_{tol}}{10} \min \left(\kappa_\mu \mu, \mu^{\Theta} \right) \right), \tag{2.11}$$

with constants $\kappa_\mu > 0, \theta \in (1, 2)$. Here, ε_{tol} denotes the overall tolerance to determine an inexact solution of (2.1) and terminate the algorithm if

$$E_0(\boldsymbol{x}, \boldsymbol{s}, \boldsymbol{\lambda}) \leqslant \varepsilon_{tol}. \tag{2.12}$$

At each iterate $(\boldsymbol{x}, \boldsymbol{s}, \boldsymbol{\lambda})$, a Newton-type system for (2.5)

$$\begin{pmatrix} \boldsymbol{W} & \boldsymbol{J}^\top \\ \boldsymbol{J} & \boldsymbol{0} \end{pmatrix} \begin{pmatrix} \boldsymbol{d} \\ \delta \end{pmatrix} = - \begin{pmatrix} \gamma + \boldsymbol{J}^\top \boldsymbol{\lambda} \\ \boldsymbol{c} \end{pmatrix} \tag{2.13}$$

is solved to compute a search direction $\boldsymbol{d} = (\boldsymbol{d}^x, \boldsymbol{d}^s)$ and $\delta = (\delta^{\mathcal{E}}, \delta^{\mathcal{I}})$ in the primal and dual space, respectively. Here, $\boldsymbol{W}$ denotes an approximation on the scaled Hessian of the Lagrangian

$$\begin{aligned} \boldsymbol{W}(\boldsymbol{x}, \boldsymbol{s}, \boldsymbol{\lambda}; \mu) := & \begin{pmatrix} \nabla^2_{xx} \boldsymbol{F}(\boldsymbol{x}) + \delta_r \boldsymbol{I} & 0 \\ 0 & \boldsymbol{\Sigma}(\Xi + \delta_r \boldsymbol{I})\boldsymbol{\Sigma} \end{pmatrix} \\ & + \sum_{i=1}^{n_{\mathcal{E}}} \boldsymbol{\lambda}_{\mathcal{E}}^{(i)} \begin{pmatrix} \nabla^2_{xx} \boldsymbol{c}_{\mathcal{E}}{}^{(i)}(\boldsymbol{x}) & 0 \\ 0 & 0 \end{pmatrix} + \sum_{i=1}^{n_{\mathcal{I}}} \boldsymbol{\lambda}_{\mathcal{I}}^{(i)} \begin{pmatrix} \nabla^2_{xx} \boldsymbol{c}_{\mathcal{I}}{}^{(i)}(\boldsymbol{x}) & 0 \\ 0 & 0 \end{pmatrix}, \end{aligned} \tag{2.14}$$

where the Hessian modification $\delta_r \boldsymbol{I}$ is introduced to enforce convexity of the subproblem model in the search direction. For $\Xi = \mu \boldsymbol{S}^{-2}$ the algorithm corresponds to the so called primal IP algorithm. However, (2.5) is better suited for Newton's method if we pre-multiply the second line of γ in (2.6) by $\boldsymbol{S}$. Thereby the nonlinear term which is approximated is not $\boldsymbol{S}^{-1}$ but $\boldsymbol{S}\,\mathrm{diag}(\boldsymbol{\lambda}_{\mathcal{I}})$. This approach, after post-multiplication with $\boldsymbol{S}^{-1}$ again yields (2.13) and (2.14) but with $\Xi = -\boldsymbol{S}^{-1}\,\mathrm{diag}(\boldsymbol{\lambda}_{\mathcal{I}})$.

Indeed, the solution of (2.13) is the most time consuming step in the optimization process. In a full-space approach for PDE-constrained optimization, the matrix is large-scale, sparse, and indefinite. The matrix structure for this class of optimization problems will be discussed in more detail in Chapter 4.1.

There are basically two options to solve this KKT system. With a sparse direct solver the system is solved exactly (up to round-off error) and knowledge about the inertia can be used to modify the Hessian matrix to guarantee global convergence; see Section 2.2. However, the fill-in, especially for three-dimensional (3D) applications, limits their use due to enormous memory requirements. The second option is the use of Krylov subspace methods. Then the system is solved inexactly, and Hessian modification and inexactness need to be controlled appropriately. This will be explained in Chapter 3.

In either way, an update of the current iterates $\boldsymbol{x}, \boldsymbol{s}, \boldsymbol{\lambda}_{\mathcal{E}}, \boldsymbol{\lambda}_{\mathcal{I}}$ is computed, where the slack variables fulfill the fraction-to-the-boundary rule

$$\boldsymbol{s}_{new}^{(i)} \geqslant (1 - \eta_{Bd}) \boldsymbol{s}_{old}^{(i)} \qquad \text{for all } i \in 1, \dots n_{\mathcal{I}}. \tag{2.15}$$

We use $\eta_{Bd} = \max\{0.99, 1 - \mu\}$. It is important to mention, that this choice of η_1 does not inhibit superlinear convergence, since it depends on $\mu \to 0$.

The overall IP framework is summarized in Algorithm 2.1

Algorithm 2.1 IP Framework

```
1: (Initialization) Choose parameters $\kappa_\varepsilon, \varepsilon_{tol}, \kappa_\mu, \Theta$, an initial barrier parameter $\mu > 0$. Initialize $(x_0, s_0, \lambda_0)$ so that the slack variables satisfy $s_0 > 0$ and $s_0 \geqslant c_{\mathcal{I}}(x_0)$.
2: (Tests for convergence) If convergence criteria (2.12) for (2.1) are satisfied, then terminate and return $x$ as an optimal solution. Else, if convergence criteria for (2.7) are satisfied and $x$ is infeasible for (2.1), then terminate and return $x$ as an infeasible stationary point.
3: (Barrier parameter update) If convergence criteria (2.10) for (2.4) are satisfied, then decrease the barrier parameter $\mu$ according to (2.11) and go to step 2.
4: (Barrier subproblem step) Compute new estimates for the primal and dual variables by applying the Algorithm 2.2 or Algorithm 3.1 which satisfy the fraction-to-the-boundary rule (2.15) and go to step 2.
```

2.1.3 Barrier Parameter and Local Convergence

We will now visualize the barrier method on the basis of a simple example,

$$\begin{aligned} \min_{x \in \mathbb{R}^2} F(x) &= x_1 + \frac{1}{2} x_2^2 \\ \text{s.t.}\, x_1 &\geqslant \mathbf{0}. \end{aligned} \tag{2.16}$$

The solution is at $x = (0, 0)$ and the objective function is visualized in Figure 2.1(a). For the ease of the example, we skip the slack variables. Thus, the barrier subproblem reads

$$\min_{x \in \mathbb{R}^2} F(x) = x_1 + \frac{1}{2} x_2^2 - \mu \log(x_1) \tag{2.17}$$

with solution $x_\mu^* = (\mu, 0)$. Thus, the local convergence rate crucially depends on the decrease rate of the barrier parameter. In the described algorithm, μ is eventually decreased exponentially; see (2.11). It has been shown that there is a solution of the barrier subproblem within a region with radius $\mathcal{O}(\mu)$ around the solution of (2.1) [27]. Thus, if the algorithm converges superlinearly to a solution of a subproblem, overall we would have superlinear convergence. To analyze this, we follow the line of [18]. In a primal-dual IP algorithm, Newton's method is applied to the optimality conditions of (2.4)

$$b(x, s, \lambda; \mu) = \begin{pmatrix} \nabla F(x) + \nabla c_{\mathcal{E}}(x)^\top \lambda_{\mathcal{E}} + \nabla c_{\mathcal{I}}(x)^\top \lambda_{\mathcal{I}} \\ -\mu e - S \operatorname{diag}(\lambda_{\mathcal{I}}) \\ c_{\mathcal{E}}(x) \\ c_{\mathcal{I}}(x) - s \end{pmatrix} = \mathbf{0},$$

and in each iteration, the KKT system

$$\boldsymbol{b}'(\boldsymbol{x},\boldsymbol{s},\boldsymbol{\lambda})\boldsymbol{p} = -\boldsymbol{b}(\boldsymbol{x},\boldsymbol{s},\boldsymbol{\lambda};\mu)$$

needs to be solved. Note, that the matrix $\boldsymbol{b}'(\boldsymbol{x},\boldsymbol{s},\boldsymbol{\lambda})$ does not depend on μ. Thus, we can consider a primal-dual IP method as an inexact Newton method for the optimality condition of (2.1), $\boldsymbol{b}(\boldsymbol{x},\boldsymbol{s},\boldsymbol{\lambda};\boldsymbol{0}) = \boldsymbol{0}$, with residual vector

$$\boldsymbol{b}'(\boldsymbol{x},\boldsymbol{s},\boldsymbol{\lambda})\boldsymbol{p} + \boldsymbol{b}(\boldsymbol{x},\boldsymbol{s},\boldsymbol{\lambda};0) = -\mu\left[\boldsymbol{0}\,\boldsymbol{e}\,\boldsymbol{0}\,\boldsymbol{0}\right] + \boldsymbol{r}_{\mu}, \tag{2.18}$$

where $\boldsymbol{r}$ accounts for the residual of inexact solutions of the KKT system. Then theorem 2.5 in [18] states:

Theorem 2.6 (Local convergence of IP methods)**.** *Let $\boldsymbol{z}^* = [\boldsymbol{x}^*,\boldsymbol{s}^*,\boldsymbol{\lambda}^*]$ denote a KKT point of (2.1) that is $\boldsymbol{b}(\boldsymbol{x}^*,\boldsymbol{s}^*,\boldsymbol{\lambda}^*;0) = \boldsymbol{0}$, and assume that the Hessians of the objective function and each constraint exist and are locally Lipschitz continuous at $\boldsymbol{x}^*$. Assume that LICQ and sufficient second order condition from Theorem 2.5 hold at $\boldsymbol{z}^*$ and also strict complementarity, i.e., $\boldsymbol{s}^* + \boldsymbol{\lambda}^*_{\mathcal{I}} > \boldsymbol{0}$. Let $\boldsymbol{z}$ be an iterate sufficiently close to $\boldsymbol{z}^*$ at which the barrier parameter is decreased from μ to μ^+. Suppose, that the residual in (2.18) satisfies*

$$\|\boldsymbol{r}_{\mu^+}\| \leqslant C\|\boldsymbol{b}(\boldsymbol{z};\mu^+)\|^{1+\kappa}$$

for some constants $C > 0, \kappa > 0$. Then if $\mu^+ = o(\|\boldsymbol{b}(\boldsymbol{z};0)\|)$ and $\kappa > 0$, the step will be superlinearly convergent to $\boldsymbol{z}^*$; moreover, if $\mu^+ = O(\|\boldsymbol{b}(\boldsymbol{z};0)\|^2)$ and $\kappa \geqslant 1$, the step will be quadratically convergent.

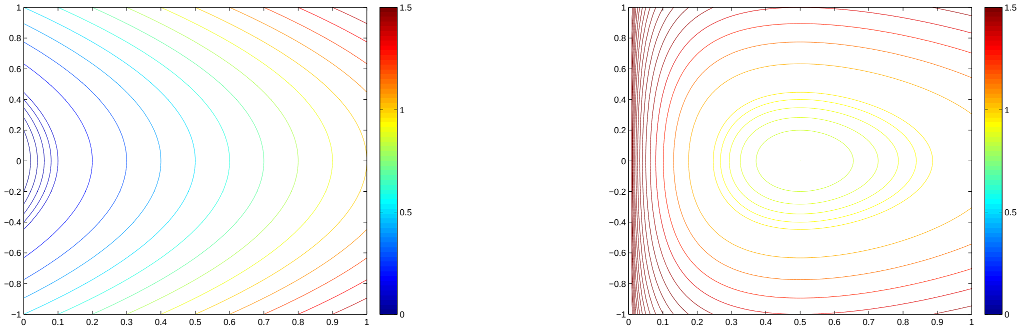
Another point of IP methods, that we can see from example (2.16) is the ill-conditioning of the KKT system. The Hessian matrix for the subproblem (2.17) in its solution is

$$\boldsymbol{H}(\boldsymbol{x}^*_{\mu}) = \begin{pmatrix} \frac{1}{\mu} & 0 \\ 0 & 1 \end{pmatrix}$$

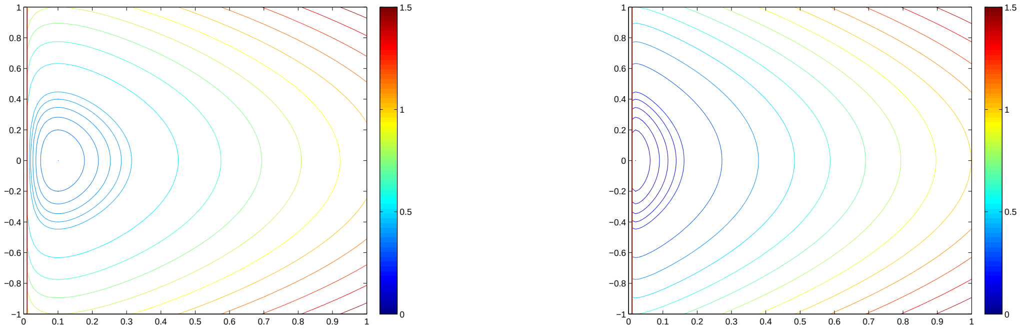
with condition $1/\mu$ for $\mu \leqslant 1$. Thus, the Hessian becomes more and more ill-conditioned as the barrier parameter is reduced. This can be also seen from Figure 2.1(b) – 2.1(d), where the contour lines close to the barrier subproblems solution become more and more stretched as $\mu \to 0$. Changing from an primal IP methods to an primal-dual approach does not improve the situation. Then the ill-conditioning does not appear in the barrier term anymore, but active inequality constraints become arbitrarily strong weights due to $\boldsymbol{\lambda}^{(i)}_{\mathcal{I}} = \mu/\boldsymbol{s}^{(i)}$; see [27].

2.2 Exact Interior-Point Method

In Section 2.1 we have seen a general framework of IP methods. We now describe our first option to compute a new iterate of Algorithm 2.1, which consists



(a) $\mu = 0$ **(b)** $\mu = 0.5$



(c) $\mu = 0.1$ **(d)** $\mu = 0.02$

Figure 2.1: *Toy example: Contour plots of the original objective function* ($\mu = 0$) *and objective of the barrier subproblems with* $\mu = 0.5, 0.1,$ *and* 0.02

in an exact Newton method combined with a filter line search. This algorithm originates from [80]. Its superlinear convergence has been proven in [78] and global convergence was analyzed in [79].

In the algorithm described in this section, a sparse direct solver is applied to (2.13). Here, an $\boldsymbol{LDL}^\top$ decomposition is computed. From this factorization, the inertia can be computed easily to check if the Hessian matrix is positive definite on the null space of the constraints. If not, a Hessian modification is performed and a new factorization is computed repeatedly.

The algorithm then seeks step lengths $\alpha^{x,s,\mathcal{E}}$ and $\alpha^{\mathcal{I}}$ to update the iterates

$$\begin{aligned} \boldsymbol{x}(\alpha) &:= \boldsymbol{x} + \alpha^{x,s,\mathcal{E}} \boldsymbol{d}^x, \\ \boldsymbol{s}(\alpha) &:= \boldsymbol{s} + \alpha^{x,s,\mathcal{E}} \boldsymbol{d}^s, \\ \boldsymbol{\lambda}^{\mathcal{E}} &:= \boldsymbol{\lambda}^{\mathcal{E}} + \alpha^{x,s,\mathcal{E}} \delta^{\mathcal{E}}, \\ \boldsymbol{\lambda}^{\mathcal{I}} &:= \boldsymbol{\lambda}^{\mathcal{I}} + \alpha^{\mathcal{I}} \delta^{\mathcal{I}}. \end{aligned} \tag{2.19}$$

First, $\alpha^{\max}$ and $\alpha^{\mathcal{I}}$ are determined by the fraction-to-the-boundary rule

$$\begin{aligned}\alpha^{\max} &= \max\{\alpha \in (0,1] : \boldsymbol{s} + \alpha \boldsymbol{d}^{s} \geqslant (1-\eta)\boldsymbol{s}\},\\ \alpha^{\mathcal{I}} &= \max\{\alpha \in (0,1] : \boldsymbol{\lambda}^{\mathcal{I}} + \alpha \delta^{\mathcal{I}} \geqslant (1-\eta)\boldsymbol{\lambda}^{\mathcal{I}}\}.\end{aligned}$$

Then a backtracking line search aims a filter accepted step length $\alpha^{x,s,\mathcal{E}}$ by checking the trial steps $2^{-l}\alpha^{\max}$. If the first trial step is not accepted and the trial step decreases feasibility, second order corrections try to improve feasibility before smaller step lengths are tested. If in the line search the step length becomes too small, a feasibility restoration phase is launched, where the algorithm is applied to the feasibility problem (2.7). In the remainder of this chapter main algorithmic ingredients will be presented. For a complete algorithmic overview, we refer to [80].

Algorithm 2.2 Barrier Subproblem Exact Newton Step

```
Input: Current estimate $\boldsymbol{x}, \boldsymbol{\lambda}, \boldsymbol{s}, \mu$,
Initialize $\delta_r = 0$
repeat
    Compute a factorization of the KKT matrix in (2.13).
    If the matrix is singular, perturb the lower left block.
    If the $\delta_r = 0$ set $\delta_r \leftarrow \delta_r^{init} = 0$, else set $\delta_r \leftarrow \max\{\delta_r^{min}, \kappa_\delta \delta_r\}$.
until Inertial is $(n, m, 0)$
Compute search direction $\boldsymbol{d}, \boldsymbol{\delta}$ by solving (2.13)
Set $\alpha^{\max}$ by the fraction-to-the-boundary-rule (2.15), set trial point $\boldsymbol{x}(\alpha^{\max})$
    and $\boldsymbol{s}(\alpha^{\max})$ according to (2.19)
repeat
    If $\boldsymbol{x}(\alpha) \in \mathcal{F}$ apply a second-order correction, recompute $\alpha^{\max}$ according to
        the fraction of the boundary rule and compute the resulting trial point.
until trial point is accepted by the filter or maximum number of second-
    order correction reached
repeat
    Set $\alpha \leftarrow \alpha/2$ and compute trial point $\boldsymbol{x}(\alpha), \boldsymbol{s}(\alpha)$
until trial point is accepted by the filer or $\alpha < \alpha^{\min}$
if $\alpha < \alpha^{\min}$ then
    Start feasibility restoration phase, apply the optimization algorithm to prob-
        lem (2.7). If it is not possible to find a point which is accepted by the filter
        terminate the optimization run, else use computed solution.
else
    Augment the filter according to (2.20)
end if
Compute $\boldsymbol{\lambda}_{\mathcal{E}}, \boldsymbol{\lambda}_{\mathcal{I}}$ by (2.19)
Output: $\boldsymbol{x}_{k+1}, \boldsymbol{\lambda}_{k+1}, \boldsymbol{s}_{k+1}$
```

2.2.1 Computation of the Newton Direction

The full-space approach in PDE-constrained optimization yields a sparse and large-scale KKT matrix in (2.13). Furthermore, it is indefinite and often ill-conditioned. Thus, a robust sparse linear solver for indefinite matrices is of utter most importance.

We apply `Pardiso` [63] to compute a $\boldsymbol{PLDL}^\top\boldsymbol{P}^\top$ factorization with a lower triangular matrix $\boldsymbol{L}$, a permutation matrix $\boldsymbol{P}$, and a block-diagonal matrix $\boldsymbol{D}$ with blocks of size 1×1 and 2×2. The permutation $\boldsymbol{P}$ reduces the fill-in and improves the accuracy by maximal matchings and pivoting strategies like Supernode-Bunch-Kaufmann. Using this factorization, the number of positive, negative and zero eigenvalues, the *inertia*, can be computed easily by investigation of the small diagonal blocks in $\boldsymbol{D}$.

The knowledge of the inertia is of great importance in the context of optimization. In unconstrained nonconvex optimization a Newton direction may be an accent direction, if the Hessian matrix is indefinite. A similar case can occur in nonconvex constrained optimization. There the Lagrangian Hessian needs to be positive definite on the null space of the constraints Jacobian.

To see this, recall, that the second-order sufficient conditions state that the $\boldsymbol{W}$-block in (2.13) is positive definite on the null space of the Jacobian. This implies a fixed inertia, as we see in the next theorem from [27]:

Theorem 2.7 (Inertia of a KKT matrix)**.** *Let*

$$\boldsymbol{K} = \begin{pmatrix} \boldsymbol{W} & \boldsymbol{J}^\top \\ \boldsymbol{J} & \boldsymbol{0} \end{pmatrix}$$

with $\boldsymbol{W} \in \mathbb{R}^{k \times k}$, $\boldsymbol{J} \in \mathbb{R}^{\ell \times k}$, $\operatorname{rank}(\boldsymbol{J}) = \ell$.

If $\boldsymbol{W}$ *is positive definite on the null space of* $\boldsymbol{J}$*, then it is regular and has* k *positive and* ℓ *negative eigenvalues.*

If K does not have this inertia, then $\boldsymbol{W}$ is semidefinite or indefinite on the null space of the linearized constraints. Thus, there exists a first-order feasible direction for which the objective can be nondecreasing. To circumvent those situations, the Hessian matrix is modified. This is handled in Algorithm 2.2, steps 2-6

2.2.2 Filter Line-Search Methods

In unconstrained optimization, the line search needs to ensure that each step reduces the objective function sufficiently. In constrained optimization two goals need to be achieved: to find a feasible point and to reduce the objective function. Thus, filters prevent cycling in an optimization algorithm by considering a constrained optimization problem as a biobjective optimization problem.

A filter contains taboo regions in $\mathbb{R}^2$ which are considered as banned for the optimization algorithm. At the beginning of each subproblem solution, the filter is initialized as

$$\mathcal{F} \leftarrow \left\{(\varphi, c) \in \mathbb{R}^2 | c \geqslant c^{\max}\right\},$$

with some maximal constraint violation $c^{\max}$. Thus, points with high infeasibility are banned in the whole algorithm. In the line search phase, the algorithm generates trial point $\boldsymbol{x}(\alpha), \boldsymbol{s}(\alpha)$ which are rejected if

$$(\varphi(\boldsymbol{x}, \boldsymbol{s}; \mu), \|\boldsymbol{c}(\boldsymbol{x})\|) \in \mathcal{F}$$

and accepted otherwise. Within algorithm 2.2, the filter saves visited points and rejects points, which are worse than one of the previously visited points in both, infeasibility and a merit function. The filter is then augmented as

$$\mathcal{F} \leftarrow \mathcal{F} \cup \{(\phi, c) \in \mathbb{R}^2 | \phi \geqslant \varphi(\boldsymbol{x}, \boldsymbol{s}; \mu) - \gamma_\varphi \|\boldsymbol{c}(\boldsymbol{x}, \boldsymbol{s})\| \text{ and } c \geqslant (1 - \gamma_c)\|\boldsymbol{c}(\boldsymbol{x}, \boldsymbol{s})\|\}, \tag{2.20}$$

with $\gamma_\varphi, \gamma_c \in (0, 1)$. A filter is visualized in Figure 2.2. On the horizontal axis we see the barrier objective function and on the vertical axis we draw the infeasibility. A solution of (2.4) lies on the x-axis as left as possible. The taboo region is marked gray and any trial point whose objective value and infeasibility are in the gray region will be rejected. In the figure, the filter was augmented with at least two iterates, each of which generates an edge in the graph.

When the algorithm has reached an almost feasible point and the search direction is a decrease direction for the objective, the filter line search is replaced by the Armijo condition

$$\varphi(\boldsymbol{x}(\alpha), \boldsymbol{s}(\alpha); \mu) \leqslant \varphi(\boldsymbol{x}, \boldsymbol{s}) + \eta_A \alpha \gamma^\top \boldsymbol{d},$$

where γ from equation (2.6) is the gradient of the barrier objective function and $\eta \in (0, 1/2)$.

2.2.3 Second-Order Correction

The famous Maratos effect shows that situations may occur, in which a full Newton step yields local quadratic convergence, but it will not be accepted by a filter or a merit function. This situation can be detected if the first trial point $\boldsymbol{x}(\alpha^{\max}), \boldsymbol{s}(\alpha^{\max})$ is rejected by the filter and the infeasibility increased. To cope with the Maratos effect, second-order correction aims to improve feasibility by an extra Newton step in the primal variables on the constraints,

$$\boldsymbol{J}(\boldsymbol{x}(\alpha), \boldsymbol{s}(\alpha))\, \boldsymbol{d}_{SOC} = -\begin{pmatrix} \boldsymbol{c}_{\mathcal{E}}(\boldsymbol{x}(\alpha)) \\ \boldsymbol{c}_{\mathcal{I}}(\boldsymbol{x}(\alpha)) - \boldsymbol{s}(\alpha) \end{pmatrix}.$$

The second-order correction is not completely determined by 2.2.3. Furthermore, an additional large-scale linear system needs to be solved. Both issues are

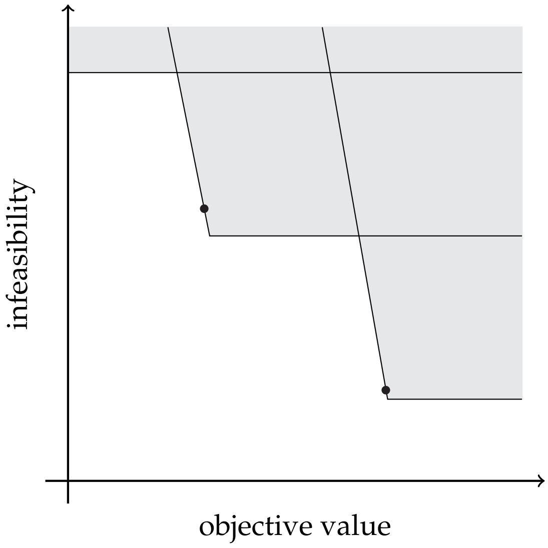


Figure 2.2: *A filter defines a taboo region (gray). A trial point* $\boldsymbol{x}(\alpha), \boldsymbol{s}(\alpha)$ *corresponds to a point in this graph by its objective function value and its infeasibility and may be accepted by the filter if it lies in the white area. Later on the filter is augmented which creates a new "corner" in this graph.*

resolved if the Jacobian matrix is not evaluated in the trial point $\boldsymbol{x}(\alpha), \boldsymbol{s}(\alpha)$ but in the current iterate $\boldsymbol{x}, \boldsymbol{s}$. Then the factorization of 2.13 can be reused. In the algorithm we solve the linear system

$$\begin{pmatrix} \boldsymbol{W} & \boldsymbol{J}^\top \\ \boldsymbol{J} & \boldsymbol{0} \end{pmatrix} \begin{pmatrix} \boldsymbol{d}^{SOC} \\ \delta \end{pmatrix} = - \begin{pmatrix} \gamma + \boldsymbol{J}^\top \boldsymbol{\lambda} \\ \alpha \boldsymbol{c} + \boldsymbol{c}(\boldsymbol{x}(\alpha)) \end{pmatrix},$$

and update the search direction

$$\boldsymbol{d} := \alpha \boldsymbol{d} + \boldsymbol{d}^{SOC},$$

as well as $\alpha^{\max}$ according to the fraction-to-the-boundary rule in the new direction.

This is described in Algorithm 2.2, step 9 – 11. There, second-order corrections are applied repeatedly, until the filter accepted the step or a maximal number of second-order corrections is reached.

Chapter 3

Inexact Interior-Point Method

In the previous chapter, we have presented the IP framework and a first option to compute a new iterate for Algorithm 2.1 by an exact Newton method in combination with a filter line search. There, the KKT system (2.13) needs to be solved, which is sparse and large scale. Sparse direct solvers suffer from enormous memory requirements due to fill-in. Thus, the idea presented in this chapter is to apply an iterative linear solver to (2.13). They compute a sequence of inexact solutions which converge to the exact solution. Since the KKT systems are solved inexactly the question is, which inexact solution is considered accurate enough. For global convergence the exact algorithm ensures a descent direction of the objective function by inspection of the inertia of the KKT matrix and modifies the Hessian block if necessary. Here we consider inexact IP methods (IIP), where no factorization is available. Thus Hessian modification needs to be controlled in the absence of knowledge about the KKT matrix inertia. These issues are accomplished by the SMART tests.

In this chapter we follow the algorithm description as presented in [35] and [22]. The optimization method is based on the series of inexact SQP, inexact Newton, and IIP algorithms that have been proposed and analyzed in [17, 23, 24, 35], though the majority relates to the latest enhancements in [35].

In [17], iterates of a linear solver were considered as search direction candidates. On the basis of a local model of a merit function, termination tests for the linear solver were developed to achieve global convergence. For the central role of the reduction of the merit function, these tests are referred to as SMART tests. In cases where the constraint Jacobians were almost rank deficient, unproductive steps may be taken. This was tackled in [23] by a step decomposition into a normal and a tangential step. The normal step was computed as the inexact solution of a trust-region subproblem aiming to reduce the primal infeasibility. The tangential step was then computed implicitly as the solution of a modified

Newton system.

In [24] the SMART tests were extended to IP methods to cope with inequality constraints. Special care was taken to the logarithmic barrier term and the fraction-to-the-boundary rule in the line search phase. All of the aforementioned inexact methods guarantee global convergence. The stabilized SMART tests in [24, 22] require the solution of two Newton systems, thus doubling the price of a Newton iteration. Even though global convergence is not guaranteed for the inexact method in [17] within an IIP framework, in practice it has shown good results. This led to the combination of both methods in [22], where the nonstabilized method was safeguarded by the globally convergent one.

The inexact step computation method is summarized in Algorithm 3.1.

Algorithm 3.1 Inexact Step Computation

```
(Initialization) If uninitialized choose line search parameters $\eta_2 \in (0,1)$, $\alpha^{\min} \in (0,1)$
if barrier parameter $\mu$ changed then
    (Reinitialization) Chose initial penalty parameter $\pi > 0$.
end if
Compute $\alpha = \alpha^{\max}$ according to the fraction-to-the-boundary rule (2.15).
Compute a SMART accepted search direction $(\boldsymbol{d}, \boldsymbol{\delta})$ and update the merit function parameter $\pi$ by Algorithm 3.2.
repeat
    Set $\alpha \leftarrow \alpha/2$ and compute a new trial point $(\boldsymbol{x}(\alpha), \boldsymbol{s}(\alpha))$.
until trial point $(\boldsymbol{x}(\alpha), \boldsymbol{s}(\alpha))$ fulfills the Armijo rule (3.10) or $\alpha < \alpha^{\min}$
if $\alpha < \alpha^{\min}$ then
    reset $\alpha \leftarrow \alpha^{\max}$
    Compute a new SMART accepted search direction $(\boldsymbol{d}, \boldsymbol{\delta})$ and update the merit function parameter $\pi$ by Algorithm 3.3.
    repeat
        Set $\alpha \leftarrow \alpha/2$ and compute a new trial point $(\boldsymbol{x}(\alpha), \boldsymbol{s}(\alpha))$.
    until trial point $(\boldsymbol{x}(\alpha), \boldsymbol{s}(\alpha))$ fulfills the Armijo rule (3.10)
end if
Compute dual step length $\beta$ according to (3.11) and update the dual variable
    $$\boldsymbol{\lambda} \leftarrow \boldsymbol{\lambda} + \beta \boldsymbol{\delta}$$
Update $\boldsymbol{x}, \boldsymbol{s}$
```

3.1 SMART Tests for an Inexact Newton Method

As mentioned earlier, when inexact solutions are used to compute an update, strategies need to be employed to cope with errors and Hessian modification. Here we first describe a search direction computation, which itself yields global convergence [17] under assumptions, which may not hold for all steps within an IP optimization method. In cases where it fails, the method changes to Algorithm 3.3 described in Section 3.2, where care is taken of almost rank deficient constraint Jacobians and the logarithmic barrier term.

The IIP methods consider a given iterate of an iterative linear solver as an *search direction candidate* and check its effect in terms of the optimization problem itself instead of just checking if a certain residual

$$\begin{pmatrix} \boldsymbol{r}_d \\ \boldsymbol{r}_p \end{pmatrix} = \begin{pmatrix} \boldsymbol{W} & \boldsymbol{J}^\top \\ \boldsymbol{J} & \boldsymbol{0} \end{pmatrix} \begin{pmatrix} \boldsymbol{d} \\ \delta \end{pmatrix} + \begin{pmatrix} \gamma + \boldsymbol{J}^\top \boldsymbol{\lambda} \\ \boldsymbol{c} \end{pmatrix} \tag{3.1}$$

has been reached. We denote the relative residual of the Newton system by $\Psi := \|\binom{\boldsymbol{r}_d}{\boldsymbol{r}_p}\| / \|\binom{\gamma+\boldsymbol{J}^\top\boldsymbol{\lambda}}{\boldsymbol{c}}\|$. To rate a search direction candidate the merit function

$$\phi(\boldsymbol{x}, \boldsymbol{s}; \mu, \pi) := \varphi(\boldsymbol{x}, \boldsymbol{s}; \mu) + \pi \|\boldsymbol{c}\| \tag{3.2}$$

is approximated,

$$m(\boldsymbol{d}; \mu, \pi) := \phi(\boldsymbol{x}, \boldsymbol{s}; \mu, \pi) + \nabla_{x,s}\varphi(\boldsymbol{x}, \boldsymbol{s})^\top \boldsymbol{d} + \pi \left\|\boldsymbol{c} + \boldsymbol{J}\boldsymbol{d}\right\|,$$

and its reduction

$$\begin{aligned} \Delta m(\boldsymbol{d}; \mu, \pi) : &= m(\boldsymbol{0}; \mu, \pi) - m(\boldsymbol{d}; \mu, \pi) \\ &= -\nabla_{x,s}\varphi(\boldsymbol{x}, \boldsymbol{s})^\top \boldsymbol{d} + \pi \left(\|\boldsymbol{c}\| - \|\boldsymbol{c} + \boldsymbol{J}\boldsymbol{d}\|\right) \end{aligned} \tag{3.3}$$

is observed.

An exact Newton step fulfills $-\nabla_{x,s}\varphi(\boldsymbol{x}, \boldsymbol{s})^\top \boldsymbol{d} \geqslant 1/2\, \boldsymbol{d}^\top \boldsymbol{W} \boldsymbol{d} + \|\boldsymbol{c}\| \|\boldsymbol{\lambda} + \boldsymbol{\delta}\|$ and thus we had $\Delta m \geqslant 1/2\, \boldsymbol{d}^\top \boldsymbol{W} \boldsymbol{d} + \epsilon_1 \pi \|\boldsymbol{c}\|$ with some constant ϵ_1. The first term can be negative for nonconvex problems and thus we safeguard for such a situation. These ideas give rise for the model reduction condition of termination test 1 in Algorithm 3.2, step 3. It can be considered the main condition to be satisfied.

In cases where the search direction candidate yields a small relative residual and the curvature $\frac{\boldsymbol{d}^\top \boldsymbol{W} \boldsymbol{d}}{\|\boldsymbol{d}\|^2}$ is positive, failure of termination test 1 may be caused by a too small merit function penalty parameter π. Thus, in the case where the primal and dual residuals are small compared to the primal infeasibility and the objective function curvature along the search direction candidate is positive, in step 4 we accept the candidate and π is increased to a value such that the model reduction condition from step 3 holds.

The third termination test in step 5 checks for almost feasible points if a step only in the dual space would improve dual feasibility. Then this dual search direction is accepted and the primal step is set to $\boldsymbol{0}$.

If a search direction candidates exhibits a reasonably small relative residual, then the curvature along this direction is inspected and if it is not sufficiently positive the Hessian matrix is modified by increasing δ_r in (2.13). This test replaces the inertia correction in the exact method. Note, that here, we only require that W is positive definite in the search direction instead of the whole Jacobian null space. This is a much more direct way to ensure positive curvature along the search direction.

These tests are usually referred to as *S*ufficient *M*erit function *A*pproximation *R*eduction *T*ermination tests (*SMART*). Algorithm 3.2 summarizes the SMART tests for the inexact Newton method. Indeed the global convergence theorem 5.1 in [17] shows that all causes of failure of termination test 1 are handled appropriately. For completeness we restate it here:

Theorem 3.1. *Suppose, the sequence* $[\boldsymbol{x}, \boldsymbol{s}]$ *generated by Algorithms 2.1, 3.1 and 3.2 is contained in a convex set over which the following properties hold:*

1. *The objective function, the constraints, and their first and second derivatives are bounded.*

2. *The constraints Jacobian* $\boldsymbol{J}$ *have full rank with smallest singular value bounded below by a positive constant.*

3. *The sequence of generated Lagrange multipliers* $[\boldsymbol{\lambda}_{\mathcal{E}} \boldsymbol{\lambda}_{\mathcal{I}}]$ *is bounded over all iterations.*

4. *The sequence of modified Hessian matrices* $\boldsymbol{W}$ *is bounded over all iterations and the Hessian modification strategy yields matrices such that* $\boldsymbol{W} - 2\theta \boldsymbol{I}$ *is positive definite after a finite number of modifications.*

5. *Each* $\boldsymbol{W}$ *yields regular KKT systems (2.13) and the linear solver is capable of solving each KKT system to an arbitrary accuracy.*

Then

$$\left\| \begin{pmatrix} \gamma(\boldsymbol{x}, \boldsymbol{s}; \mu) + \boldsymbol{J}(\boldsymbol{x})^\top \boldsymbol{\lambda} \\ \boldsymbol{c}(\boldsymbol{x}, \boldsymbol{s}) \end{pmatrix} \right\| \to 0.$$

3.2 SMART Test for a Stabilized Inexact Newton Method

The combination of Algorithms 2.1 and 3.1 with Algorithm 3.2 to compute an inexact step works very well for many practical problems. However, in cases where the constraint Jacobians are almost rank deficient the primal step length α may become very small. In [23] strategies to overcome this problem were introduced based on the decomposition of the step. In [24] these techniques were extended for IP methods. The advantage of increased stability and the opportunity to handle inequality constraints however is paid for by the necessity of solving two linear systems inexactly instead of only one. Therefore we change to this stabilized algorithm only when we have evidence Algorithm 3.2 produces

Algorithm 3.2 Inexact Newton Iteration with SMART Tests

```
1: (Initialization) Choose parameters $J \in \mathbb{N}_0$, $\kappa_{\mathrm{des}} \in (0,1)$, $\kappa \in (0,1)$, $\epsilon_1 \in (0,1)$,
   $\theta > 0$, $\zeta > 0$, $\tau \in (0,1)$, $\kappa_3 \in (0,1)$, $\epsilon_3 > 0$, $\kappa_W > 0$. Initialize $j \leftarrow 1$ and
   $(\boldsymbol{d}, \delta) \leftarrow (0,0)$.
2: (Residual test) If $j \leqslant J$ and $\Psi > \kappa_{\mathrm{des}}$, then go to step 7.
3: (Termination test 1) If $\Psi \leqslant \kappa$ and the model reduction condition
       $$\Delta m(\boldsymbol{d}; \mu, \pi) \geqslant \max\{1/2\, \boldsymbol{d}^\top \boldsymbol{W} \boldsymbol{d}, \theta \|\boldsymbol{d}\|^2\} + \epsilon_1 \pi \max\{\|\boldsymbol{c}\|, \|\boldsymbol{r_p}\| - \|\boldsymbol{c}\|\}$$
   holds, then terminate by returning $\binom{\boldsymbol{d}}{\delta}$ and the current $\pi$.
4: (Termination test 2) If the residual conditions
       $$\|\boldsymbol{r_d}(\boldsymbol{d}, \delta)\| \leqslant \kappa \|\boldsymbol{c}\| \quad \text{and} \quad \|\boldsymbol{r_p}(\boldsymbol{d})\| \leqslant \kappa \|\boldsymbol{c}\|$$
   are satisfied and the curvature condition $1/2\, \boldsymbol{d}^\top \boldsymbol{W} \boldsymbol{d} \geqslant \theta \|\boldsymbol{d}\|^2$ holds, then terminate
   by returning $\binom{\boldsymbol{d}}{\delta}$ and $\pi \leftarrow \max\{\pi, \pi^t + \zeta\}$, where
       $$\pi^t \leftarrow \frac{\nabla_{\boldsymbol{x},\boldsymbol{s}} \varphi(\boldsymbol{x}, \boldsymbol{s})^\top \boldsymbol{d} + 1/2\, \boldsymbol{d}^\top \boldsymbol{W} \boldsymbol{d}}{(1-\tau)(\|\boldsymbol{c}\| - \|\boldsymbol{r_p}(\boldsymbol{d})\|)}.$$
5: (Termination test 3) If the dual displacement and feasibility measures satisfy
       $$\|\boldsymbol{r_d}(\boldsymbol{0}, \delta)\| \leqslant \kappa_3 \|\nabla_{\boldsymbol{x},\boldsymbol{s}} \varphi(\boldsymbol{x}, \boldsymbol{s}) + \boldsymbol{J}^\top \boldsymbol{\lambda}\| \quad \text{and} \quad \|\boldsymbol{c}\| \leqslant \epsilon_3 \|\nabla_{\boldsymbol{x},\boldsymbol{s}} \varphi(\boldsymbol{x}, \boldsymbol{s}) + \boldsymbol{J}^\top \boldsymbol{\lambda}\|,$$
   then terminate by returning $(\boldsymbol{0}, \delta)$ (i.e., reset $\boldsymbol{d} \leftarrow \boldsymbol{0}$) and the current $\pi$.
6: (Hessian modification) If $\Psi \leqslant \kappa_W$ and $1/2\, \boldsymbol{d}^\top \boldsymbol{W} \boldsymbol{d} < \theta \|\boldsymbol{d}\|^2$, then increase the
   Hessian regularization parameter $\delta$, reset $j \leftarrow 1$ and $(\boldsymbol{d}, \delta) \leftarrow (0,0)$, and go
   to step 2.
7: (Search direction update) Perform one iteration of an iterative solver on (3.1)
   to compute an improved (approximate) solution $(\boldsymbol{d}, \delta)$. Increment $j \leftarrow j + 1$
   and go to step 2.
```

unproductive search directions. As an indicator for this, we check if the primal step length is larger than some lower bound $\alpha^{\min}$, which we have chosen as $\alpha^{\min} = 10^{-3}$ and use Algorithm 3.3 only in these cases. Indeed, the numerical examples show that this happens only very rarely.

According to the two goals of increasing feasibility and reducing the objective, the primal search direction $\boldsymbol{d}$ in the stabilized SMART test version is split into a *normal step* $\boldsymbol{n}$ and *tangential step* $\boldsymbol{t}$ such that

$$\boldsymbol{d} = \boldsymbol{n} + \boldsymbol{t},$$

where $\boldsymbol{J}\boldsymbol{t}$ is small (i.e., $\boldsymbol{t}$ lies in the kernel of $\boldsymbol{J}$ or close to it) and $\boldsymbol{n}$ is near the range of $\boldsymbol{J}^\top$. More specifically, the normal step $\boldsymbol{n}$ is computed as the inexact solution of the subproblem

$$\begin{aligned} \min_{\boldsymbol{n}} \, & {}^1\!/_2 \|\boldsymbol{r_p}(\boldsymbol{n})\|^2 \\ \text{s.t. } & \|\boldsymbol{n}\| \leqslant \omega \|\boldsymbol{J}^\top \boldsymbol{c}\|. \end{aligned} \tag{3.4}$$

An inexact solution of this problem is accepted if it satisfies a Cauchy decrease condition

$$\|\boldsymbol{c}\| - \|\boldsymbol{r_p}\| \geqslant \epsilon_{\boldsymbol{n}}(\|\boldsymbol{c}\| - \|\boldsymbol{r_p}(\alpha_C \boldsymbol{n}_S)\|), \tag{3.5}$$

where $\boldsymbol{n}_S = -\boldsymbol{J}^\top \boldsymbol{c}$ is the steepest descent direction of (3.4) and α_C the step length to minimize the objective function along $\boldsymbol{n}_S$ within the trust region. To find an inexact solution, we follow a dogleg approach: We first compute the Cauchy point $\boldsymbol{n}_C = \alpha_C \boldsymbol{n}_S$ and then apply an iterative solver to the Newton system of (3.4) which is very similar to (3.1) but with the (1,1) block replaced by an identity matrix and a different right-hand side. Once a normal component candidate $\boldsymbol{n}_N$ fulfills (3.5), the dogleg approximation $\boldsymbol{n}_D$ is computed as the point on the line between $\boldsymbol{n}_C$ and $\boldsymbol{n}_N$ which is closest to $\boldsymbol{n}_N$ in the trust region. Finally $\tilde{\boldsymbol{n}}_C$ and $\tilde{\boldsymbol{n}}_D$ are scaled back from $\boldsymbol{n}_C$ and $\boldsymbol{n}_D$, respectively, according to the fraction-to-the-boundary rule (2.15). In analogy to (3.5) we choose $\boldsymbol{n} \leftarrow \boldsymbol{n}_D$ if $\|\boldsymbol{c}\| - \|\boldsymbol{r_p}(\tilde{\boldsymbol{n}}_D)\| \geqslant \epsilon_{\boldsymbol{n}}(\|\boldsymbol{c}\| - \|\boldsymbol{r_p}(\tilde{\boldsymbol{n}}_C)\|)$ holds, and otherwise we set the Cauchy point $\boldsymbol{n} \leftarrow \boldsymbol{n}_C$.

Having found the normal component $\boldsymbol{n}$, we implicitly compute $\boldsymbol{t}$ by solving the system

$$\begin{pmatrix} \boldsymbol{r_d} \\ \boldsymbol{r_p} \end{pmatrix} = \begin{pmatrix} \boldsymbol{W} & \boldsymbol{J}^\top \\ \boldsymbol{J} & \boldsymbol{0} \end{pmatrix} \begin{pmatrix} \boldsymbol{d} \\ \boldsymbol{\delta} \end{pmatrix} + \begin{pmatrix} \boldsymbol{\gamma} + \boldsymbol{J}^\top \boldsymbol{\lambda} \\ \boldsymbol{J}\boldsymbol{n} \end{pmatrix}, \tag{3.6}$$

where in the second row we force $\boldsymbol{J}\boldsymbol{d} = \boldsymbol{J}\boldsymbol{n}$, i.e. $\boldsymbol{J}\boldsymbol{t} = \boldsymbol{0}$. Since we will solve (3.6) only inexactly, this equality will only hold up to a residual $\boldsymbol{r_p}$. Note that even for rank deficient Jacobians, this equation is solvable, which prevents a breakdown of the algorithm due to inconsistent linearized constraints.

To rate a tangential component candidate, similar ideas like for the inexact Newton system (3.1) are applied, but care must be taken for the search direction decomposition and the fact, that the normal component may not exactly be orthogonal to the linearized feasible plane. To do so, the quadratic model problem

$$\begin{aligned} \min_{\boldsymbol{t}} \, & (\boldsymbol{\gamma} + \boldsymbol{W}\boldsymbol{n})^\top \boldsymbol{t} + {}^1\!/_2 \, \boldsymbol{t}^\top \boldsymbol{W} \boldsymbol{t} \\ \text{s.t. } & \boldsymbol{J}\boldsymbol{t} = \boldsymbol{0} \end{aligned} \tag{3.9}$$

is considered in Algorithm 3.3, step 7. Note that a Newton step on (3.9) leads to the same KKT system (3.6) ans thus if $\boldsymbol{W}$ is positive definite on the null space of $\boldsymbol{J}$, they share the same solution.

The stabilized search direction computation with SMART tests is summarized in Algorithm 3.3. Global convergence with this algorithm has been proven in Theorem 3.13 in [24] which reads:

Algorithm 3.3 Regularized Inexact Newton Iteration with SMART Tests

```
1: (Initialization) Choose parameters $J \in \mathbb{N}_0$, $\kappa_{\text{des}} \in (0,1)$, $\psi > 0$, $\theta > 0$, $\kappa \in (0,1)$, $\epsilon_1 \in (0,1)$, $\epsilon_2 \in (0,1)$, $\zeta > 0$, $\tau \in (0,1)$, $\kappa_3 \in (0,1)$, $\epsilon_3 \in (0,1)$, $\kappa_W > 0$, and $\xi > 0$. Initialize $j \leftarrow 1$ and $(\boldsymbol{d}, \boldsymbol{\delta}) \leftarrow (\mathbf{0}, \mathbf{0})$.
2: (Normal step computation) Compute $\boldsymbol{n}$ by an inexact solution of (3.4) as described in the text.
3: (Residual test) If $j \leqslant J$ and $\Psi > \kappa_{\text{des}}$, then go to step 11.
4: (Direction decomposition) Set $\boldsymbol{t} \leftarrow \boldsymbol{d} - \boldsymbol{n}$.
5: (Tangential component test) If
       $\|\boldsymbol{t}\| \leqslant \psi \|\boldsymbol{n}\|$    (3.7)
   or if the inequalities
       $\tfrac{1}{2}\boldsymbol{t}^\top \boldsymbol{W} \boldsymbol{t} \geqslant \theta \|\boldsymbol{t}\|^2,$    (3.8)
       $(\nabla_{x,s}\varphi(\boldsymbol{x}, \boldsymbol{s})^\top + \boldsymbol{W}\boldsymbol{n})^\top \boldsymbol{t} + \tfrac{1}{2}\boldsymbol{t}^\top \boldsymbol{W} \boldsymbol{t} \leqslant 0$
   are satisfied, then continue to step 6; otherwise, go to step 9.
6: (Dual residual test) If the dual residual condition
       $\| \boldsymbol{r_d}(\boldsymbol{d}, \boldsymbol{\delta}) \| \leqslant \kappa \min \left\{ \left\| \begin{bmatrix} \nabla_{x,s}\varphi^\top + \boldsymbol{J}^\top \boldsymbol{\lambda} \\ \boldsymbol{J}\boldsymbol{n} \end{bmatrix} \right\|, \left\| \begin{bmatrix} \nabla_{x,s}\varphi_{old} + \boldsymbol{J}_{old}^\top \boldsymbol{\lambda} \\ \boldsymbol{J}_{old}\boldsymbol{n}_{k-1} \end{bmatrix} \right\| \right\}$
   is satisfied, then continue to 7; otherwise, go to step 9. Here $\nabla_{x,s}\varphi_{old}$ and $\boldsymbol{J}_{old}$ refer to values of the previous Newton iteration.
7: (Termination test 1) If the model reduction condition
       $\Delta m_k(\boldsymbol{d}; \mu, \pi) \geqslant \max\{\tfrac{1}{2}\boldsymbol{t}^\top \boldsymbol{W} \boldsymbol{t}, \theta \|\boldsymbol{t}\|^2\} + \epsilon_1 \pi (\|\boldsymbol{c}\| - \|\boldsymbol{r_p}(\boldsymbol{n})\|)$
   is satisfied, then terminate by returning $(\boldsymbol{d}, \boldsymbol{\delta})$ and the current $\pi$.
8: (Termination test 2) If the linearized constraint condition
       $\|\boldsymbol{c}\| - \|\boldsymbol{r_p}(\boldsymbol{d})\| \geqslant \epsilon_2 (\|\boldsymbol{c}\| - \|\boldsymbol{r_p}(\boldsymbol{n})\|) > 0$
   is satisfied, then terminate by returning $(\boldsymbol{d}, \boldsymbol{\delta})$ and $\pi \leftarrow \max\{\pi, \pi^t + \zeta\}$, where
       $\pi^t \leftarrow \dfrac{\nabla_{x,s}\varphi(\boldsymbol{x}, \boldsymbol{s})^\top \boldsymbol{d} + \tfrac{1}{2}\boldsymbol{t}^\top \boldsymbol{W} \boldsymbol{t}}{(1 - \tau)(\|\boldsymbol{c}\| - \|\boldsymbol{r_p}\|)}.$
9: (Termination test 3) If the dual displacement $\boldsymbol{\delta}$ yields
       $\|\boldsymbol{r_d}(\mathbf{0}, \boldsymbol{\delta})\| \leqslant \kappa_3 \min \left\{ \left\| \begin{bmatrix} \nabla_{x,s}\varphi^\top + \boldsymbol{J}^\top \boldsymbol{\lambda} \\ \boldsymbol{J}\boldsymbol{n} \end{bmatrix} \right\|, \left\| \begin{bmatrix} \nabla_{x,s}\varphi_{old} + \boldsymbol{J}_{old}^\top \boldsymbol{\lambda} \\ \boldsymbol{J}_{old}\boldsymbol{n}_{k-1} \end{bmatrix} \right\| \right\}$
   and the stationarity and dual feasibility measures satisfy
       $\|\boldsymbol{J}^\top \boldsymbol{c}\| \leqslant \epsilon_3 \|\nabla_{x,s}\varphi^\top + \boldsymbol{J}^\top \boldsymbol{\lambda}\|,$
   then terminate by returning $(\mathbf{0}, \boldsymbol{\delta})$ (i.e., reset $\boldsymbol{d} \leftarrow \mathbf{0}$) and the current $\pi$. Here $\nabla_{x,s}\varphi_{old}$ and $\boldsymbol{J}_{old}$ refer to values of the previous Newton iteration.
10: (Hessian modification) If $\Psi \leqslant \kappa_W$, but both (3.7) and (3.8) do not hold, then increase the Hessian regularization parameter $\delta$, reset $j \leftarrow 1$ and $(\boldsymbol{d}, \boldsymbol{\delta}) \leftarrow (\mathbf{0}, \mathbf{0})$, and go to step 3.
11: (Search direction update) Perform an iterative solver on (3.6) to compute an improved approximate solution $(\boldsymbol{d}, \boldsymbol{\delta})$. Increment $j \leftarrow j + 1$ and go to step 3.
```

Theorem 3.2. *Suppose, the sequence* $[\boldsymbol{x}, \boldsymbol{s}]$ *generated by Algorithms 2.1, 3.1 and 3.3 is contained in a convex set over which the following properties hold:*

1. *The objective function, the constraints, and their first derivatives are bounded and Lipschitz continuous.*
2. *The sequence of modified Hessian matrices* $\boldsymbol{W}$ *is bounded over all iterations and the Hessian modification strategy yields matrices such that* $\boldsymbol{W} - 2\theta \boldsymbol{I}$ *is positive definite after a finite number of modifications.*
3. *The linear solver is capable of solving each KKT system to an arbitrary accuracy.*

Let μ_j *denote the sequence of positive barrier parameters with* $\mu_j \to 0$*. Then one of the following statements hold:*

1. *During the outer iteration in Algorithm 2.1, it always holds that* $\|\boldsymbol{c}\|_\infty > \kappa_\varepsilon \mu_j$*. In this case the stationary condition (2.8) of the feasibility problem (2.7) is satisfied in the limit.*
2. *During an outer iteration in Algorithm 2.1, there exists an infinity inner iteration in Algorithm 3.3 where* $\|\boldsymbol{c}\|_\infty \leqslant \kappa_\varepsilon \mu_j$*, but (2.9) does not hold. Then stationary condition (2.8) of the feasibility problem (2.7) is satisfied in the limit and the merit parameter diverges,* $\pi \to \infty$*.*
3. *Each outer iteration results in an iterate* $(\boldsymbol{x}, \boldsymbol{s}, \boldsymbol{\lambda})$ *satisfying (2.9). In this case all limit points are feasible, and if a limit point satisfies the LICQ, the first-order optimality conditions from Theorem 2.2 of (2.1) hold.*

3.3 Step Length Computation

Globalization in the proposed algorithm is attained by a line search method. In contrast to the exact IP method, we do not apply a filter to accept step lengths but instead the Armijo rule is adopted on the merit function (3.2). This also overcomes the Maratos effect; it has been shown, that a SMART accepted search direction is indeed a descent direction of a local model of the merit function [17, Lemma 3], [24, Lemma 3.5].

Having found a search direction and a step length α, the primal variables are updated as

$$\boldsymbol{x}(\alpha) := \boldsymbol{x} + \alpha \boldsymbol{d}^x,$$
$$\boldsymbol{s}(\alpha) := \boldsymbol{s} + \alpha \boldsymbol{d}^s,$$

where the primal step length α needs to fulfill two conditions. First, a maximal step length according to a fraction-to-the-boundary rule (2.15) is determined. A backtracking line search then seeks a step length according to an Armijo-type rule for the merit function (3.2). The penalty parameter π is adapted automatically during the algorithm; see Algorithm 3.2, step 4 and Algorithm 3.3, step 8. With this merit function the step length condition is

$$\phi(\boldsymbol{x}(\alpha), \boldsymbol{s}(\alpha); \mu, \pi) \leqslant \phi(\boldsymbol{x}, \boldsymbol{s}; \mu, \pi) - \eta_2 \alpha \, \Delta m(\boldsymbol{x}, \boldsymbol{s}; \mu, \pi), \tag{3.10}$$

with $\eta_2 = 10^{-8}$. Here, we use the local model of the merit function reduction $\Delta m(\boldsymbol{d}; \mu, \pi)$ as introduced in (3.3).

For the dual variables, the step length β is set to the smallest value in $[\alpha, 1]$ that leads to a dual infeasibility reduction at least as large as a full Newton step, that is,

$$\beta = \min \left\{ \tilde{\beta} \in [\alpha, 1] \middle| \left\| \boldsymbol{c}(\boldsymbol{x}, \boldsymbol{s}, \boldsymbol{\lambda} + \tilde{\beta}\boldsymbol{\delta}) \right\| \leqslant \left\| \boldsymbol{c}(\boldsymbol{x}, \boldsymbol{s}, \boldsymbol{\lambda} + \boldsymbol{\delta}) \right\| \right\}. \tag{3.11}$$

Part II

Interior-Point Methods for PDE-Constrained Optimization

Chapter 4

Problem Formulation and Inherent Structure

Optimization of functions subject to partial differential equations (PDEs) plays an important role in many areas of science and industry. These optimization problems are also known as PDE-constrained optimization problems. The forward problem usually characterizes applications in which parameters of the PDE–initial conditions, boundary and domain sources, material coefficients, or domain boundary–are known, and the state variables are determined from the solution of the PDE. In PDE-constrained optimization the process is reversed; here we try to determine some PDE parameters to achieve goals in the form of an objective function and possibly inequality and equality constraints on the behavior of the system. Since the behavior of the system is modeled by a PDE, they appear as equality constraints in the optimization problem.

In this thesis we consider IP methods to solve PDE-constrained optimization problems. They are usually large-scale and in Part I two IP methods with exact and inexact step computations have been presented in a general optimization framework. In this part, we concentrate on the special structure of optimal control and parameter estimation, especially in the regime of multiple constraining PDEs.

The structure appears naturally by distinguishing independent PDE parameters from the dependent PDE solutions, the state variables. This will be discussed in Section 4.1. It is possible to eliminate the state variables which leads to the reduced-space approach discussed in Section 4.2.

4.1 Optimal Control and Parameter Estimation

We consider the general optimal control or parameter estimation problem that consists of multiple PDEs and, additional inequality constraints involving the state variable y or PDE parameter u:

$$\begin{aligned} &\min_{y,u} \mathcal{F}(y,u) \\ &\text{s.t.}\ \mathcal{A}_k(y_k,u) = 0, \qquad \text{for } k = 1,\dots,N_E, \\ &\qquad \mathcal{W}(y,u) \geqslant 0, \end{aligned}$$

where $\mathcal{A}_k(y_k,u)$ represent the kth PDE, $y = [y_1 y_2 \dots y_{N_E}]$ are the state variables that depend on the PDE parameter u, and N_E denotes the total number of continuous PDEs.

We consider both examples of optimal control and parameter estimation of PDE-constrained optimization problems. In *optimal control*, the *control* variable u appears in the PDEs on right-hand side only and the goal is to vary u such that the state variables y are as close as possible to desired states $\hat{y}_k$. Additional constraints like control box constraints or constraints on the state variable may account for technical limitations on the control u or desirable properties of the optimal state y. Such inequality constraints are denoted by $\mathcal{W}(y,u)$. The objective functional $\mathcal{F}$ is often a sum of two terms: The first one accounts for the misfit and typically consists of a convex function of norms of differences on a subset of the computational domain Ω. The second term, the *regularization term*, modifies the problem to be well-posed. In general the problem reads:

$$\begin{aligned} &\min_{y,u} \mathcal{F}(y,u) = \frac{1}{2}\sum_{k=1}^{N_E} |y_k - \hat{y}_k|^2 + \frac{\alpha}{2}\mathcal{R}(u) \\ &\text{s.t.}\ \tilde{\mathcal{A}}(y_k) = f_k(u), \qquad \text{for } k = 1,\dots,N_E, \\ &\qquad \mathcal{W}(y,u) \geqslant 0. \end{aligned}$$

Since the control variable u appears on the PDE's right-hand side only, a linear PDE and linear auxiliary constraints yields linear constraints in the optimization variable (y,u). For convex objective functionals, such optimal control problems with linear PDEs are convex optimization problems. In this thesis, we concentrate on nonconvex PDE-constrained optimization problems. Convex optimal control problems will be solved for comparison purposes, but most optimal control problems will contain some nonlinearity.

In *parameter estimation*, the PDE operator $\mathcal{A}_k$ itself depends on the *model* u. Here the goal is to find a model which is most consistent with measurements. The objective functional is of the same type as mentioned before and the problem reads

$$\begin{aligned} \min_{y,u} \mathcal{F}(y,u) &= \frac{1}{2}\sum_{k=1}^{N_E} |v(y_k) - \hat{y}_k|^2 + \frac{\alpha}{2}\mathcal{R}\,(u) \\ \text{s.t. } \tilde{\mathcal{A}}(u)\, y_k &= f_k, \qquad \text{for } k = 1,\ldots,N_E, \\ u_- &\leqslant u \leqslant u_+, \end{aligned}$$

where v denotes the evaluation functional. Parameter estimation problems are inherently nonconvex due to the mixed product in the PDE constraints. To remedy noise, multiple experiments may be measured leading to multiple PDE constraints. In the regime of parameter estimation additional inequality constraints usually consist of box constraints in the PDE parameters.

For both optimal control and parameter estimation, we specialize the auxiliary inequality constraints to the form

$$\mathcal{W}(y,u) = \widetilde{\mathcal{W}}(u) + \sum_{k=1}^{N_E} \widetilde{\mathcal{W}}_k(y_k,u), \tag{4.1}$$

and preclude cases with multiple states y_k in the same constraint. This is not a big restriction and most PDE-constrained problems fulfill (4.1).

In this work, we follow the discretize-then-optimize approach where the functions y, u, the objective functional $\mathcal{F}$, and partial differential operators $\mathcal{A}_k$ are discretized. The optimality conditions and algorithms are then applied to the finite-dimensional nonlinear optimization problem:

$$\begin{aligned} \min_{\boldsymbol{y},\boldsymbol{u}} \boldsymbol{F}(\boldsymbol{y},\boldsymbol{u}) &= \frac{1}{2}\sum_{k=1}^{N_E} \|\boldsymbol{V}\boldsymbol{y}_k - \hat{\boldsymbol{y}}_k\|^2 + \frac{\alpha}{2}\boldsymbol{R}\,(\boldsymbol{u}) \\ \text{s.t. } \boldsymbol{A}_k(\boldsymbol{y}_k,\boldsymbol{u}) &= \boldsymbol{0}, \qquad \text{for } k = 1,\ldots,N_E, \\ \boldsymbol{W}(\boldsymbol{y},\boldsymbol{u}) &\geqslant \boldsymbol{0}. \end{aligned} \tag{4.2}$$

4.2 Full-Space Approach and Reduced-Space Approach

4.2.1 Full-Space Approach

Applications of PDE-constrained optimization may utilize inequality constraints to improve optimization results by eliminating solutions that are physically unrealistic. The most promising methods for such large-scale problems are IP methods and active-set strategies. Since active-set algorithms exhibit combinatorial complexity with respect to the number of inequality constraints, they are not favorable for problems that need to satisfy a large number of inequality constraints.

For this reason our optimization framework will be based on IP methods. More precisely we adopt the IP methods from Part I and form the barrier subproblem

$$\begin{aligned}\min_{\boldsymbol{y},\boldsymbol{u},\boldsymbol{s}} \boldsymbol{F}(\boldsymbol{y},\boldsymbol{u},\boldsymbol{s};\mu) &:= \frac{1}{2}\sum_{k=1}^{N_E} \left\|\boldsymbol{V}\boldsymbol{y}_k - \hat{\boldsymbol{y}}_k\right\|^2 + \frac{\alpha}{2}\boldsymbol{R}(\boldsymbol{u}) - \mu \sum_{i=1}^{n_{\mathcal{I}}} \log(\boldsymbol{s}^{(i)}) \qquad (4.3)\\ \text{s.t. } \boldsymbol{A}_k(\boldsymbol{y}_k,\boldsymbol{u}) &= \boldsymbol{0}, \qquad \text{for } k = 1,\dots,N_E,\\ \boldsymbol{W}(\boldsymbol{y},\boldsymbol{u}) &= \boldsymbol{s},\end{aligned}$$

with $\boldsymbol{s} \in \mathbb{R}^{n_{\mathcal{I}}}$, $\boldsymbol{s} \geqslant \boldsymbol{0}$. In the context of the general optimization framework, we have

$$\boldsymbol{c}_{\mathcal{E}}(\boldsymbol{y},\boldsymbol{u},\boldsymbol{s}) = \begin{pmatrix} \boldsymbol{A}_1(\boldsymbol{y}_1,\boldsymbol{u}) \\ \vdots \\ \boldsymbol{A}_{N_E}(\boldsymbol{y}_{N_E},\boldsymbol{u}) \\ \boldsymbol{W}(\boldsymbol{y},\boldsymbol{u}) - \boldsymbol{s} \end{pmatrix}$$

with constraint Jacobian reads

$$\nabla \boldsymbol{c}_{\mathcal{E}}(\boldsymbol{y},\boldsymbol{u},\boldsymbol{s}) = \begin{pmatrix} \boldsymbol{J}_{\boldsymbol{y}_1} & & & \boldsymbol{J}_{1\boldsymbol{u}} & \\ & \ddots & & \vdots & \\ & & \boldsymbol{J}_{\boldsymbol{y}_{N_E}} & \boldsymbol{J}_{N_E\boldsymbol{u}} & \\ \boldsymbol{K}_{\boldsymbol{y}_1} & \cdots & \boldsymbol{K}_{\boldsymbol{y}_{N_E}} & \boldsymbol{K}_{\boldsymbol{u}} & -\boldsymbol{I} \end{pmatrix},$$

where $\boldsymbol{J}_{\boldsymbol{y}_k} = \nabla_{y_k}\boldsymbol{A}_k(\boldsymbol{y}_k,\boldsymbol{u})$, $\boldsymbol{J}_{k\boldsymbol{u}} = \nabla_u \boldsymbol{A}_k(\boldsymbol{y}_k,\boldsymbol{u})$, $\boldsymbol{K}_{\boldsymbol{y}_k} = \nabla_{y_k}\boldsymbol{W}(\boldsymbol{y},\boldsymbol{u})$, and $\boldsymbol{K}_{\boldsymbol{u}} = \nabla_u \boldsymbol{W}(\boldsymbol{y},\boldsymbol{u})$ denote the Jacobians of the PDE and auxiliary constraints with respect to state and parameter variables, respectively. If we assume that the discretized and linearized PDE matrices $\boldsymbol{J}_{\boldsymbol{y}_k}, k = 1,\dots,N_E$ are regular, then the Jacobian matrix has full row rank and the LICQ from (2.1) hold. This assumption is very mild, as for many problems, the discretization schemes guarantee regularity of the linearized discretized operator.

Thus, the solutions of (4.3) satisfy the first-order optimality conditions in Theorem 2.2, that is they are critical points of the full-space Lagrangian

$$\begin{aligned}\mathcal{L}^f(\boldsymbol{y},\boldsymbol{u},\boldsymbol{s},\boldsymbol{\lambda}) =& \frac{1}{2}\sum_{k=1}^{N_E} \left\|\boldsymbol{V}\boldsymbol{y}_k - \hat{\boldsymbol{y}}_k\right\|^2 + \frac{\alpha}{2}\boldsymbol{R}(\boldsymbol{u}) - \mu \sum_{i=1}^{n_{\mathcal{I}}} \log(\boldsymbol{s}^{(i)})\\ &+ \sum_{k=1}^{N_E} \boldsymbol{\lambda}_k^\top \boldsymbol{A}_k(\boldsymbol{y}_k,\boldsymbol{u}) + \boldsymbol{\lambda}_{\mathcal{I}}^\top (\boldsymbol{W}(\boldsymbol{y},\boldsymbol{u}) - \boldsymbol{s})\end{aligned}$$

and need to fulfill the KKT conditions

$$\begin{aligned}
\nabla_{y_k}\mathcal{L}^f(\boldsymbol{y},\boldsymbol{u},\boldsymbol{s},\boldsymbol{\lambda}) &= \boldsymbol{V}^\top\big(\boldsymbol{V}\boldsymbol{y}_k - \hat{\boldsymbol{y}}_k\big) + \boldsymbol{J}_{y_k}^\top\boldsymbol{\lambda}_k + \boldsymbol{K}_{y_k}^\top\boldsymbol{\lambda}_{\mathcal{I}} = \boldsymbol{0} \text{ for } k = 1,\dots,N_E, && (4.4a)\\
\nabla_{\boldsymbol{u}}\mathcal{L}^f(\boldsymbol{y},\boldsymbol{u},\boldsymbol{s},\boldsymbol{\lambda}) &= \frac{\alpha}{2}\nabla_{\boldsymbol{u}}\boldsymbol{R} + \sum_{k=1}^{N_E}\boldsymbol{J}_{ku}^\top\boldsymbol{\lambda}_k + \boldsymbol{K}_{\boldsymbol{u}}^\top\boldsymbol{\lambda}_{\mathcal{I}} = \boldsymbol{0}, &&\\
\nabla_{s}\mathcal{L}^f(\boldsymbol{y},\boldsymbol{u},\boldsymbol{s},\boldsymbol{\lambda}) &= -\mu\boldsymbol{S}^{-1}\boldsymbol{e} - \boldsymbol{\lambda}_{\mathcal{I}} = \boldsymbol{0}, &&\\
\nabla_{\lambda_k}\mathcal{L}^f(\boldsymbol{y},\boldsymbol{u},\boldsymbol{s},\boldsymbol{\lambda}) &= \boldsymbol{A}_k(\boldsymbol{y}_k,\boldsymbol{u}) = \boldsymbol{0} \quad \text{for } k = 1,\dots,N_E, && (4.4b)\\
\nabla_{\lambda_{\mathcal{I}}}\mathcal{L}^f(\boldsymbol{y},\boldsymbol{u},\boldsymbol{s},\boldsymbol{\lambda}) &= \boldsymbol{W}(\boldsymbol{y},\boldsymbol{u}) - \boldsymbol{s} = \boldsymbol{0}, &&
\end{aligned}$$

where $\boldsymbol{S} = \operatorname{diag}(\boldsymbol{s})$ denotes the diagonal matrix with entries according to the slack variables and $\boldsymbol{e}$ a vector with all its entries equal to one. Here, (4.4b) imposes the discretized PDEs and is called the *state equation*. In (4.4a), the transpose of the PDEs appears and thus it is called the *adjoint equation*.

As explained in Part I, we apply Newton's method that yields a search direction $[\boldsymbol{d}_y\ \boldsymbol{d}_u\ \boldsymbol{d}_s\ \boldsymbol{d}_{\lambda_{\mathcal{E}}}\ \boldsymbol{d}_{\lambda_{\mathcal{I}}}]$ by the solution of the linear system

$$\begin{pmatrix}
\nabla^2_{yy}\mathcal{L}^f + \delta\boldsymbol{I} & \nabla^2_{yu}\mathcal{L}^f & \boldsymbol{0} & \boldsymbol{J}_y^\top & \boldsymbol{K}_y^\top\\
\nabla^2_{yu}\mathcal{L}^{f\,\top} & \nabla^2_{uu}\mathcal{L}^f + \delta\boldsymbol{I} & \boldsymbol{0} & \boldsymbol{J}_u^\top & \boldsymbol{K}_u^\top\\
\boldsymbol{0} & \boldsymbol{0} & \boldsymbol{\Sigma}(\Xi + \delta\boldsymbol{I})\boldsymbol{\Sigma} & \boldsymbol{0} & -\boldsymbol{\Sigma}\\
\boldsymbol{J}_y & \boldsymbol{J}_u & \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0}\\
\boldsymbol{K}_y & \boldsymbol{K}_u & -\boldsymbol{\Sigma} & \boldsymbol{0} & \boldsymbol{0}
\end{pmatrix}
\begin{pmatrix}
\boldsymbol{d}_y\\ \boldsymbol{d}_u\\ \boldsymbol{d}_s\\ \boldsymbol{d}_{\lambda_{\mathcal{E}}}\\ \boldsymbol{d}_{\lambda_{\mathcal{I}}}
\end{pmatrix}
= -
\begin{pmatrix}
\nabla_{y_k}\mathcal{L}^f\\ \nabla_{\boldsymbol{u}}\mathcal{L}^f\\ \boldsymbol{\Sigma}\nabla_{\boldsymbol{s}}\mathcal{L}^f\\ \nabla_{\lambda_{\mathcal{E}}}\mathcal{L}^f\\ \nabla_{\lambda_{\mathcal{I}}}\mathcal{L}^f
\end{pmatrix}, \tag{4.5}$$

where Ξ denotes an approximation to the exact Hessian matrix (see also (2.14)) and $\boldsymbol{K}_y = \left(\boldsymbol{K}_{y_1} \cdots \boldsymbol{K}_{y_{N_E}}\right)$.

In the realm of nonconvex optimization problems, it may happen that the projection of the Hessian onto the null space of the Jacobian of the constraints is not positive definite. As a result, although the solution of (4.5) may still provide a feasible direction (up to first order), the objective along this direction is increasing instead of decreasing. To remedy this, we modify the first three diagonal blocks of the Hessian by adding a multiple of the identity matrix $\delta\boldsymbol{I}$ as described in Part I.

The matrix blocks in (4.5) are computed as follows: Denoting the ith component of the PDE constraint $\boldsymbol{A}_k$ by $\boldsymbol{A}_k^{(i)}$ and the of auxiliary constraint $\boldsymbol{W}$ by $\boldsymbol{W}^{(i)}$,

$$\nabla^2_{uu}\mathcal{L}^f = {}^{\alpha}\!/\!_{2}\nabla^2_{uu}\boldsymbol{R} + \sum_{k,i}\lambda_k^{(i)}\frac{\partial^2\boldsymbol{A}_k^{(i)}}{\partial\boldsymbol{u}^2} + \sum_i\lambda_{\mathcal{I}}^{(i)}\frac{\partial^2\boldsymbol{W}^{(i)}}{\partial\boldsymbol{u}^2} \tag{4.6}$$

is the Lagrangian Hessian matrix with respect to the PDE parameters and $\nabla^2_{uy}\mathcal{L}^f, \nabla^2_{yy}\mathcal{L}^f$ the Hessian matrices with respect to the state variables or with the mixed derivatives. Since the PDEs in (4.3) are decoupled, the objective is additive, and the auxiliary constraints $\boldsymbol{W}$ satisfy (4.1), the Lagrangian's Hessians block $\nabla^2_{uy}\boldsymbol{\mathcal{L}}^f$ and $\nabla^2_{yy}\boldsymbol{\mathcal{L}}^f$ reflect the PDE block structure, that is

$$\nabla^2_{uy}\boldsymbol{\mathcal{L}}^f = \left(\nabla^2_{uy_1}\boldsymbol{\mathcal{L}}^f \dots \nabla^2_{uy_{N_E}}\boldsymbol{\mathcal{L}}^f\right) \text{ with } \nabla^2_{uy_k}\boldsymbol{\mathcal{L}}^f = \sum_i \boldsymbol{\lambda}_k^{(i)} \frac{\partial^2 \boldsymbol{A}_k^{(i)}}{\partial \boldsymbol{u} \partial \boldsymbol{y}_k} + \sum_i \boldsymbol{\lambda}_{\mathcal{I}}^{(i)} \frac{\partial^2 \boldsymbol{W}^{(i)}}{\partial \boldsymbol{u} \partial \boldsymbol{y}_k} \tag{4.7}$$

and

$$\nabla^2_{yy}\boldsymbol{\mathcal{L}}^f = \begin{pmatrix} \nabla^2_{y_1y_1}\boldsymbol{\mathcal{L}}^f & & \\ & \ddots & \\ & & \nabla^2_{y_{N_E}y_{N_E}}\boldsymbol{\mathcal{L}}^f \end{pmatrix} \tag{4.8}$$

with

$$\nabla^2_{y_ky_k}\boldsymbol{\mathcal{L}}^f = \boldsymbol{V}^\top \boldsymbol{V} + \sum_i \boldsymbol{\lambda}_k^{(i)} \frac{\partial^2 \boldsymbol{A}_k^{(i)}}{\partial \boldsymbol{y}_k \partial \boldsymbol{y}_k} + \sum_i \boldsymbol{\lambda}_{\mathcal{I}}^{(i)} \frac{\partial^2 \boldsymbol{W}^{(i)}}{\partial \boldsymbol{y}_k \partial \boldsymbol{y}_k}.$$

The constraints' Jacobian w.r.t. the states

$$\boldsymbol{J}_y = \begin{pmatrix} \boldsymbol{J}_{y_1} & & \\ & \ddots & \\ & & \boldsymbol{J}_{y_{N_E}} \end{pmatrix} \tag{4.9}$$

is also a block diagonal matrix, where each diagonal block is the matrix obtained by the linearization of the discretized PDE. This structure is again displayed in Figure 4.1, where a PDE-constrained problem with five complex-valued PDE constraints in 2D are displayed and the blocks are inscribed.

4.2.2 Reduced-Space Approach

In the full-space approach, the number of optimization variables is large and may range in the hundreds of thousands or millions. To reduce the search space, *reduced-space methods* eliminate the state variable y. In doing so, an optimization problem with an easy objective function on an intricate manifold is transformed to an optimization problem with an intricate objective on an easy domain. The discrete reduced-space optimization problem reads

$$\begin{aligned} \min_{\boldsymbol{u}} \; & F(\boldsymbol{y}(\boldsymbol{u}), \boldsymbol{u}) = \frac{1}{2} \sum_{k=1}^{N_E} \|\boldsymbol{V}\boldsymbol{y}_k(\boldsymbol{u}) - \hat{\boldsymbol{y}}_k\|^2 + \frac{\alpha}{2}\boldsymbol{R}(\boldsymbol{u}) \\ \text{s.t.} \; & \boldsymbol{W}(\boldsymbol{y}(\boldsymbol{u}), \boldsymbol{u}) \geqslant \boldsymbol{0}, \end{aligned}$$

where $\boldsymbol{y}_k(\boldsymbol{u})$ denotes the solution of the kth PDE. The number of optimization variables reduces from $n_y + n_u$ to n_u, which is much less, say tens or hundreds of thousands. Furthermore, there are no more PDE constraints; they are now implicitly imposed by $\boldsymbol{y}(\boldsymbol{u})$. However, each objective function evaluation now

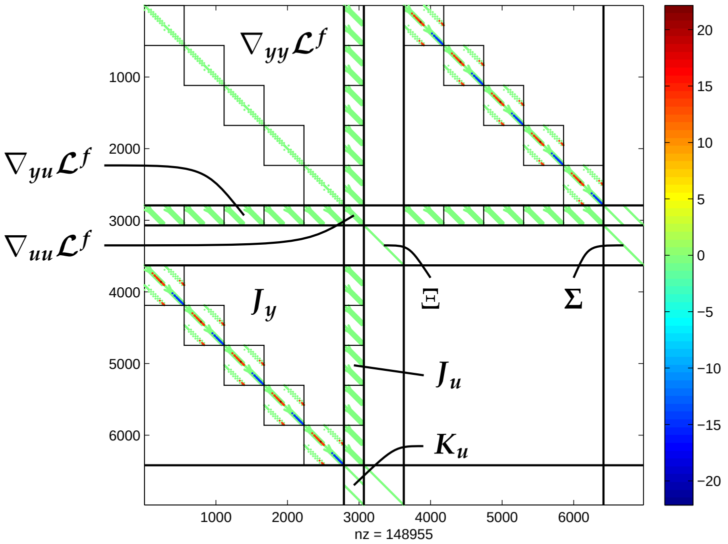


Figure 4.1: *Example KKT matrix: KKT matrix with its matrix blocks from equation* (4.5) *and subblocks from* (4.8) *and* (4.9) *(thin) for a rather small example with* $n_y = 2790, n_u = 279, n_{\mathcal{E}} = 2790, n_{\mathcal{I}} = 558$ *for a PDE-constrained optimization problem in 2D with five constraining complex-valued PDEs yielding a KKT matrix of size* 6975 *with almost* 150′000 *nonzeros.*

comprises exact solutions of the (nonlinear) PDE, while in the full-space approach, the objective evaluation was cheap. Following the IP framework from Section 2.1 yields a barrier subproblem

$$\begin{aligned} \min_{u} \; & F(y(u),u) = \frac{1}{2}\sum_{k=1}^{N_E} \|Vy_k(u) - \hat{y}_k\|^2 + \frac{\alpha}{2}R\,(u) - \mu\sum_{i=1}^{n_{\mathcal{I}}} \log(s^{(i)}) \\ \text{s.t.} \; & W(y(u),u) - s = 0 \end{aligned}$$

with the reduced-space Lagrangian

$$\mathcal{L}^r := \frac{1}{2}\sum_{k=1}^{N_E} \left\|Vy_k(u) - \hat{y}_k\right\|^2 + \frac{\alpha}{2}R\,(u) - \mu\sum_{i=1}^{n_{\mathcal{I}}} \log(s_i) + \lambda_{\mathcal{I}}^{\top}\left(W(y(u),u) - s\right).$$

Since the LICQ holds, first-order optimal points are KKT points, that is

$$\begin{aligned} \nabla_u \mathcal{L}^r &= \sum_{k=1}^{N_E}\left(\frac{\partial y_k}{\partial u}\right)^{\top} V^{\top}\left(Vy_k(u) - \hat{y}_k\right) + \frac{\alpha}{2}\nabla_u R(u) + \\ &\qquad \left(\frac{\partial W}{\partial u}^{\top} + \sum_{k=1}^{N_E}\frac{\partial y_k}{\partial u}^{\top}\frac{\partial W}{\partial y_k}^{\top}\right)\lambda_{\mathcal{I}} = 0, \qquad (4.10) \\ \nabla_s \mathcal{L}^r &= -\mu S^{-1}e - \lambda_{\mathcal{I}} = 0, \\ \nabla_{\lambda_{\mathcal{I}}} \mathcal{L}^r &= W(y(u),u) - s = 0 \end{aligned}$$

with

$$\frac{d\boldsymbol{A}_k(\boldsymbol{y}_k(\boldsymbol{u}),\boldsymbol{u})}{d\boldsymbol{u}} = \frac{\partial \boldsymbol{A}_k(\boldsymbol{y}_k,\boldsymbol{u})}{\partial \boldsymbol{u}} + \frac{\partial \boldsymbol{A}_k(\boldsymbol{y}_k,\boldsymbol{u})}{\partial \boldsymbol{y}_k}\frac{\partial \boldsymbol{y}_k}{\partial \boldsymbol{u}} = \boldsymbol{J}_{k\boldsymbol{u}} + \boldsymbol{J}_{\boldsymbol{y}_k}\frac{\partial \boldsymbol{y}_k}{\partial \boldsymbol{u}} = \boldsymbol{0},$$

and, assuming nonsingularity of the discretized and linearized PDE $\boldsymbol{J}_{\boldsymbol{y}_k} = \frac{\partial \boldsymbol{A}_k(\boldsymbol{y}_k,\boldsymbol{u})}{\partial \boldsymbol{y}_k}$, we have

$$\frac{\partial \boldsymbol{y}_k}{\partial \boldsymbol{u}} = -\boldsymbol{J}_{\boldsymbol{y}_k}^{-1}\boldsymbol{J}_{k\boldsymbol{u}}. \tag{4.11}$$

We apply a Newton-type method to achieve fast local convergence. For the ease of notation, we restrict ourself to the case $\boldsymbol{W}(\boldsymbol{y}(\boldsymbol{u}),\boldsymbol{u}) = \widetilde{\boldsymbol{W}}(\boldsymbol{u})$ and skip the Hessian modification. In each iteration, the KKT system

$$\begin{pmatrix} \nabla_{\boldsymbol{uu}}\boldsymbol{\mathcal{L}}^r & \boldsymbol{0} & \boldsymbol{K}_{\boldsymbol{u}}^\top \\ \boldsymbol{0} & \boldsymbol{\Sigma\Xi\Sigma} & -\boldsymbol{\Sigma} \\ \boldsymbol{K}_{\boldsymbol{u}} & -\boldsymbol{\Sigma} & \boldsymbol{0} \end{pmatrix} \begin{pmatrix} \boldsymbol{d}_u \\ \boldsymbol{d}_s \\ \boldsymbol{d}_{\lambda_{\mathcal{I}}} \end{pmatrix} = - \begin{pmatrix} \nabla_{\boldsymbol{u}}\boldsymbol{\mathcal{L}}^r \\ \boldsymbol{\Sigma}\nabla_{\boldsymbol{s}}\boldsymbol{\mathcal{L}}^r \\ \nabla_{\boldsymbol{\lambda}_{\mathcal{I}}}\boldsymbol{\mathcal{L}}^r \end{pmatrix}$$

needs to be solved. To compute the derivative of $\boldsymbol{J}_{k\boldsymbol{u}}^\top \boldsymbol{J}_{\boldsymbol{y}_k}^{-\top}\boldsymbol{V}^\top\left(\boldsymbol{V}\boldsymbol{y}_k(\boldsymbol{u}) - \hat{\boldsymbol{y}}_k\right)$ which appears in $\nabla_{\boldsymbol{u}}\boldsymbol{\mathcal{L}}^r$ of (4.10) we define

$$\boldsymbol{\lambda}_k = -\boldsymbol{J}_{\boldsymbol{y}_k}^{-\top}\boldsymbol{V}^\top\left(\boldsymbol{V}\boldsymbol{y}_k(\boldsymbol{u}) - \hat{\boldsymbol{y}}_k\right),$$

consistent with the full-space approach (4.4a), since $\boldsymbol{K}_{\boldsymbol{y}_k} = \boldsymbol{0}$. With this, we have

$$\begin{aligned} \frac{d\left(\boldsymbol{J}_{k\boldsymbol{u}}^\top\boldsymbol{\lambda}_k\right)}{d\boldsymbol{u}} &= -\frac{d\left(\sum_i \left(\frac{\partial \boldsymbol{A}_k^{(i)}}{\partial \boldsymbol{u}}\right)^\top \boldsymbol{\lambda}_k^{(i)}\right)}{d\boldsymbol{u}} \\ &= -\sum_i \left(\frac{\partial^2 \boldsymbol{A}_k^{(i)}}{\partial \boldsymbol{u}^2} + \frac{\partial^2 \boldsymbol{A}_k^{(i)}}{\partial \boldsymbol{u}\partial \boldsymbol{y}_k}\frac{\partial \boldsymbol{y}_k}{\partial \boldsymbol{u}}\right)\boldsymbol{\lambda}_k^{(i)} - \sum_i \frac{\partial \boldsymbol{A}_k^{(i)}}{\partial \boldsymbol{u}}\frac{\partial \boldsymbol{\lambda}_k^{(i)}}{\partial \boldsymbol{u}} \\ &= \sum_i \left(-\frac{\partial^2 \boldsymbol{A}_k^{(i)}}{\partial \boldsymbol{u}^2} + \frac{\partial^2 \boldsymbol{A}_k^{(i)}}{\partial \boldsymbol{u}\partial \boldsymbol{y}_k}\boldsymbol{J}_{\boldsymbol{y}_k}^{-1}\boldsymbol{J}_{k\boldsymbol{u}}\right)\boldsymbol{\lambda}_k^{(i)} - \boldsymbol{J}_{k\boldsymbol{u}}^\top\frac{\partial \boldsymbol{\lambda}_k}{\partial \boldsymbol{u}}. \end{aligned} \tag{4.12}$$

Defining

$$\boldsymbol{p} := \boldsymbol{J}_{\boldsymbol{y}_k}^\top\boldsymbol{\lambda}_k + \boldsymbol{V}^\top\left(\boldsymbol{V}\boldsymbol{y}_k(\boldsymbol{u}) - \hat{\boldsymbol{y}}_k\right) = \boldsymbol{0},$$

we see that

$$\frac{d\boldsymbol{p}}{d\boldsymbol{u}} = \sum_i \left(\frac{\partial^2 \boldsymbol{A}_k^{(i)}}{\partial \boldsymbol{y}_k\partial \boldsymbol{y}_k}\frac{\partial \boldsymbol{y}_k}{\partial \boldsymbol{u}} + \frac{\partial^2 \boldsymbol{A}_k^{(i)}}{\partial \boldsymbol{y}_k\partial \boldsymbol{u}}\right)\boldsymbol{\lambda}_k^{(i)} + \boldsymbol{J}_{\boldsymbol{y}_k}^\top\frac{\partial \boldsymbol{\lambda}_k}{\partial \boldsymbol{u}} + \boldsymbol{V}^\top\boldsymbol{V}\frac{\partial \boldsymbol{y}_k}{\partial \boldsymbol{u}} = \boldsymbol{0}$$

which yields

$$\frac{\partial \boldsymbol{\lambda}_k}{\partial \boldsymbol{u}} = \boldsymbol{J}_{\boldsymbol{y}_k}^{-\top}\left(\sum_i \left(\frac{\partial^2 \boldsymbol{A}_k^{(i)}}{\partial \boldsymbol{y}_k\partial \boldsymbol{y}_k}\boldsymbol{J}_{\boldsymbol{y}_k}^{-1}\boldsymbol{J}_{k\boldsymbol{u}} - \frac{\partial^2 \boldsymbol{A}_k^{(i)}}{\partial \boldsymbol{y}_k\partial \boldsymbol{u}}\right)\boldsymbol{\lambda}_k^{(i)} + \boldsymbol{V}^\top\boldsymbol{V}\,\boldsymbol{J}_{\boldsymbol{y}_k}^{-1}\boldsymbol{J}_{k\boldsymbol{u}}\right).$$

Plugging this into (4.12) together with (4.11) yields

$$
\begin{aligned}
\frac{d\left(\boldsymbol{J}_{ku}^{\top}\boldsymbol{\lambda}_k\right)}{d\boldsymbol{u}} &= -\boldsymbol{J}_{ku}^{\top}\boldsymbol{J}_{y_k}^{-\top}\boldsymbol{V}^{\top}\boldsymbol{V}\boldsymbol{J}_{y_k}^{-1}\boldsymbol{J}_{ku} \\
\sum_i \boldsymbol{\lambda}_k^{(i)} &\left(-\frac{\partial^2 \boldsymbol{A}_k^{(i)}}{\partial \boldsymbol{u}^2} + \frac{\partial^2 \boldsymbol{A}_k^{(i)}}{\partial \boldsymbol{u}\partial \boldsymbol{y}_k}\boldsymbol{J}_{y_k}^{-1}\boldsymbol{J}_{ku} + \boldsymbol{J}_{ku}^{\top}\boldsymbol{J}_{y_k}^{-\top}\frac{\partial^2 \boldsymbol{A}_k^{(i)}}{\partial \boldsymbol{y}_k\partial \boldsymbol{u}} - \boldsymbol{J}_{ku}^{\top}\boldsymbol{J}_{y_k}^{-\top}\frac{\partial^2 \boldsymbol{A}_k^{(i)}}{\partial \boldsymbol{y}_k\partial \boldsymbol{y}_k}\boldsymbol{J}_{y_k}^{-1}\boldsymbol{J}_{ku}\right).
\end{aligned}
$$

Comparing

$$
\nabla^2_{\boldsymbol{uu}}\boldsymbol{\mathcal{L}}^r = \sum_{k=1}^{N_E} -\frac{d\left(\boldsymbol{J}_{ku}^{\top}\boldsymbol{\lambda}_k\right)}{d\boldsymbol{u}} + \nabla^2_{\boldsymbol{uu}}\boldsymbol{R}(\boldsymbol{u}) + \sum_{i=1}^{n_{\mathcal{I}}}\frac{\partial^2 \boldsymbol{W}^{(i)}}{\partial \boldsymbol{u}^2}\boldsymbol{\lambda}_{\mathcal{I}}^{(i)}
$$

with the full-space Lagrangian's Hessians (4.6), (4.7), and (4.8), we see that

$$
\nabla_{\boldsymbol{uu}}\boldsymbol{\mathcal{L}}^r = \boldsymbol{J}_u^{\top}\boldsymbol{J}_y^{-\top}\nabla_{\boldsymbol{yy}}\boldsymbol{\mathcal{L}}^f\boldsymbol{J}_y^{-1}\boldsymbol{J}_u + \nabla_{\boldsymbol{uu}}\boldsymbol{\mathcal{L}}^f - \boldsymbol{J}_u^{\top}\boldsymbol{J}_y^{-\top}\nabla_{\boldsymbol{yu}}\boldsymbol{\mathcal{L}}^f - \nabla_{\boldsymbol{uy}}\boldsymbol{\mathcal{L}}^f\boldsymbol{J}_y^{-1}\boldsymbol{J}_u. \tag{4.13}
$$

For most PDE-constrained optimization problems, this matrix cannot be stored explicitly since $\nabla_{\boldsymbol{uu}}\boldsymbol{\mathcal{L}}^r$ is dense of size $n_{\boldsymbol{u}} \times n_{\boldsymbol{u}}$. Even if $n_{\boldsymbol{u}}$ is small, say several thousands, and $\nabla_{\boldsymbol{uu}}\boldsymbol{\mathcal{L}}^r$ could be stored, then to compute it, $n_{\boldsymbol{u}}$ discretized and linearized PDEs need to be solved to compute $\boldsymbol{J}_y^{-1}\boldsymbol{J}_u$. To cope with this, there are basically two strategies: The first strategy takes advantage of the fact that Krylov subspace methods can solve a linear system inexactly without storing the matrix explicitly; it suffices to apply the matrix to a vector. Another approach is to use approximations for the Hessian matrix based on first-order derivatives only. This is followed in Gauss-Newton or Quasi-Newton approaches as well as nonlinear CG methods like Fletcher-Reeves and Polak-Ribière.

Another disadvantage of reduced-space methods is that it is hard to cover additional constraints including the state variable. In Section 5.2 we will see how the ideas of reduced-space methods and full-space methods can be combined to cope with additional state constraints as well.

Chapter 5

Iterative Solution of the KKT System and Preconditioning

In Chapter 3 we have presented an IIP algorithm to avoid high memory requirements of direct linear solvers. Iterative linear solvers are an alternative to direct linear solvers. Krylov subspace methods are nowadays the most successful and commonly used iterative solvers. Their convergence depends on the eigenvalue distribution and can be improved significantly by preconditioners.

A preconditioner $\boldsymbol{M}$ is a linear mapping which aims to cluster the eigenvalues of a matrix, thereby making it more suitable for Krylov subspace methods. Instead of solving the system

$$\boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}$$

the linear system

$$\boldsymbol{M}^{-1}\boldsymbol{A}\boldsymbol{x} = \boldsymbol{M}^{-1}\boldsymbol{b} \quad \text{or} \quad \boldsymbol{A}\boldsymbol{M}^{-1}\boldsymbol{y} = \boldsymbol{b} \text{ with } \boldsymbol{x} = \boldsymbol{M}^{-1}\boldsymbol{y}$$

is solved. On the one extreme, $\boldsymbol{M} = \boldsymbol{A}$ would lead to convergence in one iteration but solution of the preconditioning system $\boldsymbol{x} = \boldsymbol{M}^{-1}\boldsymbol{y}$ would be as difficult as the original system. The other extreme is $\boldsymbol{M} = \boldsymbol{I}$, which does not precondition at all. The challenge is to find a preconditioner which is easy to solve and at the same time clusters the matrix spectrum, yielding a small number of Krylov subspace iterations.

5.1 Algebraic Multilevel Incomplete $LDL^\top$ Preconditioner

In this section, we discuss a general-purpose preconditioning approach and compare the IIP method from Chapter 3 with the exact IP method from Section 2.2. This work was published in [22].

Our approach is based on an algebraic multilevel preconditioner recently designed for symmetric highly indefinite systems. It uses symmetric maximum weight matchings to improve the block diagonal dominance of the system, followed by an inverse-based pivoting strategy to compute an inexact factorization. In order to bound the norm of the inverse, the factorization of some rows and columns might be postponed to the end. This leaves a Schur complement to which the procedure is applied recursively within a multilevel preconditioning framework.

In this section, we denote $\bar{\boldsymbol{A}} \in \mathbb{R}^{m\times m}$ as the matrix from (4.5). Symmetric weighted matching can be viewed as a preprocessing step that computes a positive diagonal matrix $\boldsymbol{D} \in \mathbb{R}^{m\times m}$ and a permutation matrix $\boldsymbol{P} \in \mathbb{R}^{m\times m}$ to obtain a new rescaled and reordered matrix

$$\hat{\boldsymbol{A}} = \boldsymbol{P}^{\top}\boldsymbol{D}\bar{\boldsymbol{A}}\boldsymbol{D}\boldsymbol{P} \tag{5.1}$$

such that the block diagonal dominance of the matrix is improved. All entries $\hat{a}^{(i,j)}$ of $\hat{\boldsymbol{A}}$ are at most one in size. Moreover, the diagonal blocks are either 1×1 scalars $\hat{a}^{(i,i)}$ with $|\hat{a}^{(i,i)}| = 1$ (in exceptional cases one has $\hat{a}^{(i,i)} = 0$) or symmetric 2×2 blocks

$$\begin{pmatrix} \hat{a}^{(i,i)} & \hat{a}^{(i+1,i)} \\ \hat{a}^{(i+1,i)} & \hat{a}^{(i+1,i+1)} \end{pmatrix} \text{ such that } |a^{(i,i)}|, |\hat{a}^{(i+1,i+1)}| \leqslant 1 \text{ and } |\hat{a}^{(i+1,i)}| = 1.$$

In addition, a further permutation based on nested dissection [47] (w.l.o.g. included in $\boldsymbol{P}$) is computed to reduce the fill-in in a sparse $\boldsymbol{L}\,\boldsymbol{D}\,\boldsymbol{L}^{\top}$ factorization. Numerical experiments in [61, 63, 65] indicate that this preprocessing step, in the context of computing a factorization of $\bar{A}$, makes dynamic pivoting strategies unnecessary.

Once the system is reordered and rescaled according to (5.1), the preconditioner is computed by an incomplete factorization $\boldsymbol{L}\,\boldsymbol{D}\,\boldsymbol{L}^{\top} = \hat{\boldsymbol{A}} + \boldsymbol{E}$ of $\hat{\boldsymbol{A}}$. Suppose that at step j of the factorization algorithm we have

$$\hat{\boldsymbol{A}} = \begin{pmatrix} \boldsymbol{B} & \boldsymbol{F}^{\top} \\ \boldsymbol{F} & \boldsymbol{C} \end{pmatrix} = \begin{pmatrix} \boldsymbol{L_B} & \boldsymbol{0} \\ \boldsymbol{L_F} & \boldsymbol{I} \end{pmatrix} \begin{pmatrix} \boldsymbol{D_B} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{S_C} \end{pmatrix} \begin{pmatrix} \boldsymbol{L}_{\boldsymbol{B}}^{\top} & \boldsymbol{L}_{\boldsymbol{F}}^{\top} \\ \boldsymbol{0} & \boldsymbol{I} \end{pmatrix},$$

where $\boldsymbol{L_B} \in \mathbb{R}^{j\times j}$ is unit lower triangular, $\boldsymbol{D_B} \in \mathbb{R}^{j\times j}$ is block diagonal with diagonal blocks of size 1×1 and 2×2, and $\boldsymbol{S_C} = \boldsymbol{C} - \boldsymbol{L_F}\boldsymbol{D_B}\boldsymbol{L}_{\boldsymbol{F}}^{\top}$ denotes the approximate Schur complement. Following [15, 62], one can easily estimate

$$\left\| \begin{pmatrix} \boldsymbol{L_B} & \boldsymbol{0} \\ \boldsymbol{L_F} & \boldsymbol{I} \end{pmatrix}^{-1} \right\| \leqslant \kappa_L \tag{5.2}$$

in every step for a prescribed bound $\kappa_L > 1$ based on a sparse adaption of the method presented in [20]. If at step j the approximate factorization fails to satisfy (5.2), then row and column j are permuted to the end. Otherwise we proceed with the approximate factorization where small entries in $\boldsymbol{L}$ are dropped according to a relative drop tolerance $\epsilon_L \in (0,1]$. When the approximate $\boldsymbol{L}\,\boldsymbol{D}\,\boldsymbol{L}^\top$ decomposition has finally passed all rows and columns of $\hat{\boldsymbol{A}}$, we are faced with a system of the form

$$\boldsymbol{Q}^\top \hat{\boldsymbol{A}}\, \boldsymbol{Q} = \begin{pmatrix} \boldsymbol{L}_{11} & \boldsymbol{0} \\ \boldsymbol{L}_{21} & \boldsymbol{I} \end{pmatrix} \begin{pmatrix} \boldsymbol{D}_{11} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{S}_{22} \end{pmatrix} \begin{pmatrix} \boldsymbol{L}_{11}^\top & \boldsymbol{L}_{21}^\top \\ \boldsymbol{0} & \boldsymbol{I} \end{pmatrix}.$$

The Schur complement $\boldsymbol{S}_{22}$, corresponding to all postponed updates, is then computed explicitly, and the strategies for reordering, scaling, and factoring are recursively applied to $\boldsymbol{S}_{22}$, leading to a multi-level factorization. To improve sparsity, small elements of the Schur complement $\boldsymbol{S}_{22}$ are dropped before the recursion according to a relative drop tolerance $\epsilon_S \in (0,1]$. Once the Schur complement has a sufficiently small size (less than 5000 rows and columns in our implementation) or a maximum number of levels is reached, the Schur complement is factored exactly. More details of the preconditioner can be found in [61, 64].

For the iterative solution of linear systems we use the symmetric quasi-minimum residual (SQMR) method [30] which has been found to work well with this preconditioner. Here, we allow a depth up to 30 in the multi-level approach, the constant bounding the norm of the inverse of the factor in (5.2) is chosen to be $\kappa_L = 2$, and the drop tolerances for the factor and the Schur complement are set to be $\epsilon_L = 10^{-2}$ and $\epsilon_S = 10^{-3}$, respectively. The SQMR method is allowed a maximum number of 1500 iterations. If this number is exceeded, then the preconditioner is recomputed with tightened drop tolerances (both divided by 3) and the iteration counter is reset. If necessary, the tolerances are tightened repeatedly. If an SMART test acceptable solution for the inexact Newton method in Algorithm 3.2 has not been computed after 4 such attempts, then the method reverts to the regularized inexact Newton method in Algorithm 3.3. If an acceptable solution for Algorithm 3.3 has not been computed after 4 such attempts, then the last computed inexact solution is used (without guarantees for a successful line search). In either of these latter two cases, before a new linear system is solved, the drop tolerances are multiplied by 3, though they are never set higher than the default values given above.

5.2 Reduced Space Preconditioned GMRES

In the previous section, we have presented, a general purpose multilevel algebraic incomplete $\boldsymbol{L}\,\boldsymbol{D}\,\boldsymbol{L}^\top$ preconditioner to solve (4.5). Even though the speedup compared to the exact IP method is remarkable, it could be improved even further, if the linear solver is tailored to the KKT system. In this section, we describe a linear solver, which takes advantage of the sparsity pattern of the KKT system as well as the fact that in Krylov subspace methods there is no need to form or store the matrix explicitly but instead it suffices to compute a matrix vector product. We also present several preconditioners including a very simple but efficient preconditioner based on the Schur complement, preconditioners following a multi-grid approach as well as preconditioners bases on ideas from simultaneous sources in seismic imaging. Parts of the work presented here were also described in [35]. The thesis author main contribution has been the simultaneous source preconditioning approach, the application of sparse approximate inverse preconditioners, as well as the development and implementation of the software, and performing the numerical experiments.

We now present out linear solver framework which is a reduced-space preconditioned Krylov subspace method. It is similar to Lagrange-Newton-Krylov methods presented in [55, Section 16.2] and [12, 13, 11]. The above methods, however, differ in the globalization strategy. In [55] the technique is presented for convex quadratic problems. Biros and Ghattas cope with nonconvexity by a Gauss-Newton approximation to the reduced-space Hessian. The resulting linear mapping is positive semidefinite and is preconditioned by the BFGS method which corresponds also to a positive definite mapping. Furthermore, these works do not include inequality constraints.

Here, we take advantage of exact Hessian matrices yielding fast local convergence. To cope with indefiniteness of the reduced Hessian and inexactness issues, we apply the SMART tests and Hessian modification. At each Newton iteration, it is required to solve the linear system (4.5). The matrix blocks $\nabla^2_{yy}\boldsymbol{\mathcal{L}}^f$ and $\nabla^2_{uu}\boldsymbol{\mathcal{L}}^f$ are symmetric and both $\nabla^2_{yy}\boldsymbol{\mathcal{L}}^f$ and $\boldsymbol{J}_y$ are block diagonal. Let $\mathbf{D}$ denote a reordered KKT matrix from (4.5),

$$\mathbf{D} = \begin{pmatrix} \nabla^2_{yy}\boldsymbol{\mathcal{L}}^f & \boldsymbol{J}_y^\top & \mathbf{0} & \nabla^2_{yu}\boldsymbol{\mathcal{L}}^f & \boldsymbol{K}_y^\top \\ \boldsymbol{J}_y & \mathbf{0} & \mathbf{0} & \boldsymbol{J}_u & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \boldsymbol{\Sigma\Xi\Sigma} & \mathbf{0} & -\boldsymbol{\Sigma} \\ \nabla^2_{yu}\boldsymbol{\mathcal{L}}^{f\top} & \boldsymbol{J}_u^\top & \mathbf{0} & \nabla^2_{uu}\boldsymbol{\mathcal{L}}^f & \boldsymbol{K}_u^\top \\ \boldsymbol{K}_y & \mathbf{0} & -\boldsymbol{\Sigma} & \boldsymbol{K}_u & \mathbf{0} \end{pmatrix},$$

where we omitted the Hessian modification for the ease of notation. This can also be written as

$$\mathbf{D} = \begin{pmatrix} \mathbf{Q} & \mathbf{B} \\ \mathbf{B}^\top & \mathbf{P} \end{pmatrix},$$

with

$$\mathbf{Q} = \begin{pmatrix} \nabla^2_{yy}\boldsymbol{\mathcal{L}}^f & \boldsymbol{J}_y^\top \\ \boldsymbol{J}_y & \mathbf{0} \end{pmatrix}, \mathbf{P} = \begin{pmatrix} \boldsymbol{\Sigma\Xi\Sigma} & \mathbf{0} & -\boldsymbol{\Sigma} \\ \mathbf{0} & \nabla^2_{uu}\boldsymbol{\mathcal{L}}^f & \boldsymbol{K}_u^\top \\ -\boldsymbol{\Sigma} & \boldsymbol{K}_u & \mathbf{0} \end{pmatrix}, \mathbf{B} = \begin{pmatrix} \mathbf{0} & \nabla^2_{yu}\boldsymbol{\mathcal{L}}^f & \boldsymbol{K}_y^\top \\ \mathbf{0} & \boldsymbol{J}_u & \mathbf{0} \end{pmatrix}.$$

In many applications of practical interest the size of $\mathbf{P}$ is much smaller than the size of $\mathbf{Q}$. For this type of problems it is particularly attractive to pursue the solution of the original linear system by solving the Schur-complement system with respect to the $(2,2)$ block of $\mathbf{D}$, $\mathbf{P}$, given by

$$G = \mathbf{P} - \mathbf{B}^\top \mathbf{Q}^{-1} \mathbf{B}.$$

The particular structure of $\mathbf{Q}$ allows its inverse to be written explicitly as follows:

$$\mathbf{Q}^{-1} = \begin{pmatrix} \mathbf{0} & J_y^{-1} \\ J_y^{-\top} & -J_y^{-\top} \nabla^2_{yy} \mathcal{L}^f J_y^{-1} \end{pmatrix}, \tag{5.3}$$

suggesting that solving linear systems with $\mathbf{Q}$ as the left-hand side matrix is trivial provided that a very robust and efficient linear solver exists for the inversion of J_y. So, assuming our original system reads

$$\begin{pmatrix} \mathbf{Q} & \mathbf{B} \\ \mathbf{B}^\top & \mathbf{P} \end{pmatrix} \begin{pmatrix} d_{y\lambda_\mathcal{E}} \\ d_{us} \end{pmatrix} = \begin{pmatrix} w_1 \\ w_2 \end{pmatrix}, \tag{5.4}$$

where $d_{y\lambda_\mathcal{E}}^\top$ denotes $(d_y^\top d_{\lambda_\mathcal{E}}^\top)$ and $d_{us}^\top$ for $(d_s^\top, d_u^\top, d_{\lambda_\mathcal{I}}^\top)$, we first transform it to

$$\begin{pmatrix} \mathbf{Q} & \mathbf{B} \\ \mathbf{0} & \mathbf{P} - \mathbf{B}^\top \mathbf{Q}^{-1} \mathbf{B} \end{pmatrix} \begin{pmatrix} d_{y\lambda_\mathcal{E}} \\ d_{us} \end{pmatrix} = \begin{pmatrix} w_1 \\ p_2 \end{pmatrix} \tag{5.5}$$

with $p_2 = w_2 - \mathbf{B}^\top \mathbf{Q}^{-1} w_1$, which is then solved by using Algorithm 5.1. We never form the dense Schur complement G in the third step of Algorithm 5.1 explicitly, but instead we use preconditioned GMRES.

Algorithm 5.1 Reduced-Space Preconditioned GMRES Solver (RSP-GMRES)

```
1: Solve $\mathbf{Q} p_1 = w_1$
2: Form $p_2 \leftarrow w_2 - \mathbf{B}^\top p_1$
3: Solve $G d_{us} = p_2$, where $G = \mathbf{P} - \mathbf{B}^\top \mathbf{Q}^{-1} \mathbf{B}$
4: Form $p_1 \leftarrow w_1 - \mathbf{B} d_{us}$
5: Solve $\mathbf{Q} d_{y\lambda_\mathcal{E}} = p_1$
```

Note that the Schur complement

$$G = \mathbf{P} - \mathbf{B}^\top \mathbf{Q}^{-1} \mathbf{B} = \begin{pmatrix} \mathbf{\Sigma \Xi \Sigma} & \mathbf{0} & -\mathbf{\Sigma} \\ \mathbf{0} & \nabla^2_{uu} \mathcal{L}^f + Z & K_u^\top + J_u^\top J_y^{-\top} K_y^\top \\ -\mathbf{\Sigma} & K_u + K_y J_y^{-1} J_u & \mathbf{0} \end{pmatrix}$$

with

$$Z = -\nabla^2_{uy} \mathcal{L}^f J_y^{-1} J_u - J_u^\top J_y^{-\top} \nabla^2_{yu} \mathcal{L}^f + J_u^\top J_y^{-\top} \nabla^2_{yy} \mathcal{L}^f J_y^{-1} J_u$$

is the reduced-space Lagrangian Hessian (4.13). Thus, with RSP-GMRES we solve the linear system in the reduced-space while we still perform the line search in the full-space.

In the line search the objective function is evaluated repeatedly at trial points. This is expensive in the reduced-space, where the objective function evaluation involves solutions of possibly nonlinear PDEs. In the full-space approach, PDEs do not need to be solved but it suffices to compute PDE residuals for the merit function evaluation which is much cheaper than solving PDEs. In contrast to the reduced-space approach, this reduced-space preconditioned IIP (RSP-IIP) method allows for additional constraints also involving the state variable y.

Due to the block diagonal structure of $\boldsymbol{J_y}$ the PDEs can be solved simultaneously. Often the PDEs $A_k(y_k, u)$ are linear in y_k and the diagonal blocks $\boldsymbol{J}_{y_k}, k = 1, \ldots, N_E$ are all the same. In such cases, we only need to compute and store the factorization of a single block, provided that a direct sparse solver has been employed, or if an iterative solver is used the preconditioner needs to be constructed only once for each KKT system. Thus the presented approach shows a linear run-time complexity with respect to the number of PDEs.

Moreover, the block diagonal structure of $\boldsymbol{J_y}$ is ideal for a distributed-memory solution of the related linear systems. Each diagonal block along with the corresponding right-hand sides, can be assigned to different nodes. Because of the fact that all these linear systems share the same sparsity structure, factorization and solution times are similar among all processors, which is essential for high scalability. The remaining operators involve sparse matrix-vector products and can be easily distributed and applied in parallel.

5.2.1 Reduced-Space Preconditioning

We now introduce different reduced-space preconditioners and compare them on the basis of a challenging inverse medium problem for the Helmholtz equation, which is described in more detail in Part III.

Block Preconditioner

The first preconditioner we apply in our RSP-GMRES solver is based on the Schur complement and we set

$$\boldsymbol{M}_{\mathbf{P}} = \mathbf{P}.$$

This preconditioner is very simple and it is not even guaranteed that $\boldsymbol{M}_{\mathbf{P}}$ is regular. However, as we will see in the numerical experiments, it performs well in practice. Since $\boldsymbol{M}_{\mathbf{P}}$ is sparse a direct solver can be used to factorize it. The factorization is computed at the beginning of each linear system solve, and during the GMRES iterations only cheap solutions of triangular systems need to be performed. An advantage of this preconditioner is that it scales linearly with the number of PDEs N_E.

Opt. step	$\boldsymbol{M}_{\mathbf{P}}$	$\boldsymbol{M}_{SPAI}$
1	73 (3)	177 (3)
2	150 (10)	254 (10)
3	223 (17)	327 (17)
4	271 (22)	393 (22)
5	267 (23)	379 (39)
6	453 (43)	578 (44)
7	651 (65)	864 (74)
8	670 (58)	805 (70)
9	670 (60)	866 (76)
10	858 (89)	1338 (123)
11	1123 (113)	1740 (169)
12	1220 (128)	2111 (206)
13	1181 (120)	2027 (200)
14	962 (100)	1189 (111)
15	1183 (124)	1724 (168)

Table 5.1: *Run-time (s) and number of RSP-GMRES iterations for RSP-GMRES with two different preconditioners to solve each KKT system of a Marmousi problem with one PDE.*

Sparse Approximate Inverse Preconditioner

More sophisticated preconditioners need to incorporate an approximation of the dense term $\mathbf{B}^\top\mathbf{Q}^{-1}\mathbf{B}$. Our first approach is a sparse approximation of this term. The fill-in is generated by the operators $\boldsymbol{J}_y^{-1}$ and $\boldsymbol{J}_y^{-\top}$ in (5.3). We approximate these inverse operators by sparse matrices [36], that is we find $\boldsymbol{T}_1 \approx \boldsymbol{J}_y^{-1}$ and $\boldsymbol{T}_2 \approx \boldsymbol{J}_y^{-\top}$, then form

$$\mathbf{Q}^{-1} \approx \boldsymbol{T}_3 = \begin{pmatrix} \mathbf{0} & \boldsymbol{T}_1 \\ \boldsymbol{T}_2 & -\boldsymbol{T}_2\nabla^2_{yy}\boldsymbol{\mathcal{L}}^f\boldsymbol{T}_1 \end{pmatrix},$$

and set $\boldsymbol{M}_{SPAI} = \mathbf{P} - \mathbf{B}^\top\boldsymbol{T}_3\mathbf{B}$. Since $\mathbf{P}, \mathbf{B}$, and $\boldsymbol{T}_3$ are sparse, we can form and factorize the small and sparse approximation $\boldsymbol{M}_{SPAI}$ explicitly at the beginning of each KKT system solve.

To compare the sparse approximate inverse preconditioner to the block preconditioner, we generate KKT matrices for the inverse medium problem (9.1) with $\omega = 20$ and only a single PDE. The state variable is discretized with bilinear Q^1 elements on a mesh with $h = 10\,\text{m}$ and the model parameters also with Q^1 elements on a mesh with $h = 40\,\text{m}$. The whole optimization run yields 15 KKT systems of size $n = 417'999$. Each linear system is solved with a MATLAB implementation of RSP-GMRES with both preconditioners, $\boldsymbol{M}_{\mathbf{P}}$ and $\boldsymbol{M}_{SPAI}$, to a tolerance of $tol = 10^{-4}$. The sparse approximations $\boldsymbol{T}_1$ and $\boldsymbol{T}_2$ are computed by the SPAI[2] library with default parameters except $eps = 0.2$. In table 5.1 we see the run-time in seconds and the number of RSP-GMRES iterations in parentheses. For the first KKT systems both preconditioners perform equally well in terms of the number of RSP-GMRES iterations. The costs of computing the preconditioner $\boldsymbol{M}_{SPAI}$ can be observed from the increased run-time. Starting from optimization step 5, we even see an increase of the number of RSP-GMRES iterations. This may occur when sparse approximation is not exact enough. For the matrix of optimization step 10, we computed the preconditioner also with an increased SPAI parameter $eps = 1$, which indeed led to a reduction of the run-time (1338 s to 1192 s) and iteration count (123 to 112 iterations). However, the block preconditioner remains superior in both number of iterations as well as run-time. An improvement in the sparse approximation up to $eps = 5$ did not lead to better results.

Two Level Preconditioner

In the previous approach, we approximated the dense block $\mathbf{B}^\top \mathbf{Q}^{-1} \mathbf{B}$ by a sparse matrix. In the two level approach, we approximate this term with a dense matrix, without forming it explicitly. This can be realized efficiently by use of the following fact: If in (5.4), $\boldsymbol{w}_1 = \mathbf{0}$ and $\boldsymbol{w}_2 = \boldsymbol{p}_2$, then $\boldsymbol{d}_{\boldsymbol{us}}$ also solves the Schur complement system (5.5) with some $\boldsymbol{d}_{y\lambda_\varepsilon}$. Thus, to implement

$$\boldsymbol{M}_{TL} = \mathbf{P} - \tilde{\mathbf{B}}^\top \tilde{\mathbf{Q}}^{-1} \tilde{\mathbf{B}} \approx \mathbf{P} - \mathbf{B}^\top \mathbf{Q}^{-1} \mathbf{B}$$

we set up the linear system

$$\tilde{\mathbf{D}}_{TL} = \begin{pmatrix} \tilde{\mathbf{Q}}_{TL} & \tilde{\mathbf{B}}_{TL} \\ \tilde{\mathbf{B}}_{TL}^\top & \mathbf{P} \end{pmatrix} \begin{pmatrix} \tilde{\boldsymbol{d}}_{y\lambda_\varepsilon} \\ \tilde{\boldsymbol{d}}_{\boldsymbol{us}} \end{pmatrix} = \begin{pmatrix} \mathbf{0} \\ \boldsymbol{p}_2 \end{pmatrix}. \tag{5.6}$$

Even though the approximation $\boldsymbol{M}_{TL}$ is dense, we only need to solve the sparse linear system (5.6). To reduce the size of the preconditioner compared to the original linear system (2.13), we follow a two level preconditioning approach; we assemble a smaller KKT system with the same PDE parameter but coarser state variable discretization. We increase the mesh size $h_{TL} = C_h \cdot h$ and reduce the finite element order $p_{TL} = p - \Delta p$. This smaller system is then solved by the multi-frontal direct solver `Pardiso`. For $C_h = 1$ and $\Delta p = 0$ we precondition with the same matrix which is equivalent to solving the Schur complement system directly. In Table 5.2 we see total run-time of the linear solver and total number of RSP-GMRES iterations (in parentheses) for different two level preconditioners applied to the inverse medium problem (9.1) with one PDE and with $\omega = 20$ and $\omega = 40$. On the fine level, the state variable was discretized on a $N_h \times N_h$ mesh with $N_h = 10\,\mathrm{m}$ and biquadratic elements. For comparison, we also added

---

[2] http://cccs.unibas.ch/lehre/software-packages

results when the linear system is solved with the direct solver `Pardiso`. To assure the same optimization path for all preconditioners, we solved all KKT system with a tolerance of 10^{-8}.

Due to this very small tolerance, the direct solver outperforms the iterative solver with block preconditioner by a factor of about 3. When incorporating information about the constraining PDEs, the preconditioner becomes more accurate and the number of RSP-GMRES iterations reduces remarkably. However, the preconditioner is now more expensive to apply and run-time reduces by a different amount. For $\omega = 20$ preconditioning with $C_h = 2, \Delta p = 0$, $C_h = 4, \Delta p = 0$, or $C_h = 1, \Delta p = 1$ (that is, precondition biquadratic elements with bilinear ones), the two level preconditioner outperforms the block preconditioner and is even faster than the direct solver. For the combination of h- and p-coarsening, $C_h = 2, \Delta p = 1$, leads to an increase in the iteration numbers and thus the run-time increases. For $\omega = 40$, only h-coarsening is effective, while for the cases with $\Delta p \neq 0$ the run-time increased significantly.

Linear solver	preconditioner			$\omega = 20$	$\omega = 40$
	Type	C_h	Δp	14 opt. steps	16 opt. steps
`Pardiso` *	-	-	-	10489 (14)	14734 (16)
RSP-GMRES	$\boldsymbol{M}_{\mathbf{P}}$	-	-	35037 (1430)	44775 (2299)
RSP-GMRES	$\boldsymbol{M}_{TLP}$	1	1	7556 (190)	37947 (857)
RSP-GMRES	$\boldsymbol{M}_{TLP}$	2	0	8827 (120)	10818 (151)
RSP-GMRES	$\boldsymbol{M}_{TLP}$	4	0	7155 (219)	13907 (460)
RSP-GMRES	$\boldsymbol{M}_{TLP}$	2	1	11960 (436)	57171 (2502)

Table 5.2: *Run-time of linear solver and number of RSP-GMRES iterations (in parentheses) to solve Marmousi problems with one PDE with differently preconditioned RSP-GMRES for varying temporal frequencies ω. The run-times for* `Pardiso` *are achieved by preconditioning with $C_h = 1$ and $\Delta p = 0$.*

For a single PDE constraint with $\omega = 20$ these numbers look promising. However, there are three major drawbacks. A further increased ω would reduce the efficiency of the two level preconditioner on the same mesh, as multi-grid approaches perform well only for small wave numbers for the Helmholtz equation. The preconditioner necessitates solutions of smaller KKT systems. However, for large-scale problems in 3D, it may not be affordable to factorize even this smaller KKT system. Furthermore, the preconditioner also becomes increasingly expensive with the numbers of PDEs N_E and thus it cannot be parallelized efficiently. Indeed, when the number of PDEs is increased, the preconditioner costs increase as well. We applied two preconditioners with $(C_h, \Delta p) = (1, 1)$ and $(2, 0)$ to the problem (9.1) with eleven PDEs and the KKT systems were solved until the SMART tests accepted the solution, which corresponds to $tol \approx 10^{-4}$. In Table 5.3 we list run-times and total RSP-GMRES iteration counts for the problem

state mesh	preconditioner	# opt. steps	run-time [s] (# RSP-GMRES it.)
$Q^1, h = 20\,\mathrm{m}$	$\boldsymbol{M}_{\mathbf{P}}$	10	825 (496)
$Q^1, h = 20\,\mathrm{m}$	$\boldsymbol{M}_{TL}, C_h = 2, \Delta p = 0$	10	21198 (179)
$Q^2, h = 40\,\mathrm{m}$	$\boldsymbol{M}_{\mathbf{P}}$	10	582 (482)
$Q^2, h = 40\,\mathrm{m}$	$\boldsymbol{M}_{TL}, C_h = 1, \Delta p = 1$	10	22192 (164)

Table 5.3: *Run-time of linear solver and number of RSP-GMRES iterations (in parentheses) to solve Marmousi problems with eleven PDEs with differently preconditioned RSP-GMRES for* $\omega = 20$*.*

with different discretizations. The number of iterations indeed decreases by a factor of more than two. However, the preconditioner is now very expensive to apply and the run-time increases by a factor of 25 for linear finite elements with h-coarsening and a factor of 14 for quadratic finite elements with p-coarsening.

Simultaneous Source Preconditioner

The main drawback of the two level preconditioner is the increasing complexity with respect to the number of PDEs N_E due to the increasingly larger matrix $\tilde{\mathbf{D}}_{TL}$. For linear PDEs this can be overcome by ideas from seismic data processing [37], where single PDE constraints are replaced by a linear combination of them, thus reducing the number of state variables and equality constraints. More precisely, let $w = (w_{ij})_{i,j=1}^{\widetilde{N_E}, N_E}$ denote the weights of source j for the new PDE constraint $i, i = 1, \dots, \widetilde{N_E}$, that is

$$\tilde{\boldsymbol{f}}_i := \sum_{j=1}^{N_E} w_{ij} \boldsymbol{f}_j \quad \text{for } i = 1, \dots, \widetilde{N_E}$$

Then a new optimization problem

$$\begin{aligned} \min_{\tilde{\boldsymbol{y}}, \boldsymbol{u}} F(\tilde{\boldsymbol{y}}, \boldsymbol{u}) &= \frac{1}{2} \sum_{k=1}^{N_E} \|\boldsymbol{V} \tilde{\boldsymbol{y}}_k - \hat{\tilde{\boldsymbol{y}}}_k\|^2 + \frac{\alpha}{2} \boldsymbol{R}(\boldsymbol{u}) \\ \text{s.t. } \tilde{\boldsymbol{A}}_k(\boldsymbol{u}) \tilde{\boldsymbol{y}}_k &= \tilde{\boldsymbol{f}}_k \qquad \text{for } k = 1, \dots \widetilde{N_E}, \\ \boldsymbol{u}_- &\leqslant \boldsymbol{u} \leqslant \boldsymbol{u}_+, \end{aligned} \tag{5.7}$$

with fewer PDEs constraints yields a smaller KKT system

$$\tilde{\mathbf{D}}_{SS} = \begin{pmatrix} \tilde{\mathbf{Q}} & \tilde{\mathbf{B}} \\ \tilde{\mathbf{B}}^\top & \tilde{\mathbf{P}} \end{pmatrix}.$$

To compute the constraint Jacobian and Hessian block, the state variable $\tilde{\boldsymbol{y}}$ and adjoint variable $\tilde{\boldsymbol{\lambda}}$ are necessary. According to the state equation (4.4b) and adjoint equation (4.4a) we set $\tilde{\boldsymbol{y}}_i = \sum_{j=1}^{N_E} w_{ij}\boldsymbol{y}_j$ and Lagrange multipliers $\tilde{\boldsymbol{\lambda}}_i = \sum_{j=1}^{N_E} w_{ij}\boldsymbol{\lambda}_j$. Note that the problem (5.7) and the original problem (4.2) don't share the same solution, since

$$\sum_{k=1}^{N_E} \boldsymbol{J}_{k\boldsymbol{u}}^\top \boldsymbol{\lambda}_k \neq \sum_{k=1}^{\widetilde{N_E}} \tilde{\boldsymbol{J}}_{k\boldsymbol{u}}^\top \tilde{\boldsymbol{\lambda}}_k$$

in general.

Following the approach in the two level preconditioner 5.2.1, we apply the Schur complement of $\tilde{\mathbf{D}}_{SS} = \left(\begin{smallmatrix} \tilde{\mathbf{Q}} & \tilde{\mathbf{B}} \\ \tilde{\mathbf{B}}^\top & \mathbf{P} \end{smallmatrix}\right)$, that is

$$\boldsymbol{M}_{SS} = \mathbf{P} - \tilde{\mathbf{B}}_{SS}^\top \tilde{\mathbf{Q}}_{SS}^{-1} \tilde{\mathbf{B}}_{SS}$$

as preconditioner. In a different context in [37] weights w_{ij} are chosen randomly with mean 0 in each iteration. Here, we fix the weights, since we want to precondition the linear system and no averaging over different optimization steps can occur. Furthermore, we would like to use a fixed preconditioner for a system to reuse the preconditioners factorization.

We applied this preconditioner to the problem (9.1) at $\omega = 20$ with 11 PDEs. The linear systems were solved until the current inexact solution was accepted by the SMART tests, which corresponds to $tol \approx 10^{-3} - 10^{-4}$. In Table 5.4 we list number of optimization steps, overall run-time, and total number of RSP-GMRES iterations. Here the additional matrix or vector index denotes the weight matrix w, i.e. $\boldsymbol{M}_{SS,{}^1\!/_{11}\boldsymbol{e}}$ denotes a preconditioner with $\widetilde{N_E} = 1$ and $w_{1j} = {}^1\!/_{11}, j = 1, \dots 11$, a " ± " in front of a vector denotes alternating signs in the columns, e.g., $\boldsymbol{M}_{SS,\pm\boldsymbol{e}}$ is a simultaneous source preconditioner with $w_{1j} = (-1)^j j = 1, \dots, 11$. By $\boldsymbol{M}_{SS,w_3}$ we mean a preconditioner with the weights

$$w_3 = \frac{1}{11}\begin{pmatrix} -1 & & & 1 & & & -1 & & & 1 & \\ & 1 & & & -1 & & & 1 & & & -1 \\ & & -1 & & & 0 & & & -1 & & \end{pmatrix}.$$

The weights in $\boldsymbol{M}_{SS,\pm\boldsymbol{e}}$, $\boldsymbol{M}_{SS,\pm 1/11\,\boldsymbol{e}}$, and $\boldsymbol{M}_{SS,w_3}$ are chosen to have a mean close to zero, as ideas from stochastic optimization suggest [37]. We see from Table 5.4 and Figure 5.1 that simultaneous sources indeed can reduce the number of RSP-GMRES iterations. The preconditioner $\boldsymbol{M}_{SS,w_3}$ performs best in this collection since 3 PDE constraints are used and most information of the PDEs is incorporated in the preconditioner. Preconditioning with $\boldsymbol{M}_{SS,\pm\boldsymbol{e}}$ led to an increase of RSP-GMRES iterations. Notably, the preconditioners increased the RSP-GMRES iterations most at the end of the optimization run, where most RSP-GMRES iterations are needed. This may be promising for problems, where the number of RSP-GMRES iterations are higher. However, in this example, the most efficient preconditioner was the block preconditioner $\boldsymbol{M}_{\mathbf{P}}$.

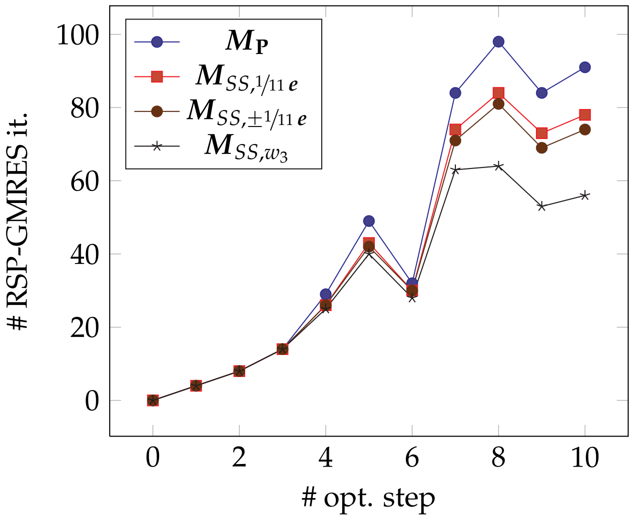


Figure 5.1: *RSP-GMRES for the parameter estimation problem* (9.1) *with* 11 *PDEs: comparison of block preconditioner* $\boldsymbol{M}_{\mathbf{P}}$ *and simultaneous source preconditioners* $\boldsymbol{M}_{SS,w}$ *with varying weights* w_{ij}*.*

preconditioner	# opt. steps	run-time (s) (# RSP-GMRES it.)
$\boldsymbol{M}_{\mathbf{P}}$	10	825 (496)
$\boldsymbol{M}_{SS,1/11\,\boldsymbol{e}}$	10	3995 (434)
$\boldsymbol{M}_{SS,\pm\boldsymbol{e}}$	10	17147 (2407)
$\boldsymbol{M}_{SS,\pm 1/11\,\boldsymbol{e}}$	10	3682 (419)
$\boldsymbol{M}_{SSw_3}$	10	7097 (355)

Table 5.4: *RSP-GMRES for problem* (9.1) *with* 11 *PDEs, for* $\omega = 20$*: number of optimization steps, overall run-time, and number of RSP-GMRES iterations (in parentheses) with different simultaneous source preconditioners.*

Chapter 6

Numerical Results

6.1 Software Implementation Aspects

The PDE-constrained optimization problems presented in this chapter and in Part III contain several millions of optimization variables. To work with such problems on a daily basis, fast and efficient implementations of various methods are needed. Since the target platform is a `Unix` based parallel distributed memory architecture we mainly used libraries tailored to MPI-parallel execution.

All PDE-constrained optimization problems are implemented in the programming language C++. The IP algorithms, both, exact and inexact, are implemented in the open source optimization package `IPOPT`.[3] This library is also available in an MPI-parallel version. Its open source character allows for extensions of internal software components, like we have done by the RSP-GMRES solver in Section 5.2. For the easy use of the parallel `IPOPT` version, we also contributed to the software package by an interface to the software package `PETSc`.

The major memory and computational expenses in PDE-constrained optimization are related to linear algebra objects like matrices and vectors and operations on them. For the parallel representation of vectors and matrices we used the open source library `PETSc`,[4] which has proven efficiency and scalability in many large-scale PDE applications [7].

The finite element discretization was implemented with the open source library `libMesh`.[5] This library can assemble into `PETSc` matrices and vectors. It provides classes for distributed unstructured meshes with various element types, $h-$ and $p-$mesh refinement, and many more tools which are frequently needed when working with finite elements. The interface to this library allows

---

[3] http://www.coin-or.org/Ipopt/

[4] http://www.mcs.anl.gov/petsc/

[5] http://libmesh.sourceforge.net/

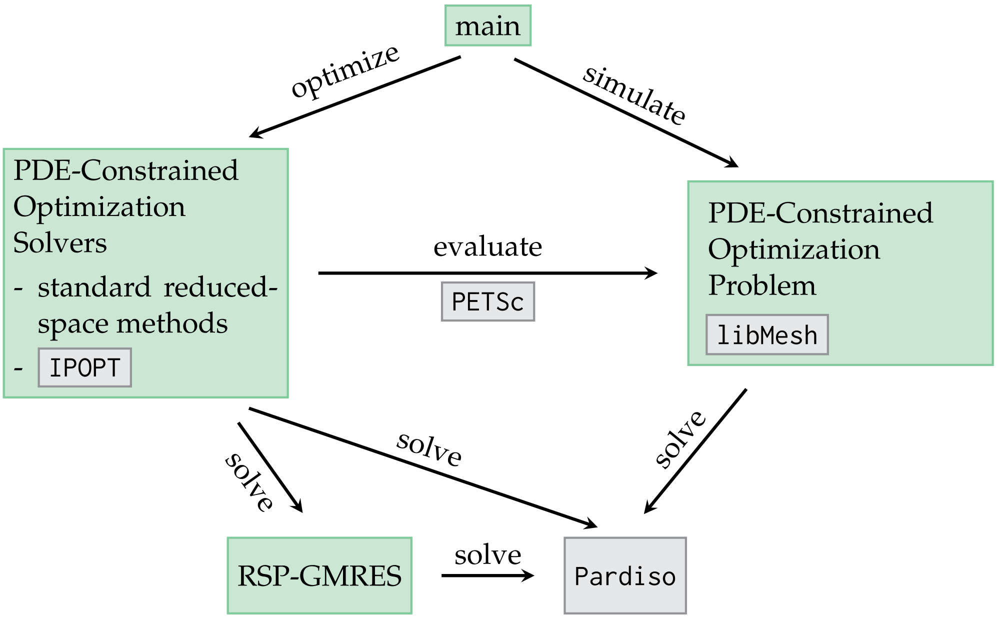

main
optimize
simulate
PDE-Constrained Optimization Solvers
- standard reduced-space methods
- IPOPT
evaluate
PETSc
PDE-Constrained Optimization Problem
libMesh
solve
solve
solve
RSP-GMRES
solve
Pardiso

Figure 6.1: *Interaction of different software components used to implement the PDE-constrained optimization problems in Chapter 6 and in Part III*

for application codes which are independent of the physical problem space dimension and also of the actual element order. The kernel of the matrix and vector assembling code can be written in a very intuitive manner.

The computationally most expensive cost of the whole process is the solution of linear systems. Here we used the shared memory parallel sparse solver library `Pardiso`,[6] which can be considered as the "working horse" in these programs. This library implements a sparse direct solver which is used in the exact IP algorithm and an algebraic multilevel incomplete $\boldsymbol{L}\boldsymbol{D}\boldsymbol{L}^\top$ preconditioner which is described in Section 5.1.

Figure 6.1 visualizes the interaction and calls of the software modules. Modules with major contribution from the thesis author are marked in green. The main routine creates an instance of a class which implements the PDE-constrained optimization problem at hand. If just a forward simulation is queried, its "SolveFwd" routine is called to solve the forward problem with given PDE parameters.

Normally the programs are called to solve an optimization problem. Then an optimizer object is generated which implements a standard reduced-space optimization algorithm (projected reduced gradient, projected limited memory BFGS or projected Polak-Ribière) or an `IPOPT`-solver. The standard algorithms are currently only available for problem (9.1). For an `IPOPT` optimization instance an `IPOPT-PETSc` interface is instantiated to convert assembled `PETSc` ma-

---

[6] http://www.pardiso-project.org/

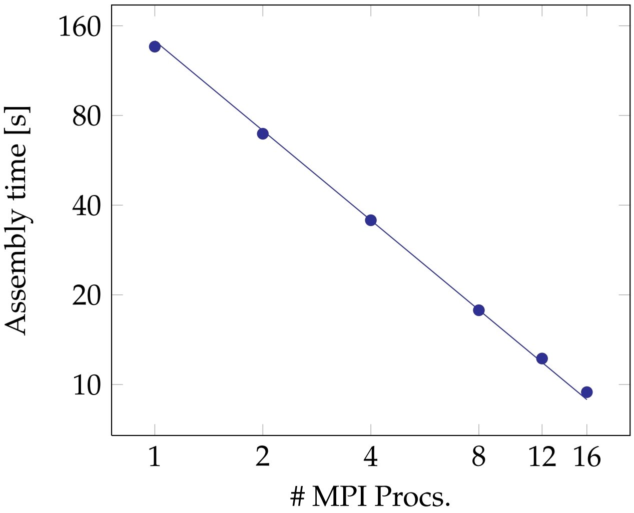


Figure 6.2: *Scaling of forward problem matrix assembly: Run-time to assemble a single simulation matrix* $\boldsymbol{A}(\boldsymbol{u})$ *in* (7.7) *with* $206'082$ *degrees of freedom.*

trices and `PETSc` vectors into the distributed matrix and vector format in `IPOPT` (a distributed triplet format). Also the concepts of control and state variables and of PDE constraints and auxiliary constraints are implemented in this interface class. The object is then accessed by `IPOPT` to evaluate the objective function, constraints, and their derivatives. `IPOPT` generates the matrices and vectors to set up the KKT system (4.5). It calls a linear solver, which in this thesis is either a RSP-GMRES solver, an algebraic multilevel incomplete $\boldsymbol{LDL}^\top$ preconditioned SQMR method, or a shared memory parallel sparse direct solver. The latter two are implemented in the `Pardiso` library. Also the RSP-GMRES solver calls `Pardiso` to solve PDEs, PDE adjoints, or the preconditioner. At the current state, we did not implement a distributed parallel linear solver yet. However, RSP-GMRES is easily parallelizable which is subject to future work.

We would like to emphasize, that the generated software programs are more than a compilation of the above libraries. Especially when it is about the finite element discretization, there are many issues we are confronted with and all of them need to be resolved in order to successfully solve a large-scale optimization problem. Finite element libraries are mainly tailored to simulation of PDEs instead of the solution of PDE-constrained optimization problems. One of those issues is, e.g., the occurrence of two different finite element meshes. For example, in the optimal boundary control problem in 6.2.2 a single mesh was used and additional degrees of freedom were added to the boundary nodes to discretize the control variable.

Another strategy is followed in the inverse medium problem in Part III, where a class for a general finite element discretized function is implemented.

Two instances are created, one for the control and one for the state variable, both with different meshes and different finite elements. For fast assembly, we concentrate on the case where each element of the state variable mesh is part of a single element of the control mesh. Otherwise, it would be necessary to generate a joint mesh with elements generated by the intersection of control and state mesh elements which can be very memory and time consuming in both runtime and development time. To assure, that the vector components for control and state variables of a certain element are available on a processor, the meshes and degrees of freedom of the state and control variables cannot be distributed independently. Instead, the control mesh is distributed first by the graph partitioning library `METIS` [46] and afterwards the state mesh is distributed such that each state element is saved on the same processor as its corresponding control element. To achieve this, a class ″AlignedPartitioner″ was derived from the ″Paritioner″ class in `libMesh` and an instance was passed to the mesh before its ″DoFMap″ distributes the degrees of freedom. Resolving those problems and implementing a solution involves an understanding of the libraries' internal structure and one has to work through the details of the library implementation.

As mentioned above, we implemented the PDE-constrained problems using these MPI-parallel libraries to be able to evaluate the objective function, the constraints, and their derivatives on distributed memory computers. As a sanity check we show in Figure 6.2 weak scaling of the matrix assembly for a 3D problem of (7.7) with a regular control mesh of only 10^3 hexahedral Q^1 elements and a state mesh consisting of 96′000 tetrahedral P^2 elements leading to 563′442 degrees of freedoms. The experiments were made on a shared memory 16 core machine and the codes scales very well, as expected.

6.2 Algebraic Multilevel Incomplete $LDL^\top$ Preconditioner

In this section we present numerical results, mainly on PDE-constrained optimization problems, for the IIP method described in Chapter 3 in combination with SQMR preconditioned by the general purpose algebraic multilevel incomplete $\boldsymbol{LDL}^\top$ decomposition from Section 5.1. For these experiments, we use revision 1954 of the `branches/parallel` development branch in `IPOPTs` `svn` repository. The linear systems are solved using the iterative linear system solvers and preconditioners implemented in the `Pardiso` software package version 4.1.1. The finite element discretization of the PDEs in Sections 6.2.2 and 6.2.3 was implemented using the open-source `libmesh` library [48], revision 3881 in its `trunk` branch, together with the PETSc library [6] version 3.1-p3. The 3D meshes for the example in Section 6.2.3 are generated with the `tetgen` software.[7]

---

[7] http://tetgen.berlios.de/

In `IPOPT`, we use the default parameter settings with a termination tolerance of $\varepsilon_{tol} = 10^{-6}$, together with the parameter choices given in the previous sections. The iterative linear solver in `Pardiso` uses the SQMR algorithm [30] with the preconditioner described above.

The numerical experiments in this section illustrate the performance of our implementation on a large nonlinear optimization test set and on two PDE-constrained problems. Overall, we show that the method is robust and provides improved computation times compared to the default `IPOPT` algorithm as problem sizes grow large. The results were obtained on 8-core Intel Xeon machines with 2.33 GHz clock speed and 32 GB RAM, running Ubuntu Linux with GNU 4.4.1 compilers. To avoid tainted CPU times caused by memory bus contention, we ran only one serial process at a time.

6.2.1 Standard Nonlinear Programming Test Sets

To assess the robustness of the algorithm we compare its performance with the exact IP method in Chapter 2.2 implemented in `IPOPT` on problems from the CUTEr test set [33, 34] for which AMPL models [28] are available.[8] We include all feasible problems that have at least one degree of freedom, are not unbounded, and which do not have inequality constraints with both lower and upper bounds in the formulation (the latter is a purely superficial limitation of our current implementation).

Since these problems are not very large, we changed the setting for the `Pardiso` preconditioner so that the multi-level strategy continues until the Schur complement matrix reaches the size 10 (instead of the default 5000). This has the effect that we will always obtain a multilevel iterative preconditioner for matrices that have more than 10 equations. Allowing a CPU time limit of 30 minutes and a limit of 3000 `IPOPT` iterations, the default algorithm in `IPOPT` [80] using a direct factorization with `PARDISO` and a filter line-search procedure is able to find a point satisfying the termination criteria in 592 out of a total of 617 optimization problems, giving a success rate of 96%. Note that some of the problems do not satisfy the regularity assumptions made for the global convergence analysis in [79]. Failures are also due to exceeding the iteration limit (12 cases), and to numerical issues caused by ill-conditioning. The CPU time limit was not reached for any problem by the default algorithm.

Algorithm 3.1 terminated successfully for 549 problems (89% success rate), exceeding the iteration limit in 12 and the CPU time limit in 20 cases. In the majority of the remaining cases, the algorithm broke down because no suitable preconditioner could be computed and the iterative linear system solver did not converge. We note that for 142 problems, the algorithm switched to Algorithm 3.3 at some point.

---

[8] http://orfe.princeton.edu/~rvdb/ampl/nlmodels/cute/

We also compare this performance with that of the original algorithm in [24], which always uses Algorithm 3.3 (i.e., it decomposes the step computation) in every `IPOPT` iteration. That method was successful for only 518 problems, yielding a success rate of 84%. The iteration limit was exceeded in 18 cases, and the CPU time limit was hit for 33 problems. This demonstrates that an increase in robustness was obtained for our implementation with the addition of the switching strategy described in chapter 3. Specifically, the switching strategy increased the percentage of successful solves from 84% to 89%.

We also note that our algorithm is able to solve the counterexample from [77] in 20 iterations, reverting to Algorithm 3.3 twice (for one iteration each time), following the adaptive step computation strategy described in Chapter 3. By contrast, the algorithm fails if steps are always computed using Algorithm 3.2 because the step sizes α_k converge to zero, as expected from the analysis in [77].

6.2.2 Optimal Boundary Control

Our first PDE-constrained optimization problem is an optimal control problem motivated by the "heating with radiation boundary conditions" example in Section 1.3.1 of [74]:

$$\min_{u,y} \int_\Gamma u \, da$$

s.t.

$$\begin{aligned} -\Delta y &= 0 && \text{in } \Omega, && (6.1a)\\ \frac{\partial y}{\partial n} &= \alpha(u - y^4) && \text{on } \Gamma, && (6.1b)\\ y &\geqslant y_j^{\min} && \text{in } \Omega_j \text{ for } j = 1, \dots, N_S, &&\\ u &\geqslant 0 && \text{on } \Gamma. && \end{aligned}$$

Here, y denotes temperature in a domain $\Omega \subseteq \mathbb{R}^3$, and the term $\frac{\partial y}{\partial n}$ denotes the outward-pointing normal derivative of the temperature on the boundary Γ of Ω. The boundary condition (6.1b) expresses the radiation heat loss according to the Stefan-Boltzmann law with a Stefan's constant $\alpha > 0$, where the control u dictates heat that can be resupplied on Γ. The goal is to minimize the amount of heat supplied while attaining a temperature of at least $y_i^{\min}$ within N_S subregions $\Omega_j \subseteq \Omega$. Following the common finite element approach, we multiply (6.1a) with a test function $v \in H^1(\Omega)$ and apply Green's formula together with (6.1b). The weak formulation of the PDE is then to find $y \in H^1(\Omega)$ such that

$$0 = -\int_\Omega \Delta y v \, dx = \int_\Omega \nabla y \cdot \nabla v \, dx - \alpha \int_\Gamma (y^4 - u) \, v \, da \quad \forall v \in H^1(\Omega). \qquad (6.2)$$

We generate a regular mesh of tetrahedrons, each with volume $h^3/24$ for a discretization parameter $h > 0$, and use the standard linear finite element basis functions $\{\varphi_i\}_{i=1,\dots,n_h}$. Projecting (6.2) onto the generated finite dimensional subspace V^h by approximating y with $y^h = \sum_i \boldsymbol{y}^{(i)}\varphi_i$ and u by $u^h = \sum_i \boldsymbol{u}^{(i)}\varphi_i$ (the latter requires discretized values $\boldsymbol{u}^{(i)}$ only corresponding to the boundary Γ), we solve the finite-dimensional problem

$$\min_{\boldsymbol{u}^{(i)},\boldsymbol{y}^{(i)}} \sum_i \boldsymbol{u}^{(i)} \int_\Gamma \varphi_i \, da$$

s.t.

$$\begin{aligned} 0 = & \int_\Omega \sum_i \boldsymbol{y}^{(i)} \nabla\varphi_i \cdot \nabla\varphi_j \, dx \\ & - \alpha \int_\Gamma \left((\sum_i \boldsymbol{y}^{(i)}\varphi_i)^4 - \sum_i \boldsymbol{u}^{(i)}\varphi_i \right) \varphi_j \, da \quad \text{for } j = 1,\dots,n_h \qquad (6.3a) \\ \boldsymbol{y}^{(i)} \geqslant \ & y_j^{\min} \quad \text{for } j \in \{1,\dots,N_S\} \text{ and } i \in \{\hat{i} \mid \exists x \in \Omega_j : \varphi_{\hat{i}}(x) = 1\} \qquad (6.3b) \\ \boldsymbol{u}^{(i)} \geqslant \ & 0. \end{aligned}$$

We choose $\alpha = 1$ and $\Omega = (0,1)^3$ and define two regions to be heated, $\Omega_1 = [0.1, 0.2] \times [0.05, 0.3] \times [0, 0.1]$ and $\Omega_2 = [0.8, 1] \times [0.75, 1] \times [0.7, 1]$, with associated threshold temperatures of $y_1^{\min} = 2.5$ and $y_2^{\min} = 2$. In (6.3b), we used the fact that a nodal finite element basis was chosen, so that $\max_{x\in\Omega} \varphi_i(x) = 1$, and for all $x \in \Omega$ we have $\sum_i \varphi_i(x) = 1$. Since $\nabla\varphi_i \cdot \nabla\varphi_j = \mathcal{O}(1/h^2)$ and $\int_E dx = \mathcal{O}(h^3)$ for a tetrahedron E, we multiply (6.3a) by $10^{-2}/h$ in our implementation, to ensure that the gradients of these constraints do not vanish as $h \to 0$. Similarly, the objective function was scaled internally by the factor $10^{-2}/(h^2)$.

We executed our implementation of the optimization algorithm for four choices of the discretization level. As initial point, we chose $\boldsymbol{y} = \boldsymbol{y}_{\text{init}}$ with $\boldsymbol{y}_{\text{init}} = 1.1(\boldsymbol{y}_1^{\min} + \boldsymbol{y}_2^{\min})$ and $\boldsymbol{u} = (\boldsymbol{y}_{\text{init}})^4$. Table 6.1 shows the discretization parameter (h), number of optimization variables (#var), number of simple bound constraints (#bds), number of equality constraints (#eq), and number of inequality constraints (#ineq) for various instances of this example. Tables 6.2 and 6.3 provide performance measures in the form of number of iterations (it), final objective value ($f(x_*)$), CPU seconds (CPUs), and CPU seconds per iteration (CPUs/it) for exact IP and for the IIP method. The last column in Table 6.3 shows the overall CPU time speedup of the inexact algorithm compared to the default method. Figure 6.3 shows the optimal solution for the finest discretization $h = 0.02$.

We clearly see a significant gain in computation speed that becomes more pronounced as the problem size increases. For the largest problem with a discretization parameter $h = 0.02$, the speedup is a factor of 7.85.

The tables list the average CPU time per iteration, but it should be noted that the step computation requires considerably more time towards the end of the optimization procedure than at the beginning. Taking the $h = 0.02$ case as

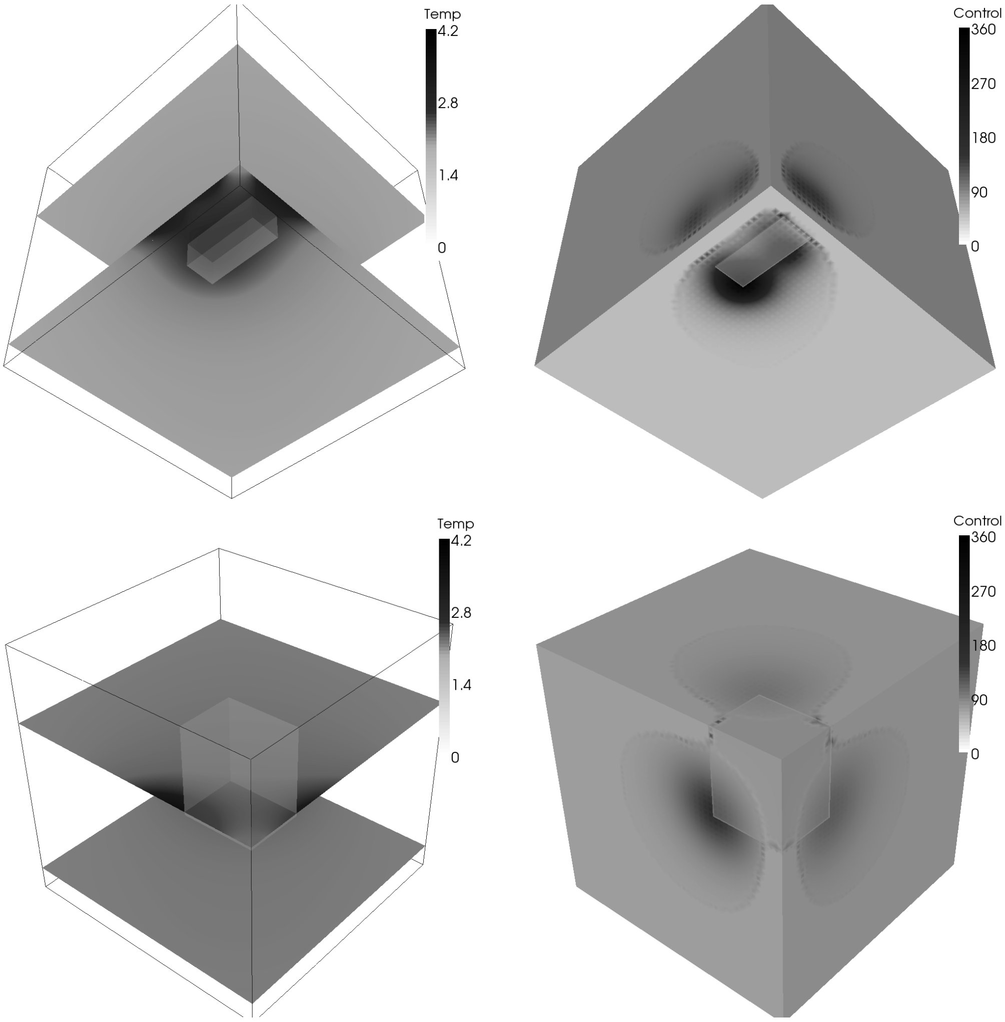


Figure 6.3: *Optimal state (left) and control (right) for the boundary control example. The regions Ω_1 (top) and Ω_2 (bottom) are visualized as a box. It is interesting to note that the corners of the regions Ω_1 and Ω_2 are heated most, instead of the inner part of its surface.*

h	#var	#bds	#eq	#ineq
0.05	47263	5724	42461	0
0.04	89453	9183	81951	0
0.03	199389	16498	186319	0
0.02	670153	42313	640151	0

Table 6.1: *Problem sizes for instances of the boundary control example.*

h	it	$f(x_*)$	CPUs	CPUs/it
0.05	29	40.6349	675.85	23.31
0.04	33	39.9458	2806.16	85.04
0.03	34	37.7909	16330.56	480.31
0.02	46	40.9115	304780.45	6625.66

Table 6.2: *Performance measures for the exact IP algorithm applied to the boundary control example.*

h	it	$f(x_*)$	CPUs	CPUs/it	speedup
0.05	33	40.6349	374.17	11.34	1.81
0.04	33	39.9458	646.77	19.60	4.34
0.03	37	37.7909	4495.83	121.51	3.63
0.02	47	40.9115	38824.33	826.05	7.85

Table 6.3: *Performance measures for the IIP method applied to the boundary control example.*

an example, in the first 22 `IPOPT` iterations the preconditioners (each computed in less than one minute) have fill-in factors of at most 3 and SQMR requires only between 35 and 200 iterations, leading to times of less than 4 minutes for each step computation. However, in the last `IPOPT` iterations, the dropping tolerances have to be tightened (down to about $3 \cdot 10^{-4}$ and $3 \cdot 10^{-5}$, respectively). At the tightest level of these tolerances, the preconditioner (computed in up to 9 minutes) has a fill-in factor of almost 10 and still SQMR requires more than 1000 iterations, leading to times up to 35 minutes at this level. Even though our results demonstrate significant improvements due to the use of iterative linear system solvers, this illustrates that finding preconditioners that are less dependent on the conditioning of the saddle point matrix in (4.5) as μ approaches zero is still an area of active research (see, e.g., [3, 9]).

6.2.3 Server Room Cooling

Our second example is motivated by the real-life problem of cooling computer equipment in a server room. In our simplified model, we assume that (cold) air is blown into the room from air conditioners (AC), and that (hot) air leaves the room at exhausts (Ex); see Figure 6.4. Inside the domain lies equipment with hot surfaces that need to be cooled by sufficient airflow passing alongside.

For simplicity, we suppose that air is incompressible, has no internal friction, and that all velocities are far below the speed of sound. Under these assumptions, we can model air velocity as the gradient of a potential $y(x)$ satisfying the

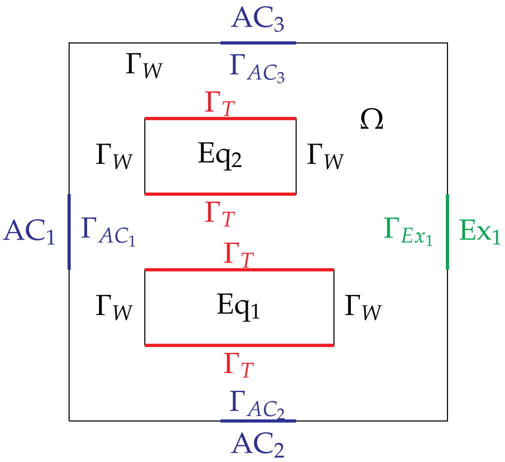


Figure 6.4: *Illustration of the geometry of the server room cooling model projected onto the $x_3 = 0$ axis. Cool air is pumped into the room via the AC units on the boundary in order to cool the hot surfaces of the equipment Γ_T. Air flows out of the room via the exhaust at Γ_{Ex_1}.*

Laplace equation

$$-\Delta y = 0 \qquad \text{in } \Omega \tag{6.4}$$

for a domain $\Omega \subseteq \mathbb{R}^3$. Appropriate boundary conditions for the walls (and non-heat producing surfaces of the equipment) Γ_W, cold air inlets Γ_{AC_i}, exhausts Γ_{Ex_i}, and heat producing surfaces of the equipment Γ_T, respectively, are

$$\begin{aligned} \frac{\partial y}{\partial n} &= 0 && \text{on } \Gamma_W, \\ \frac{\partial y}{\partial n} &= -u_{AC_i}\Psi_{\Gamma_{AC_i}} && \text{on } \Gamma_{AC_i}, \\ \frac{\partial y}{\partial n} &= +u_{Ex_i}\Psi_{\Gamma_{Ex_i}} && \text{on } \Gamma_{Ex_i}, \\ \frac{\partial y}{\partial n} &= 0 && \text{on } \Gamma_T; \end{aligned} \tag{6.5}$$

see also Figure 6.4. Here, $\frac{\partial y}{\partial n}$ denotes the outward-pointing normal derivative of the potential, and $\Psi_{\Gamma_{AC_i}}(x)$ $(i = 1, \ldots, N_{AC})$ and $\Psi_{\Gamma_{Ex_i}}(x)$ $(i = 1, \ldots, N_{Ex})$ define airflow velocity profiles on the surfaces of the air conditioners and exhausts, respectively. Similarly, $u_{AC_i} \in \mathbb{R}$ and $u_{Ex_i} \in \mathbb{R}$ denote control parameters for the maximal flow rates at these air inlets and outlets. The weak formulation of (6.4) with (6.5) is to find $y \in H^1(\Omega)$ such that

$$\begin{aligned} 0 &= -\int_\Omega \Delta y v \, dx \\ &= \int_\Omega \nabla y \cdot \nabla v \, dx + \sum_{i=1}^{N_{AC}} \int_{\Gamma_{AC_i}} u_{AC_i} \Psi_{\Gamma_{AC_i}} v \, da \\ &\qquad - \sum_{i=1}^{N_{Ex}} \int_{\Gamma_{Ex_i}} u_{Ex_i} \Psi_{\Gamma_{Ex_i}} v \, da \qquad\qquad \forall v \in H^1(\Omega). \end{aligned} \tag{6.6}$$

It is important to note that (6.6) has a solution only if the controls satisfy the mass balance equation

$$\sum_{i=1}^{N_{AC}} \int_{\Gamma_{AC_i}} u_{AC_i} \Psi_{\Gamma_{AC_i}} da - \sum_{i=1}^{N_{Ex}} \int_{\Gamma_{Ex_i}} u_{Ex_i} \Psi_{\Gamma_{Ex_i}} da = 0, \tag{6.7}$$

and in that case (6.6) only determines the potential $y \in H^1(\Omega)$ up to an additive constant. Therefore, a normalization condition will be introduced below.

As a constraint, we require that the air speed at the heat-producing surfaces has a minimum velocity so that heat is carried away. More precisely, recalling that the velocity is the gradient of the potential function y, we impose a pointwise state constraint

$$\|\nabla y(x)\|_2^2 \geqslant y_{\min}^2 \qquad \text{for all } x \in \Gamma_T \tag{6.8}$$

with a constant $y_{\min} > 0$.

To obtain the discretized problem, we generate an irregular mesh of tetrahedrons, each with maximal volume h^3, again choose a finite-dimensional subset $V^h \subseteq H^1(\Omega)$ with a basis $\{\varphi_i\}_{i=1,\dots,n_h}$, and express the finite-dimensional approximation $y_h = \sum_i \boldsymbol{y}^{(i)} \varphi_i$ of y with coefficients $\boldsymbol{y} \in \mathbb{R}^{n_h}$. Defining $u = (u_{AC}, u_{Ex})$ as the vector consisting of all control parameters, the discretized PDE (6.6) then becomes

$$\boldsymbol{A}\boldsymbol{y} - \boldsymbol{B}u = 0,$$

where $\boldsymbol{A}$ denotes the stiffness matrix $\boldsymbol{A}^{(i,j)} = \int_\Omega \nabla\varphi_i \cdot \nabla\varphi_j dx$, and $\boldsymbol{B} = [\boldsymbol{B}_{AC}\ \boldsymbol{B}_{Ex}]$ implements the boundary conditions with $\boldsymbol{B}_{AC}^{(i,j)} = -\int_{\Gamma_{AC_j}} \Psi_{\Gamma_{AC_j}} \varphi_i \, da$ and $\boldsymbol{B}_{Ex}^{(i,j)} = \int_{\Gamma_{Ex_j}} \Psi_{\Gamma_{AC_j}} \varphi_i \, da$.

Thus, the finite-dimensional optimization problem is

$$\min_{\boldsymbol{y}_i, u_i, \bar{u}} \sum \beta_j u_{AC_j}$$

s.t.

$$\begin{aligned} \boldsymbol{A}\boldsymbol{y} - \boldsymbol{B}u + \gamma e \bar{u} &= 0, && \text{(6.9a)}\\ \gamma e^\top \boldsymbol{y} - \bar{\gamma}\bar{u} &= 0, && \text{(6.9b)}\\ e^\top \boldsymbol{B} u &= 0, && \text{(6.9c)}\\ \int_{\Gamma_e} \nabla y_h(x) \cdot \nabla y_h(x)\, da - y_{\min}^2 \left(\int_{\Gamma_e} da \right) &\geqslant 0 \text{ for } \Gamma_e \subseteq \Gamma_T, && \text{(6.9d)}\\ u &\geqslant 0 \end{aligned}$$

with weights $\beta_i > 0$ in the objective function and $e = (1, \dots, 1)^\top \in \mathbb{R}^{n_h}$. Here, (6.9c) is a compact way of writing (6.7), and (6.9d) is the discretized version of (6.8), which is posed for all element faces Γ_e contained in a heat producing surface Γ_T. Note that the constraint (6.9d) is nonlinear and nonconvex. Again, in our implementation of the above problem, we scaled the constraints (6.9a) and (6.9d) by factors $10^{-2}/h$ and $10^{-1}/h$, respectively, to ensure that the gradients of those functions do not vanish as $h \to 0$.

To overcome the ill-posedness of the PDE, an auxiliary variable $\bar{u} \in \mathbb{R}$ has been added to the problem statement. Equation (6.9a) includes the discretized PDE, where the term $\gamma e \bar{u}$ acts as a constant virtual source or sink all over Ω. Since we impose the mass conservation in (6.9c) explicitly, this term eventually yields $\bar{u} = 0$. Furthermore, an integral-type equation is imposed in (6.9b). Indeed, $e^\top \boldsymbol{y}$ can be understood as a discretization of $\int_\Omega y\, d\mu$ for some measure μ depending on the finite-element discretization and is eventually set to zero in (6.9b) since $\bar{u} = 0$, therefore normalizing the velocity potential y. Arguing alternatively from a linear algebra point of view, while the linear system $\boldsymbol{A}\boldsymbol{y} = \boldsymbol{b}$ determining the state variables $\boldsymbol{y}$ is singular with e being an eigenvector corresponding to the eigenvalue 0, it can be shown that the linear system

$$\begin{pmatrix} \boldsymbol{A} & \gamma e \\ \gamma e^\top & -\bar{\gamma} \end{pmatrix} \begin{pmatrix} \boldsymbol{y} \\ \bar{u} \end{pmatrix} = \begin{pmatrix} \boldsymbol{b} \\ 0 \end{pmatrix}$$

is non singular and provides a solution satisfying $\boldsymbol{A}\boldsymbol{y} = \boldsymbol{b}$.

For our experiments we choose $\beta_i = 1, \gamma = 1, \bar{\gamma} = 10^8, y_{\min} = 1, \Gamma_{AC_1} = \{0\} \times [0.4, 0.6] \times [0.2, 0.4]$, $\Gamma_{AC_2} = [0.4, 0.6] \times \{0\} \times [0.2, 0.4]$, $\Gamma_{AC_3} = [0.4, 0.6] \times \{1\} \times [0.2, 0.4]$, and $\Gamma_{Ex_1} = \{1\} \times [0.4, 0.6] \times [0.6, 0.8]$. The equipment is placed so that $\Omega_{Eq_1} = [0.2, 0.7] \times [0.2, 0.4] \times [0, 0.8]$ and $\Omega_{Eq_2} = [0.2, 0.6] \times [0.6, 0.8] \times [0, 0.8]$ with the remaining boundary components Γ_T and Γ_W defined accordingly as illustrated in Figure 6.4. The airflows at the inlets and outlets are assumed to have quadratic profiles, e.g., on $\Gamma_{AC_i} = \{a^{(1)}\} \times [a^{(2)}, b^{(2)}] \times [a^{(3)}, b^{(3)}]$ we choose

$$\Psi_\Gamma(x) = \frac{4(x^{(2)} - a^{(2)})(b^{(2)} - x^{(2)})}{(b^{(2)} - a^{(2)})^2} \cdot \frac{4(x^{(3)} - a^{(3)})(b^{(3)} - x^{(3)})}{(b^{(3)} - a^{(3)})^2}.$$

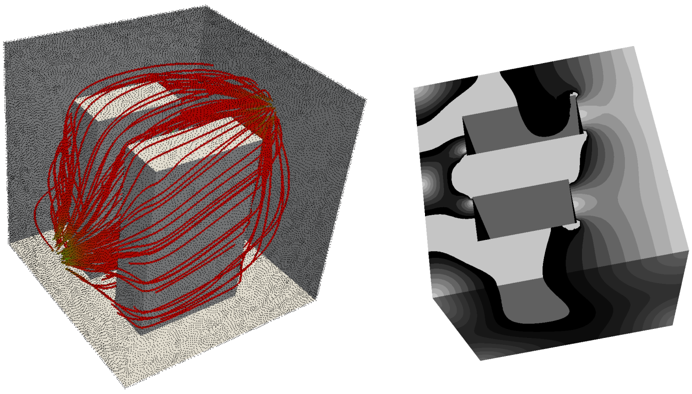


Figure 6.5: *Optimal solution of the server room cooling optimization example. On the left, we see the streamlines of the airflow, going from the main AC on the left to the exhaust on the right. On the right, we have a bottom view of the domain Ω, where the colors have been chosen to be dark if the air velocity is close to the threshold $y_{\min} = 1$. One can clearly see a region at the wall of the larger piece of equipment, at which the velocity is close to critical, indicating the location of the active constraints* (6.9d) *in* Γ_T.

Due to the nooks in Ω created by the equipment, numerical experiments with linear finite elements showed only linear L^2 convergence of the PDE solution as $h \to 0$. However, since (6.8) involves the gradient of the state variable, superlinear convergence is crucial. Thus, we have chosen quadratic finite elements and observed quadratic convergence for the PDE solution. Specifically, for three choices of the mesh size parameter $h = 0.2, 0.1, 0.05$, we computed the state variables from (6.9a)–(6.9b) for fixed values of the control parameters. Then we refined the mesh, corresponding to a value of $h/2$, and recomputed the state variables. The L^2-differences for the refined and original mesh were calculated as $3.87 \cdot 10^{-3}$, $1.04 \cdot 10^{-3}$ and $2.50 \cdot 10^{-4}$. Thus we observed factors of 3.7 and 4.2 for a bisection in h, which indicates quadratic convergence.

Table 6.4 shows problem size information for various instances of this problem. As the starting point for our experiments, we calculated the solution of (6.9a)–(6.9c) for $u_{AC_i} = 20$. Tables 6.5 and 6.6 provide performance measures for the default `IPOPT` algorithm and for our implementation, respectively, where now we break down the optimal objective values into those for the control variables u_{AC_i} for each of the three air conditioners. Also here we see a clear reduction in computation time achieved by using the inexact algorithm, without a loss in solution accuracy. Specifically, the computation time for the largest instance with more than 600′000 variables was reduced from more than 6 days to 8.2 hours, a speedup by a factor of 17.89. Figure 6.5 shows the optimal solution for the finest discretization.

In this example, the default settings for the preconditioner thresholds were

h	#var	#bds	#eq	#ineq
0.04	38582	4	38579	869
0.03	88398	4	88395	1528
0.02	285510	4	285507	3409
0.015	663886	4	663883	6110

Table 6.4: *Problem sizes for instances of the server room cooling example.*

h	it	$f(x_*)$	AC_1	AC_2	AC_3	CPUs	CPUs/it
0.04	23	15.4372	15.102	0.335	0.0	1019.16	44.31
0.03	21	15.5283	15.235	0.293	0.0	4511.48	214.83
0.02	33	15.5694	15.311	0.258	0.0	69427.33	2103.86
0.015	32	15.6509	15.428	0.223	0.0	528320.22	16510.01

Table 6.5: *Performance measures for the default algorithm applied to the server room cooling example.*

h	it	$f(x_*)$	AC_1	AC_2	AC_3	CPUs	CPUs/it	speedup
0.04	20	15.4372	15.102	0.335	0.0	622.31	31.12	1.64
0.03	24	15.5283	15.235	0.293	0.0	1710.30	71.26	2.64
0.02	28	15.5694	15.311	0.258	0.0	10008.50	357.45	6.94
0.015	27	15.6509	15.428	0.223	0.0	29526.53	1093.58	17.89

Table 6.6: *Performance measures for the inexact algorithm applied to the server room cooling example.*

sufficient in each iteration, so that no tightening occurred. For the $h = 0.015$ case, the computation time for the preconditioner ranged from 164 to 234 seconds (with an average of 204 seconds), the number of SQMR iterations was 204–1129 with an average of 396, and the time spent in SQMR ranged from 365 to 2018 seconds (with an average of 719 seconds). While there is some variation, we did not observe such a clear degeneration of computation time per iteration towards the end of the optimization as we saw for the example in 6.2.2.

6.3 Numerical Results for RSP-IIP with Block Preconditioner

We now present numerical results on PDE-constrained optimization problems for the IIP method from Chapter 3, where the linear systems (4.5) were solved by the RSP-GMRES method described in Section 5.2. As the comparison in that chapter showed best efficiency using the block preconditioner $\boldsymbol{M}_{\mathbf{P}}$, here we run all numerical experiments with this preconditioner.

To demonstrate the effectiveness of the IIP method with RSP-GRMES, we shall now apply it to four different PDE-constrained optimization problems. In the first two examples from optimal control, the control variable u appears on the right-hand side of the PDE constraint and attempts to steer the system's state y towards a desired state $\hat{y}$. In the last example from parameter estimation, the spatially distributed model parameter u appears inside the differential operator itself, which typically leads to a nonconvex inverse problem. Here, additional knowledge about the true model $\hat{u}$, such as inequality constraints or multiple measurements, not only reduces the number of false local minima, but also mitigates the effect of noise in the observations $\hat{y}$.

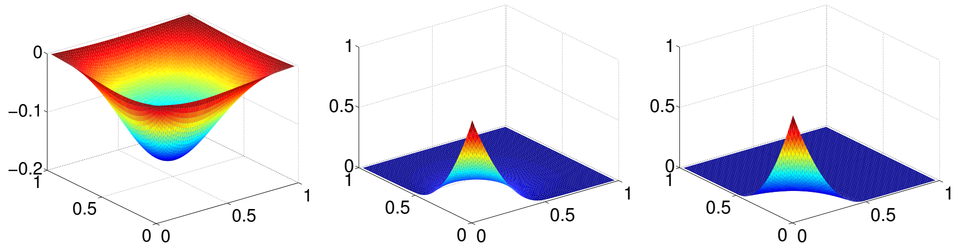
In all examples, we follow the "discretize-then-optimize" approach, where the objective function and the PDE-constraints are first approximated numerically before applying our IIP method to the resulting finite-dimensional nonlinear optimization problem. To determine the new search direction at each Newton iteration, we apply to the (linear) KKT system (4.5) the RSP-GMRES method preconditioned with the block preconditioner $\boldsymbol{M}_{\mathbf{P}}$, described in Section 5.2; in particular, prior to the GMRES iteration, the PDE block matrix $\boldsymbol{J}_{\boldsymbol{y}}$ in (5.3) is factorized.

Once the GMRES iterate meets the desired tolerance, it is evaluated by the SMART tests. If the termination tests in Algorithm 3.2 or Algorithm 3.3 accept the new search direction, the optimization method proceeds with the step length computation; see steps 7 and 13 of Algorithm 3.1. If a Hessian modification is required, the Hessian block of the KKT system is modified as in (4.5)—see also step 6 of Algorithm 3.2 and Step 10 of Algorithm 3.3—and the RSP-GMRES method is restarted. If none of the previous cases apply, the desired relative residual is reduced by a factor of 10 while the RSP-GMRES iteration and SMART acceptance procedure are repeated. Due to Lemma 3.2 in [24] this process will eventually terminate.

Both the IIP algorithm and the exact IP method, later used for the sake of comparison, have been implemented in the `IPOPT` open-source optimization package (rev. 2094). Unless noted otherwise, we always set the first desired relative residual to 10^{-2} and use default parameter values elsewhere for the inexact algorithm.

The RSP preconditioner is implemented in C++ and uses a linear solver based on an LU-factorization from the `Pardiso` software package (version 4.1.2.); again we use default settings unless noted otherwise. The IIP algorithm stops once the optimality conditions (2.10) are satisfied within a tolerance of 10^{-8} in the maximum norm.

All comparisons were preformed on an Intel Xeon architecture with 128 GB main memory using Intel's compiler version 10.1 under CentOS 5.8.



(a) *Optimal Control u^** **(b)** *Optimal State y^** **(c)** *Desired State $\hat{y}$*

Figure 6.6: *Corner load problem: optimal control u^*, optimal state y^*, and desired state $\hat{y}$ with $N_h = 64$.*

6.3.1 2D Distributed Control

To illustrate the efficiency of our RSP-IIP approach, in particular with respect to the mesh size h, we now consider a PDE-constrained quadratic program for which a multigrid based preconditioned projected conjugate gradient (PPCG) method was developed in [57]. Note that the PPCG algorithm requires a positive definite Hessian on the null space of the Jacobian in (4.5); therefore, it cannot be applied to more difficult nonconvex problems, as we shall discuss in Sections 6.3.2–6.3.3.

Hence, we consider the following convex optimal control problem, either with a "corner load" or a "centered load" desired state $\hat{y}$:

$$\begin{aligned} \min_{y,u} F(y,u) &= \frac{1}{2}||y-\hat{y}||^2_{L^2(\Omega)} + \frac{\alpha}{2}||u||^2_{L^2(\Omega)} \\ \text{s.t. } -\Delta y &= u \quad \text{in } \Omega = (0,1)^2, \\ y &= g \quad \text{on } \partial\Omega, \end{aligned}$$

with $\alpha = 0.02$. For the *corner load* problem we set $g = \hat{y}|_{\partial\Omega}$, where

$$\hat{y}(x_1,x_2) = \begin{cases} (2x_1-1)^2(2x_2-1)^2 & \text{if } (x_1,x_2) \in \left[0,\frac{1}{2}\right]^2, \\ 0 & \text{otherwise,} \end{cases}$$

whereas for the *centered load* problem we let $g = 0$ and

$$\hat{y}(x_1,x_2) = \exp\left(-\frac{(x_1-0.5)^2+(x_2-0.5)^2}{0.125^2}\right).$$

The state and control variables y,u are discretized on a regular $N_h \times N_h$ grid with Q_1 finite elements.

Next, we apply the RSP-IIP method with $\mu_{init} = 10^{-11}$ to both problems and compare run-times and iteration counts with those from the `MATLAB` implementation of the PPCG algorithm in [58]. The KKT systems (4.5) are solved iteratively either with PPCG or RSP-GMRES until the relative residual has reached

Table 6.7: *Corner load problem: Run-time (s) and iteration count (in parentheses) for PPCG and RSP-GMRES, with varying tolerance tol and mesh size $N_h \times N_h$, leading to a KKT system of size n. All optimization problems were solved within a single (inexact) optimization step.*

N_h	n	PPCG		RSP-GMRES	
		$tol = 10^{-6}$	$tol = 10^{-12}$	$tol = 10^{-6}$	$tol = 10^{-12}$
4	18	0.04 (3)	0.05 (4)	0.01 (3)	0.01 (5)
8	98	0.05 (3)	0.05 (4)	0.01 (3)	0.01 (5)
16	450	0.05 (2)	0.06 (4)	0.02 (3)	0.02 (5)
32	1′922	0.07 (2)	0.10 (4)	0.05 (3)	0.05 (5)
64	7′938	0.17 (1)	0.27 (3)	0.24 (3)	0.24 (5)
128	32′258	0.73 (1)	1.30 (3)	1.30 (3)	1.37 (5)
256	130′050	4.41 (1)	7.89 (3)	6.33 (3)	6.68 (5)
512	522′242	23.0 (1)	40.7 (3)	30.2 (3)	31.9 (5)

Table 6.8: *Centered load problem: Run-time (s) and iteration count (in parentheses) for PPCG and RSP-GMRES with varying tolerance tol and mesh size $N_h \times N_h$ leading to a KKT system of size n. All optimization problems were solved within a single inexact optimization step.*

N_h	n	PPCG		RSP-GMRES	
		$tol = 10^{-6}$	$tol = 10^{-12}$	$tol = 10^{-6}$	$tol = 10^{-12}$
4	18	0.04 (2)	0.04 (3)	0.01 (3)	0.01 (3)
8	98	0.05 (2)	0.05 (3)	0.01 (3)	0.01 (5)
16	450	0.05 (2)	0.05 (3)	0.02 (3)	0.02 (5)
32	1′922	0.07 (2)	0.08 (3)	0.05 (3)	0.05 (5)
64	7′938	0.22 (2)	0.26 (3)	0.24 (3)	0.31 (5)
128	32′258	0.99 (2)	1.24 (3)	1.30 (3)	1.37 (5)
256	130′050	5.54 (2)	7.20 (3)	6.35 (3)	6.70 (5)
512	522′242	27.2 (2)	35.7 (3)	30.3 (3)	31.8 (5)

a desired tolerance $tol = 10^{-6}$ or 10^{-12}. Regardless of problem size, the IIP algorithm, though inexact, always found a solution within a single optimization step, thus demonstrating the remarkable accuracy of its search direction.

In Tables 6.7 and 6.8 we compare run-times and iteration counts (in parentheses) of the PPCG and the RSP-IIP method to solve the KKT system for varying mesh size and tolerance. In all cases, the number of iterations remains independent of N_h; hence, our RS preconditioner also exhibits optimal h-independent behavior. The run-times of both algorithms are comparable, while the iteration counts differ by at most two. Recall, however, that the PPCG method cannot be applied to more general nonconvex problems, possibly with inequality constraints, as we shall discuss in the following.

6.3.2 2D Boundary Control Problem

Next, to demonstrate the robustness of the RSP-IIP method with respect to increasing mesh size and number of PDE constraints, we consider a 2D boundary control problem (example 5.7 in [51]) with multiple nonlinear PDEs and inequality constraints. Since the PPCG algorithm no longer applies here, we shall compare the RSP-IIP method to the exact IP method from [80] and to the IIP method with preconditioned SQMR from Section 6.2 instead, where the KKT systems are solved by the sparse direct solver `Pardiso` [63]. By exploiting the sparsity structure of the KKT system, the RSP-IIP method achieves a significant speedup over the exact IP method, even more so as the KKT system increases. Hence, we consider the optimal boundary control problem:

$$\begin{aligned}
&\min_{y,u} F(y,u) = \frac{1}{2N_E}\sum_{k=1}^{N_E} ||y_k - \hat{y}_k||^2_{L^2(\Omega)} + \frac{\alpha}{2}||u||^2_{L^2(\partial\Omega)} \\
&\text{s. t. } -\Delta y_k - y_k + y_k^3 = 0 \quad \text{in } \Omega = (0,1)^2, \qquad (6.10a)\\
&\qquad \frac{\partial y_k}{\partial n} = ku \quad \text{on } \partial\Omega = (0,1)^2, \\
&\qquad 1.8 \leqslant u \leqslant 2.5, \qquad (6.10b)
\end{aligned}$$

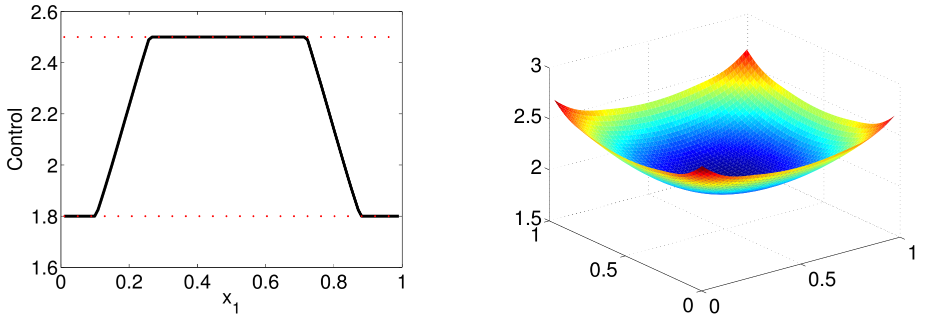
where

$$\hat{y}_k = k(2 - x_1(x_1 - 1) + x_2(x_2 - 1)), \qquad k = 1, \dots, N_E,$$

and $\alpha = 0.01$. Here (6.10a) describes the interaction of normalized quantum mechanical wave functions of electrons y_k in a superconductor on the basis of a simplified Ginzburg-Landau model [73]. For each $k = 1, \dots, N_E$ the corresponding PDE constraint (6.10a) is discretized with standard second-order finite differences on a regular $N_h \times N_h$ grid.

In Figure 6.7 the optimal control u^* and optimal state y^* are shown for a single PDE constraint, that is, $N_E = 1$ and $N_h = 100$. Both are initialized as $u \equiv 2.15$ and $y = \hat{y}_k$, respectively. Note that the bounds (6.10b) on u are partially active and the KKT system thus becomes increasingly ill-conditioned as $\mu \to 0$ because of the barrier term in (4.3). Next, in Figure 6.8 we show the number of RSP-GMRES iterations ($\times$) and the relative residual of each iterate accepted by the SMART test ($\diamond$) for varying N_E and N_h. Independently of N_E or the mesh size $N_h \times N_h$, the RSP-IIP method always converges within 20 optimization steps, while the RSP-GMRES iterates satisfy the SMART test within only five iterations.

In Table 6.9 we list for varying N_h and N_E the size of the resulting KKT system n, the total run-time, and the number of optimization steps for the RSP-IIP and the exact IP method. The number of optimization steps needed by the RSP-IIP method exceeds at most by four that of the exact IP method; it barely increases as the KKT system grows one thousandfold. By taking advantage of the KKT system's sparsity structure, the RSP-IIP method achieves a speedup of



(a) *Optimal Control u^* at $x_2 = 0$. The upper and lower bounds are indicated by the dotted lines.*

(b) *Optimal State y_1^**

Figure 6.7: *2D boundary control: optimal state and control for $N_E = 1$ PDE constraint discretized on a $N_h \times N_h$ mesh with $N_h = 100$.*

36, and even more so with growing problem size. As the run-time per IIP iteration scales linearly with respect to the number of PDE constraints N_E while the number of optimization steps barely increases, we obtain essentially linear run-time complexity. Moreover, if the different diagonal blocks in $\boldsymbol{J}_y$ and $\boldsymbol{J}_y^\top$ in (4.9), all independent of one another, were solved in parallel, the total execution time would essentially become constant and thus independent of N_E.

To compare the iterative linear solvers RSP-GMRES to the iterative linear solver from Section 6.2, we see in Table 6.10 the same quantities for smaller problems for the RSP-IIP and for the IIP method with an SQMR solver preconditioner an inverse based multilevel incomplete $\boldsymbol{LDL}^\top$ decomposition. The number of optimization steps for both IIP methods differs only slightly by at most two iterations. With growing problem size, we observe an increasing difference in run-time. While the RSP-GMRES solver takes advantage of the problem inherent block structure, the general purpose SQMR method needs increasingly more memory to compute an efficient preconditioner.

6.3.3 2D and 3D Parameter Estimation

We now consider two problems from groundwater modeling [38, 76], the first in two and the second in three space dimensions, where the log conductivity $u(x)$ needs to be estimated from noisy measurements of the fluid pressure, $y(x)$. Since the influence of noise can be further reduced by including measurements from multiple sources, practical applications often lead to multiple PDE constraints. Then, the (optimal) linear complexity achieved by the RSP-IIP algorithm with respect to the number of PDE constraints N_E becomes a key ingredient for its efficiency.

Thus, we let $\Omega = (0,1)^d, d = 2, 3$, and consider the spatially distributed

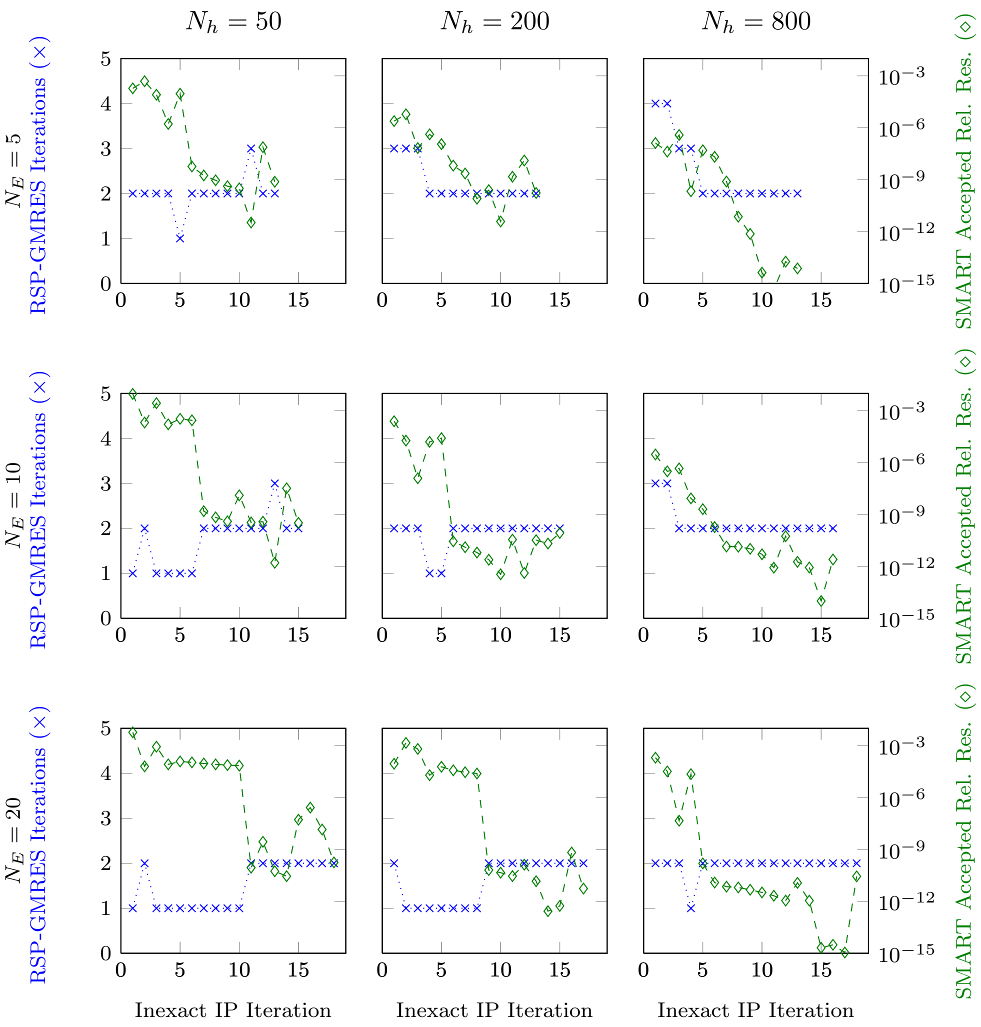


Figure 6.8: *2D boundary control: number of RSP-GMRES iterations (×) and relative residual (◇) at SMART acceptance for varying number of PDE constraints N_E and mesh sizes $N_h \times N_h$. Note the different scales on the left and right axes.*

parameter estimation problem

$$\begin{aligned} \min_{y,u} F(y,u) &= \frac{1}{2N_E 64} \sum_{k=1}^{N_E} ||v(y_k) - \hat{y}_k||^2_{\ell^2} + \frac{\alpha}{2}\left(V(u) + \beta ||u||^2_{L^2(\Omega)}\right) \\ \text{s.t. } -\nabla \cdot (\exp(u)\nabla y_k) &= q_k \quad \text{in } \Omega, k = 1, \dots, N_E, \\ y_k &= 0 \text{ on } \partial\Omega, \\ -2 \leqslant u \leqslant 2. \end{aligned} \tag{6.11}$$

Here, the true log conductivity is

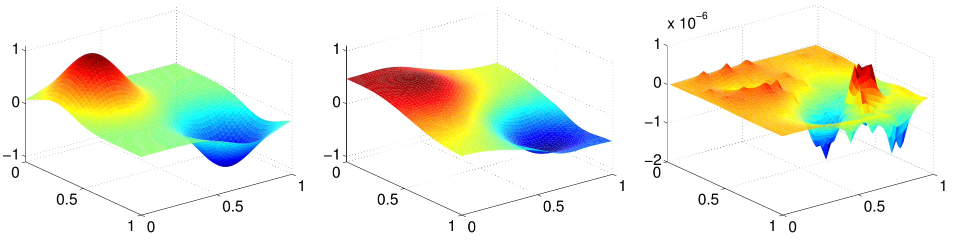
$$\hat{u}(x) = \exp\left(-\frac{||\tilde{x}_a - x||^2}{0.05}\right) - \exp\left(-\frac{||\tilde{x}_b - x||^2}{0.05}\right),$$

Table 6.9: *2D boundary control problem: size of the KKT system n, number of optimization steps and total run-time for the RSP-IIP method, and the exact IP algorithm for varying numbers of PDE constraints N_E and mesh size $N_h \times N_h$.*

Problem size			RSP-IIP		exact IP	
N_h	N_E	n	# IIP steps	run-time (s)	# IP steps	run-time (s)
50	5	26′000	13	3	12	3
50	10	51′000	15	7	13	10
50	20	101′000	18	17	14	10
200	5	404′000	13	78	12	125
200	10	804′000	15	173	13	697
200	20	1′604′000	17	382	14	3′829
800	5	6′416′000	13	1′661	13	10′479
800	10	12′816′000	16	4′106	14	68′750
800	20	25′616′000	18	9′132	14	335′470

Table 6.10: *2D boundary control problem: size of the KKT system n, number of optimization steps, and total run-time for the RSP-IIP method and the IIP algorithm with the preconditioned SQMR method from Section 6.2 for varying numbers of PDE constraints N_E and mesh size $N_h \times N_h$.*

Problem size			RSP-IIP		PSQMR-IIP	
N_h	N_E	n	# IIP steps	run-time (s)	# IIP steps	run-time (s)
25	3	4′250	11	0.4	11	0.4
25	5	6′750	12	0.6	13	1.3
50	3	16′000	11	1.7	13	4.7
50	5	26′000	13	3.2	13	8.6
100	3	62′000	12	9.5	13	52.0
100	5	102′000	13	16.8	14	113.0



(a) *True model $\hat{u}$* **(b)** *Optimal model u^** **(c)** *Optimal adjoint variable λ^**

Figure 6.9: *2D log conductivity estimation: true model and optimal solution with $N_E = 10$ PDE constraints on a $N_h \times N_h$ mesh with $N_h = 100$.*

Table 6.11: *2D log conductivity estimation: size of the KKT system n, number of optimization steps and total run-time for the RSP-IIP method for varying number of PDE constraints N_E, and mesh size $N_h \times N_h$.*

N_h	N_E	n	# IIP steps	run-time (s)
100	5	151'005	9	25
100	10	251'005	9	38
100	20	451'005	9	60
200	5	602'005	9	109
200	10	1'002'005	9	155
200	20	1'802'005	9	256
400	5	2'404'005	11	579
400	10	4'004'005	9	649
400	20	7'204'005	9	1'073

with $\tilde{x}_a = (0.25,\, 0.25)^\top, \tilde{x}_b = (0.75,\, 0.75)^\top$ in 2D or $\tilde{x}_a = (0.25,\, 0.25,\, 0.25)^\top$, $\tilde{x}_b = (0.75,\, 0.75,\, 0.75)^\top$ in 3D, respectively. To generate the synthetic measurements $\hat{y}_k, k = 1, \ldots, N_E$, we numerically solve the forward problem for a given source q_k, and add 1% white noise to its solution. In 2D, the source terms are given by

$$\begin{aligned} q_{4l+1}(x) &= \sin(\nu_1(l)\, 2\pi x_1) \sin(\nu_2(l)\, 2\pi x_2), \\ q_{4l+2}(x) &= \cos(\nu_1(l)\, 2\pi x_1) \sin(\nu_2(l)\, 2\pi x_2), \\ q_{4l+3}(x) &= \sin(\nu_1(l)\, 2\pi x_1) \cos(\nu_2(l)\, 2\pi x_2), \\ q_{4l+4}(x) &= \cos(\nu_1(l)\, 2\pi x_1) \cos(\nu_2(l)\, 2\pi x_2), \end{aligned}$$

with $(\nu_1(1), \nu_2(1)) = (1, 1)$, $(\nu_1(2), \nu_2(2)) = (1, 2)$, $(\nu_1(3), \nu_2(3)) = (2, 1)$, $(\nu_1(4), \nu_2(4)) = (1, 3)$, $(\nu_1(5), \nu_2(5)) = (2, 2), \ldots$ and similarly in 3D by triple products of trigonometric functions. For each k, the corresponding pressure field $y_k(x)$ is measured at 64 fixed locations irregularly distributed throughout Ω; those measurements are then collected in the vector $v(y_k)$.

In (6.11) each elliptic PDE is discretized on a regular grid using Q^0 finite elements for the model variable u and Q^1 finite elements for the state variable y_k. As a measure of variation in the piecewise constant log conductivity coefficient, we include the penalty term

$$V(u) = h^{2d} \int_{\mathcal{F}_h} [\![u]\!]^2 ds,$$

where $\mathcal{F}_h$ denotes all interelement boundaries and $[\![u]\!]$ denotes jumps across interfaces.

First, we consider the 2D case and apply our RSP-IIP method to (6.11) with $\alpha = 10^{-7}$ and $\beta = 0$. As an initial guess, we always set u identically to zero and y_k to the corresponding forward solutions. Figure 6.9 shows the optimal solution for $N_E = 10$ PDE constraints on an $N_h \times N_h$ mesh with $N_h = 100$. Indeed, the optimal model approximately follows the true model, while the measurements'

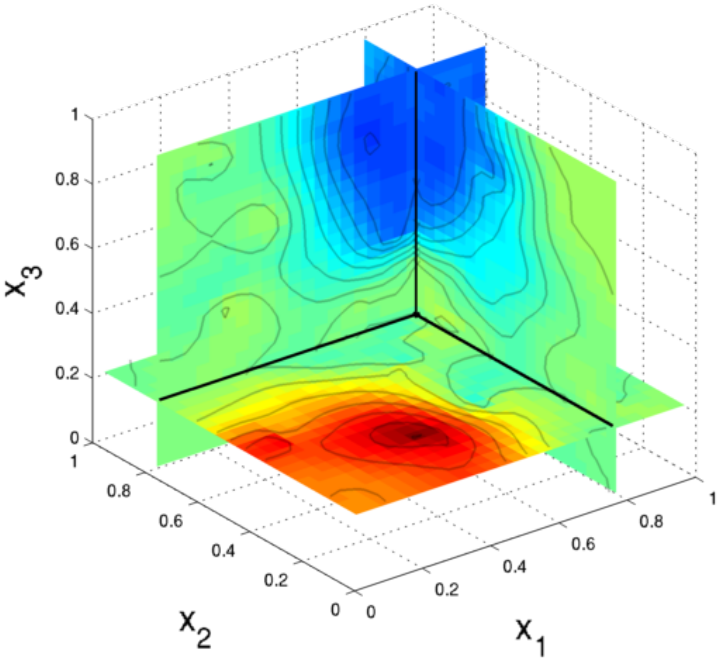


Figure 6.10: *3D log conductivity estimation: optimal model u^* computed by the RSP-IIP method applied to* (6.11) *with* $N_E = 6$ *PDE constraints on a* $25 \times 25 \times 25$ *grid.*

misfits act as point sources in the adjoint problem for the Lagrange multiplier λ. In Table 6.11 we list the number of optimization steps and total run-time for varying mesh size $N_h \times N_h$ and number of PDE constraints, N_E, thereby resulting in KKT systems up to $n = 7.2 \cdot 10^6$ in size. Remarkably, every problem was solved within merely 11 optimization steps, independently of N_h and N_E, while the total run-time increases only linearly with N_E.

Since the PDE in (6.11) is linear in the state variable y, as is often the case, the submatrices $\boldsymbol{J}_{y_k}$ in (4.9) are all identical; hence, the RSP-GMRES solver only needs to factorize a single PDE block matrix, such as $\boldsymbol{J}_{y_1}$.

Next, we consider the 3D case, and apply the RSP-IIP method to (6.11) with $\alpha = 10^{-7}$ and $\beta = 1$. Figure 6.10 shows slices of a typical optimal model u^* for $N_E = 6$ PDE constraints on an $N_h \times N_h \times N_h$ mesh with $N_h = 25$, which also exhibits a maximum about $(0.25, 0.25, 0.25)$ and a minimum about $(0.75, 0.75, 0.75)$. In Table 6.12 we list the number of optimization steps and total run-time for varying N_E and mesh sizes $N_h \times N_h \times N_h$. Again, all problems were solved within merely 15 optimization steps, while the number of optimization steps barely increases with growing mesh size. Moreover, the RSP-IIP method efficiently takes advantage of the added information from increasingly many PDE constraints, as the number of optimization steps actually decreases for increasing N_E. For comparison we also list number of optimization steps and run-time for the IIP method with the general purpose preconditioned SQMR from Sec-

Table 6.12: *3D log conductivity estimation: size of the KKT system n, number of optimization steps and total run-time for the RSP-IIP method and for the IIP method with preconditioned SQMR from Section 6.2 for a varying number of PDE constraints N_E, and mesh size $N_h \times N_h \times N_h$. A '-' denotes for early terminated runs due to long run-time.*

Problem size			RSP-IIP		PSQMR-IIP	
N_h	N_E	n	# IIP steps	run-time (s)	# IIP steps	run-time (s)
20	4	110′305	11	59	19	286
20	8	174′305	11	80	13	324
20	16	302′305	11	129	10	527
20	32	558′305	9	202	10	932
40	4	856′605	15	1′452	15	9′351
40	8	1′368′605	12	1′345	14	15′380
40	16	2′392′605	12	1′913	13	20′612
40	32	4′440′605	12	3′102	13	29′454
60	4	2′862′905	15	10′180	–	–
60	8	4′590′905	14	11′491	–	–
60	16	8′046′905	14	14′331	–	–
60	32	14′958′905	13	18′595	–	–

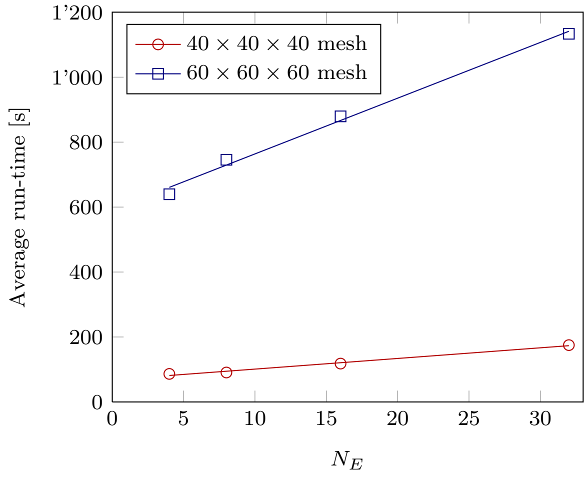


Figure 6.11: *3D log conductivity estimation: average run-time per optimization step of the RSP-IIP method as a function of the number of PDE constraints N_E for an $N_h \times N_h \times N_h$ mesh with $N_h = 40, 60$.*

tion 6.2. We see a significant speedup for the RSP-IIP method, even more so with increasing problem sizes.

In Figure 6.11 we again observe the linear average run-time complexity of the RSP-IIP method with respect to N_E for two different $N_h \times N_h \times N_h$ meshes. Remarkably, the total run-time is even sublinear due to the mild decrease in the number of optimization steps with increasing N_E. There we also follow the evolution of the number of RSP-GMRES iterations ('$\times$') and the relative residual of the accepted search direction ('$\diamond$') at each optimization step. The relative residual of the accepted iterate, never below 10^{-6}, typically lies above 10^{-2}. Hence, the SMART tests do not demand increasingly accurate solutions from the RSP-GMRES solver during the optimization process. Although the number of RSP-GMRES iterations typically increases in the course of any optimization run, all KKT systems were solved within 25 RSP-GMRES iterations, independently of N_h or N_E.

6.4 Conclusion

In Part I we presented descriptions of primal-dual IP methods for large-scale nonconvex optimization where the search directions are computed exactly by a sparse direct linear system solver or inexactly by means of an iterative linear system solver.

For the IIP method a general purpose algebraic multigrid preconditioned SQMR method was applied to the KKT systems. Numerical experiments on a large set of test problems and on two PDE-constrained optimization problems have also been presented. These results demonstrate the robustness of the approach and illustrate the significant speedup the IIP algorithm attains when compared to the exact IP method. As mentioned at the end of Section 6.2.2, we have observed a decrease in effectiveness of the preconditioner as the barrier parameter approaches zero.

Next we introduced a reduced-space preconditioned GMRES (RSP-GMRES) solver and tested several reduced-space preconditioners, including a block preconditioner, a sparse approximation of the inverse, a two-level preconditioner, and a preconditioner based on simultaneous source ideas for a challenging inverse medium problem for the Helmholtz equation. While the two-level preconditioner efficiently reduced the iteration numbers, its application becomes expensive, even more so for an increasing number of PDEs. The simultaneous source preconditioner circumvents this drawback but does not reduce the iteration numbers effectively. Thus, astonishingly, the simple block preconditioner, has proven most efficient.

We have evaluated the RSP-IIP method on the bases of three instances in 2D and 3D. These examples include distributed control and boundary control

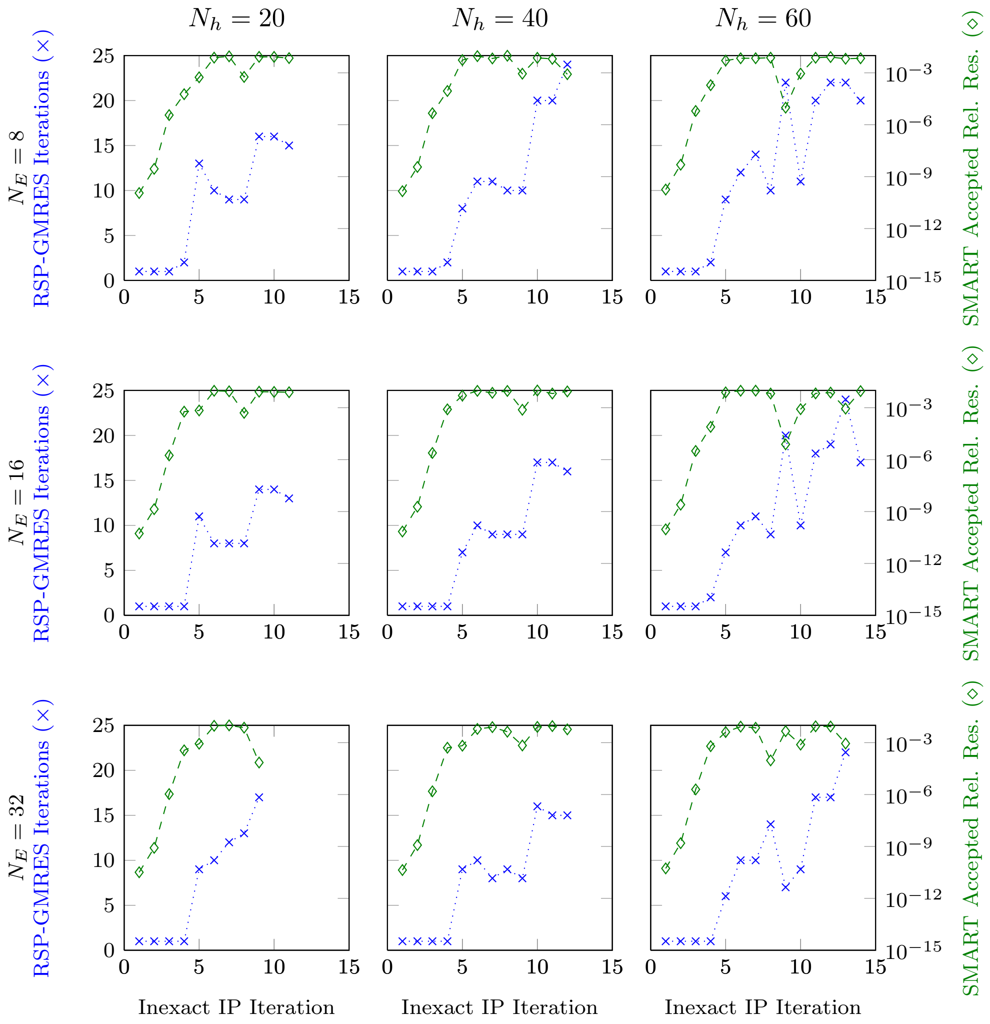


Figure 6.12: *3D log conductivity estimation: number of RSP-GMRES iterations (×) and relative residual (◇) at SMART acceptance for varying number of PDE constraints N_E, and mesh sizes $N_h \times N_h \times N_h$. Note the different scales on the left and right axes.*

with a nonlinear PDE constraint, and parameter estimation for the Laplace equation with inequality constraints on the controls or parameters. The RSP-GMRES solver takes advantage of the KKT matrix internal structure which yields a significant speedup for example 6.3.2. For the proposed RSP-GMRES solver we have shown h-independent preconditioning properties. In comparison with an optimal preconditioner 6.3.1 we observed similar iteration counts. For the boundary control problem considered in 6.3.2, the number of reduced-space preconditioned GMRES iterations per linear system did not increase with finer meshes despite the presence of active inequality constraints. Thus, the h-independent preconditioning properties have been observed repeatedly. A comparison revealed the advantages of the block structure exploitation approach in RSP-GMRES compared to the general purpose preconditioned SQMR method and even for small problems a speedup factor of 10 was observed, increasing for growing problem sizes.

In the context of parameter estimation, linear run-time complexity is of major importance for noise robust multiple experiment problems. This optimal run-time behavior has been observed for the parameter estimation problem for the Laplace equation in 6.3.3 in 2D and 3D. Since the PDEs are independent of one another, RSP-IIP can be parallelized easily. For linear PDEs, only a single PDE block needs to be factorized, which even reduces the constant in the run-time complexity.

Part III

Inverse Medium Problem for the Helmholtz Equation

Chapter 7

Formulation of an Inverse Medium Problem for the Helmholtz Equation

In the previous chapter we have seen how IP methods can be applied to large-scale nonconvex PDE-constrained optimization problems and evaluated an exact IP method and an inexact IP method with different inexact solver and preconditioners. In this part, we concentrate on a special PDE-constrained optimization problem, namely, an inverse medium problem for the Helmholtz equation.

In Chapter 7 we derive the block matrices and vectors of the optimality conditions (4.5) in detail and study regularization and nonconvexity issues in more detail. In Section 7.1 we first derive the Helmholtz equation from the wave equation and consider its finite element discretization in Section 7.2. This is than used to formulate the discretized optimization problem, where each block of the KKT system (4.5) is computed explicitly.

7.1 Derivation of the Helmholtz Equation

Waves are ubiquitous in medical and engineering applications because they can travel over long distances while retaining much of their shape; hence, they are natural carriers of information. When a wave encounters an inhomogeneity, it is partly scattered in the surrounding medium. From that scattered wave, usually recorded at remote sensors, the nature, location, and shape of the obstacle hidden inside the medium can be reconstructed, as in computer tomography, ultrasound and seismic imaging, nondestructive testing, and remote sensing. We formulate the inverse medium problem as a PDE-constrained optimization problem, where the underlying wave field solves the (time-harmonic) Helmholtz equation. In this chapter, we first derive the Helmholtz equation from the wave equation and consider its numerical discretization.

We consider a scalar valued wave $\tilde{y}(x,t)$ for which the wave equation

$$\frac{\partial^2 \tilde{y}(x,t)}{\partial t^2} - \nabla \cdot \big(u(x)^2 \nabla \tilde{y}(x,t)\big) = \tilde{f}(x,t) \tag{7.1}$$

holds. Here, $u(x)$ is called the wave speed. In the regime of parameter estimation, $u(x)$ is not particularly constant, and to emphasize this we call u the *wave speed profile*. If $\tilde{f}$ is of the form

$$\tilde{f}(x,t) = f_\omega(x) e^{-i\omega t} \tag{7.2}$$

with temporal frequency ω, the ansatz

$$\tilde{y}(x,t) = y(x) e^{-i\omega t} \tag{7.3}$$

yields the *Helmholtz equation*

$$-\omega^2 y(x) - \nabla \cdot (u(x)^2 \nabla y(x)) = f_\omega(x). \tag{7.4}$$

Every arbitrary right-hand side can be expressed as a superposition of right-hand sides of the form (7.2) by a Fourier transform. Due to linearity the solution of (7.1) can also be computed as a superposition of the corresponding solutions of (7.4). The obvious advantage is that (7.4) has one dimension less, since it is time independent.

Here, we state the Helmholtz equation in divergence form. In the literature we often find the Helmholtz equation in the form $\Delta y + k(x)^2 y(x) = f_\omega(x)$, where $k = \omega/u$ is the wave number. For a constant wave speed profile, they are equivalent. However, since we are confronted with nonconstant wave speed profiles, we stick to the formulation in (7.4).

Note that in (7.4) the Laplace operator $-\nabla \cdot (y^2 \nabla \odot)$ is coercive and we subtract the coercive operator $\omega^2 \odot$. Thus, the Helmholtz operator is not coercive any more. This fact makes it particularly difficult to solve the Helmholtz equation. Often in the literature we find the Helmholtz equation also with a different sign in front of the operator $\omega^2 \odot$, and it is then referred to as the “good” Helmholtz equation. In this thesis, we always refer to the noncoercive “bad” Helmholtz equation. Despite the real valued solution of (7.1) the solution of (7.4) is complex valued due to the ansatz function in (7.3).

7.2 Finite Element Discretization of the Helmholtz Equation

We consider a finite subregion Ω, embedded in a 2D or 3D unbounded medium. Inside Ω, the wave field then satisfies the Helmholtz equation (7.4). In this thesis $f(x)$ consists of a collection of point sources at locations $x_s \in \Omega$. Since all sources are assumed to be inside the computational domain Ω, the waves are purely outgoing as they reach the artificial boundary $\partial\Omega$. To prevent spurious reflections from it, we impose the Sommerfeld-type first-order absorbing boundary condition [8]

$$\frac{\partial y(x)}{\partial n} = i\,\frac{\omega}{u(x)},\, y(x) \qquad x \in \partial\Omega \tag{7.5}$$

for simplicity. We now separate the real and imaginary parts of $y = y_r + iy_i$ and formulate the complex-valued PDE (7.4) with (7.5) as a system of two real-valued PDEs

$$\begin{aligned} -\nabla\cdot(u(x)^2\nabla y_r(x)) - \omega^2 y_r &= f_{\omega r} \quad \text{in } \Omega \quad & \frac{\partial y_r}{\partial n} &= -\frac{\omega}{u(x)} y_i \quad \text{on } \partial\Omega, \\ -\nabla\cdot(u(x)^2\nabla y_i(x)) - \omega^2 y_i &= f_{\omega i} \quad \text{in } \Omega \quad & \frac{\partial y_i}{\partial n} &= \frac{\omega}{u(x)} y_r \quad \text{on } \partial\Omega. \end{aligned} \tag{7.6}$$

Following the standard finite element approach, the weak formulation of 7.6 is then to find $y_r, y_i \in H^1(\Omega)$ such that

$$\begin{aligned} \int_\Omega u^2\nabla y_r\cdot\nabla v_r - \omega^2\, y_r\, v_r\, dx + \omega\int_{\partial\Omega} u\, y_i\, v_r\, do &= \int_\Omega f_r\, v_r\, dx, \\ \int_\Omega u^2\nabla y_i\cdot\nabla v_i - \omega^2\, y_i\, v_i\, dx - \omega\int_{\partial\Omega} u\, y_r\, v_i\, do &= \int_\Omega f_i\, v_i\, dx \end{aligned}$$

for all $v_r, v_i \in H^1(\Omega)$. To discretize the wave speed profile model and the state variable, we subdivide Ω into triangles or quadrangles $T_i^u, i = 1,...N_{\mathcal{T}^u}$ and may subdivide these model elements further into triangles or quadrangles $T_j^y, j = 1,...N_{\mathcal{T}^y}$. Then a standard Galerkin finite element discretization of (7.6) with globally continuous, piecewise polynomial Lagrange basis functions $\varphi_y^{(j)}(x), j = 1,\dots,N_y$, $\varphi_u^{(j)}(x), j = 1,\dots,N_u$ leads to the conditions

$$\begin{aligned} \boldsymbol{c}(\boldsymbol{u},\boldsymbol{y}) &= \boldsymbol{A}(\boldsymbol{u})\,\boldsymbol{y} - \boldsymbol{f} \\ &= \begin{pmatrix} \boldsymbol{K}(\boldsymbol{u}) + \omega^2\boldsymbol{M} & \omega\boldsymbol{B}(\boldsymbol{u}) \\ \omega\boldsymbol{B}(\boldsymbol{u}) & -\boldsymbol{K}(\boldsymbol{u}) - \omega^2\boldsymbol{M} \end{pmatrix} \begin{pmatrix} \boldsymbol{y}_r \\ \boldsymbol{y}_i \end{pmatrix} - \begin{pmatrix} \boldsymbol{f}_r \\ -\boldsymbol{f}_i \end{pmatrix} = \mathbf{0}. \end{aligned} \tag{7.7}$$

Here, both the wave field and the wave speed profile are expanded in the finite element basis as $y_{rh} = \sum_j \boldsymbol{y}_r^{(j)}\varphi_y^{(j)}, y_{ih} = \sum_j \boldsymbol{y}_i^{(j)}\varphi_y^{(j)}, u_h = \sum_i \boldsymbol{u}^{(j)}\varphi_u^{(j)}$. The stiffness matrix $\boldsymbol{K}(\boldsymbol{u})$, the mass matrix $\boldsymbol{M}$, the boundary matrix $\boldsymbol{B}(\boldsymbol{u})$, and the load vector $\boldsymbol{f}_r$ are defined as

$$\begin{aligned} (\boldsymbol{K}(\boldsymbol{u}))^{(i,j)} &= \int_\Omega u_h^2\nabla\varphi_y^{(j)}\cdot\nabla\varphi_y^{(i)}\,dx, & (\boldsymbol{M})^{(i,j)} &= \int_\Omega \varphi_y^{(j)}\varphi_y^{(i)}\,dx, \\ (\boldsymbol{B}(\boldsymbol{u}))^{(i,j)} &= \int_{\partial\Omega} u_h\varphi_y^{(j)}\varphi_y^{(i)}\,do, & (\boldsymbol{f}_r)^{(t)} &= \varphi_y^{(t)}(x_s), \end{aligned}$$

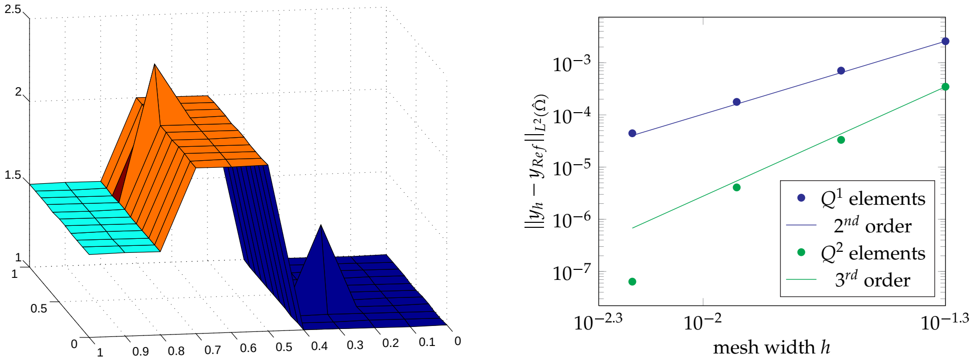


Figure 7.1: *Convergence of implemented parallel simulation: left: fixed nonconstant model parameter; right: numerical and theoretical convergence of bilinear Q^1 and biquadratic Q^2 elements.*

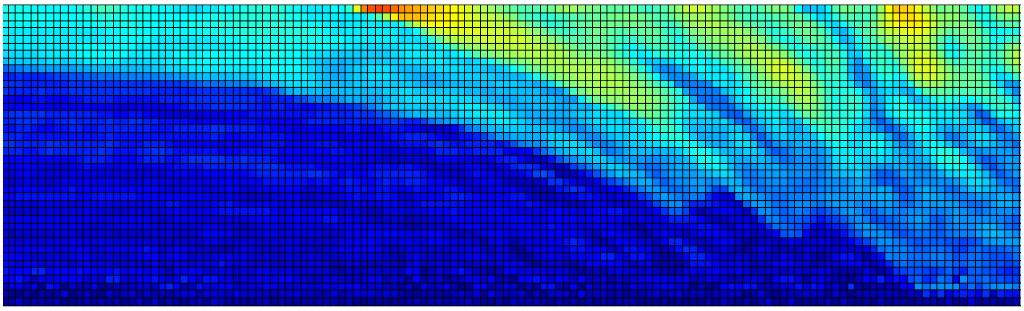
where x_s denotes the location of the point source. We set $\boldsymbol{f}_i = \mathbf{0}$. By splitting the integrals into elementwise contributions, the matrix entries can be efficiently computed in parallel.

This has been implemented in C++, using the finite element library `libMesh` together with the distributed memory parallel library `PETSc`. As a sanity check, we see in figure 7.1 a convergence graph for bilinear Q^1 and biquadratic Q^2 elements for the state variable y. The model u was discretized with bilinear elements on a fixed mesh with mesh width $h = 0.1$. The model is depicted in Figure 7.1 on the left. Only the right-hand side, we set a point source at $(0.1, 0.1)$. To check convergence, the L^2 difference on a subdomain $\hat{\Omega} = (0.2, 1) \times (0, 1)$ between the simulation result and a reference solution, computed with biquadratic elements, $h = 0.05$ was computed using 4 MPI processes. The point for $h = 6.25 \cdot 10^{-3}$ with Q^2 elements does not lie on the line for 3rd order convergence, since the solutions are compared to a reference solution as described above instead of an analytic solution.

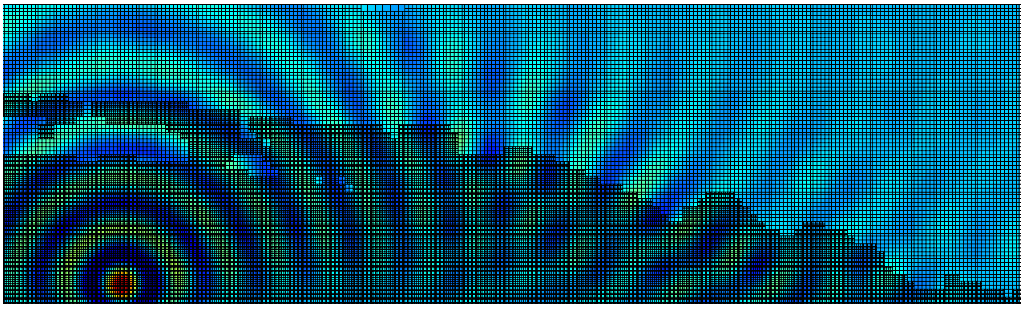
We also applied adaptive mesh refinement techniques to the state variable discretization. Here, the mesh was refined such that a certain number of elements per wavelength was achieved. In Figure 7.2 we see the wave speed model and the imaginary part of the wave field generated by a point source at $(1.4, 0.1)$ with $\omega = 20$. The state variable mesh was refined such that 10 points per wave length are achieved, that is

$$10 \cdot h \leqslant \lambda(x) = 2\pi \frac{c(x)}{\omega}. \tag{7.8}$$

If an element did not fulfill this rule, it is refined by halving its size. Global continuity is enforced by extra constraints on the hanging nodes. In Figure 7.2 we see a mesh with 55′362 degrees of freedom. A regular mesh with fixed $h = h_{min}$ needs 96′761 degrees of freedom to fulfill rule (7.8). Note that an optimization



(a) *wave speed profile* u_h



(b) *imaginary part of PDE solution at* $\omega = 20$ *and adapted discretization mesh*

Figure 7.2: *Adaptive mesh refinement for the state variable. The mesh was refined to follow the 10-points-per-wavelength rule, where the local wavelength was estimated by* (7.8) *with the wave speed profile depicted in 7.2(a). The mesh is finer in the lower right region with smaller wave speed and shorter wavelength.*

variable and the KKT system contains several real and imaginary parts each of which is of this size.

7.3 Formulation as PDE-Constrained Problem

In the previous section we have derived the Helmholtz equation and its standard finite element discretization. In this section, we formulate the inverse medium problem for the Helmholtz equation and derive the terms in (4.5) in detail.

The setup for the inverse medium problem consists of multiple point sources at positions $x_{s_k}, k = 1, \dots, N_E$ each of which generates a wave field y_k according to the Helmholtz equation (7.4). Each wave is recorded at positions $x_{m_i}, i = 1, \dots N_m$ yielding, after Fourier transformation, a vector $\hat{y}_k \in \mathbb{R}^{2N_m}$ of real and imaginary parts. To reduce the danger of false solutions, we follow two strategies. The first one is a continuation approach along the temporal frequency ω. We start from low frequencies, where more global information is contained, to higher frequencies with increased local information but also more false solutions. The second strategy restricts the search space by inequalities, as they are often available for applications like seismic imaging. To tackle ill-posedness, a regularization term $R(u)$ is added to the objective function. We will discuss regularization in more detail in Section 8.1. The optimization problem reads

$$\begin{aligned}
\min_{y,u} F(y,u) &= \frac{1}{2}\sum_{k=1}^{N_E} ||v(y_k) - \hat{y}_k||_{\ell^2}^2 + \frac{\alpha}{2}R(u) \\
\text{s.t.} -\nabla\cdot(u^2\nabla y_k) - \omega^2 y_k &= \delta_k \qquad \text{in } \Omega, \\
\frac{\partial y_k}{\partial n} &= i\frac{\omega}{u} y_k \quad \text{on } \partial\Omega, \\
u_- \leqslant u \leqslant u_+ .
\end{aligned}$$

In the full-space approach, the optimization variable $\boldsymbol{x}$ consists of all wave fields together with the model parameters, $\boldsymbol{x} = (\boldsymbol{y}_1, \dots, \boldsymbol{y}_{N_E}, \boldsymbol{u})$. The goal is to determine the wave speed profile $\boldsymbol{u}$ that minimizes the measurements' mismatch. This leads to the nonconvex programming problem

$$\begin{aligned}
\min_{\boldsymbol{y}\in\mathbb{R}^{N_y}, \boldsymbol{u}\in\mathbb{R}^{N_u}} \boldsymbol{F}(\boldsymbol{y},\boldsymbol{u}) &= \frac{1}{2}\sum_{k=1}^{N_E} \|\boldsymbol{V}\boldsymbol{y}_k - \hat{\boldsymbol{y}}_k\|_{\ell^2}^2 + \frac{\alpha}{2}\boldsymbol{R}(\boldsymbol{u}) \\
\text{s.t. } \boldsymbol{A}(\boldsymbol{u})\,\boldsymbol{y}_k &= \boldsymbol{f}_k \qquad \text{for } k = 1, \dots N_E, \\
\boldsymbol{u}_- \leqslant \boldsymbol{u} \leqslant \boldsymbol{u}_+ .
\end{aligned} \tag{7.9}$$

Here, $\boldsymbol{V}$ denotes the evaluation matrix

$$\boldsymbol{V}\boldsymbol{y}_k = \begin{pmatrix} \tilde{\boldsymbol{V}} & \boldsymbol{0} \\ \boldsymbol{0} & \tilde{\boldsymbol{V}} \end{pmatrix} \begin{pmatrix} \boldsymbol{y}_{rk} \\ \boldsymbol{y}_{ik} \end{pmatrix} \text{ with } \tilde{\boldsymbol{V}}^{(ij)} = \varphi_y^{(j)}(x_{m_i})$$

at receiver locations x_m. To setup the Newton system (4.5), the constraint Jacobians need to be assembled, that is,

$$\boldsymbol{J}_{\boldsymbol{y}_k} = \boldsymbol{A}(\boldsymbol{u}) \qquad \text{for } k = 1, \dots N_E,$$

with $\boldsymbol{A}(\boldsymbol{u})$ from (7.7) and

$$\boldsymbol{J}_{\boldsymbol{u}k} = \begin{pmatrix} \boldsymbol{K}_{r\boldsymbol{u}} \\ \boldsymbol{K}_{i\boldsymbol{u}} \end{pmatrix} \text{ with } (\boldsymbol{K}_{r\boldsymbol{u}})_{st} = 2\int_\Omega u_h \varphi_u^{(t)} \nabla y_{krh} \nabla \varphi_y^{(s)}\, dx + \omega \int_{\partial\Omega} y_{kih} \varphi_u^{(t)} \varphi_y^{(s)} do,$$

and $\boldsymbol{K}_{i\boldsymbol{u}}$ correspondingly. Here, y_{krh} and y_{kih} denotes the discretization of the real and imaginary part of field y_k. Since the PDE is linear, $\nabla^2_{\boldsymbol{y}_k\boldsymbol{y}_k}\boldsymbol{\mathcal{L}}^f = \boldsymbol{V}^\top\boldsymbol{V}$ in (4.5) consists only of the objective function's Hessian. To denote the Hessian blocks $\nabla^2_{\boldsymbol{u}\boldsymbol{y}_{kr}}\boldsymbol{\mathcal{L}}^f$ and $\nabla^2_{\boldsymbol{u}\boldsymbol{y}_{kr}}\boldsymbol{\mathcal{L}}^f$ we introduce the discretized Lagrangian multipliers

$$\lambda_{krh}(x) := \sum_{s=1}^{n_y} (\boldsymbol{\lambda}_{kr})^{(s)}\, \varphi_y^{(s)}(x) \quad \text{and } \lambda_{kih}(x) := \sum_{s=1}^{n_y} (\boldsymbol{\lambda}_{ki})^{(s)}\, \varphi_y^{(s)}(x),$$

where $\boldsymbol{\lambda}_{kr}, \boldsymbol{\lambda}_{ki}$ denote the Lagrange multiplier vectors for the real and imaginary part of the kth PDE constraint. With this, the Hessian blocks can be written explicitly as

$$\left(\nabla^2_{\boldsymbol{uu}}\boldsymbol{\mathcal{L}}^f\right)_{st} = \frac{\alpha}{2}\nabla^2_{\boldsymbol{uu}}\boldsymbol{R} + 2\int_\Omega \varphi_u^{(s)}\varphi_u^{(t)}\left(\nabla y_{krh}\cdot\nabla\lambda_{krh} + \nabla y_{kih}\cdot\nabla\lambda_{kih}\right)dx$$

and $\nabla^2_{\boldsymbol{uy}_k}\boldsymbol{\mathcal{L}}^f = \left(\nabla^2_{\boldsymbol{uy}_{kr}}\boldsymbol{\mathcal{L}}^f \;\; \nabla^2_{\boldsymbol{uy}_{ki}}\boldsymbol{\mathcal{L}}^f\right)$ with

$$\left(\nabla^2_{\boldsymbol{uy}_{kr}}\boldsymbol{\mathcal{L}}^f\right)_{st} = 2\int_\Omega u_h\varphi_u^{(s)}\nabla\varphi_y^{(t)}\cdot\nabla\lambda_{krh}\,dx - \omega\int_{\partial\Omega}\varphi_u^{(s)}\varphi_y^{(t)}\lambda_{kih}\,do, \text{ and}$$

$$\left(\nabla^2_{\boldsymbol{uy}_{ki}}\boldsymbol{\mathcal{L}}^f\right)_{st} = 2\int_\Omega u_h\varphi_u^{(s)}\nabla\varphi_y^{(t)}\cdot\nabla\lambda_{kih}\,dx + \omega\int_{\partial\Omega}\varphi_u^{(s)}\varphi_y^{(t)}\lambda_{krh}\,do.$$

Chapter 8

Numerical Studies

In this chapter we will discuss several issues of the inverse medium problem for the Helmholtz equation on the basis of an easy 2D example. The true wave speed profile (model) $\hat{u}$ and the real and imaginary part of y_2 for $\omega = 60$ is depicted in Figure 8.1. The wave speed profile mimics a layered material with two regions of different wave speed inscribed. The locations of point sources x_s and receivers x_m are marked in Figure 8.1 as stars and circles respectively.

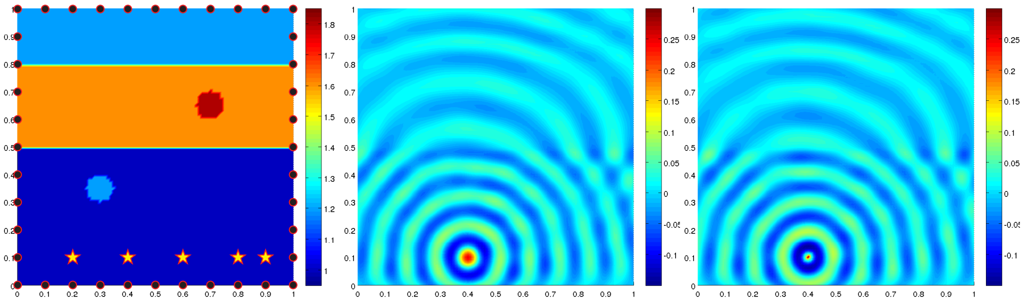


Figure 8.1: *True model $\hat{u}$ and real and imaginary part of y_2 for a model problem at $\omega = 60$.*

8.1 Ill-Posedness and Regularization

In problem (9.1) the objective functional is extended by a regularization term $\frac{\alpha}{2}\boldsymbol{R}(\boldsymbol{u})$. In this section, we explain, the idea of regularization and explore some regularization terms numerically.

According to Hadamars definition the problem to find z such that $K(z) = d$ is well-posed, if the following three assertions hold:

1. For each right-hand side d, there exists a solution z.

2. For each right hand-side d, the solution z is unique.
3. The solution z depends continuously on d.

For most instances of PDE-constrained optimization problems, assertion 3 does not hold in the continuous function space setting. In the context of our inverse medium problem, we seek a function u to minimize $\|v(A(u)^{-1}f) - \hat{y}\|^2$, with the Helmholtz operator A. That is, we need to solve the nonlinear normal equation

$$K(u, \hat{y}) = 0 \tag{8.1}$$

which is a continuous version of (4.10). Even if we had data $\hat{y}$ on the whole boundary for infinitely many illuminating waves this equation would still be ill-posed. In the continuous setting, the linearization of K is a compact operator, and thus it has singular values converging to 0. Thus, its inverse is unbounded and solution components in subspaces with small singular values can be amplified arbitrarily. This principal problem cannot be circumvented by adding more measurements.

After discretization, we are confronted with two issues. The first is inherent to the approximated unbounded inverse operator yielding unstable methods which become worse for increasingly finer meshes. The second issue is simply based on the fact that there may be not enough measurements to determine all model parameters uniquely. This nontrivial null space then strongly depends on the geometry of source and receiver locations. The latter problem could be resolved by increasing the number of measurements, but this is not always possible in practice. Either way, additional information needs to be used to yield good reconstruction results.

Next, we will explore numerically the nature of the inherent ill-posedness. High spatial frequency components in u do not influence the wave y_k severely. Thus, measurements $\hat{y}_k$ are not sensitive to such perturbations. We evaluated for $\omega = 20$ and $N_m = 40$ receivers the reduced-space objective in perturbation directions with increasingly higher spatial frequency. More specifically, we consider an interpolation of

$$u(\beta; \nu) = \hat{u} + \beta/2 \sin(2\pi\nu x_1)\sin(2\pi\nu x_2)$$

and evaluate the reduced-space objective

$$\tilde{\boldsymbol{F}}_{N_m}(\beta; \nu) = \boldsymbol{F}(\boldsymbol{y}(\boldsymbol{u}(\beta, \nu)), \boldsymbol{u}(\beta, \nu)).$$

In Figure 8.2 we plot $\tilde{\boldsymbol{F}}_{N_m}(\beta; \nu)$ for $\omega = 20, N_m = 40, \beta \in [-1, 1]$, and $\nu = 1, 2, \ldots 64$. We observe a decreasing sensitivity for high spatial frequency perturbations ν. To quantify this, the second derivative of $\tilde{\boldsymbol{F}}$ is listed in Table 8.1. The sensitivity reduces by a factor of $6 \cdot 10^{-5}$ for a spatial frequency increase from

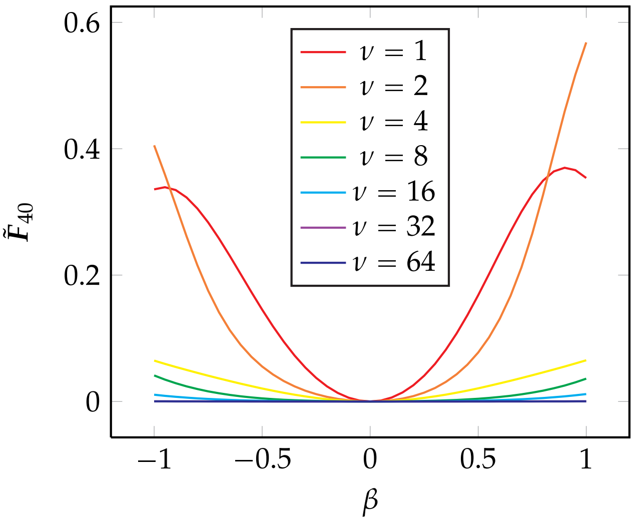


Figure 8.2: *Objective function for model perturbations with frequency ν.*

$\nu = 1$ to $\nu = 64$. The same experiments have been repeated with $N_m = 1'280$ receivers. Now the number of model parameters $n_{\boldsymbol{u}} = 10'201$ is exceeded by the number of measurement points from real and imaginary parts of five waves and we still see a sensitivity reduction of $1.3 \cdot 10^{-4}$.

ν	1	4	16	64
$\tilde{F}''_{40}(0;\nu)$	$1.2 \cdot 10^{0}$	$1.8 \cdot 10^{-1}$	$5.4 \cdot 10^{-3}$	$2.9 \cdot 10^{-4}$
$\tilde{F}''_{1'280}(0;\nu)$	$3.8 \cdot 10^{1}$	$6.0 \cdot 10^{0}$	$1.3 \cdot 10^{-1}$	$5.0 \cdot 10^{-3}$

Table 8.1: *Sensitivities for different number of measurement points for model perturbations of varying spatial frequencies ν.*

To tackle ill-posedness, several strategies are common [49]. One strategy is the reduction of the model space dimension. The most efficient approach of these strategies consists in a subspace generated by eigenvectors of the reduced Hessian (4.13) with eigenvalues larger than a certain threshold. However, the computation of eigenvalues and eigenvectors of the reduced Hessian is very expensive. Furthermore, since the problem is nonlinear, the reduced Hessian at the optimal point is not available at the beginning of the computation. Another approach to reduce the model space dimension is based in a coarse discretization of u. In this general purpose approach the model space is not tailored to the problem and fine model details are not contained in the low-dimensional model space. If additional information about the model is available, e.g., position and shape of regions with similar wave speed, then a low-dimensional model space can be generated which also contains all detailed features. This will be used later to generate very low-dimensional model spaces.

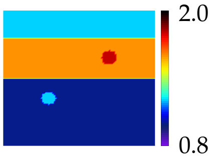
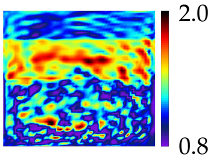
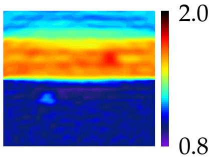
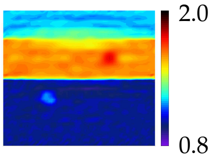
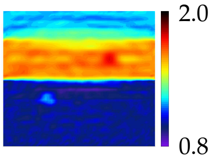
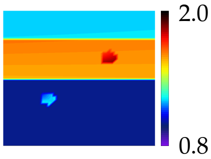
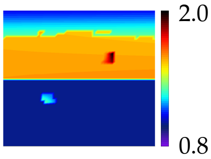
True wave speed profile		$\|\hat{u}\| = 1.25$ $\bar{\boldsymbol{m}} = 2.7 \cdot 10^{-3}$
$\alpha = 0$		$\|u_h^* - \hat{u}_h\| = 2.2 \cdot 10^{-1}$ $\boldsymbol{m} = 5.2 \cdot 10^{-3}$ $\nabla \boldsymbol{R} = 0$ $\nabla \boldsymbol{m} = 5.2 \cdot 10^{-2}$
R_{∇} $\alpha^* = 10^{-2}$		$\|u_h^* - \hat{u}_h\| = 5.3 \cdot 10^{-2}$ $\boldsymbol{m} = 3.1 \cdot 10^{-2}$ $\nabla \boldsymbol{R} = 3.1 \ \cdot 10^{-3}$ $\nabla \boldsymbol{m} = 3.1 \cdot 10^{-3}$
R_{TV} $\alpha^* = 10^{-3}, \varepsilon_{TV} = 27$		$\|u_h^* - \hat{u}_h\| = 3.8 \cdot 10^{-2}$ $\boldsymbol{m}(u) = 2.5 \cdot 10^{-3}$ $\nabla \boldsymbol{R} = 4.0 \cdot 10^{-4}$ $\nabla \boldsymbol{m} = 4.0 \cdot 10^{-4}$
R_{Gauss} $\alpha^* = 10^{-1}, \sigma = 27$		$\|u_h^* - \hat{u}_h\| = 4.8 \cdot 10^{-2}$ $\boldsymbol{m}(u) = 2.6 \cdot 10^{-3}$ $\nabla \boldsymbol{R} = 8.4 \cdot 10^{-4}$ $\nabla \boldsymbol{m} = 9.5 \cdot 10^{-4}$
Cluster $n_u = 20$		$\|u_h^* - \hat{u}_h\| = 1.2 \cdot 10^{-2}$ $\boldsymbol{m}(u) = 1.4 \cdot 10^{-2}$ $\nabla \boldsymbol{R} = 0$ $\nabla \boldsymbol{m} = 8.6 \cdot 10^{-2}$
Cluster $n_u = 20$		$\|u_h^* - \hat{u}_h\| = 6.5 \cdot 10^{-2}$ $\boldsymbol{m}(u) = 2.8 \cdot 10^{-2}$ $\nabla \boldsymbol{R} = 0$ $\nabla \boldsymbol{m} = 1.1 \cdot 10^{-2}$

Table 8.2: *Reconstruction results of the inverse medium problem for the Helmholtz equation with different regularization strategies.*

A different strategy is the modification of the normal equation operator (8.1) by adding a regularization term, which yields a well-posed problems. Those strategies are usually summarized as Tikhonov regularization. In the optimization context, this is realized by adding a regularization term to the objective function. The strategy can also be considered as incorporating an a priori model probability distribution [45] or penalizing certain characteristics of models. In many applications of the inverse medium problem we can assume that the true model consists of different regions where u does not vary much and can be well approximated by a smooth function. Furthermore, since the objective function is not sensitive to high frequency perturbations they are contained mainly in subspaces with small eigenvalues. To penalize these high oscillatory models the gradient is included in the regularization term, e.g.,

$$R_{\nabla}(u) = \|G\nabla u\|^2_{L^2(\Omega)}.$$

Here $G \in \mathbb{R}^d, d = 2,3$, is a weighting factor for gradient directions. For example in seismic imaging one can expect a layered material, where the layers are horizontal, and thus gradients in the vertical direction should not be penalized as much as gradients in horizontal directions. Models, e.g., in seismic imaging often are not even differentiable or they exhibit locally large gradients, which are over-penalized by R_{∇}. To circumvent this with a differentiable regularization term, approaches like total variation (TV)

$$R_{TV}(u) = \int_{\Omega} \sqrt{|G\nabla u|^2 + \varepsilon}\, dx$$

or the Huber norm [5] can be applied. These regularization functions are convex. In this thesis, we also apply a regularization term which is nonconvex,

$$R_{\text{Gauss}}(u) = \int_{\Omega} 1 - \exp(-|G\nabla u|^2/\sigma^2)\, dx.$$

The idea of R_{Gauss} is to increase penalization only for small gradients. Once the algorithm has detected a large gradient or an edge in the model, the penalization of this edge only weakly depends on the actual value of the gradient or jump.

There are several techniques for the choice of the regularization parameter α, e.g., approaches based on filter factors or L-curve analysis. However, in practice the regularization parameter is often determined by numerical experimentation.

For a fair comparison of the different regularization terms in the sequel of this section, we apply the following heuristics for the choice of α. To avoid overfitting, we first introduce an additional termination criterion for the optimization method based on noise estimation. At the solution, we then impose a criterion to balance the regularization term to the misfit term. The optimization may then be restarted with an adapted regularization parameter.

The additional optimization termination criterion is based on the knowledge of measurement noise ξ. In the absence of noise, we had ideally $\boldsymbol{V}\boldsymbol{y}_k = \hat{\hat{y}}_k$. With noise $\hat{y}_k = \hat{\hat{y}}_k + \xi$ the expectation of the misfit $\boldsymbol{m} := \sum_{k=1}^{N_E} \|\boldsymbol{V}\boldsymbol{y}_k - \hat{\hat{y}}_k + \xi\|^2$ is

$$\mathbb{E}(\boldsymbol{m}) = \mathbb{E}(\|\xi\|^2).$$

For real world measurements, the noise level can be estimated from sensor specification or from data themselves. For our synthetic data, noise was generated by uniform distribution relative to the simulated value, that is $\xi_t \sim (y_k)_t \, \eta \, U(-1,1), \eta \geqslant 0$ with expectation value $\mathbb{E}(\xi_t^2) = 1/3\,\eta^2 (y_k)_t^2$. Thus, we estimate the expected misfit as

$$\bar{\boldsymbol{m}} = \frac{1}{3}\eta^2 \sum_{k=1}^{N_E} \|\hat{y}_k\|^2. \tag{8.2}$$

After each optimization step, the current misfit is checked, to see if

$$\sum_{k=1}^{N_E} \|\boldsymbol{V}\boldsymbol{y}_k - \hat{y}_k\|^2 < \zeta_1 \bar{\boldsymbol{m}} \tag{8.3}$$

holds, for $\zeta_1 > 1$. In this case, the reduced-space misfit with $\boldsymbol{y} = \boldsymbol{y}(\boldsymbol{u})$ is computed. If (8.3) still holds, the optimization problem with current regularization parameter α is considered as solved since a smaller misfit would probably generate worse reconstruction results due to overfitting.

At a solution of (7.9), the condition (4.10) must hold. A dominant misfit term in this sum then indicates a strong dependence on the error-prone measurements. On the other hand, a dominant regularization gradient implies only weak dependence on the measured data. Therefore, to estimate the regularization parameter α, we demand that both terms are of the same order of magnitude. We check if

$$\zeta_2 \leqslant \frac{\|\nabla_{\boldsymbol{u}} \boldsymbol{m}(y(\boldsymbol{u}))\|}{{}^{\alpha}\!/\!{}_{2}\|\nabla_{\boldsymbol{u}} \boldsymbol{R}\|} \leqslant \zeta_3, \tag{8.4}$$

with $0 < \zeta_2 < 1 < \zeta_3$, and if (8.4) holds we terminate the algorithm successfully. Otherwise, α is increased or reduced towards the satisfaction of (8.4) and the optimization algorithm is restarted to solve the optimization problem with the new regularization parameter.

With this heuristic if the noise reduces to 0, $\eta \to 0$, it follows that $\mathbb{E}(\|\boldsymbol{V}\boldsymbol{y} - \hat{y}_k\|^2) \to 0$, $\mathbb{E}(\|\nabla \boldsymbol{m}\|^2) = \mathbb{E}(\|\boldsymbol{V}^\top(\boldsymbol{V}\boldsymbol{y} - \hat{y}_k)\|^2) \to 0$ and thus, if $\|\nabla_{\boldsymbol{u}} \boldsymbol{R}\|$ is bounded away from 0, we have that $\alpha \to 0$, which is desirable if enough data are given.

To test different regularization strategies we applied this heuristic with $\zeta_1 = 2, \zeta_2 = 1/10, \zeta_3 = 10$ to an inverse medium problem for the Helmholtz equation at $\omega = 60$ and 5% noise. Lower and upper bounds were fixed to $u_- = 0.8, u_+ =$

2.0. The barrier parameter was set to $\mu = 10^{-7}$ and the algorithm was initialized with a three layered model with constant wave speed in each layer, $1, 1.5, 1$ (true model: $1, 1.6, 1.2$) and without the two hidden objects. The 1280 receivers were placed equidistantly on the border. The results are summarized in Table 8.2. There, we depict the reconstructed model u_h^* and list the final model error $\|\hat{u}_h - u_h^*\|^2_{L^2(\Omega)}$, the final misfit $\boldsymbol{m}(\boldsymbol{y}(\boldsymbol{u}))$, and the final gradient norms of the misfit $\nabla \boldsymbol{m}$ and regularization term $\alpha/2\nabla \boldsymbol{R}$. We seek the true profile in the upper line. There we see the estimated final misfit $\bar{\boldsymbol{m}}$ according to (8.2). If we do not regularize ($\alpha = 0$), the noise creates spurious highly oscillatory perturbations and the reconstruction is not good. Gradient regularization terms damp these oscillatory components and the reconstructions are much better. While edges are smoothed too much by the R_∇ term, they appear sharper for TV regularization R_{TV}. The regularization term R_{Gauss} yields a reconstruction with approximately the same quality as TV regularization.

Visually, the best result is generated if additional knowledge about regions with similar wave speed can be incorporated. In the penultimate row in Table 8.2, we model the PDE parameter as a piecewise bilinear function in the five regions including the three layers without the objects and the two circular objects themselves. Since high oscillatory perturbations are contained in the model space and the number of measurement data points exceeds the number of model parameters, there is no need for the Tikhonov regularization term and also the heuristic described above was not applied. The wave speed in the regions are reconstructed successfully. Since in practice the data to generate such reduced model spaces may be error-prone, we also generated the five regions from the solution with TV regularization and built the model space based on these regions. The result is depicted in the lower row. To make results comparable, the reduced gradient for the later two approaches is not computed with respect to the model parameters but with respect to a discretized wave speed model as was also reported for the other examples.

We have seen for several Tikhonov regularization terms how they perform in the reconstruction of discontinuous models. The contribution of additional knowledge in the form of regions with smooth model behavior is remarkable. However, this information may not be available. TV regularization has shown the best reconstruction result. Thus, for the latter test, we stick to TV regularization if not otherwise stated.

8.2 Nonconvexity, Hessian Modification, and Frequency Continuation

Due to the mixed product of the model u and the state variable y, the inverse medium problem for the Helmholtz equation (9.1) is nonconvex. Thus a unique

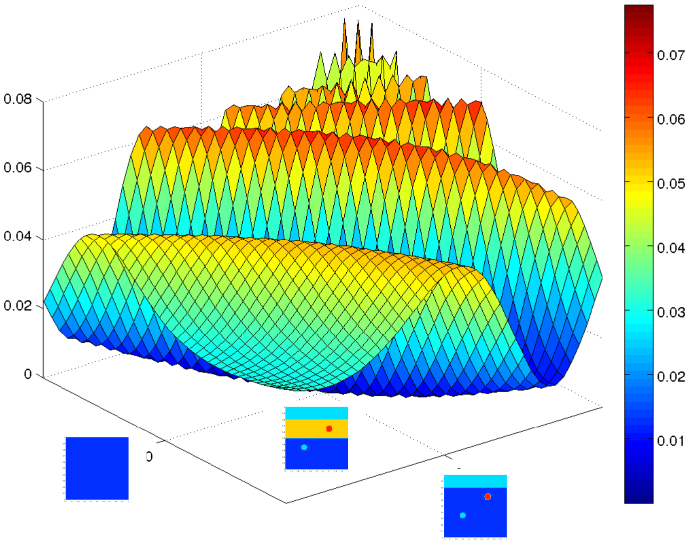


Figure 8.3: *Reduced-space objective function in a two dimensional subspace with the true solution in the origin*

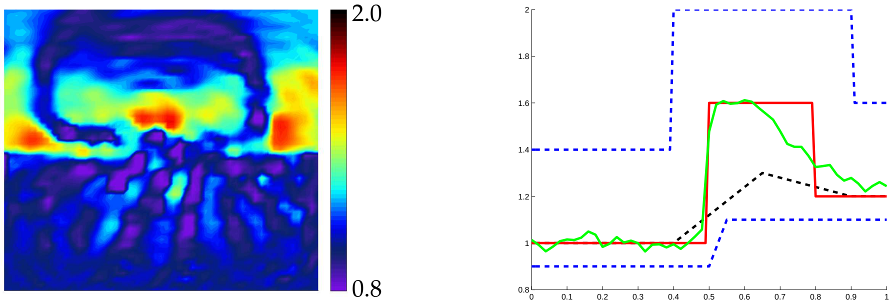
local solution cannot be guaranteed. Furthermore, the objective or merit function may increase in the Newton direction. These potential pitfalls can appear in nonconvex optimization. Here we show that they really occur for the problem considered in this chapter. We first show that the inverse medium problem for the Helmholtz equation indeed is highly nonconvex. We discuss the issues that arise from this fact, including convergence to false local solutions and Newton directions with increasing merit function. We also show, how to overcome these problems.

To have an insight into the nonconvexity of the problem, we first visualize the reduced-space objective in a 2D subspace. We set

$$u_h(s,t) = \hat{u}_h + s(u_{h,1} - \hat{u}_h) + t(u_{h,2} - \hat{u}_h),$$

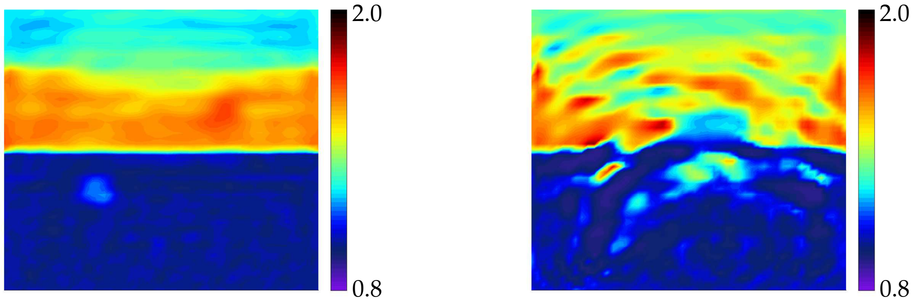
where $\hat{u}_h$ is the true control which was used to generate the measurements $\hat{y}$ without noise, and $u_{h,1}, u_{h,2}$ are models as depicted on the axis. In Figure 8.3 we show the reduced-space objective function $g(s,t) = F(\boldsymbol{y}(\boldsymbol{u}(s,t)), \boldsymbol{u}(s,t))$ with $\omega = 20, N_m = 40$, and $\alpha = 0$ on a 2D subspace and we clearly see that the reduced-space objective function indeed is highly nonconvex. The local minima in the presented 2D subspace are not necessarily local minima in the whole model space, but still the graphic shows that nonconvexity cannot be neglected for this problem.

To show nonconvexity issues for the problem at hand, we initialize the exact IP algorithm with a model which is constant in the x_1 direction and its x_2 de-



(a) *optimal model u^*, no box constraints*

(b) *box constraints u_+, u_-, true model $\hat{u}_h$, and optimal model u_h^* at $x_1 = 0.5$*



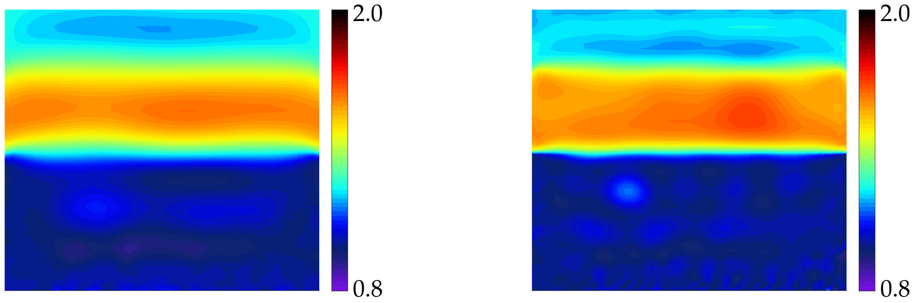
(c) *optimal model u^*, reconstructed with box constraints*

(d) *u after 300 iterations, stronger box constraints, no Hessian modification*

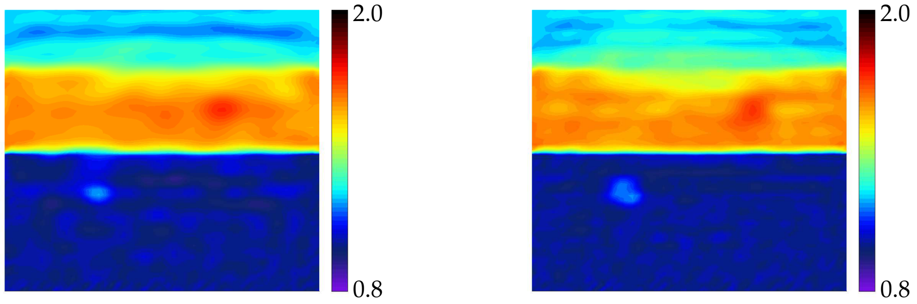
Figure 8.4: *Nonconvexity issues for the inverse medium problem for the Helmholtz equation: (a) local false solution without box constraints, (b) initial model, true model, reconstructed model, and model box constraints at $x_1 = 0.5$, (c) reconstruction with box constraints, (d) model after 300 iterations without Hessian modification.*

pendence is depicted in Figure 8.4(b). We apply the exact IP method described in Chapter 2 to the problem at $\omega = 60$. The method converges in 20 iterations to the false local minimum depicted in Figure 8.4(a). This shows the existence of false (inexact) solutions. If we add knowledge in terms of inequality constraints, also plotted in Figure 8.4(b), the method converges in 25 optimization steps to the good solution in Figure 8.4(c), where the layers are reconstructed and the two hidden objects can be observed.

As mentioned in Section 2.2, the exact algorithm computes a factorization to solve the KKT system (2.13). This factorization is also used to check if the inertia indicates a semidefinite of indefinite Hessian on the null space of the constraints Jacobian. If necessary the Hessian block is then modified. In the optimization run with optimal model, shown in Figure 8.4(c), this occurs in the optimization



(a) *Optimal model u^* at $\omega = 10$* **(b)** *Optimal model u^* at $\omega = 20$*



(c) *Optimal model u^* at $\omega = 40$* **(d)** *Optimal model u^* at $\omega = 60$*

Figure 8.5: *Reconstruction results in the frequency stepping strategy with $\omega = 10, 20, 40$ and $\omega = 60$.*

steps 2 and 4–18. If the Hessian modification is not performed, the computed Newton directions may or may not be accent directions for a merit function. Without Hessian modification, in optimization steps 1–16 the line search algorithm still finds a step length accepted by the filter. However, in iteration 17 this is no longer possible. While the inexact algorithm terminates in the line search phase with an error, the exact algorithm falls into a restoration phase for two optimization steps and follows the main IP algorithm afterwards. This then happens repeatedly. We terminated the optimization after 300 optimization steps. The model in iteration 300 is shown in Figure 8.4(d). Neither the existence of the two upper layers nor their wave speed can be estimated correctly, not to speak about the two hidden objects.

Clearly, the above inclusion of a priori information through box constraints can be combined with a standard frequency stepping strategy for improved robustness. In this parameter continuation method along the temporal angular frequency ω we assume that for each ω the optimal model $\boldsymbol{u}^*$ is a strict local minimum. It is implicitly defined by (4.4),

$$\nabla \mathcal{L}^f(\boldsymbol{y}, \boldsymbol{u}, \boldsymbol{s}, \boldsymbol{\lambda}; \omega) = \mathbf{0}$$

with Jacobian matrix $\left(\nabla^2 \mathcal{L}^f \mid \frac{\partial \nabla \mathcal{L}^f}{\partial \omega}\right)$, where $\nabla^2 \mathcal{L}^f$ is the regular KKT matrix from (4.5). Then the implicit function theorem states the existence of a continuous path $\boldsymbol{u}^* = \boldsymbol{u}^*(\omega)$.

For illustration of the frequency stepping method, we consider the same problem at $\omega = 10$. Starting from an initial uniform background $u = 1.25$, the IP method rapidly identified in 13 optimization steps the overall layered structure of the medium, see Figure 8.5(a). To detect the two smaller buried objects, we now optimize at the higher frequencies $\omega = 20, 40$ and 60, initializing each optimization run with the solution obtained from the lower frequency. For $\omega = 20, 40$, and 60, the method converges in only $8, 11$, and 6 optimization steps to the reconstruction in Figures 8.5(b), 8.5(c), and 8.5(d). In the final reconstruction, we can see the wave speed and location of the three layers as well as the two hidden objects.

8.3 Model Parametrization

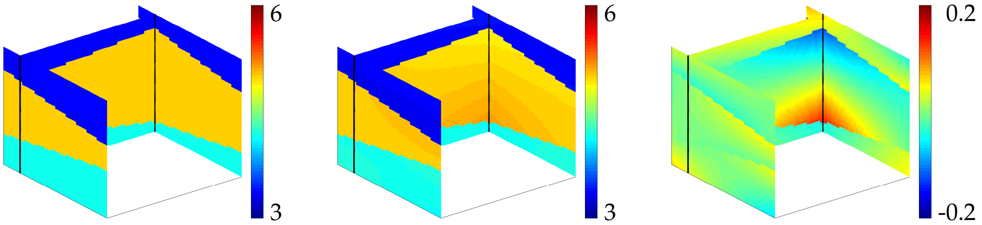
As mentioned in Section 8.1 one possibility to overcome the inherent ill-posedness as well as the problem of underdetermination due to insufficient measurements is the reduction of the model parameter space dimension. In the context of the RSP-IIP algorithm introduced in Chapter 3 and Section 5.2 this has also the advantage that the Schur complement can become very small, e.g., several hundreds instead of tens or hundreds of thousands.

Here, we assume, that subregions $\Omega_i \subset \Omega \subset \mathbb{R}^d, i = 1, \ldots, N_p$ are given where the model u is smooth. Then, on each subregion, we model the wave speed profile by only a few parameters. More specifically, we apply bilinear or trilinear basis functions with coefficient vector $\boldsymbol{p} \in \mathbb{R}^{N_p 2^d}$. In the implementation, we still have the model u discretized on a mesh and thus the corresponding coefficients $\boldsymbol{u} = \boldsymbol{B}\boldsymbol{p}$ can be computed easily. The derivatives with respect to the new model parameters are computed as

$$\nabla_p \mathcal{L}^f = \boldsymbol{B}^\top \nabla_u \mathcal{L}^f, \ \nabla^2_{pp} \mathcal{L}^f = \boldsymbol{B}^\top \nabla^2_{uu} \mathcal{L}^f \boldsymbol{B}, \text{ and } \nabla^2_{yp} \mathcal{L}^f = \nabla^2_{yu} \mathcal{L}^f \boldsymbol{B}.$$

We would like to emphasize, that a region can be very general and it even does not need to be connected. For an example with such a region, we remind to the lower row in Table 8.2.

Here, we also apply the parametrization strategy to a 3D problem originated from [14]. The true model is shown in Figure 8.6(a). It consists of the layers in



(a) *True model $\hat{u}$* **(b)** *Optimal model u^** **(c)** *Reconstruction Error $u^* - \hat{u}$*

Figure 8.6: *Reconstruction for a 3D inverse medium problem for the Helmholtz equation at $\omega = 90$. Prior knowledge was included into the model space consisting of trilinear functions in each region.*

$\Omega = (0,1)^3$ with constant wave speed,

$$\hat{u}(x_1, x_2, x_3) = \begin{cases} 4.16667 & \text{if } 1/2x_1 + 5/2x_2 + 3/8x_3 - 1 < 0, \\ 3.33333 & \text{if } -1/6x_1 + 5/3x_2 + 1/3x_3 - 1 > 0, \\ 5 & \text{otherwise.} \end{cases}$$

Three slices of $\hat{u}$ at $x_1 = 0, x_1 = 1$, and $x_2 = 1$ are shown in Figure 8.6.

This example is especially appropriate for the reduced model space approach, since a model space with trilinear functions needs only 12 parameters. The underlying full discretization with standard trilinear finite elements on a regular grid involves 132′651 degrees of freedom instead. In the RSP-IIP method, the reduced-space KKT system is only of size 60, including the blocks for upper and lower box constraints and their Lagrange multipliers.

Five point sources with $\omega = 90$ at locations $x_{s_1} = (0.2, 0.2, 0.1), x_{s_2} = (0.2, 0.8, 0.1), x_{s_3} = (0.8, 0.2, 0.1), x_{s_4} = (0.8, 0.8, 0.1)$, and $x_{s_5} = (0.5, 0.5, 0.7)$ are simulated and the solution was evaluated at 113 measurement points on a grid close to the upper boundary and also in four wells at the corners of Ω along the z-axis. To simulate measurements, 1% percent noise has been added.

The RSP-IIP method was used to take advantage of the very small model space dimension. The method converged after only 13 iterations to a model which is shown in Figure 8.6(b). By the choice of the model space the three layers are predefined. However, the algorithm correctly identifies the wave speed in each layer as almost constant. Also the values are reconstructed nicely. The overall error of the model is $\|u_h^* - \hat{u}_h\| = 2.5 \cdot 10^{-2}$. It is depicted in Figure 8.6(c) and also reflects the predefined regions, as expected.

Chapter 9

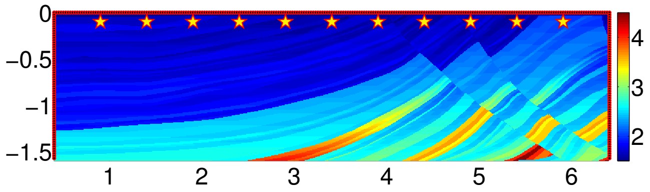
Real World Problem in Seismic Imaging

Full waveform seismic inversion leads to some of the most challenging nonlinear PDE-constrained optimization problems. Here we consider the Marmousi model [67, 81, 16], a standard benchmark in seismic imaging. It corresponds to a vertical slice through the Cuanza basin in Angola [75, 44] delimited by the domain $\Omega = (0.4\,\text{km}, 6.4\,\text{km}) \times (0\,\text{km}, 1.6\,\text{km})$.

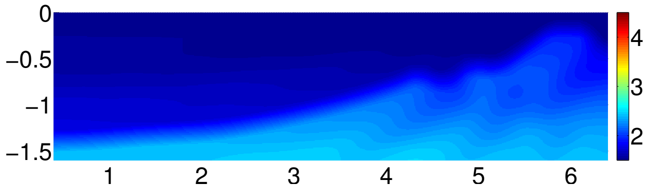
Given measurements, $\hat{y}_k, k = 1, \ldots, N_E$ from $N_E = 11$ seismic events ("shots"), we shall attempt to reconstruct the true velocity profile $\hat{u}(x)$ in Ω—see Figure 9.1(a). The measurements are obtained for each point source δ_k by recording the amplitude of the corresponding (complex-valued) wave field y_k at 367 observation points located at the surface and the two lateral boundaries (vertical wells); those values are then collected in the vector $v(y_k)$. Hence, we consider the PDE-constrained optimization problem:

$$
\begin{aligned}
\min_{y,u} F(y,u) &= \frac{1}{2}\sum_{k=1}^{N_E} ||v(y_k) - \hat{y}_k||_{\ell^2}^2 + \frac{\alpha}{2} R_{TV}(u) \\
\text{s.t.} -\nabla \cdot (u^2 \nabla y_k) - \omega^2 y_k &= \delta_k \qquad \text{in } \Omega, \\
\frac{\partial y_k}{\partial n} &= i\frac{\omega}{u} y_k \quad \text{on } \partial\Omega, \\
u_- \leqslant u \leqslant u_+.&
\end{aligned}
\tag{9.1}
$$

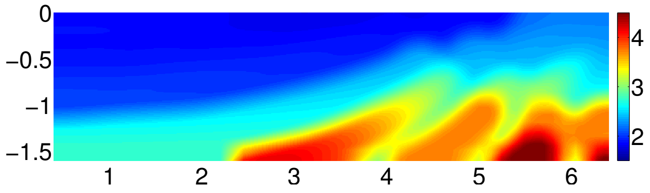
Here again $R_{TV}(u) = \int_\Omega \sqrt{|\nabla u|^2 + \varepsilon}\,dx$ denotes TV regularization with $\varepsilon = 18.75$ and $\alpha = 10^{-4}$. Each wave field y_k satisfies the Helmholtz equation (forward problem) for a given velocity profile $u(x)$ and frequency $\omega > 0$. At the boundary of the computational domain $\partial\Omega$ we impose a first-order Sommerfeld-like absorbing boundary condition. To generate the synthetic data $\hat{y}_k$, we first solve



(a) True model $\hat{u}$)



(b) Lower bound u_-



(c) Upper bound u_+

Figure 9.1: *Geophysical imaging: true Marmousi model $\hat{u}$ (in km/s) with upper and lower bounds u_+, u_-.The stars indicate the location of the different point sources δ_k.*

the forward problem on a fine mesh with mesh size $h = 8\,\mathrm{m}$ using Q^1 elements for u and Q^2 elements for y_k; then we add 1% white noise to the recorded values. The finite element discretization was implemented using `libMesh` [48] and `PETSc` [7].

Since the problem is nonconvex, any local optimization method may converge to a false solution. Prior knowledge, in practice often available as inequality constraints, reduces the search space and thereby prevents the algorithm from becoming trapped too easily in a (false) local minimum. Here, the upper and lower bounds u_+, u_- are determined from $\hat{u}$ by local mean and maximum, or minimum, filtering [66], respectively, followed by 10% further expansion of the feasible interval—see Figures 9.1(b) and (c).

As the number of extrema increases with frequency, we further reduced the risk of ending up in a (false) local minimum through frequency stepping: starting at a lower frequency ω, we progressively increase ω while initializing each optimization run from the previous optimal model at lower frequency. In all cases we set $\varepsilon_{tol} = 10^{-4}$ in (2.12), as smaller values did not yield any further improvement in the reconstruction. Moreover, we discretize both u and y_k with Q^1 finite elements on a coarser mesh, thus avoiding any potential "inverse crime".

Starting at the lowest frequency $\omega = 20$, we now apply the RSP-IIP method to (9.1) with mesh size $h = 40\,\mathrm{m}$ for u and $h = 20\,\mathrm{m}$ for y_k. We initialize the algorithm with $\mu = 10^{-3}$ and $u = (u_+ + u_-)/2$ – see Figure 9.3. After only

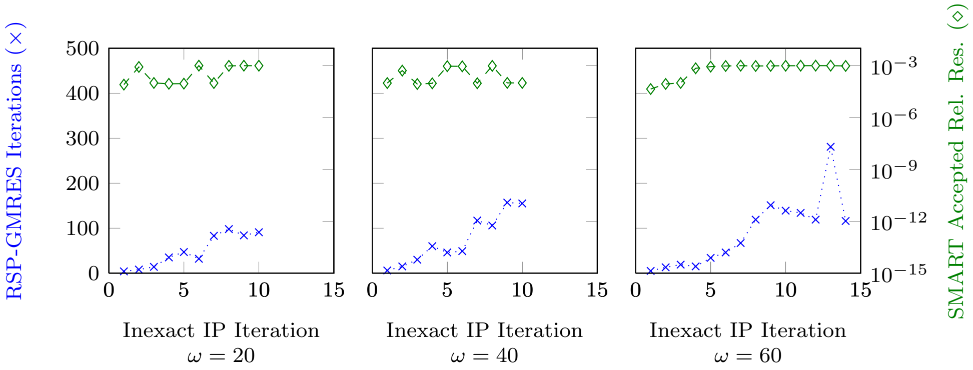


Figure 9.2: *Geophysical imaging: number of RSP-GMRES iterations (×) and relative residual (◇) at SMART acceptance for $\omega = 20, 40$, and 60. Note the different scales on the left and right axes.*

Table 9.1: *Geophysical imaging: time frequency ω, problem size, size of the KKT system n, number of IIP iterations, and run-times.*

ω	# States	# Model Param.	n	# IIP steps	run-time (s)
20	$5.36 \cdot 10^5$	$6.19 \cdot 10^3$	$1.10 \cdot 10^6$	10	825
40	$5.36 \cdot 10^5$	$6.19 \cdot 10^3$	$1.10 \cdot 10^6$	10	1′001
60	$5.36 \cdot 10^5$	$2.44 \cdot 10^4$	$1.19 \cdot 10^6$	14	1′734

10 optimization steps, the RSP-IIP method converges to the optimal model u^*, displayed in Figure 9.3; the imaginary part of one typical wave field y_2 is also shown there. At such low frequency, the wave length is still large and hence unable to detect smaller features in the medium—see also Figure 9.4. The total run-time and problem size are summarized in Table 9.1. In Figure 9.2, we follow the number of RSP-GMRES iterations required to solve the $n \times n$ system with $n = 1.1 \cdot 10^6$.

Next, we let $\omega = 40$ and initialize the RSP-IIP algorithm with the optimal model from the previous run at $\omega = 20$. Again, we observe in Figure 9.2 an increase in the number of RSP-GMRES iterations during the optimization process. Although the problem size and the number of optimization steps are identical, the overall larger number of RSP-GMRES iterations results in a slight increase in total run-time—see Table 9.1.

Finally, we let $\omega = 60$, $\mu = 10^{-5}$, and choose a finer mesh width $h = 20\,\text{m}$ for the model parameter, too. As a consequence, the KKT system barely increases, whereas the KKT Schur complement now increases by a factor four. Nontheless, the RSP-IIP method converges in only 14 steps, while the number of RSP-GMRES iterations never exceeds 300 iterations. The reconstructed model, shown in Figure 9.3, now reproduces the sharp transitions in the velocity field from the true model, with even smallest details revealed in Figure 9.4.

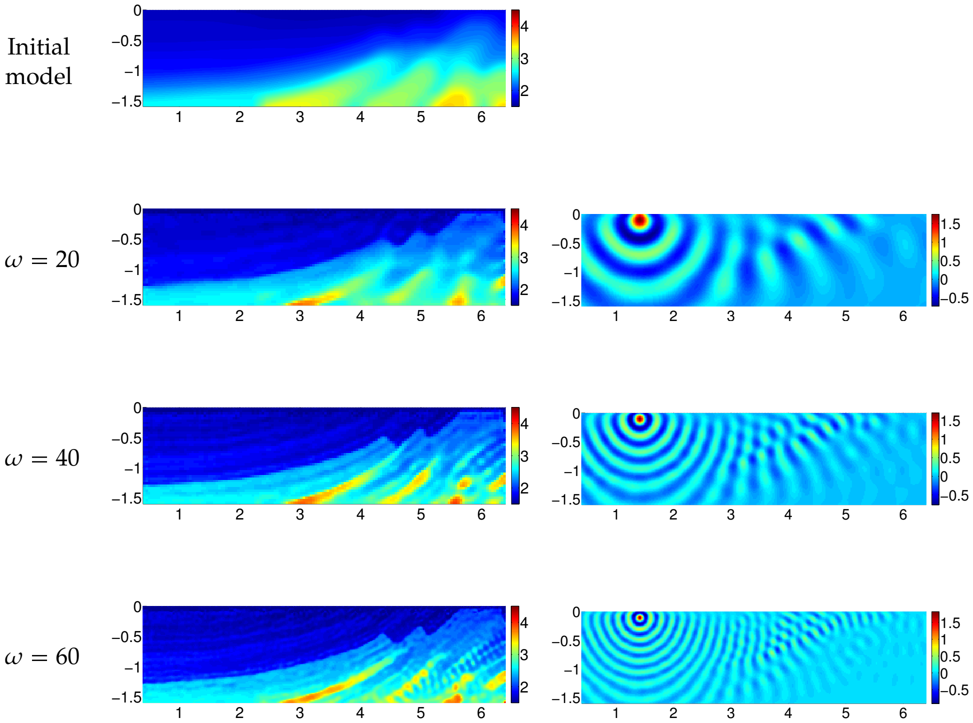


Figure 9.3: *Geophysical imaging: initial model for $\omega = 20$, and reconstructed optimal model for $\omega = 20, 40,$ and 60 (left); Imaginary part of the y_2 for $\omega = 20, 40,$ and 60 (right). Compare with true model in Figure 9.1.*

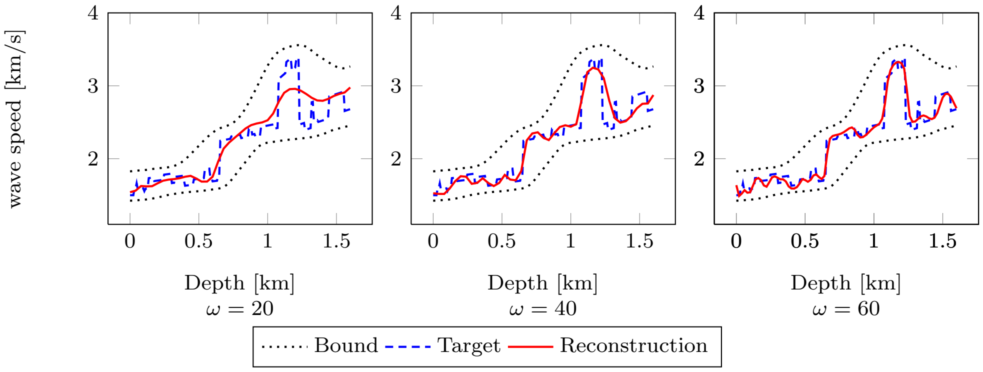


Figure 9.4: *Geophysical imaging: vertical cross section of the true wave speed $\hat{u}$, its reconstruction u^*, and lower and upper bounds u_+ and u_- at $x_1 = 4.0$ at increasingly higher frequencies $\omega = 20, 40,$ and 60.*

For comparison we now also apply the exact IP method described in 2.2 to (9.1) with $\omega = 40$. The linear systems are solved using the direct solver `Pardiso` with 4 OpenMP threads. The exact IP method converges in 9 optimization steps, instead of 10 for the RSP-IIP method, yet its total run-time of $14'993$ s now results in a fifteenthfold increase, despite the multithreaded hardware! This illustrates that the SMART tests efficiently control the inexactness, thus leading to very few additional optimization steps, while the RSP-GMRES solver takes advantage of the sparsity structure of the KKT systems.

Part IV

Conclusions & Future Work

Chapter 10

Conclusion and Future Work

10.1 Conclusion

In this thesis the superiority of an IIP method over an exact IP method for large-scale nonconvex PDE-constrained optimization problems with inequality constraints has been demonstrated. A parallelizable RSP-IIP method is introduced which takes advantage of the inherent sparsity pattern of the linear systems arising in PDE-constrained optimization problems.

In many applied sciences, PDEs model systems, which we aim to optimize in a certain sense. These pose PDE-constrained optimization problems which are in general large-scale and nonconvex. Inequality constraints are useful for guiding the algorithm towards a good local solution or they are necessary to account for application specific constraints. In this thesis, IIP methods have been proven to be an efficient approach to solve these class of problems.

A detailed description of IP methods with both exact and inexact step computations has been given. To handle nonconvexity, exact IP methods take advantage of the computed inertia, but suffer from fill-in. IIP methods circumvent this by use of iterative solvers. To control model convexification and inexactness, SMART tests are applied. They rate a linear solver iterate on the basis of a merit function approximation and accept the search direction, request an iterative solver for improvement, or indicate Hessian modification. With SMART tests, global convergence guarantees are achieved. While the IIP method in [24] needs to solve two sparse large-scale linear systems, the IIP method proposed in Chapter 3 comes along with only a single linear system solution in most optimization steps.

Efficiency and robustness of the proposed IIP method has been shown on the basis of a large test set of general optimization problems and for various PDE-constrained optimization problems. The IIP method with a general purpose preconditioned SQMR solver was assessed by solvable problems of the CUTEr test set and a success rate of 89% has been achieved. An 3D optimal boundary control problem with a nonlinear PDE and with box constraints on the state variable has been solved for up to $7 \cdot 10^5$ optimization variables and a speedup of about 8 with respect to the exact IP method was observed. For another 3D example of placing exhausts and air conditioners optimally in a server room, a speedup of 17 has been achieved. This shows that the permission of controlled errors in the KKT system solution allows for fast iterative solvers. The mild increase of optimization steps demonstrates that the fast local convergence of the exact IP method is also observed for the IIP method.

We have introduced an RSP-GMRES linear solver into the IIP method, which is tailored to the KKT sparsity pattern. Several preconditioners have been compared based on various ideas from the negligence of the dense term in the Schur complement, a sparse approximation of this dense term, a multilevel approximation, as well as a simultaneous source approximation. Despite its simplicity, the negligence of the dense term turned out to be the most efficient preconditioner. With this preconditioner the RSP-IIP method has confirmed the expected linear run-time complexity with respect to the number of PDEs for 2D and 3D parameter estimation problems. As the proposed RSP-IIP method is parallelizable, this could even be improved to a method with a run-time complexity almost independent of the number of PDEs. For a convex distributed control problem we compared RSP-IIP with a preconditioned projected CG method, which is tailored to such optimal control problems. Despite inexactness, RSP-IIP converged in a single optimization step. For the solution of the corresponding KKT system we observed only a mild increase in the iteration counts and an h-independent RSP-GMRES iteration number. This demonstrates h-independent preconditioning properties of the proposed preconditioner.

The speedup factors for RSP-IIP methods with respect to the exact IP method confirms its efficiency, which is superior to the IIP method with a general purpose preconditioner. We assessed this method by a boundary control problem with nonlinear constraining PDEs, yielding a speedup factor of 36 for a KKT systems of size $25.6 \cdot 10^6$. For an inverse medium problem for the Helmholtz equation we observed a speedup factor of 15, despite the parallel execution of the direct linear solver on 4 OpenMP threads.

For this problem the matrix and vector entries have been derived in detail. Here, a parallel code was developed and numerical examples confirmed scaling of the MPI parallelism. Mesh adaption for the state variable has been presented yielding a reduction of the number of optimization variables by a factor of 2 for a real world problem. The reason for ill-posedness of the inverse medium problem has been demonstrated numerically and different regularization strategies including three Tikhonov regularization terms and a regularization strategy by a priori knowledge of smooth profile regions were compared. For this comparison a heuristic for the regularization parameter adaption based on the estimated misfit has been introduced. For this problem we evaluated the impact of nonconvexity and presented convergence to a local false solution in the absence of inequality constraints. Also, the importance of Hessian regularization has been demonstrated for a problem where the regular algorithm converges in 25 optimization steps. Without Hessian modification, the algorithm has not succeeded after 300 iterations and is still far away from a good reconstruction. The combination of inequality constraints and a frequency continuation approach yield good reconstruction results. For a 3D parameter estimation problem for the Helmholtz equation, the RSP-IIP method was used taking further advantage of a priori knowledge to regularize and also to reduce the size of the Schur complement.

The combination of frequency stepping with inequality constraints to guide the algorithm leads to good reconstruction results for a real world seismic imaging problem. The problems have been solved efficiently by the RSP-IIP method and the number of optimization steps for each run of three temporal frequencies remained bounded below 15. However, the number of RSP-GMRES iterations per KKT system increased within an optimization run.

10.2 Future Work and Research Directions

As summarized above, the IIP method was assessed by several PDE-constrained optimization problems and especially the RSP-IIP method has proven to be a highly efficient approach for nonconvex, large-scale PDE-constrained optimization problems with inequalities. However, there are still open questions that remain to be answered by future developments and some of the ideas in this thesis might give a starting point for further enhancements.

This thesis concentrates on computational results and various PDE-constrained optimization problems have been considered. Given this generality and the complexity of the algorithm, it is hard to actually prove theorems. For instance, fast local convergence has been observed by comparison of iteration numbers of the exact and inexact IP methods. While, for the exact IP method superlinear local convergence is proven, for the inexact method this proof is missing. As a basic idea of this local convergence proof, the IIP method may be considered as an inexact Newton method similar to the approach followed in [18]. Another subject of future research might be the behavior for increasingly finer meshes. The results in this thesis has shown robustness of iteration numbers for h refinement; however, theoretical results are lacking. The implemented optimization method uses different discrete norms and stability might become an issue for other instances of PDE-constrained optimization problems. Since we have followed the discretize-then-optimize approach, all norms for a single discretization problem are equivalent. With finer meshes, the equivalence constants converge to 0 and ∞, respectively. This poses the question, of whether the number of optimization steps remains bounded for $h \rightarrow 0$.

Regularization in general is an active area of research. Regularization can be considered as an a priori distribution in the PDE parameter space and techniques from artificial intelligence, like a support vector machine, may be used to model this a priori knowledge. For Tikhonov regularization, an automatic regularization parameter adaption may be interesting to investigate. In Section 8.1 a heuristic based on the estimated final misfit has been introduced. This approach seems to be very interesting and can be extended to a reformulation of the optimization problem. For instance one may maximize the regularization parameter subject to an inner optimization problem. This inner optimization problem may then minimize the difference of the achieved and estimated misfit instead of minimizing the misfit itself. The KKT conditions of the inner optimization problem may then appear as equality constraints or, to allow inexact solutions, as inequality constraints. In general, the idea of utilizing the estimated misfit and possibly the misfit variance for optimality measure seems to be very interesting.

Even though many of the instances of the PDE-constrained optimization problems are implemented to assemble and process in parallel, the main computational expenses which consist in the solution of the linear system, is currently implemented only for shared memory parallelism by OpenMP. The RSP-GMRES method can be parallelized to solve larger instances of PDE-constrained problems. Since the PDEs in the Schur complement computation are decoupled, the distribution of single PDEs for each MPI process is natural. If a sparse direct solver is used for forward and adjoint solves, this data distribution leads to very similar solution times in each process and a very good scaling is to be expected. For assembling the vectors and matrices, another data distribution according to the computation domain is more convenient, and it depends on the problem at hand if the communication costs for vector and matrix redistribution is beneficial. If the forward and adjoint problems are solved iteratively, which is necessary for large-scale 3D problems, the solvers may need very different iteration numbers for different PDE right-hand sides which causes unbalanced work load. In this context, a distribution of the PDEs as for matrix and vector assembly may be reconsidered. To control the inexactness of PDE solutions, the SMART tests may be extended to distinguish requests for improved solutions for the whole KKT system or for some PDE systems only.

We have applied the RSP-IIP method to various PDE-constrained optimization problems. However, all instances for the RSP-IIP method contain inequality constraints on the PDE parameter only. Here, applying RSP-IIP to inequality constraints also involving the state variable like the ones presented in Section

6.2 is an interesting extension. We have seen that RSP-IIP can solve large-scale PDE-constrained optimization problems very efficiently. However, in the inverse medium problem for $\omega = 60$, there is a single optimization step with a higher iteration count (about twice as high as the for the other KKT systems). For increasingly higher temporal frequencies, this issue appears more often, which poses the desire for more efficient preconditioners for the KKT system. In general the question how to precondition KKT systems is an active area of research.

Bibliography

[1] F. Abraham, M. Behr, and M. Heinkenschloss. Shape optimization in steady blood flow: a numerical study of non-newtonian effects. *Computer Methods in Biomechanics and Biomedical Engineering*, 8(2):127–137, 2005. [cited at p. 4]

[2] F. Abraham, M. Behr, and M. Heinkenschloss. Shape optimization in unsteady blood flow: A numerical study of non-newtonian effects. *Computer Methods in Biomechanics and Biomedical Engineering*, 8(3):201–212, 2005. [cited at p. 4]

[3] G. Al-Jeiroudi, J. Gondzio, and J. Hall. Preconditioning indefinite systems in interior point methods for large scale linear optimisation. *Optimization Methods and Software*, 23(3):345–364, 2008. [cited at p. 75]

[4] G. Allaire, F. Jouve, and A.-M. Toader. Structural optimization using sensitivity analysis and a level-set method. *Journal of Computational Physics*, 194(1):363–393, 2004. [cited at p. 4]

[5] U. Ascher and E. Haber. Computational methods for large distributed parameter estimation problems with possible discontinuities. In *Proc. Symp. Inverse Problems, Design & Optimization*, pages 201–208, 2004. [cited at p. 109]

[6] S. Balay, J. Brown, K. Buschelman, V. Eijkhout, W. D. Gropp, Kaushik D., M. G. Knepley, L. C. McInnes, B. F. Smith, and Zhang H. PETSc users manual. Technical Report ANL-95/11 - Revision 3.1, Argonne National Laboratory, 2010. [cited at p. 70]

[7] S. Balay, J. Brown, K. Buschelman, W. D. Gropp, D. Kaushik, M. G. Knepley, L. C. McInnes, B. F. Smith, and H. Zhang. PETSc web page, 2012. http://www.mcs.anl.gov/petsc. [cited at p. 67, 118]

[8] A. Bayliss, M. Gunzburger, and E. Turkel. Boundary conditions for the numerical solution of elliptic equations in exterior regions. *SIAM Journal on Applied Mathematics*, 42(2):430–451, 1982. [cited at p. 99]

[9] M. Benzi, G. H. Golub, and J. Liesen. Numerical solution of saddle point problems. *Acta Numerica*, 14(14):1–137, 2005. [cited at p. 75]

[10] D. P. Bertsekas. Projected Newton methods for optimization problems with simple constraints. *SIAM Journal on Control and Optimization*, 20(2):221–246, 1982. [cited at p. 13]

[11] G. Biros and O. Ghattas. Inexactness issues in the lagrange-Newton-Krylov-Schur method for PDE-constrained optimization. In L. T. Biegler, O. Ghattas, M. Heinkenschloss, and B. Van Bloemen Waanders, editors, *Large-Scale PDE-Constrained Optimization*, pages 93–114, New York, NY, USA, 2003. Springer. [cited at p. 58]

[12] G. Biros and O. Ghattas. Parallel lagrange-Newton-Krylov-Schur methods for PDE-constrained optimization. part i: The Krylov-Schur solver. *SIAM Journal on Scientific Computing*, 27(2):687–713, 2005. [cited at p. 58]

[13] G. Biros and O. Ghattas. Parallel lagrange-Newton-Krylov-Schur methods for PDE-constrained optimization. part ii: The lagrange-Newton solver and its application to optimal control of steady viscous flows. *SIAM Journal on Scientific Computing*, 27(2):714–739, 2005. [cited at p. 58]

[14] M. Bollhöfer, M. J. Grote, and O. Schenk. Algebraic multilevel preconditioner for the Helmholtz equation in heterogeneous media. *SIAM Journal on Scientific Computing*, 31(5):3781–3805, 2009. [cited at p. 115]

[15] M. Bollhöfer and Y. Saad. Multilevel preconditioners constructed from inverse-based ilus. *SIAM Journal on Scientific Computing*, 27(5):1627–1650, 2006. [cited at p. 56]

[16] C. Bunks, F. M. Saleck, S. Zaleski, and G. Chavent. Multiscale seismic waveform inversion. *Geophysics*, 60(5):1457–1473, 1995. [cited at p. 117]

[17] R. H. Byrd, F. E. Curtis, and J. Nocedal. An inexact Newton method for nonconvex equality constrained optimization. *Mathematical Programming*, 122(2):273–299, 2010. [cited at p. 33, 34, 35, 36, 40]

[18] R. H. Byrd, G. Liu, and J. Nocedal. On the local behavior of an interior point method for nonlinear programming. *Numerical analysis*, 1997:37–56, 1997. [cited at p. 25, 26, 127]

[19] J. F. Claerbout and I. Green. *Basic Earth Imaging*. Citeseer, 2008. [cited at p. 6]

[20] A. K. Cline, C. B. Moler, G. W. Stewart, and J. H. Wilkinson. An estimate for the condition number of a matrix. *SIAM Journal on Numerical Analysis*, 16(2):368–375, 1979. [cited at p. 56]

[21] A. I. Cohen. Rate of convergence of several conjugate gradient algorithms. *SIAM Journal on Numerical Analysis*, 9(2):248–259, 1972. [cited at p. 10]

[22] F. E. Curtis, J. Huber, O. Schenk, and A. Wächter. A note on the implementation of an interior-point algorithm for nonlinear optimization with inexact step computations. *Mathematical Programming*, pages 1–19, 2012. [cited at p. 33, 34, 55]

[23] F. E. Curtis, J. Nocedal, and A. Wächter. A matrix-free algorithm for equality constrained optimization problems with rank-deficient jacobians. *SIAM Journal on Optimization*, 20(3):1224–1249, 2009. [cited at p. 33, 36]

[24] F. E. Curtis, O. Schenk, and A. Wächter. An interior-point algorithm for large-scale nonlinear optimization with inexact step computations. *SIAM Journal on Scientific Computing*, 32(6):3447–3475, 2010. [cited at p. v, 22, 33, 34, 36, 38, 40, 72, 81, 125]

[25] M. Fisher, J. Nocedal, Y. Trémolet, and S. J. Wright. Data assimilation in weather forecasting: a case study in PDE-constrained optimization. *Optimization and Engineering*, 10(3):409–426, 2009. [cited at p. 4]

[26] C. A. Floudas and C. E. Gounaris. A review of recent advances in global optimization. *Journal of Global Optimization*, 45(1):3–38, 2009. [cited at p. 6]

[27] A. Forsgren, P. E. Gill, and M. H. Wright. Interior methods for nonlinear optimization. *SIAM review*, 44(4):525–597, 2002. [cited at p. 25, 26, 29]

[28] R. Fourer, D. M. Gay, and B. W. Kernighan. *AMPL: A Modeling Language for Mathematical Programming*. Brooks/Cole, 2002. [cited at p. 71]

[29] M. Frank, A. Klar, and R. Pinnau. Optimal control of glass cooling using simplified pn theory. *Transport Theory and Statistical Physics*, 39(2-4):282–311, 2011. [cited at p. 4]

[30] R. W. Freund. Preconditioning of symmetric, but highly indefinite linear systems. In A. Sydow, editor, *15th IMACS World Congress on Scientific Computation, Modelling and Applied Mathematics*, pages 551–556, Berlin, 1997. Wissenschaft & Technik. [cited at p. 57, 71]

[31] C. Geiger and C. Kanzow. *Numerische Verfahren zur Lösung Unrestringierter Optimierungsaufgaben*. Springer Verlag, 1999. [cited at p. 7]

[32] C. Geiger and C. Kanzow. *Theorie und Numerik Restringierter Optimierungsaufgaben*, volume 72. Springer, 2002. [cited at p. 7]

[33] N. I. M. Gould, I. Bongartz, A. R. Conn, and Ph. L. Toint. CUTE: Constrained and unconstrained testing environment. *ACM Transactions on Mathematical Software*, 21(1):123–160, 1995. [cited at p. 71]

[34] N. I. M. Gould, D. Orban, and Ph. L. Toint. CUTEr and SifDec: A constrained and unconstrained testing environment, revisited. *ACM Transactions on Mathematical Software*, 29(4):373–394, 2003. [cited at p. 71]

[35] M. J. Grote, J. Huber, D. Kourounis, and O. Schenk. Inexact interior-point methods for nonlinear PDE-constrained optimization. unpublished. [cited at p. 33, 58]

[36] M. J. Grote and T. Huckle. Parallel preconditioning with sparse approximate inverses. *SIAM Journal on Scientific Computing*, 18(3):838–853, 1997. [cited at p. 61]

[37] E. Haber, M. Chung, and F. Herrmann. An effective method for parameter estimation with PDE constraints with multiple right hand sides. `http://www.math.ubc.ca/~haber/EldadHaberWebPage/Publications.html`, 2010. 15.02.2013. [cited at p. 64, 65]

[38] E. Haber and L. Hanson. Model problems in PDE-constrained optimization. Technical report, Emory University, 2007. [cited at p. 5, 8, 85]

[39] M. Heinkenschloss, L. N. Vincente, and L. S. Fernandes. Numerical PDE constrained optimization (lecture notes in computational science & engineering, vol. 64), 2008. [cited at p. 7]

[40] R. Herzog and K. Kunisch. Algorithms for PDE-constrained optimization. *GAMM-Mitteilungen*, 33(2):163–176, 2010. [cited at p. 7]

[41] M. Hinze, R. Pinnau, M. Ulbrich, and S. Ulbrich. *Optimization with PDE Constraints*, volume 23. Springer, 2008. [cited at p. 4, 7]

[42] J. Huber. Zur übertragung des optimierungskonzeptes der support-vector-machine vom reproduzierenden hilbertraum in den originalraum. Master's thesis, University Karlsruhe, 2007. [cited at p. 7]

[43] J. Huber, U. Naumann, O. Schenk, E. Varnik, and A. Wächter. Algorithmic differentiation and nonlinear optimization for an inverse medium problem. In U. Naumann and O. Schenk, editors, *Combinatorial scientific computing*, volume 12, chapter 8, pages 203–231. Chapman & Hall, 2012. [cited at p. 8]

[44] T. Irons. Marmousi model, 2007. `http://www.reproducibility.org/RSF/book/data/marmousi/paperhtml/node1.html`. [cited at p. 117]

[45] J. Kaipio and E. Somersalo. *Statistical and Computational Inverse Problems*, volume 160. Springer, 2004. [cited at p. 109]

[46] G. Karypis. Multi-constraint mesh partitioning for contact/impact computations. In *Proceedings of the 2003 ACM/IEEE conference on Supercomputing*, page 56. ACM, 2003. [cited at p. 70]

[47] G. Karypis and V. Kumar. A fast and high quality multilevel scheme for partitioning irregular graphs. *SIAM Journal on Scientific Computing*, 20(1):359–392, 1998. [cited at p. 56]

[48] B. S. Kirk, J. W. Peterson, R. H. Stogner, and G. F. Carey. `libMesh`: A C++ library for parallel adaptive mesh refinement/coarsening simulations. *Engineering with Computers*, 22(3–4):237–254, 2006. [cited at p. 70, 118]

[49] A. Kirsch. *An Introduction to the Mathematical Theory of Inverse Problems*, volume 120. Springer, 2011. [cited at p. 107]

[50] R. V. Kohn and A. McKenney. Numerical implementation of a variational method for electrical impedance tomography. *Inverse Problems*, 6(3):389, 1999. [cited at p. 5]

[51] H. Maurer and H. D. Mittelmann. Optimization techniques for solving elliptic control problems with control and state constraints: Part 1. boundary control. *Computational Optimization and Applications*, 16:29–55, 2000. 10.1023/A:1008725519350. [cited at p. 84]

[52] D. Meyer. Newton-Krylov methods for inverse acoustic 1d wave propagation. Master's thesis, University Basel, 2010. [cited at p. 8]

[53] F. Natterer. Reflection imaging without low frequencies. *Inverse Problems*, 27(3):035011, 2011. [cited at p. 6]

[54] A. Neumaier. Complete search in continuous global optimization and constraint satisfaction. *Acta Numerica*, 13(1):271–369, 2004. [cited at p. 6]

[55] J. Nocedal and S. J. Wright. *Numerical Optimization*. Springer verlag, 1999. [cited at p. 7, 20, 21, 58]

[56] G. Pratt, C. Shin, et al. Gauss–Newton and full Newton methods in frequency–space seismic waveform inversion. *Geophysical Journal International*, 133(2):341–362, 2002. [cited at p. 6]

[57] T. Rees, H. S. Dollar, and A. J. Wathen. Optimal solvers for PDE-constrained optimization. *SIAM Journal on Scientific Computing*, 32(1):271–298, 2010. [cited at p. 82]

[58] T. Rees, H. S. Dollar, and A. J. Wathen. Tyrone rees, 2010. [cited at p. 82]

[59] K. Roser. Optimierungsverfahren zur lösung der inversen Helmholtz-gleichung. Master's thesis, University Basel, 2011. [cited at p. 10]

[60] J. A. Samareh. A survey of shape parameterization techniques. In *NASA CONFERENCE PUBLICATION*, pages 333–344. Citeseer, 1999. [cited at p. 4]

[61] O. Schenk, M. Bollhöfer, and R. A. Römer. On large scale diagonalization techniques for the Anderson model of localization. *SIAM Journal on Scientific Computing*, 28(3):963–983, 2006. [cited at p. 56, 57]

[62] O. Schenk, M. Bollhöfer, and R. A. Römer. On large-scale diagonalization techniques for the Anderson model of localization. *SIAM Review*, 50(1):91–112, 2008. [cited at p. 56]

[63] O. Schenk and K. Gärtner. On fast factorization pivoting methods for sparse symmetric indefinite systems. *Electronic Transactions on Numerical Analysis*, 23:158–179, 2006. [cited at p. 29, 56, 84]

[64] O. Schenk, A. Wächter, and M. Hagemann. Matching-based preprocessing algorithms to the solution of saddle-point problems in large-scale nonconvex interior-point optimization. *Computational Optimization and Applications*, 36(2-3):321–341, April 2007. [cited at p. 57]

[65] O. Schenk, A. Wächter, and M. Weiser. Inertia revealing preconditioning for large-scale nonconvex constrained optimization. *SIAM Journal on Scientific Computing*, 31(2):939–960, 2008. [cited at p. 56]

[66] R.A. Schowengerdt. *Remote Sensing: Models and Methods for Image Processing*. Academic Press, 2006. [cited at p. 118]

[67] L. Sirgue and R.G. Pratt. Efficient waveform inversion and imaging: A strategy for selecting temporal frequencies. *Geophysics*, 69(1):231–248, 2004. [cited at p. 6, 117]

[68] F. Sourbier, S. Operto, J. Virieux, P. Amestoy, and J.-Y. LExcellent. Fwt2d: A massively parallel program for frequency-domain full-waveform tomography of wide-aperture seismic datapart 1: Algorithm. *Computers & Geosciences*, 35(3):487–495, 2009. [cited at p. 6]

[69] F. Sourbier, S. Operto, J. Virieux, P. Amestoy, and J.-Y. LExcellent. Fwt2d: A massively parallel program for frequency-domain full-waveform tomography of wide-aperture seismic datapart 2: Numerical examples and scalability analysis. *Computers & Geosciences*, 35(3):496–514, 2009. [cited at p. 6]

[70] T. Steihaug. The conjugate gradient method and trust regions in large scale optimization. *SIAM Journal on Numerical Analysis*, 20(3):626–637, 1983. [cited at p. 11]

[71] A. Tarantola. Inversion of seismic reflection data in the acoustic approximation. *Geophysics*, 49(8):1259–1266, 1984. [cited at p. 6]

[72] A. Tarantola. Linearized inversion of seismic reflection data. *Geophysical Prospecting*, 32:9981015, 1984. [cited at p. 6]

[73] M. Tinkham. *Introduction to Superconductivity: Second Edition*. Dover books on physics and chemistry. Dover Publications, 2004. [cited at p. 84]

[74] F. Tröltzsch. *Optimal Control of Partial Differential Equations: Theory, Methods, and Applications*, volume 112. American Mathematical Society, 2010. [cited at p. 72]

[75] R. Versteeg. The Marmousi experience: Velocity model determination on a synthetic complex data set. *The Leading Edge*, 13:927, 1994. [cited at p. 117]

[76] C. R. Vogel. Sparse matrix computations arising in distributed parameter identification. *SIAM J. Matrix Analysis and Applications*, 20:1027–1037, 1999. [cited at p. 85]

[77] A. Wächter and L. T. Biegler. Failure of global convergence for a class of interior point methods for nonlinear programming. *Mathematical Programming*, 88(3):565–574, 2000. [cited at p. 72]

[78] A. Wächter and L. T. Biegler. Line search filter methods for nonlinear programming: Local convergence. *SIAM Journal on Optimization*, 16(1):32–48, 2005. [cited at p. 27]

[79] A. Wächter and L. T. Biegler. Line search filter methods for nonlinear programming: Motivation and global convergence. *SIAM Journal on Optimization*, 16(1):1–31, 2005. [cited at p. 27, 71]

[80] A. Wächter and L. T. Biegler. On the implementation of an interior-point filter fine search algorithm for large-scale nonlinear programming. *Mathematical Programming*, 106(1):25–57, 2006. [cited at p. 22, 27, 28, 71, 84]

[81] S. Xu and G. Lambare. Fast migration/inversion with multivalued rayfields: Part imethod, validation test, and application in 2d to marmousi. *Geophysics*, 69(5):1311–1319, 2004. [cited at p. 117]

Curriculum Vitae

Personal Data

Name	Johannes Huber
Date of birth	November 18, 1973, Oppenau, Germany
Nationality	German
Family Status	unmarried

Practical Experience

1995 – 1996	Internship at Motan Holding GmbH, Konstanz, Germany
1997 – 1998	Assistant at Fraunhofer IPM, Freiburg, Germany
1998 – 2002	Software development at Erwin Junker Maschinenfabrik, Nordrach, Germany
2002 – 2008	Software development at Fraunhofer IOSB (formely FOM), Ettlingen, Germany
2006	Software development in the Interdisciplinary Research Center at A&T North Carolina State University, Greensboro, USA
2010	Internship at IBM Watson Research Center, White Plains, USA

Education

1980 – 1985	Grund- und Hauptschule Oppenau, Germany
1985 – 1990	Realschule Oberkirch, Germany
1990 – 1992	Apprenticeship as chemical-technical assistant, NTA Isny, Germany
1993 – 1997	Diplom Engineer in physics, NTA Isny, Germany
2002 – 2008	Diploma in Mathematics, minors in Computer Science and Physics at the University of Karlsruhe, Germany
2008 – 2013	Ph.D. candidate in Mathematics at the University of Basel, Switzerland

Scholarships and Honors

1993 – 1997	Scholarship from the "Förderverein der NTA Isny"
2005 – 2006	Fulbright Scholarship Program
2008	Award of the Department of Mathematics at Karlsruhe Institute of Technology
2008	Scholarship from the "Heinrich-Hertz-Gesellschaft"

I enjoyed attending the lectures of the following professors and lecturers:
Alefeld, Aumann, Berger, Bergler, Bililign, Bohlender, Borah, Donges, Gantert, Gasparian, Grote, Kirsch, Klein, Kühnlein, Längle, Leuzinger, Lipps, v. Löhneysen, Plum, v. Renteln, Rieder, Röschentaler, Schönauer, Sorber, Tang, Xu